راهنمای مرجع و پروژه‌محور

آموزش جامع HTML
از صفر تا پیشرفته

ساختار، معنای عناصر، فرم‌ها، رسانه، Semantic HTML،
Accessibility، SEO، Performance، امنیت و پروژه‌های عملی؛ همراه
با مثال‌های قابل کپی و نکته‌های کاربردی.

مبتدی تا پیشرفته
مثال کدنویسی
پروژه عملی
چک‌لیست نهایی

فهرست مطالب

24. فصل ۲۴ - طراحی فرم خرید و فروشگاه

25. فصل ۲۵ - پروژهٔ عملی: صفحهٔ معرفی شخصی

26. فصل ۲۶ - پروژهٔ عملی: صفحهٔ محصول

27. فصل ۲۷ - پروژهٔ عملی: FAQ و منوی بازشونده بدون JavaScript

28. فصل ۲۸ - خطاهای بسیار رایج و نسخهٔ درست آن‌ها

29. فصل ۲۹ - HTML مدرن و چیزهایی که باید بررسی کنی

30. فصل ۳۰ - چک‌لیست نهایی یک فایل HTML حرفه‌ای

31. فصل ۳۱ - ویژگی‌های سراسری و Attributeهای حرفه‌ای

32. فصل ۳۲ - کارگاه پروژهٔ جامع HTML واقعی

33. فصل ۳۳ - نحو و مدل پردازش HTML در عمق

34. فصل ۳۴ - مرجع کامل عناصر متنی HTML

35. فصل ۳۵ - لینک‌ها، URLها و ناوبری پیشرفته

36. فصل ۳۶ - تصاویر پیشرفته و Responsive Images

37. فصل ۳۷ - صوت، ویدئو و زیرنویس در سطح حرفه‌ای

38. فصل ۳۸ - فهرست‌ها و جدول‌های پیچیده

39. فصل ۳۹ - مرجع کامل فرم‌ها و کنترل‌های ورودی

40. فصل ۴۰ - فرم و مرورگر: داده، نام، ارسال و FormData

41. فصل ۴۱ - Head حرفه‌ای: base، script، link و metadata

42. فصل ۴۲ - Accessibility پیشرفته و طراحی برای فناوری کمکی

43. فصل ۴۳ - SEO عمیق، ساختار محتوا و دادهٔ ساختاریافته

44. فصل ۴۴ - امنیت وب در لایهٔ HTML

45. فصل ۴۵ - HTML، DOM، Browser APIs و JavaScript

46. فصل ۴۶ - SVG، MathML و Canvas در HTML

47. فصل ۴۷ - Web Components در سطح حرفه‌ای

48. فصل ۴۸ - PWA، manifest و قابلیت نصب

49. فصل ۴۹ - بین‌المللی‌سازی، RTL و چندزبانی حرفه‌ای

50. فصل ۵۰ - اعتبارسنجی، تست، دیباگ و استاندارد حرفه‌ای HTML

51. مرجع نهایی - عناصر، ویژگی‌ها و الگوهای سریع

52. ضمیمه A - Cheat Sheet سریع HTML

53. ضمیمه B - مسیر یادگیری سریع از صفر تا تسلط

فصل 01

فصل ۱ - HTML چیست؟ و چرا مهم است؟

HTML زبان نشانه‌گذاری استاندارد برای ساختاردهی محتوای صفحات وب است. در این فصل از تصویر ذهنی درست HTML، نقش آن در مرورگر و تفاوت آن با CSS و JavaScript شروع می‌کنیم.

HTML دقیقاً چه کاری انجام می‌دهد؟

HTML مشخص می‌کند هر بخش از صفحه چه معنایی دارد: عنوان، پاراگراف، لینک، تصویر، فرم، جدول، مقاله، ناوبری و... . مرورگر این نشانه‌ها را می‌خواند و درختی از عناصر به نام DOM می‌سازد.

```
<h1>سلام دنیا</h1>
<p>این اولین صفحهٔ من است.</p>
```

کاربرد عملی: در پروژهٔ واقعی فقط نمایش تصویر مهم نیست؛ باید متن جایگزین، اندازهٔ مناسب، نسبت تصویر، بارگذاری تنبل در موارد مناسب و نسخهٔ مناسب برای صفحه‌نمایش‌های مختلف را هم مدیریت کنی.
نکات دقیق و خطاهای رایج: برای تصاویر تزئینی معمولاً alt خالی بهتر از متن ساختگی است؛ برای تصویرِ دارای اطلاعات، alt باید همان اطلاعات مهم را منتقل کند. همچنین width و height از تغییر ناگهانی چیدمان جلوگیری می‌کنند.

```
<img src="product.webp" alt="گوشی مشکی با نمایشگر 6.4 اینچی" width="640"
height="480" loading="lazy">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

HTML، CSS و JavaScript چه فرقی دارند؟

HTML = ساختار و معنا

CSS = ظاهر و چیدمان

JavaScript = رفتار و تعامل

برای یک سایت حرفه‌ای معمولاً هر سه کنار هم قرار می‌گیرند؛ اما یادگیری صحیح با HTML شروع می‌شود.

```
<!-- HTML -->
<button>خرید</button>

/* CSS */
button { padding: 12px; }

// JavaScript
document.querySelector("button").addEventListener("click", () =>
alert("سلام!"));
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «HTML، CSS و JavaScript چه فرقی دارند؟» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

قانون طلایی

HTML را برای **معنا** بنویس، نه برای ظاهر. اگر چیزی تیتر است از heading استفاده کن؛ اگر لینک است از a؛ اگر دکمه است از button. ظاهر را بعداً با CSS بساز.

```
<h2>محصولات</h2>
<a href="/products">مشاهدهٔ همه</a>
<button type="button">افزودن به سبد</button>
```

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.
نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 02

فصل ۲ - ساختار یک سند استاندارد HTML

این فصل ستون فقرات هر فایل HTML را می‌سازد. هدف این است که بدون حفظ کردن کورکورانه، بدانید هر خط چرا وجود دارد.

قالب استاندارد

اعلام DOCTYPE باعث می‌شود مرورگر سند را در حالت استاندارد پردازش کند. عنصر html ریشهٔ سند است و head اطلاعاتی برای مرورگر و موتور جست‌وجو نگه می‌دارد؛ body محتوای قابل مشاهده را در خود دارد.

```
<!doctype html>
<html lang="fa" dir="rtl">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>صفحهٔ من</title>
</head>
<body>
  <h1>سلام وب!</h1>
</body>
</html>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «قالب استاندارد» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

lang و dir

برای صفحهٔ فارسی، lang="fa" زبان را معرفی می‌کند و dir="rtl" جهت متن را راست‌به‌چپ می‌کند. این اطلاعات برای دسترس‌پذیری، خوانش صحیح و برخی ابزارها مهم‌اند.

```
<html lang="fa" dir="rtl">
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

meta charset

UTF-8 را برای پشتیبانی از فارسی، انگلیسی، ایموجی و طیف گسترده‌ای از نویسه‌ها مشخص کن. بهتر است این meta در ابتدای head قرار گیرد.

```
<meta charset="utf-8">
```

کاربرد عملی: اطلاعات head بخشی از قرارداد صفحه با مرورگر، موتور جست‌وجو، شبکه‌های اجتماعی و ابزارهای اشتراک‌گذاری است. یک title خوب، canonical صحیح و metadata تمیز از مهم‌ترین نقاط شروع‌اند.
نکات دقیق و خطاهای رایج: هیچ metaای جای محتوای واقعی را نمی‌گیرد. canonical باید به URL مرجع همان محتوا اشاره کند و در صفحات مشابه باید آگاهانه تنظیم شود.

```
<title>آموزش HTML نام سایت | پیشرفته تا صفر</title>
<meta name="description" content="راهنمای جامع و پروژه‌محور HTML.">
<link rel="canonical" href="https://example.com/html">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

viewport برای موبایل

بدون viewport بسیاری از صفحات در موبایل مثل یک صفحهٔ دسکتاپ کوچک‌شده رفتار می‌کنند. مقدار استاندارد زیر نقطهٔ شروع طراحی واکنش‌گراست.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «viewport برای موبایل» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.
نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 03

فصل ۳ - عناصر، تگ‌ها، محتوا و ویژگی‌ها

در HTML باید بین عنصر، تگ، محتوا و attribute تفاوت بگذاری. این تفاوت کوچک، پایهٔ فهم درست کل زبان است.

عنصر و تگ

در عبارت `<p>سلام</p>` تگ شروع و پایان، عنصر را می‌سازند و «سلام» محتوای آن است.

```
<p>سلام</p>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «عنصر و تگ» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.
نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

ویژگی‌ها یا Attributes

attribute اطلاعات اضافی دربارهٔ عنصر می‌دهد. معمولاً به شکل نام="مقدار" نوشته می‌شود. بعضی ویژگی‌ها Boolean هستند و با حضورشان فعال می‌شوند.

```
<a href="https://example.com" target="_blank" rel="noopener">سایت</a>
<input disabled>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «ویژگی‌ها یا Attributes» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.
نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

id و class

id باید برای یک عنصر در صفحه یکتا باشد؛ class برای گروه‌بندی عناصر و اعمال سبک یا انتخاب چند عنصر کاربرد دارد.

```
<section id="about" class="card featured">...</section>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «id و class» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.
نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

Global attributes

ویژگی‌هایی مثل id، class، title، hidden، lang، dir، data-* و tabindex در عناصر متنوعی قابل استفاده‌اند. از tabindex فقط وقتی واقعاً نیاز داری استفاده کن.

```
<div id="box" class="card" data-product-id="42" title="محصول کوتاه">توضیح کوتاه</div>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 04

فصل ۴ - Heading، پاراگراف، متن و قالب‌بندی معنایی

تیترها فقط برای بزرگ‌کردن فونت نیستند؛ آن‌ها ساختار سند را تعریف می‌کنند و به کاربر، موتور جست‌وجو و فناوری‌های کمکی کمک می‌کنند.

تیترهای h1 تا h6

هر صفحه باید یک ساختار منطقی از headingها داشته باشد. از h2 برای زیرعنوان‌های h1 و از h3 برای زیربخش h2 استفاده کن. تعداد h1 به‌تنهایی مسئلهٔ اصلی نیست؛ انسجام ساختار مهم‌تر است.

```
<h1>راهنمای HTML</h1>
<h2>مبانی</h2>
<h3>عناصر</h3>
<h2>فرم‌ها</h2>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

تأکید معنایی

اهمیت برای تأکید معنایی و تأکید برای stress استفاده می‌شود. صرفاً برای ضخیم یا ایتالیک کردن متن سراغ این عناصر نرو؛ CSS برای ظاهر است.

```
<p><strong>هشدار:</strong> قبل از حذف فایل پشتیبان بگیر.</p>
<p>این بخش <em>خیلی</em> مهم است.</p>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

br، hr و white-space

از br برای شکستن طبیعی خط در مواردی مثل شعر یا آدرس استفاده کن، نه برای فاصله‌سازی. hr یک جداکنندهٔ معنایی است. برای فاصله و چیدمان از CSS استفاده می‌شود.

```
<p>خیابان اصلی<br>کوچهٔ دوم<br>پلاک ۱۰</p>
<hr>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «br، hr و white-space» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.
نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

sup و sub

برای توان و اندیس شیمیایی مناسب‌اند و معنای متن را حفظ می‌کنند.

```
<p>x<sup>2</sup> + H<sub>2</sub>0</p>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

blockquote و cite

blockquote برای نقل‌قول بلوکی و cite برای نام اثر یا منبع استفاده می‌شوند.

```
<blockquote cite="https://example.com">دانش یعنی توانایی استفاده از اطلاعات.</
blockquote>
<cite>یک نقل‌قول آموزشی</cite>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل
آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 05

فصل ۵ - لینک‌ها و ناوبری

لینک، یکی از مهم‌ترین ایده‌های وب است. یاد می‌گیریم لینک داخلی، خارجی، ایمیل، تلفن، فایل، anchor و ویژگی‌های امنیتی آن را درست بنویسیم.

لینک ساده

ویژگی href مقصد لینک را مشخص می‌کند. متن لینک باید قابل فهم باشد؛ «اینجا کلیک کن» معمولاً اطلاعات کافی نمی‌دهد.

```
<a href="/contact">تماس با ما</a>
```

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.
نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

لینک خارجی امن

برای لینک بازشونده در تب جدید، rel="noopener" از دسترسی صفحهٔ مقصد به window.opener جلوگیری می‌کند. در پروژه‌های واقعی این الگو انتخاب مطمئنی است.

```
<a href="https://example.com" target="_blank" rel="noopener
noreferrer">وب‌سایت مرجع</a>
```

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.
نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع
خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

Anchor یا پرش داخل صفحه

یک id روی مقصد قرار بده و با #id به آن لینک بده.

```
<a href="#faq">سؤالات متداول</a>
<section id="faq"><h2>سؤالات متداول</h2></section>
```

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.

نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

mailto و tel

برای شروع نرم‌افزار ایمیل و تماس در دستگاه‌های پشتیبانی‌شده استفاده می‌شوند.

```
<a href="mailto:test@example.com">ایمیل</a>
<a href="tel:+989121234567">تماس</a>
```

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.

نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

دانلود فایل

وجود download مرورگر را به دانلود فایل راهنمایی می‌کند. نام پیشنهادی نیز می‌تواند در همین ویژگی تعیین شود.

```
<a href="/files/guide.pdf" download="guide.pdf">دانلود راهنما</a>
```

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.
نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 06

فصل ۶ - تصاویر و رسانه‌ها

تصویر فقط یک فایل تزئینی نیست. نام جایگزین، ابعاد، فرمت و روش بارگذاری همگی بر دسترس‌پذیری و عملکرد اثر می‌گذارند.

img و alt

alt برای توصیف محتوای معنادار تصویر است. اگر تصویر صرفاً تزئینی است، alt="" مناسب‌تر است تا صفحه‌خوان آن را دوباره اعلام نکند.

```
<img src="images/product.webp" alt="مدل گوشی مشکی روی میز X" width="800"
height="600">
```

کاربرد عملی: در پروژهٔ واقعی فقط نمایش تصویر مهم نیست؛ باید متن جایگزین، اندازهٔ مناسب، نسبت تصویر، بارگذاری تنبل در موارد مناسب و نسخهٔ مناسب برای صفحه‌نمایش‌های مختلف را هم مدیریت کنی.
نکات دقیق و خطاهای رایج: برای تصاویر تزئینی معمولاً alt خالی بهتر از متن ساختگی است؛ برای تصویرِ دارای اطلاعات، alt باید همان اطلاعات مهم را منتقل کند. همچنین width و height از تغییر ناگهانی چیدمان جلوگیری می‌کنند.

```
<img src="product.webp" alt="گوشی مشکی با نمایشگر 6.4 اینچی" width="640"
height="480" loading="lazy">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

ابعاد تصویر

تعیین width و height یا نسبت تصویر در CSS به مرورگر اجازه می‌دهد فضای تصویر را از قبل رزرو کند و از جابه‌جایی شدید layout جلوگیری شود.

```
<img src="photo.jpg" alt="منظرهٔ کوهستان" width="1200" height="800"
loading="lazy">
```

کاربرد عملی: در پروژهٔ واقعی فقط نمایش تصویر مهم نیست؛ باید متن جایگزین، اندازهٔ مناسب، نسبت تصویر، بارگذاری تنبل در موارد مناسب و نسخهٔ مناسب برای صفحه‌نمایش‌های مختلف را هم مدیریت کنی.
نکات دقیق و خطاهای رایج: برای تصاویر تزئینی معمولاً alt خالی بهتر از متن ساختگی است؛ برای تصویرِ دارای اطلاعات، alt باید همان اطلاعات مهم را منتقل کند. همچنین width و height از تغییر ناگهانی چیدمان جلوگیری می‌کنند.

```
<img src="product.webp" alt="گوشی مشکی با نمایشگر 6.4 اینچی" width="640"
height="480" loading="lazy">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

تصویر واکنش‌گرا با picture

picture اجازه می‌دهد برای صفحه‌نمایش‌ها یا فرمت‌های مختلف منبع مناسب ارائه شود.

```
<picture>
  <source srcset="photo.avif" type="image/avif">
  <source srcset="photo.webp" type="image/webp">
  <img src="photo.jpg" alt="منظرهٔ کوهستان">
</picture>
```

کاربرد عملی: در پروژهٔ واقعی فقط نمایش تصویر مهم نیست؛ باید متن جایگزین، اندازهٔ مناسب، نسبت تصویر، بارگذاری تنبل در موارد مناسب و نسخهٔ مناسب برای صفحه‌نمایش‌های مختلف را هم مدیریت کنی.
نکات دقیق و خطاهای رایج: برای تصاویر تزئینی معمولاً alt خالی بهتر از متن ساختگی است؛ برای تصویرِ دارای اطلاعات، alt باید همان اطلاعات مهم را منتقل کند. همچنین width و height از تغییر ناگهانی چیدمان جلوگیری می‌کنند.

```
<img src="product.webp" alt="گوشی مشکی با نمایشگر 6.4 اینچی" width="640"
height="480" loading="lazy">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

audio و video

برای پخش رسانه از عناصر بومی مرورگر استفاده کن. controls رابط کاربری پخش را فعال می‌کند.

```
<video controls width="720" preload="metadata">
  <source src="lesson.mp4" type="video/mp4">
  مرورگر شما از ویدئو پشتیبانی نمی‌کند.
</video>
```

کاربرد عملی: رسانهٔ وب یک موضوع فنی چندبخشی است: منبع فایل، فرمت، کنترل پخش، دسترس‌پذیری، زیرنویس، بارگذاری، پوستر و سازگاری مرورگر باید با هم دیده شوند.

نکات دقیق و خطاهای رایج: برای محتوای آموزشی زیرنویس، کنترل‌های واضح و fallback مناسب مهم‌اند. autoplay به‌خصوص برای صدا باید با احتیاط استفاده شود.

```
<video controls preload="metadata" poster="poster.jpg" width="720">
  <source src="lesson.mp4" type="video/mp4">
  <track src="fa.vtt" kind="subtitles" srclang="fa" label="فارسی">
</video>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

track برای زیرنویس

زیرنویس و کپشن به دسترس‌پذیری و فهم ویدئو کمک می‌کنند.

```
<video controls>
  <source src="course.mp4" type="video/mp4">
  <track src="fa.vtt" kind="subtitles" srclang="fa" label="فارسی">
</video>
```

کاربرد عملی: رسانهٔ وب یک موضوع فنی چندبخشی است: منبع فایل، فرمت، کنترل پخش، دسترس‌پذیری، زیرنویس، بارگذاری، پوستر و سازگاری مرورگر باید با هم دیده شوند.

نکات دقیق و خطاهای رایج: برای محتوای آموزشی زیرنویس، کنترل‌های واضح و fallback مناسب مهم‌اند. autoplay به‌خصوص برای صدا باید با احتیاط استفاده شود.

```
<video controls preload="metadata" poster="poster.jpg" width="720">
  <source src="lesson.mp4" type="video/mp4">
  <track src="fa.vtt" kind="subtitles" srclang="fa" label="فارسی">
</video>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 07

فصل ۷ - فهرست‌ها

فهرست‌ها در منوها، ویژگی‌ها، مراحل و اطلاعات دسته‌بندی‌شده کاربرد زیادی دارند.

لیست نامرتب

برای مواردی که ترتیبشان مهم نیست.

```
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «لیست نامرتب» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

لیست مرتب

برای مراحل، رتبه‌ها یا دستورالعمل‌های دارای ترتیب.

```
<ol>
  <li>فایل را بساز.</li>
  <li>کد را بنویس.</li>
  <li>در مرورگر باز کن.</li>
</ol>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «لیست مرتب» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

لیست توضیحی

برای زوج‌های term/description مانند واژه‌نامه.

```
<dl>
  <dt>HTML</dt>
  <dd>ساختار و معنای صفحهٔ وب.</dd>
  <dt>CSS</dt>
  <dd>ظاهر و چیدمان صفحه.</dd>
</dl>
```

کاربرد عملی: اطلاعات head بخشی از قرارداد صفحه با مرورگر، موتور جست‌وجو، شبکه‌های اجتماعی و ابزارهای اشتراک‌گذاری است. یک title خوب، canonical صحیح و metadata تمیز از مهم‌ترین نقاط شروع‌اند.

نکات دقیق و خطاهای رایج: هیچ metaای جای محتوای واقعی را نمی‌گیرد. canonical باید به URL مرجع همان محتوا اشاره کند و در صفحات مشابه باید آگاهانه تنظیم شود.

```
<title>آموزش HTML نام سایت | از صفر تا پیشرفته</title>
<meta name="description" content="راهنمای جامع و پروژه‌محور HTML.">
<link rel="canonical" href="https://example.com/html">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 08

فصل ۸ - جدول‌ها

جدول برای دادهٔ جدولی است، نه چیدمان صفحه. با table، caption، thead، tbody، tfoot، th و scope جدول قابل فهم‌تری می‌سازیم.

جدول استاندارد

استفاده از بخش‌بندی جدول خوانایی و دسترس‌پذیری را بهتر می‌کند.

```
<table>
  <caption>نتایج آزمون</caption>
  <thead>
    <tr><th scope="col">نام</th><th scope="col">نمره</th></tr>
  </thead>
  <tbody>
    <tr><th scope="row">علی</th><td>19.5</td></tr>
    <tr><th scope="row">سارا</th><td>20</td></tr>
  </tbody>
</table>
```

کاربرد عملی: جدول برای دادهٔ دوبعدی و رابطهٔ سطر/ستون مناسب است، نه برای چیدمان صفحه. ساختار صحیح جدول شامل عنوان، سرستون‌های واضح و در صورت نیاز ارتباط دقیق headerهاست.

نکات دقیق و خطاهای رایج: هرچه جدول پیچیده‌تر شود، استفاده از scope و در موارد خاص id/headers مهم‌تر می‌شود. برای موبایل نیز رفتار overflow را با CSS حل کن، نه با حذف داده‌ها.

```
<table>
  <caption>فروش ماهانه</caption>
  <thead><tr><th scope="col">ماه</th><th scope="col">فروش</th></tr></thead>
  <tbody><tr><td>فروردین</td><td>120</td></tr></tbody>
</table>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

colspan و rowspan

برای ادغام سلول‌ها در دادهٔ واقعاً جدولی استفاده می‌شوند؛ با دقت زیاد چون پیچیدگی خواندن جدول را بالا می‌برند.

```
<tr><th colspan="2">گزارش ماهانه</th></tr>
<tr><td rowspan="2">مهر</td><td>فروش</td></tr>
```

کاربرد عملی: جدول برای دادهٔ دوبعدی و رابطهٔ سطر/ستون مناسب است، نه برای چیدمان صفحه. ساختار صحیح جدول شامل عنوان، سرستون‌های واضح و در صورت نیاز ارتباط دقیق headerهاست.
نکات دقیق و خطاهای رایج: هرچه جدول پیچیده‌تر شود، استفاده از scope و در موارد خاص id/headers مهم‌تر می‌شود. برای موبایل نیز رفتار overflow را با CSS حل کن، نه با حذف داده‌ها.

```
<table>
  <caption>فروش ماهانه</caption>
  <thead><tr><th scope="col">ماه</th><th scope="col">فروش</th></tr></thead>
  <tbody><tr><td>فروردین</td><td>120</td></tr></tbody>
</table>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

اشتباه رایج

ساخت layout با table امروزه روش درستی نیست. برای layout از CSS Grid/Flexbox استفاده کن.

```
<!-- از table برای layout نکن استفاده -->
```

کاربرد عملی: جدول برای دادهٔ دوبعدی و رابطهٔ سطر/ستون مناسب است، نه برای چیدمان صفحه. ساختار صحیح جدول شامل عنوان، سرستون‌های واضح و در صورت نیاز ارتباط دقیق headerهاست.
نکات دقیق و خطاهای رایج: هرچه جدول پیچیده‌تر شود، استفاده از scope و در موارد خاص id/headers مهم‌تر می‌شود. برای موبایل نیز رفتار overflow را با CSS حل کن، نه با حذف داده‌ها.

```
<table>
  <caption>فروش ماهانه</caption>
  <thead><tr><th scope="col">ماه</th><th scope="col">فروش</th></tr></thead>
  <tbody><tr><td>فروردین</td><td>120</td></tr></tbody>
</table>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 09

فصل ۹ - فرم‌ها از پایه تا حرفه‌ای

فرم‌ها جایی هستند که HTML مستقیماً با دادهٔ کاربر ارتباط می‌گیرد. انتخاب type درست input، label و name مهم‌تر از ظاهر آن است.

ساختار فرم

هر ورودی meaningful باید label داشته باشد. name برای ارسال داده بسیار مهم است.

```
<form action="/signup" method="post">
  <label for="name">نام</label>
  <input id="name" name="name" type="text">
  <button type="submit">ثبت نام</button>
</form>
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

input typeهای مهم

از type مناسب استفاده کن تا مرورگر validation و keyboard مناسب‌تری در اختیار کاربر قرار دهد.

```
<input type="email" name="email">
<input type="password" name="password">
<input type="number" name="age">
<input type="date" name="birthdate">
<input type="file" name="avatar" accept="image/*">
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

required، minlength، maxlength

بخشی از اعتبارسنجی اولیهٔ سمت مرورگر را می‌توان با attributeها انجام داد؛ ولی validation سمت سرور همچنان ضروری است.

```
<input name="username" required minlength="3" maxlength="30">
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت‌نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

radio و checkbox

radio برای انتخاب یک گزینه از گروه و checkbox برای انتخاب صفر یا چند گزینه مناسب است.

```
<fieldset>
  <legend>نوع حساب</legend>
  <label><input type="radio" name="plan" value="free"> رایگان</label>
  <label><input type="radio" name="plan" value="pro"> حرفه‌ای</label>
</fieldset>
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت‌نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

select و textarea

select برای انتخاب از گزینه‌های از پیش تعریف‌شده و textarea برای متن چندخطی است.

```
<label for="city">شهر</label>
<select id="city" name="city">
  <option value="tehran">تهران</option>
  <option value="shiraz">شیراز</option>
</select>
<label for="message">پیام</label>
<textarea id="message" name="message" rows="5"></textarea>
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت‌نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

datalist

داده‌های پیشنهادی را نشان می‌دهد ولی همچنان به کاربر اجازهٔ وارد کردن مقدار می‌دهد.

```
<label for="browser">مرورگر</label>
<input id="browser" name="browser" list="browsers">
<datalist id="browsers">
  <option value="Chrome">
  <option value="Firefox">
</datalist>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

button type

درون form، دکمهٔ بدون type به طور پیش‌فرض ممکن است submit باشد. type را صریح بنویس.

```
<button type="submit">ارسال</button>
<button type="reset">پاک کردن</button>
<button type="button">عملیات جاوااسکریپت</button>
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت‌نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

fieldset و legend

برای گروه‌بندی منطقی کنترل‌های فرم بسیار مفیدند، به‌خصوص برای کاربران صفحه‌خوان.

```
<fieldset>
  <legend>اطلاعات تماس</legend>
  ...
</fieldset>
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نام قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 10

فصل ۱۰ - Semantic HTML و ساختار حرفه‌ای صفحه

Semantic HTML یعنی به‌جای divهای بی‌معنا از عناصر دارای نقش معنایی استفاده کنیم. این کار نگهداری، SEO و accessibility را بهتر می‌کند.

الگوی استاندارد صفحه

header برای سربرگ، nav برای ناوبری، main برای محتوای اصلی، article برای محتوای مستقل، section برای بخش موضوعی، aside برای محتوای جانبی و footer برای پاورقی مناسب‌اند.

```
<body>
<header>...</header>
<nav aria-label="اصلی">...</nav>
<main>
  <article>...</article>
  <aside>...</aside>
</main>
<footer>...</footer>
</body>
```

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.
نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

section یا div؟

section معمولاً یک بخش موضوعی با heading دارد؛ div یک container عمومی بدون معنای خاص است. هر جا عنصر معنایی مناسب وجود دارد، ابتدا همان را بررسی کن.

```
<section aria-labelledby="features-title">
  <h2 id="features-title">ویژگی‌ها</h2>
</section>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

article

محتوایی که به‌تنهایی قابل توزیع یا درک است مثل مقاله، خبر، پست وبلاگ، نقد محصول یا کارت محتوایی.

```
<article>
  <h2>راهنمای انتخاب لپ‌تاپ</h2>
  <p>...</p>
</article>
```

کاربرد عملی: ساختار معنایی باعث می‌شود محتوای صفحه برای انسان، موتور جست‌وجو و فناوری کمکی قابل درک‌تر باشد. سؤال اصلی همیشه این است که «این بخش چه نقشی دارد؟».

نکات دقیق و خطاهای رایج: از تودرتو کردن sectionها بدون عنوان معنادار پرهیز کن و div را زمانی نگه دار که فقط یک ظرف عمومی لازم است.

```
<main>
  <article>
    <h2>عنوان مقاله</h2>
    <p>محتوای مستقل مقاله...</p>
  </article>
</main>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

figure و figcaption

برای رسانه‌ای که توضیح یا caption دارد بسیار مناسب است.

```
<figure>
  <img src="diagram.png" alt="نمودار رشد فروش">
  <figcaption>نمودار فروش سه‌ماههٔ اول</figcaption>
</figure>
```

کاربرد عملی: رسانهٔ وب یک موضوع فنی چندبخشی است: منبع فایل، فرمت، کنترل پخش، دسترس‌پذیری، زیرنویس، بارگذاری، پوستر و سازگاری مرورگر باید با هم دیده شوند.

نکات دقیق و خطاهای رایج: برای محتوای آموزشی زیرنویس، کنترل‌های واضح و fallback مناسب مهم‌اند. autoplay به‌خصوص برای صدا باید با احتیاط استفاده شود.

```
<video controls preload="metadata" poster="poster.jpg" width="720">
  <source src="lesson.mp4" type="video/mp4">
  <track src="fa.vtt" kind="subtitles" srclang="fa" label="فارسی">
</video>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 11

فصل ۱۱ - Head حرفه‌ای، SEO و metadata

بخش head صفحه در ظاهر دیده نمی‌شود، اما برای عنوان، اشتراک‌گذاری، موتورهای جست‌وجو و رفتار مرورگر بسیار مهم است.

title

title اصلی‌ترین عنوان سند در تب مرورگر و یکی از مهم‌ترین metadataهای صفحه است. برای هر صفحه عنوان توصیفی و یکتا بنویس.

```
<title>آموزش HTML وب | مدرسهٔ پیشرفته از صفر تا</title>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل
آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

description

meta description خلاصهٔ صفحه است و می‌تواند در نتایج جست‌وجو نمایش داده شود. متن طبیعی و مرتبط بنویس و از keyword stuffing دوری کن.

```
<meta name="description" content="آموزش جامع HTML از پایه تا مباحث پیشرفته، همراه با مثال و پروژهٔ عملی.">
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

canonical

برای مشخص کردن URL ترجیحی در سایت‌هایی که یک محتوا از چند مسیر قابل دسترسی است استفاده می‌شود.

```
<link rel="canonical" href="https://example.com/html-guide">
```

کاربرد عملی: اطلاعات head بخشی از قرارداد صفحه با مرورگر، موتور جست‌وجو، شبکه‌های اجتماعی و ابزارهای اشتراک‌گذاری است. یک title خوب، canonical صحیح و metadata تمیز از مهم‌ترین نقاط شروع‌اند.
نکات دقیق و خطاهای رایج: هیچ metaای جای محتوای واقعی را نمی‌گیرد. canonical باید به URL مرجع همان محتوا اشاره کند و در صفحات مشابه باید آگاهانه تنظیم شود.

```
<title>آموزش HTML نام سایت | پیشرفته تا صفر</title>
<meta name="description" content="راهنمای جامع و پروژه‌محور HTML.">
<link rel="canonical" href="https://example.com/html">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

robots

برای راهنمایی crawlerها دربارهٔ index و follow استفاده می‌شود، اما سیاست واقعی سایت باید با دقت و در کنار robots.txt مدیریت شود.

```
<meta name="robots" content="index,follow">
```

کاربرد عملی: اطلاعات head بخشی از قرارداد صفحه با مرورگر، موتور جست‌وجو، شبکه‌های اجتماعی و ابزارهای اشتراک‌گذاری است. یک title خوب، canonical صحیح و metadata تمیز از مهم‌ترین نقاط شروع‌اند.
نکات دقیق و خطاهای رایج: هیچ metaای جای محتوای واقعی را نمی‌گیرد. canonical باید به URL مرجع همان محتوا اشاره کند و در صفحات مشابه باید آگاهانه تنظیم شود.

```
<title>آموزش HTML نام سایت | پیشرفته تا صفر</title>
<meta name="description" content="راهنمای جامع و پروژه‌محور HTML.">
<link rel="canonical" href="https://example.com/html">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

Open Graph

برای کنترل نمایش لینک در بسیاری از شبکه‌های اجتماعی و پیام‌رسان‌ها رایج است.

```
<meta property="og:title" content="راهنمای HTML">
<meta property="og:description" content="یادگیری HTML با مثال و پروژه">
<meta property="og:type" content="website">
<meta property="og:url" content="https://example.com/">
```

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.

نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

favicon

آیکون تب مرورگر. فرمت و سایز مناسب را متناسب با نیاز پروژه فراهم کن.

```
<link rel="icon" href="/favicon.svg" type="image/svg+xml">
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 12

فصل ۱۲ - عناصر متنی پیشرفته و کاربردهای کمتر شناخته‌شده

HTML عناصر ظریف‌تری هم دارد که در پروژه‌های واقعی گاهی بسیار مفیدند.

pre، code و kbd

برای نمایش کد و ورودی صفحه‌کلید معنا را حفظ می‌کنند. ظاهر نهایی را با CSS کنترل کن.

```
<pre><code>npm run dev</code></pre>
<p>برای ذخیره <kbd>Ctrl</kbd> + <kbd>S</kbd> بزن را.</p>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

mark و small

mark برای برجسته‌سازی مرتبط با زمینه و small برای توضیحات جانبی یا چاپ ریز استفاده می‌شود.

```
<p>نتیجه : <mark>موفق</mark></p>
<small>آخرین به‌روزرسانی: امروز</small>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

time و datetime

زمان را ماشین‌خوان‌تر می‌کند.

```
<time datetime="2026-08-25">۲۵ اوت ۲۰۲۶</time>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

data

مقدار قابل نمایش را به یک مقدار ماشینی متصل می‌کند.

```
<data value="IR-42">محصول شماره ۴۲</data>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل
آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

bdi و bdo

برای متن‌های دارای جهت متفاوت، مثلاً نام کاربری عربی یا فارسی در جملهٔ انگلیسی، مفیدند.

```
<p>کاربر: <bdi>علی123</bdi></p>
<bdo dir="rtl">Hello</bdo>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

ruby

برای annotation و تلفظ برخی زبان‌ها کاربرد دارد و کمتر در پروژه‌های فارسی مورد نیاز است.

```
<ruby>漢<rt>かん</rt></ruby>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 13

فصل ۱۳ - عناصر تعاملی بومی HTML

قبل از نوشتن JavaScript پیچیده، بررسی کن آیا HTML خودش ابزار مناسب را در اختیارت گذاشته است.

details و summary

برای بخش‌های بازشونده، سوالات متداول و توضیحات تکمیلی عالی هستند و بدون JS هم کار می‌کنند.

```
<details>
  <summary>HTML چیست؟</summary>
  <p>زبان نشانه‌گذاری وب است.</p>
</details>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «details و summary» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.
نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

dialog

برای ساخت dialogهای بومی مرورگر استفاده می‌شود و با JS می‌توان آن را باز و بسته کرد.

```
<dialog id="info">
  <p>پیام مهم</p>
  <form method="dialog"><button>بستن</button></form>
</dialog>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «dialog» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

progress

برای نمایش پیشرفت یک فرایند قابل اندازه‌گیری مناسب است.

```
<label for="download">دانلود</label>
<progress id="download" max="100" value="65">65%</progress>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «progress» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

meter

برای نمایش مقدار در یک بازهٔ شناخته‌شده مثل سطح باتری یا امتیاز مناسب است؛ برای progress از progress استفاده کن.

```
<label for="score">امتیاز</label>
<meter id="score" min="0" max="100" value="82">82%</meter>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «meter» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.
نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

popover

Popover API امکان ساخت پاپ‌اورهای بومی‌تر را فراهم می‌کند. در پروژهٔ واقعی پشتیبانی مرورگرها را بررسی کن.

```
<button popovertarget="help">راهنما</button>
<div id="help" popover>متن راهنما</div>
```

کاربرد عملی: پروژهٔ کوچک بهترین جایی است که مفهوم HTML را به تصمیم‌های واقعی تبدیل کنی: ساختار، فرم، رسانه، دسترس‌پذیری، لینک‌سازی و مسیر فایل‌ها باید هم‌زمان درست باشند.

نکات دقیق و خطاهای رایج: بعد از ساخت پروژه، آن را با کیبورد، موبایل، HTML validator و چند مرورگر بررسی کن؛ پروژهٔ خوب فقط کدی نیست که در یک مرورگر «نمایش داده شود».

```
<main>
  <h1>صفحهٔ اصلی پروژه</h1>
  <section aria-labelledby="features">
    <h2 id="features">ویژگی‌ها</h2>
  </section>
</main>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل ۱۴ - دسترس‌پذیری (Accessibility) از پایه تا حرفه‌ای

صفحه‌ای که فقط برای کاربر ماوس‌دار طراحی شده، کامل نیست. HTML معنایی صحیح اولین و مهم‌ترین قدم accessibility است.

alt و label

تصویر معنادار alt داشته باشد و هر کنترل فرم label قابل‌فهم داشته باشد.

```
<label for="email">ایمیل</label>
<input id="email" name="email" type="email">
<img src="logo.svg" alt="لوگوی فروشگاه">
```

کاربرد عملی: در پروژهٔ واقعی فقط نمایش تصویر مهم نیست؛ باید متن جایگزین، اندازهٔ مناسب، نسبت تصویر، بارگذاری تنبل در موارد مناسب و نسخهٔ مناسب برای صفحه‌نمایش‌های مختلف را هم مدیریت کنی.
نکات دقیق و خطاهای رایج: برای تصاویر تزئینی معمولاً alt خالی بهتر از متن ساختگی است؛ برای تصویرِ دارای اطلاعات، alt باید همان اطلاعات مهم را منتقل کند. همچنین width و height از تغییر ناگهانی چیدمان جلوگیری می‌کنند.

```
<img src="product.webp" alt="گوشی مشکی با نمایشگر 6.4 اینچی" width="640"
height="480" loading="lazy">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

عنوان‌های منطقی

ساختار heading را به‌هم نریز. صفحه‌خوان‌ها از headingها برای پیمایش سریع استفاده می‌کنند.

```
<h1>...</h1>
<h2>...</h2>
<h2>...</h2>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

لینک یا button؟

اگر کاربر به URL یا منبع دیگری می‌رود، link؛ اگر عملیاتی روی همان صفحه انجام می‌شود، button.

```
<a href="/account">حساب کاربری</a>
<button type="button" id="open-cart">سبد خرید</button>
```

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.

نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

aria را بی‌دلیل اضافه نکن

ARIA جای HTML semantic را نمی‌گیرد. ابتدا عنصر بومی مناسب را انتخاب کن؛ سپس در صورت نیاز aria-label، aria-expanded، aria-controls و غیره را اضافه کن.

```
<button aria-expanded="false" aria-controls="menu">منو</button>
<nav id="menu">...</nav>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

keyboard first

تمام تعاملات مهم باید با صفحه‌کلید قابل انجام باشند. از حذف outline بدون جایگزین مناسب اجتناب کن.

```
<button type="button">فعال‌سازی با کیبورد</button>
```

کاربرد عملی: دسترس‌پذیری یک لایهٔ تزئینی نیست؛ هدف این است که همان عملکرد با ورودی‌های مختلف مانند صفحه‌کلید، خوانشگر صفحه و بزرگ‌نمایی نیز قابل استفاده بماند.

نکات دقیق و خطاهای رایج: HTML بومی را قبل از ARIA امتحان کن. عنصرهای واقعی مثل button و label معمولاً رفتار، فوکوس و معنای بهتری از divهای تقلیدی دارند.

```
<a class="skip-link" href="#main">پرش به محتوای اصلی</a>
<main id="main">...</main>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

skip link

برای کاربران صفحه‌خوان یا کیبورد، پرش مستقیم به main باعث کاهش عبور از منوی تکراری می‌شود.

```
<a class="skip-link" href="#main">پرش به محتوای اصلی</a>
<main id="main">...</main>
```

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.
نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 15

فصل ۱۵ - فرم‌های حرفه‌ای و اعتبارسنجی بومی

HTML می‌تواند بخشی از validation را بدون JavaScript انجام دهد. اما هیچ‌وقت validation سمت کلاینت را جایگزین validation و امنیت سمت سرور نکن.

pattern

برای الگوهای سادهٔ متنی استفاده می‌شود؛ regex پیچیده را با احتیاط به کار ببر و پیام خطای قابل فهم فراهم کن.

```
<input name="code" pattern="[A-Z]{3}-[0-9]{4}" required>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

min و max و step

برای number و date و برخی ورودی‌های دیگر محدوده و گام تعریف می‌کنند.

```
<input type="number" min="0" max="100" step="5" value="50">
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «min و max و step» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.
نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

autocomplete

به مرورگر می‌گوید اطلاعات قبلی کاربر چگونه به فیلد مربوط می‌شود. درست تنظیم‌کردن آن تجربهٔ فرم را بهتر می‌کند.

```
<input name="email" type="email" autocomplete="email">
<input name="given-name" autocomplete="given-name">
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

novalidate و formnovalidate

در موارد خاص validation بومی را خاموش می‌کنند. این ویژگی‌ها را صرفاً به‌خاطر رفع خطاهای فرم استفاده نکن.

```
<form novalidate>...</form>
<button type="submit" formnovalidate>ارسال بدون validation بومی</button>
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

method و enctype

GET برای داده‌هایی که بخشی از URL هستند و برای جست‌وجو مناسب است؛ POST برای ارسال دادهٔ ایجادکننده/تغییردهنده متداول‌تر است. برای فایل معمولاً multipart/form-data لازم می‌شود.

```
<form action="/upload" method="post" enctype="multipart/form-data">
  <input type="file" name="file">
  <button type="submit">آپلود</button>
</form>
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت‌نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 16

فصل ۱۶ - HTML و عملکرد (Performance)

HTML روی سرعت ادراکی و واقعی اثر دارد. تصمیم‌های کوچک دربارهٔ تصویر، script و منابع می‌توانند تفاوت بزرگی ایجاد کنند.

lazy loading

برای تصاویر و iframeهایی که خارج از viewport هستند مناسب است. همه‌چیز را lazy نکن؛ محتوای حیاتی بالای صفحه معمولاً باید سریع‌تر در دسترس باشد.

```
<img src="gallery.webp" alt="..." loading="lazy">
<iframe src="https://example.com" loading="lazy" title="نقشه"></iframe>
```

کاربرد عملی: محتوای جاسازی‌شده مرز امنیتی و عملکردی دارد؛ permissions policy، sandbox، origin، referrer policy و اندازهٔ قاب همگی باید در نظر گرفته شوند.

نکات دقیق و خطاهای رایج: هر iframe را به‌صورت پیش‌فرض قابل اعتماد فرض نکن. حداقل سطح دسترسی و قابلیت لازم را به آن بده و در صورت امکان عنوان واضح برایش قرار بده.

```
<iframe src="https://example.com/widget" title="ویجت نمونه" loading="lazy"
sandbox></iframe>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

script defer

برای اسکریپت‌های معمولی، defer اجازه می‌دهد HTML parsing ادامه یابد و اجرای اسکریپت پس از parse انجام شود.

```
<script src="app.js" defer></script>
```

کاربرد عملی: عملکرد HTML بیشتر از «سریع لود شدن فایل HTML» است؛ ترتیب درخواست منابع، parser blocking، اولویت منابع و هزینهٔ تصاویر و اسکریپت‌ها هم مهم‌اند.

نکات دقیق و خطاهای رایج: منابع critical را زودتر و منابع غیرضروری را دیرتر بارگذاری کن. preload فقط زمانی مفید است که دقیقاً بدانی منبع مهم است؛ استفادهٔ بی‌رویه می‌تواند نتیجهٔ معکوس داشته باشد.

```
<link rel="preload" href="hero.webp" as="image">
<script src="app.js" defer></script>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

async

برای اسکریپت‌هایی که مستقل هستند و وابستگی به ترتیب اجرای اسکریپت‌های دیگر ندارند مفید است.

```
<script src="analytics.js" async></script>
```

کاربرد عملی: عملکرد HTML بیشتر از «سریع لود شدن فایل HTML» است؛ ترتیب درخواست منابع، parser blocking، اولویت منابع و هزینهٔ تصاویر و اسکریپت‌ها هم مهم‌اند.
نکات دقیق و خطاهای رایج: منابع critical را زودتر و منابع غیرضروری را دیرتر بارگذاری کن. preload فقط زمانی مفید است که دقیقاً بدانی منبع مهم است؛ استفادهٔ بی‌رویه می‌تواند نتیجهٔ معکوس داشته باشد.

```
<link rel="preload" href="hero.webp" as="image">
<script src="app.js" defer></script>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

preload

برای منابع حیاتی با نیاز روشن به بارگذاری زودهنگام استفاده می‌شود. preload زیاد می‌تواند نتیجهٔ معکوس داشته باشد.

```
<link rel="preload" href="font.woff2" as="font" type="font/woff2"
crossorigin>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل
آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

preconnect

برای originهای خارجی مورد نیاز می‌تواند هزینهٔ اتصال اولیه را کاهش دهد؛ فقط برای منابع مهم.

```
<link rel="preconnect" href="https://fonts.example.com" crossorigin>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 17

فصل ۱۷ - بارگذاری CSS و JavaScript درست

در اینجا مرز HTML با منابع خارجی و ترتیب بارگذاری را یاد می‌گیریم.

CSS خارجی

CSS را به‌صورت فایل جدا نگه دار تا کش‌پذیر و قابل نگهداری باشد.

```
<link rel="stylesheet" href="styles.css">
```

کاربرد عملی: بخش بزرگی از خطاهای HTML در پروژه‌های واقعی به مسیر نادرست فایل‌ها مربوط است. مسیر نسبی نسبت به URL سند فعلی و مسیر مطلق نسبت به ریشهٔ origin تفسیر می‌شود.
نکات دقیق و خطاهای رایج: روی Windows ممکن است تفاوت حروف را نبینی ولی روی بسیاری از سرورها نام فایل case-sensitive است. نام‌گذاری ثابت از خطاهای عجیب جلوگیری می‌کند.

```
<img src="./images/logo.svg" alt="لوگو">
<a href="../index.html">بازگشت</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

JavaScript خارجی

script را معمولاً با defer بارگذاری کن مگر سناریوی خاصی داشته باشی.

```
<script src="app.js" defer></script>
```

کاربرد عملی: عملکرد HTML بیشتر از «سریع لود شدن فایل HTML» است؛ ترتیب درخواست منابع، parser blocking، اولویت منابع و هزینهٔ تصاویر و اسکریپت‌ها هم مهم‌اند.

نکات دقیق و خطاهای رایج: منابع critical را زودتر و منابع غیرضروری را دیرتر بارگذاری کن. preload فقط زمانی مفید است که دقیقاً بدانی منبع مهم است؛ استفادهٔ بی‌رویه می‌تواند نتیجهٔ معکوس داشته باشد.

```
<link rel="preload" href="hero.webp" as="image">
<script src="app.js" defer></script>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

module

برای ماژول‌های ES از "type="module استفاده می‌شود. ماژول‌ها رفتار deferگونه دارند.

```
<script type="module" src="app.js"></script>
```

کاربرد عملی: عملکرد HTML بیشتر از «سریع لود شدن فایل HTML» است؛ ترتیب درخواست منابع، parser blocking، اولویت منابع و هزینهٔ تصاویر و اسکریپت‌ها هم مهم‌اند.

نکات دقیق و خطاهای رایج: منابع critical را زودتر و منابع غیرضروری را دیرتر بارگذاری کن. preload فقط زمانی مفید است که دقیقاً بدانی منبع مهم است؛ استفادهٔ بی‌رویه می‌تواند نتیجهٔ معکوس داشته باشد.

```
<link rel="preload" href="hero.webp" as="image">
<script src="app.js" defer></script>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

noscript

می‌تواند پیام جایگزین برای محیط‌هایی ارائه کند که JavaScript اجرا نمی‌شود؛ ولی سایت را بی‌دلیل وابسته به noscript نکن.

```
<noscript>برای استفاده کامل از این بخش، JavaScript را فعال کنید.</noscript>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «noscript» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.
نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 18

فصل ۱۸ - iframe، embed، object و محتواهای خارجی

نمایش محتوای خارجی نیازمند توجه به امنیت، performance و تجربهٔ کاربر است.

iframe امن‌تر

همیشه title معنادار بده. در سناریوهای حساس از sandbox و محدودسازی ویژگی‌ها استفاده کن.

```
<iframe src="https://example.com" title="صفحهٔ نمونه" loading="lazy"
sandbox="allow-scripts allow-forms"></iframe>
```

کاربرد عملی: اطلاعات head بخشی از قرارداد صفحه با مرورگر، موتور جست‌وجو، شبکه‌های اجتماعی و ابزارهای اشتراک‌گذاری است. یک title خوب، canonical صحیح و metadata تمیز از مهم‌ترین نقاط شروع‌اند.

نکات دقیق و خطاهای رایج: هیچ metaای جای محتوای واقعی را نمی‌گیرد. canonical باید به URL مرجع همان محتوا اشاره کند و در صفحات مشابه باید آگاهانه تنظیم شود.

```
<title>آموزش HTML از صفر تا پیشرفته | نام سایت</title>
<meta name="description" content="راهنمای جامع و پروژه‌محور HTML.">
<link rel="canonical" href="https://example.com/html">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

sandbox

sandbox قابلیت‌های iframe را محدود می‌کند. هر permission را فقط در صورت نیاز اضافه کن.

```
<iframe src="embedded.html" sandbox="allow-scripts"></iframe>
```

کاربرد عملی: محتوای جاسازی‌شده مرز امنیتی و عملکردی دارد؛ permissions policy ،sandbox ،origin، referrer policy و اندازهٔ قاب همگی باید در نظر گرفته شوند.

نکات دقیق و خطاهای رایج: هر iframe را به‌صورت پیش‌فرض قابل اعتماد فرض نکن. حداقل سطح دسترسی و قابلیت لازم را به آن بده و در صورت امکان عنوان واضح برایش قرار بده.

```
<iframe src="https://example.com/widget" title="ویجت نمونه" loading="lazy"
sandbox></iframe>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

embed و object

برای بعضی انواع محتوای خارجی و اسناد قدیمی یا خاص دیده می‌شوند. در پروژه‌های جدید اول بررسی کن آیا روش بومی بهتر وجود دارد.

```
<object data="file.pdf" type="application/pdf" width="800" height="600">
  <a href="file.pdf">دریافت PDF</a>
</object>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 19

فصل ۱۹ - Template، slot و Web Components

این فصل مرز HTML مدرن با Web Components را معرفی می‌کند. برای پروژه‌های بزرگ، مفهوم component اهمیت زیادی دارد.

template

محتوای template تا زمانی که با JavaScript clone نشود به شکل عادی render نمی‌شود.

```
<template id="card-template">
  <article class="card">
    <h2 class="title"></h2>
  </article>
</template>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

slot

slot در Shadow DOM برای تزریق محتوای بیرونی به component استفاده می‌شود.

```
<my-card>
  <span slot="title">محصول ویژه</span>
</my-card>
```

کاربرد عملی: HTML بعد از parse به ساختار DOM تبدیل می‌شود و JavaScript معمولاً با همین درخت کار می‌کند. تفاوت attributeهای HTML با propertyهای DOM در پروژه‌های پیچیده اهمیت پیدا می‌کند.

نکات دقیق و خطاهای رایج: هر attribute الزاماً به همان شکلی که نوشته شده در JavaScript منعکس نمی‌شود؛ برای رفتار دقیق، مستندات عنصر و API مربوط به DOM را ببین.

```
<button id="save" data-state="draft">ذخیره</button>
<script>
const b = document.querySelector("#save");
console.log(b.dataset.state);
</script>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

custom element

نام عناصر سفارشی باید شامل خط تیره باشد؛ تعریف واقعی آن با JavaScript انجام می‌شود.

```
<product-card data-id="42"></product-card>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

چه زمانی استفاده کنیم؟

برای کتابخانه‌های UI، componentهای تکرارشونده و سیستم طراحی می‌تواند مناسب باشد؛ برای یک صفحهٔ ساده نیازی به پیچیده‌سازی بی‌دلیل نیست.

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «چه زمانی استفاده کنیم؟» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 20

فصل ۲۰ - Global attributes مهم و نکات دقیق DOM

شناخت attributeهای عمومی در بسیاری از خطاهای پروژه جلوی دوباره‌کاری را می‌گیرد.

hidden

محتوا را از rendering معمول خارج می‌کند. برای stateهای UI مناسب است؛ اما اگر محتوا باید از نظر دسترس‌پذیری صرفاً مخفی شود، انتخاب تکنیک باید با دقت انجام شود.

```
<div hidden>محتوای پنهان</div>
```

کاربرد عملی: دسترس‌پذیری یک لایهٔ تزئینی نیست؛ هدف این است که همان عملکرد با ورودی‌های مختلف مانند صفحه‌کلید، خوانشگر صفحه و بزرگ‌نمایی نیز قابل استفاده بماند.
نکات دقیق و خطاهای رایج: HTML بومی را قبل از ARIA امتحان کن. عنصرهای واقعی مثل button و label معمولاً رفتار، فوکوس و معنای بهتری از divهای تقلیدی دارند.

```
<a class="skip-link" href="#main">پرش به محتوای اصلی</a>
<main id="main">...</main>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

contenteditable

به کاربر اجازهٔ ویرایش محتوا می‌دهد. برای ویرایشگرهای حرفه‌ای، مدیریت state و امنیت پیچیده‌تر از این attribute ساده است.

```
<div contenteditable="true">متن قابل ویرایش</div>
```

کاربرد عملی: جدول برای دادهٔ دوبعدی و رابطهٔ سطر/ستون مناسب است، نه برای چیدمان صفحه. ساختار صحیح جدول شامل عنوان، سرستون‌های واضح و در صورت نیاز ارتباط دقیق headerهاست.
نکات دقیق و خطاهای رایج: هرچه جدول پیچیده‌تر شود، استفاده از scope و در موارد خاص id/headers مهم‌تر می‌شود. برای موبایل نیز رفتار overflow را با CSS حل کن، نه با حذف داده‌ها.

```
<table>
  <caption>فروش ماهانه</caption>
  <thead><tr><th scope="col">ماه</th><th scope="col">فروش</th></tr></thead>
  <tbody><tr><td>فروردین</td><td>120</td></tr></tbody>
</table>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

spellcheck

فعال/غیرفعال کردن بررسی املا در فیلدهای متنی.

```
<textarea spellcheck="true"></textarea>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

translate

برای راهنمایی برخی ابزارهای ترجمه دربارهٔ یک بخش استفاده می‌شود.

```
<span translate="no">BrandName</span>
```

کاربرد عملی: HTML بعد از parse به ساختار DOM تبدیل می‌شود و JavaScript معمولاً با همین درخت کار می‌کند. تفاوت attributeهای HTML با propertyهای DOM در پروژه‌های پیچیده اهمیت پیدا می‌کند.
نکات دقیق و خطاهای رایج: هر attribute الزاماً به همان شکلی که نوشته شده در JavaScript منعکس نمی‌شود؛ برای رفتار دقیق، مستندات عنصر و API مربوط به DOM را ببین.

```
<button id="save" data-state="draft">ذخیره</button>
<script>
const b = document.querySelector("#save");
console.log(b.dataset.state);
</script>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

inert

یک subtree را از تعامل کاربر و برخی مسیرهای accessibility خارج می‌کند و برای مدیریت modalها مفید است.

```
<main inert>محتوای غیرفعال</main>
```

کاربرد عملی: دسترس‌پذیری یک لایهٔ تزئینی نیست؛ هدف این است که همان عملکرد با ورودی‌های مختلف مانند صفحه‌کلید، خوانشگر صفحه و بزرگ‌نمایی نیز قابل استفاده بماند.
نکات دقیق و خطاهای رایج: HTML بومی را قبل از ARIA امتحان کن. عنصرهای واقعی مثل button و label معمولاً رفتار، فوکوس و معنای بهتری از divهای تقلیدی دارند.

```
<a class="skip-link" href="#main">پرش به محتوای اصلی</a>
<main id="main">...</main>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 21

فصل ۲۱ - مسیرها، فایل‌ها و ساختار پروژه

ساختار فایل درست، مخصوصاً هنگام استفاده از تصاویر، CSS و JavaScript، جلوی تعداد زیادی خطای 404 و مسیر اشتباه را می‌گیرد.

ساختار پیشنهادی

پروژهٔ کوچک را خوانا نگه دار. ساختار زیر نقطهٔ شروع خوبی است.

```
project/
├── index.html
├── about.html
├── css/
│   └── styles.css
├── js/
│   └── app.js
└── assets/
    ├── images/
    └── icons/
```

کاربرد عملی: پروژهٔ کوچک بهترین جایی است که مفهوم HTML را به تصمیم‌های واقعی تبدیل کنی: ساختار، فرم، رسانه، دسترس‌پذیری، لینک‌سازی و مسیر فایل‌ها باید هم‌زمان درست باشند.

نکات دقیق و خطاهای رایج: بعد از ساخت پروژه، آن را با کیبورد، موبایل، HTML validator و چند مرورگر بررسی کن؛ پروژهٔ خوب فقط کدی نیست که در یک مرورگر «نمایش داده شود».

```
<main>
  <h1>صفحهٔ اصلی پروژه</h1>
  <section aria-labelledby="features">
    <h2 id="features">ویژگی‌ها</h2>
  </section>
</main>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

مسیر نسبی و مطلق

در href/src باید فرق بین / مسیر ریشه و ./ یا ../ مسیر نسبی را بفهمی.

```
<img src="assets/images/logo.svg" alt="لوگو">
<a href="../index.html">صفحهٔ اصلی</a>
```

کاربرد عملی: بخش بزرگی از خطاهای HTML در پروژه‌های واقعی به مسیر نادرست فایل‌ها مربوط است. مسیر نسبی نسبت به URL سند فعلی و مسیر مطلق نسبت به ریشهٔ origin تفسیر می‌شود.

نکات دقیق و خطاهای رایج: روی Windows ممکن است تفاوت حروف را نبینی ولی روی بسیاری از سرورها نام فایل case-sensitive است. نام‌گذاری ثابت از خطاهای عجیب جلوگیری می‌کند.

```
<img src="./images/logo.svg" alt="لوگو">
<a href="../index.html">بازگشت</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

حساسیت حروف

روی برخی سرورها Logo.svg و logo.svg دو فایل متفاوت‌اند. نام‌گذاری یکدست انتخاب کن.

```
<img src="assets/logo.svg" alt="لوگو">
```

کاربرد عملی: بخش بزرگی از خطاهای HTML در پروژه‌های واقعی به مسیر نادرست فایل‌ها مربوط است. مسیر نسبی نسبت به URL سند فعلی و مسیر مطلق نسبت به ریشهٔ origin تفسیر می‌شود.
نکات دقیق و خطاهای رایج: روی Windows ممکن است تفاوت حروف را نبینی ولی روی بسیاری از سرورها نام فایل case-sensitive است. نام‌گذاری ثابت از خطاهای عجیب جلوگیری می‌کند.

```
<img src="./images/logo.svg" alt="لوگو">
<a href="../index.html">بازگشت</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 22

فصل ۲۲ - HTML برای SEO واقعی: اصول محتوایی

SEO با چند meta تگ حل نمی‌شود. ساختار، محتوای مفید، لینک‌سازی داخلی، عملکرد و تجربهٔ کاربر هم مهم‌اند.

یک H1 و عنوان دقیق

عنوان و محتوای اصلی صفحه باید با هدف جست‌وجو هماهنگ باشند.

```
<h1>آموزش HTML از صفر</h1>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

لینک داخلی توصیفی

متن anchor را معنادار بنویس تا کاربر و موتور جست‌وجو مقصد را بهتر بفهمند.

```
<a href="/forms">آموزش فرم‌های HTML</a>
```

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.

نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

تصاویر قابل فهم

alt خوب برای تصاویر محتوایی هم accessibility را بهتر می‌کند و هم زمینهٔ معنایی فراهم می‌کند.

```
<img src="form.webp" alt="فرم ثبت‌نام با فیلدهای ایمیل و رمز عبور">
```

کاربرد عملی: دسترس‌پذیری یک لایهٔ تزئینی نیست؛ هدف این است که همان عملکرد با ورودی‌های مختلف مانند صفحه‌کلید، خوانشگر صفحه و بزرگ‌نمایی نیز قابل استفاده بماند.

نکات دقیق و خطاهای رایج: HTML بومی را قبل از ARIA امتحان کن. عنصرهای واقعی مثل button و label معمولاً رفتار، فوکوس و معنای بهتری از divهای تقلیدی دارند.

```
<a class="skip-link" href="#main">پرش به محتوای اصلی</a>
<main id="main">...</main>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

structured data

دادهٔ ساختاریافته در پروژه‌های واقعی معمولاً با JSON-LD و JavaScript یا HTML مناسب پیاده می‌شود. انتخاب schema باید فقط وقتی انجام شود که واقعاً مرتبط باشد.

```
<script type="application/ld+json">{ "@context": "https://schema.org", "@type": "Article", "headline": "راهنمای HTML" }</script>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 23

فصل ۲۳ - امنیت در HTML و اطراف آن

HTML به‌تنهایی سیستم امنیتی کامل نیست. با این حال چند انتخاب درست می‌تواند سطح ریسک را کاهش دهد.

target=_blank

در سناریوهای قدیمی‌تر، noopener مهم بود؛ امروز مرورگرها رفتارهای امن‌تری دارند اما explicit بودن rel همچنان خوانایی و کنترل بیشتری می‌دهد.

```
<a href="https://example.com" target="_blank" rel="noopener noreferrer">باز کردن</a>
```

کاربرد عملی: HTML به‌تنهایی همهٔ امنیت وب را حل نمی‌کند، اما انتخاب صحیح عنصرها، جلوگیری از تزریق markup، محدود کردن iframeها و ترکیب مناسب با CSP و هدرهای HTTP سطح حمله را کاهش می‌دهد.
نکات دقیق و خطاهای رایج: هر چیزی که کاربر ارسال می‌کند «دادهٔ غیرقابل اعتماد» فرض می‌شود. escaping و sanitization را متناسب با context انجام بده و روی validation سمت مرورگر حساب امنیتی باز نکن.

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

form action و دادهٔ کاربر

هیچ دادهٔ فرم را فقط به validation HTML اعتماد نکن. سمت سرور باید داده‌ها را بررسی، نرمال و معتبرسازی کند.

```
<form action="/login" method="post">...</form>
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت‌نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

iframe sandbox

محتوای خارجی یا untrusted را تا حد امکان محدود کن.

```
<iframe src="/preview.html" sandbox="allow-scripts"></iframe>
```

کاربرد عملی: محتوای جاسازی‌شده مرز امنیتی و عملکردی دارد؛ permissions policy ،sandbox ،origin، referrer policy و اندازهٔ قاب همگی باید در نظر گرفته شوند.

نکات دقیق و خطاهای رایج: هر iframe را به‌صورت پیش‌فرض قابل اعتماد فرض نکن. حداقل سطح دسترسی و قابلیت لازم را به آن بده و در صورت امکان عنوان واضح برایش قرار بده.

```
<iframe src="https://example.com/widget" title="ویجت نمونه" loading="lazy"
sandbox></iframe>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

HTML injection و XSS

نمایش مستقیم دادهٔ کاربر به‌صورت HTML می‌تواند مسیر حمله ایجاد کند. در JS از innerHTML فقط وقتی استفاده کن که ورودی trust شده یا کاملاً sanitize شده باشد؛ textContent انتخاب امن‌تری برای متن خام است.

```
const message = document.querySelector("#message");
message.textContent = userInput;
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 24

فصل ۲۴ - طراحی فرم خرید و فروشگاه

یک پروژهٔ واقعی ترکیبی از عناصر مختلف HTML است. در این پروژه یک صفحهٔ سفارش کوچک می‌سازیم.

فرم سفارش

ساختار پیشنهادی برای اطلاعات تماس و سفارش:

```
<form action="/orders" method="post">
  <fieldset>
    <legend>اطلاعات مشتری</legend>
    <label for="full-name">نام و نام خانوادگی</label>
    <input id="full-name" name="full_name" required autocomplete="name">

    <label for="phone">شماره موبایل</label>
    <input id="phone" name="phone" type="tel" required autocomplete="tel">
  </fieldset>

  <fieldset>
    <legend>محصول</legend>
    <label for="product">محصول</label>
    <select id="product" name="product" required>
      <option value="">انتخاب کنید</option>
      <option value="laptop">لپ‌تاپ</option>
      <option value="phone">گوشی</option>
    </select>
  </fieldset>

  <button type="submit">ثبت سفارش</button>
</form>
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.

نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت‌نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

نکتهٔ مهم

قیمت، موجودی، تخفیف و مجوز خرید باید سمت سرور تعیین و کنترل شوند. مقدار hidden یا HTML قابل اعتماد نیست و کاربر می‌تواند آن را تغییر دهد.

```
<input type="hidden" name="discount" value="50">
<!-- این مقدار را هرگز منبع حقیقت قیمت در نظر نگیر -->
```

کاربرد عملی: HTML بعد از parse به ساختار DOM تبدیل می‌شود و JavaScript معمولاً با همین درخت کار می‌کند. تفاوت attributeهای HTML با propertyهای DOM در پروژه‌های پیچیده اهمیت پیدا می‌کند.
نکات دقیق و خطاهای رایج: هر attribute الزاماً به همان شکلی که نوشته شده در JavaScript منعکس نمی‌شود؛ برای رفتار دقیق، مستندات عنصر و API مربوط به DOM را ببین.

```
<button id="save" data-state="draft">ذخیره</button>
<script>
const b = document.querySelector("#save");
console.log(b.dataset.state);
</script>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 25

فصل ۲۵ - پروژهٔ عملی: صفحهٔ معرفی شخصی

این پروژه برای جمع‌بندی semantic HTML، لینک‌ها، تصویر، لیست و فرم تماس طراحی شده است.

نسخهٔ کامل

کد زیر یک اسکلت حرفه‌ای و قابل توسعه می‌سازد. سپس می‌توانی CSS و JS را جداگانه اضافه کنی.

```
<!doctype html>
<html lang="fa" dir="rtl">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="description" content="صفحهٔ شخصی یک توسعه‌دهندهٔ وب">
  <title>نام من | توسعه‌دهندهٔ وب</title>
  <link rel="stylesheet" href="css/styles.css">
  <script src="js/app.js" defer></script>
</head>
<body>
  <a href="#main">پرش به محتوای اصلی</a>
  <header>
    <nav aria-label="اصلی">
      <a href="#about">درباره</a>
      <a href="#skills">مهارت‌ها</a>
      <a href="#contact">تماس</a>
    </nav>
  </header>

  <main id="main">
    <section id="about" aria-labelledby="about-title">
      <h1 id="about-title">سلام، من یک توسعه‌دهندهٔ وب هستم</h1>
      <p>ساخت وب‌سایت‌های سریع و قابل استفاده را دوست دارم.</p>
      <img src="assets/profile.webp" alt="تصویر پروفایل من" width="320"
height="320">
    </section>

    <section id="skills" aria-labelledby="skills-title">
      <h2 id="skills-title">مهارت‌ها</h2>
      <ul><li>HTML</li><li>CSS</li><li>JavaScript</li></ul>
    </section>

    <section id="contact" aria-labelledby="contact-title">
      <h2 id="contact-title">تماس</h2>
      <form action="/contact" method="post">
        <label for="email">ایمیل</label>
        <input id="email" name="email" type="email" required
autocomplete="email">
        <label for="message">پیام</label>
        <textarea id="message" name="message" required></textarea>
        <button type="submit">ارسال</button>
```

```
      </form>
    </section>
  </main>

  <footer>© 2026 نام من</footer>
</body>
</html>
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «نسخهٔ کامل» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 26

فصل ۲۶ - پروژهٔ عملی: صفحهٔ محصول

این پروژه استفادهٔ هم‌زمان از semantic HTML، تصویر واکنش‌گرا، قیمت، ویژگی‌ها و CTA را تمرین می‌دهد.

کارت محصول

ساختار پیشنهادی:

```
<article class="product-card">
  <a href="/products/phone-x" aria-label="مشاهدهٔ گوشی مدل X">
    <picture>
      <source srcset="phone-x.avif" type="image/avif">
      <img src="phone-x.webp" alt="گوشی مدل X با رنگ مشکی" width="600"
height="600" loading="lazy">
    </picture>
  </a>
  <h2><a href="/products/phone-x">گوشی مدل X</a></h2>
  <p>نمایشگر 120 هرتز، حافظهٔ 256 گیگابایت.</p>
  <data value="24990000">۲۴٬۹۹۰٬۰۰۰ تومان</data>
  <div>
    <button type="button">افزودن به سبد</button>
  </div>
</article>
```

کاربرد عملی: پروژهٔ کوچک بهترین جایی است که مفهوم HTML را به تصمیم‌های واقعی تبدیل کنی: ساختار، فرم، رسانه، دسترس‌پذیری، لینک‌سازی و مسیر فایل‌ها باید هم‌زمان درست باشند.

نکات دقیق و خطاهای رایج: بعد از ساخت پروژه، آن را با کیبورد، موبایل، HTML validator و چند مرورگر بررسی کن؛ پروژهٔ خوب فقط کدی نیست که در یک مرورگر «نمایش داده شود».

```
<main>
  <h1>صفحهٔ اصلی پروژه</h1>
  <section aria-labelledby="features">
    <h2 id="features">ویژگی‌ها</h2>
  </section>
</main>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 27

فصل ۲۷ - پروژهٔ عملی: FAQ و منوی بازشونده بدون JavaScript

با HTML بومی می‌توان بخشی از تعاملات را بدون کدنویسی پیچیده انجام داد.

FAQ با details

راهکاری ساده، قابل دسترس و کم‌کد:

```
<section aria-labelledby="faq-title">
  <h2 id="faq-title">سؤالات متداول</h2>
  <details>
    <summary>HTML چیست؟</summary>
    <p>زبان نشانه‌گذاری برای ساختاردهی محتوای وب است.</p>
  </details>
  <details>
    <summary>آیا HTML یک زبان برنامه‌نویسی است؟</summary>
    <p>HTML زبان نشانه‌گذاری است، نه زبان برنامه‌نویسی عمومی.</p>
  </details>
</section>
```

کاربرد عملی: پروژهٔ کوچک بهترین جایی است که مفهوم HTML را به تصمیم‌های واقعی تبدیل کنی: ساختار، فرم، رسانه، دسترس‌پذیری، لینک‌سازی و مسیر فایل‌ها باید هم‌زمان درست باشند.

نکات دقیق و خطاهای رایج: بعد از ساخت پروژه، آن را با کیبورد، موبایل، HTML validator و چند مرورگر بررسی کن؛ پروژهٔ خوب فقط کدی نیست که در یک مرورگر «نمایش داده شود».

```
<main>
  <h1>صفحهٔ اصلی پروژه</h1>
  <section aria-labelledby="features">
    <h2 id="features">ویژگی‌ها</h2>
  </section>
</main>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل ۲۸ - خطاهای بسیار رایج و نسخهٔ درست آن‌ها

بسیاری از مشکلات HTML از چند عادت ساده می‌آیند. این فصل را مثل چک‌لیست مرور کن.

استفاده از div به‌جای همه‌چیز

div همیشه غلط نیست، اما جایگزین section, article, nav, button یا a نشود.

```
<!-- بد -->
<div onclick="openMenu()">منو</div>

<!-- بهتر -->
<button type="button" onclick="openMenu()">منو</button>
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت‌نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

لینک با onclick

برای عملیاتی که URL ندارند button مناسب‌تر است.

```
<!-- بد -->
<a href="#" onclick="save()">ذخیره</a>

<!-- بهتر -->
<button type="button" onclick="save()">ذخیره</button>
```

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.
نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

تصویر بدون alt

نمونهٔ نادرست:

```
<img src="logo.png">
```

```
<img src="logo.png" alt="لـوگوی شرکت">
```

کاربرد عملی: در پروژهٔ واقعی فقط نمایش تصویر مهم نیست؛ باید متن جایگزین، اندازهٔ مناسب، نسبت تصویر، بارگذاری تنبل در موارد مناسب و نسخهٔ مناسب برای صفحه‌نمایش‌های مختلف را هم مدیریت کنی.
نکات دقیق و خطاهای رایج: برای تصاویر تزئینی معمولاً alt خالی بهتر از متن ساختگی است؛ برای تصویرِ دارای اطلاعات، alt باید همان اطلاعات مهم را منتقل کند. همچنین width و height از تغییر ناگهانی چیدمان جلوگیری می‌کنند.

```
<img src="product.webp" alt="گوشی مشکی با  نما یشگر 6.4 ا ینچی" width="640"
height="480" loading="lazy">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

input بدون label

نمونهٔ نادرست:

```
<input type="email">
```

```
<label for="email">ایمیل</label>
<input id="email" name="email" type="email">
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

button بدون type

نمونهٔ نادرست:

```
<button>ارسال</button>
```

```
<button type="submit">ارسال</button>
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت‌نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

استفادهٔ نابجا از br

نمونهٔ نامناسب:

```
<p>عنوان<br><br><br>متن</p>
```

```
<h2>عنوان</h2>
<p>متن</p>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

تکیه بر placeholder به‌عنوان label

placeholder جای label نیست. placeholder معمولاً hint است و ممکن است ناپدید شود.

```
<label for="username">نام کاربری</label>
<input id="username" name="username" placeholder="مثلاً ali123">
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «تکیه بر placeholder به‌عنوان label» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 29

فصل ۲۹ - HTML مدرن و چیزهایی که باید بررسی کنی

استاندارد وب در طول زمان رشد می‌کند. برای APIهای جدید، همیشه پشتیبانی مرورگر و سازگاری پروژه را بررسی کن.

ویژگی‌های جدید

نمونه‌ها شامل inert، dialog، popover و کنترل‌های بومی جدیدترند. وجود یک attribute به معنی پشتیبانی برابر در همهٔ مرورگرهای قدیمی نیست.

```
<button popovertarget="menu">باز کردن</button>
<div id="menu" popover>منو</div>
```

کاربرد عملی: HTML بعد از parse به ساختار DOM تبدیل می‌شود و JavaScript معمولاً با همین درخت کار می‌کند. تفاوت attributeهای HTML با propertyهای DOM در پروژه‌های پیچیده اهمیت پیدا می‌کند.
نکات دقیق و خطاهای رایج: هر attribute الزاماً به همان شکلی که نوشته شده در JavaScript منعکس نمی‌شود؛ برای رفتار دقیق، مستندات عنصر و API مربوط به DOM را ببین.

```
<button id="save" data-state="draft">ذخیره</button>
<script>
const b = document.querySelector("#save");
console.log(b.dataset.state);
</script>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

Progressive enhancement

ابتدا نسخه‌ای بساز که با HTML درست کار کند، سپس CSS و JavaScript را برای بهبود تجربه اضافه کن. این رویکرد resilience پروژه را بیشتر می‌کند.

```
<form action="/search" method="get">
  <label for="q">جست‌وجو</label>
  <input id="q" name="q">
  <button type="submit">جست‌وجو</button>
</form>
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.
نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 30

فصل ۳۰ - چک‌لیست نهایی یک فایل HTML حرفه‌ای

قبل از تحویل هر صفحه، این موارد را بررسی کن. این چک‌لیست عمداً عملی و کوتاه نگه داشته شده است.

ساختار

DOCTYPE داری؟ lang و dir درست‌اند؟ title مناسب است؟ charset و viewport وجود دارند؟

کاربرد عملی: اطلاعات head بخشی از قرارداد صفحه با مرورگر، موتور جست‌وجو، شبکه‌های اجتماعی و ابزارهای اشتراک‌گذاری است. یک title خوب، canonical صحیح و metadata تمیز از مهم‌ترین نقاط شروع‌اند.

نکات دقیق و خطاهای رایج: هیچ metaای جای محتوای واقعی را نمی‌گیرد. canonical باید به URL مرجع همان محتوا اشاره کند و در صفحات مشابه باید آگاهانه تنظیم شود.

```
<title>آموزش HTML نام سایت | پیشرفته تا صفر</title>
<meta name="description" content="راهنمای جامع و پروژه‌محور HTML.">
<link rel="canonical" href="https://example.com/html">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

معنا

headingها منطقی‌اند؟ از semantic elements مناسب استفاده شده؟ div فقط برای container عمومی است؟

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

لینک و فرم

href واقعی است؟ لینک‌ها متن توصیفی دارند؟ label و name برای ورودی‌ها وجود دارد؟ type دکمه‌ها مشخص است؟

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.

نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

رسانه

alt مناسب است؟ width/height یا aspect ratio باعث جلوگیری از layout shift می‌شود؟ رسانه‌های غیرحیاتی lazy شده‌اند؟

کاربرد عملی: رسانهٔ وب یک موضوع فنی چندبخشی است: منبع فایل، فرمت، کنترل پخش، دسترس‌پذیری، زیرنویس، بارگذاری، پوستر و سازگاری مرورگر باید با هم دیده شوند.

نکات دقیق و خطاهای رایج: برای محتوای آموزشی زیرنویس، کنترل‌های واضح و fallback مناسب مهم‌اند. autoplay به‌خصوص برای صدا باید با احتیاط استفاده شود.

```
<video controls preload="metadata" poster="poster.jpg" width="720">
  <source src="lesson.mp4" type="video/mp4">
  <track src="fa.vtt" kind="subtitles" srclang="fa" label="فارسی">
</video>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

دسترس‌پذیری

با کیبورد قابل استفاده است؟ focus حذف نشده؟ رنگ تنها راه انتقال معنا نیست؟ عنوان‌ها و landmarkها درست‌اند؟

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

SEO

title و description مرتبط‌اند؟ canonical در صورت نیاز وجود دارد؟ لینک‌های داخلی منطقی‌اند؟

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.

نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

عملکرد

scriptها درست بارگذاری می‌شوند؟ preload/preconnect بیش از حد نیستند؟ تصاویر بهینه‌اند؟

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

امنیت

ورودی کاربر را trusted فرض نکرده‌ای؟ iframeها بی‌دلیل آزاد نیستند؟ فرم روی validation سمت سرور هم حساب می‌کند؟

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.

نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت‌نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فصل 31

فصل ۳۱ - ویژگی‌های سراسری و Attributeهای حرفه‌ای

این فصل به Attributeهایی می‌پردازد که معمولاً در جزوه‌های مقدماتی نادیده گرفته می‌شوند، اما در پروژه‌های واقعی روی دسترس‌پذیری، ورودی کاربر، تعامل، مرورگر و کنترل رفتار صفحه اثر دارند.

autofocus را کجا و چگونه استفاده کنیم؟

autofocus باعث می‌شود یک کنترل هنگام آماده شدن صفحه focus بگیرد. این ویژگی برای فرم‌های کوتاه مثل جست‌وجوی اصلی مفید است، اما استفادهٔ افراطی می‌تواند focus را از کاربر بگیرد و مخصوصاً برای کاربر صفحه‌خوان گیج‌کننده باشد.

```
<label for="search">جست‌وجو</label>
<input id="search" name="q" type="search" autofocus>
```

کاربرد عملی: فقط وقتی focus خودکار واقعاً شروع طبیعی تعامل را مشخص می‌کند از آن استفاده کن؛ مثل یک فرم جست‌وجوی اصلی یا دیالوگ تازه‌گشوده‌شده.

نکتهٔ دقیق: چند عنصر نباید هم‌زمان autofocus داشته باشند؛ در عمل باید فقط یک مقصد منطقی برای focus اولیه داشته باشی.

inputmode و enterkeyhint

این دو Attribute به‌خصوص روی موبایل مهم‌اند. `inputmode` صفحه‌کلید مناسب‌تری پیشنهاد می‌دهد و `enterkeyhint` متن/عمل پیشنهادی کلید Enter را راهنمایی می‌کند.

```
<input type="text" inputmode="numeric" enterkeyhint="next" name="code">
```

کجا به کار می‌آید؟ فرم‌های موبایل، شمارهٔ تماس، کد تأیید و فرم‌های چندمرحله‌ای.

خطای رایج: inputmode نوع واقعی داده را تعریف نمی‌کند؛ برای اعتبارسنجی از type و constraint مناسب هم استفاده کن.

autocapitalize و autocorrect

این Attributeها برای کنترل پیشنهادهای ورود متن روی ابزارهای دارای سیستم تصحیح یا بزرگ‌نمایی حروف مفیدند. برای نام کاربری، کد، ایمیل و شناسه‌ها معمولاً رفتار پیش‌بینی‌شدهٔ متن مهم‌تر از تصحیح خودکار است.

```
<input name="username" autocapitalize="none" autocorrect="off"
spellcheck="false">
```

نکته: تنظیمات را فقط برای جاهایی تغییر بده که واقعاً با نوع داده تضاد دارند؛ برای متن آزاد، تصحیح خودکار می‌تواند مفید باشد.

dir، lang، translate و نوشتار چندزبانه

Attributeهای زبانی فقط تزئینی نیستند. `lang` به فناوری کمکی و پردازش زبان کمک می‌کند، `dir` جهت را مشخص می‌کند و `translate="no"` می‌تواند یک نام برند یا قطعهٔ کد را از ترجمهٔ خودکار دور نگه دارد.

```
<p lang="fa" dir="rtl">متن فارسی</p>
<code translate="no" dir="ltr">npm install</code>
```

contenteditable و spellcheck

با `contenteditable` یک عنصر می‌تواند قابل ویرایش شود. `spellcheck` هم رفتار بررسی املایی را راهنمایی می‌کند. این قابلیت‌ها بیشتر برای ویرایشگرها، یادداشت‌ها و ابزارهای مدیریت محتوا دیده می‌شوند.

```
<div contenteditable="true" spellcheck="true">متن را ویرایش کن</div>
```

هشدار: contenteditable به‌تنهایی یک ویرایشگر کامل و امن نیست؛ مدیریت paste، پاک‌سازی محتوا، ساختار selection و ذخیره‌سازی به منطق JavaScript و backend نیاز دارد.

hidden و inert چه فرقی دارند؟

`hidden` برای محتوایی است که فعلاً نباید نمایش داده شود؛ `inert` بیشتر روی غیرفعال کردن تعامل و حذف موقت از مسیر focus و فناوری کمکی اثر می‌گذارد. این دو را بر اساس هدف انتخاب کن، نه صرفاً ظاهر.

```
<section hidden>محتوای کاملاً پنهان</section>
<div inert>این بخش موقتاً غیرقابل تعامل است</div>
```

tabindex و مدیریت focus

مقدار 0 عنصر را وارد ترتیب طبیعی focus می‌کند و مقدارهای مثبت ترتیب مصنوعی می‌سازند و معمولاً توصیه نمی‌شوند. برای کنترل custom بهتر است تا جای ممکن از عناصر native مثل button و input استفاده شود.

```
<button type="button">بهتر</button>
<div role="button" tabindex="0">فقط در صورت ضرورت</div>
```

خطای مهم: role و tabindex به‌تنهایی رفتار کامل keyboard را ایجاد نمی‌کنند. اگر عنصر custom ساختی باید تعامل keyboard و state آن را نیز درست پیاده‌سازی کنی.

headingoffset و headingreset

در استاندارد زندهٔ HTML Attributeهای جدیدی برای محاسبهٔ سطح heading وجود دارند. `headingoffset` می‌تواند سطح منطقی headingهای زیرمجموعه را جابه‌جا کند و `headingreset` مرز محاسبه ایجاد می‌کند. پشتیبانی مرورگر و ابزارهای کمکی را قبل از استفاده در پروژهٔ production بررسی کن.

```
<section headingoffset="1">
  <h1>عنوان منطقی این بخش</h1>
</section>
```

writingsuggestions

Attribute `writingsuggestions` برای راهنمایی رفتار پیشنهادهای نوشتاری مرورگر تعریف شده است. این مورد برای محیط‌های ویرایش متن می‌تواند جالب باشد، ولی مانند هر قابلیت جدید باید compatibility را بررسی کرد.

```
<textarea writingsuggestions="false"></textarea>
```

command و commandfor در تعامل بومی

در نسخه‌های جدید HTML، خانوادهٔ commandها راهی برای بیان بعضی تعاملات به شکل معنایی‌تر ارائه می‌کند. این قابلیت به‌ویژه کنار dialog و popover ارزش دارد، اما برای پروژه‌های گسترده باید وضعیت پشتیبانی مرورگر را جداگانه ارزیابی کرد.

```
<button command="show-modal" commandfor="settings">تنظیمات</button>
<dialog id="settings">...</dialog>
```

نکته: این موضوع را با APIهای JavaScript و polyfillها قاطی نکن؛ HTML مسئول بیان intent است و پشتیبانی اجرایی مرورگر موضوع جداگانه‌ای است.

Attributeهای سفارشی data-* و نام‌گذاری درست

برای داده‌های خصوصیِ صفحه از `data-*` استفاده کن. نام‌گذاری باید کوتاه، معنادار و پایدار باشد تا JavaScript بتواند با `dataset` آن را بخواند.

```
<article data-product-id="42" data-state="sale">محصول</article>
<script>
const card = document.querySelector("article");
console.log(card.dataset.productId);
</script>
```

فراموش نکن: Attribute با Property یکی نیست

Attribute بخشی از نشانه‌گذاری HTML است؛ property بخشی از شیء DOM در JavaScript است. این تفاوت برای checked، value و چند کنترل فرم اهمیت ویژه دارد.

```
<input id="age" value="16">
<script>
const age = document.querySelector("#age");
console.log(age.getAttribute("value"));
console.log(age.value);
</script>
```

نکتهٔ دقیق: ممکن است مقدار attribute اولیه با state فعلی control یکی نباشد. در فرم‌های پویا باید فرق «default value» و «current value» را بفهمی.

فصل ۳۲ - کارگاه پروژهٔ جامع HTML واقعی

در این فصل، HTML را از حالت «حفظ تگ‌ها» خارج می‌کنیم و به یک پروژهٔ چندبخشی واقعی نزدیک می‌شویم؛ از اسکلت سند تا محتوا، فرم، رسانه، SEO و accessibility.

معماری فایل‌های پروژه

قبل از نوشتن markup، مسیرها را مشخص کن. یک پروژهٔ ساده می‌تواند پوشه‌های icons، images، pages و data داشته باشد. اسم‌گذاری ثابت، خطاهای مسیر را کم می‌کند.

```
project/
├── index.html
├── pages/
│   ├── about.html
│   └── product.html
├── images/
├── icons/
└── data/
```

ساخت اسکلت صفحه

از یک سند استاندارد شروع کن و سپس landmarkها را اضافه کن. یک صفحهٔ واقعی معمولاً header، nav، main و footer دارد و داخل main با section یا article سازمان‌دهی می‌شود.

```
<!doctype html>
<html lang="fa" dir="rtl">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>فروشگاه نمونه</title>
</head>
<body>
  <header>...</header>
  <main id="main">...</main>
  <footer>...</footer>
</body>
</html>
```

ناوبری حرفه‌ای و skip link

کاربر باید بتواند سریع به محتوای اصلی برسد. لینک پرش، متن لینک توصیفی و ساختار nav از ساده‌ترین بهبودهای accessibility هستند.

```
<a class="skip-link" href="#main">پرش به محتوای اصلی</a>
<nav aria-label="ناوبری اصلی">
  <a href="/">خانه</a>
  <a href="/products">محصولات</a>
</nav>
```

کارت محصول با HTML معنایی

برای نمایش محصول، تصویر، نام، توضیح، قیمت و اقدام اصلی را با ساختار مشخص کنار هم قرار بده. ظاهر کارت بعداً با CSS ساخته می‌شود؛ HTML باید معنا و رابطهٔ محتوا را حفظ کند.

```
<article>
  <img src="phone.webp" alt="گوشی مشکی مدل X" width="640" height="640">
  <h2>گوشی مدل X</h2>
  <p>توضیح کوتاه محصول</p>
  <data value="12990000">۱۲٬۹۹۰٬۰۰۰ تومان</data>
  <a href="/products/x">مشاهدهٔ محصول</a>
</article>
```

فرم ثبت‌نام که واقعاً قابل ارسال باشد

هر control قابل ارسال باید name مناسب داشته باشد. label باید به control وصل شود و type و autocomplete تا حد ممکن اطلاعات درست را منتقل کنند.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ساخت حساب</button>
</form>
```

تصویر قهرمان صفحه و LCP

تصویر اصلی بالای صفحه را با اندازهٔ واقعی، نسبت تصویر مشخص و منبع مناسب قرار بده. برای تصویر حیاتی، lazy loading معمولاً انتخاب خوبی نیست و باید اولویت بارگذاری را با احتیاط مدیریت کرد.

```
<img src="hero.webp"
     alt="نمای اصلی فروشگاه"
     width="1280" height="720"
     fetchpriority="high">
```

FAQ بدون JavaScript

وقتی تعامل شما فقط باز و بسته شدن توضیحات است، details و summary انتخاب native و ساده‌ای هستند.

```
<details>
  <summary>هزینهٔ ارسال چقدر است؟</summary>
  <p>هزینه بر اساس شهر و روش ارسال محاسبه می‌شود.</p>
</details>
```

SEO حداقلی اما واقعی

title باید توصیفی باشد، description با محتوای صفحه هماهنگ باشد و canonical فقط زمانی قرار بگیرد که URL مرجع را واقعاً می‌دانی. ساختار headingها نیز باید بازتاب محتوای واقعی باشد.

```
<title>مشخصات و قیمت | گوشی مدل X</title>
<meta name="description" content="مشخصات کامل، قیمت و جزئیات گوشی مدل X.">
<link rel="canonical" href="https://example.com/products/x">
```

متادیتای شبکه‌های اجتماعی

Open Graph و Twitter/X Card بخشی از HTML head هستند و برای نحوهٔ نمایش URL هنگام اشتراک‌گذاری مفیدند. این metadata جای محتوای خوب را نمی‌گیرد.

```
<meta property="og:title" content="گوشی مدل X">
<meta property="og:description" content="مشخصات و قیمت گوشی مدل X">
<meta property="og:image" content="https://example.com/images/x.jpg">
```

آزمون نهایی پروژه

پروژه را فقط در حالت عادی بررسی نکن. با keyboard، موبایل، zoom، سرعت شبکهٔ پایین، بدون JavaScript و با دادهٔ نامعتبر تست کن. بعد Lighthouse/DevTools، validator و بررسی دستی accessibility را انجام بده.

```
□ می‌توانم حرکت کنم؟ بدون mouse
□ تمام inputها label دارند؟
□ عنوان‌ها ترتیب منطقی دارند؟
□ تصاویر مهم alt مناسب دارند؟
□ مسیرها و لینک‌ها سالم‌اند؟
□ فرم بدون JS رفتار پایه دارد؟
```

چگونه از این پروژه برای ورود به CSS و JavaScript استفاده کنیم؟

ابتدا HTML را کامل و semantic نگه دار؛ سپس CSS را برای layout، رنگ، typography و responsive design اضافه کن. بعد با JavaScript رفتارهایی مانند جست‌وجوی زنده، سبد خرید، dialogهای پویا و ارسال asynchronous را اضافه کن.

اصل حرفه‌ای: JavaScript نباید جایگزین markup معنایی شود. هرجا یک عنصر native نیاز تو را برآورده می‌کند، معمولاً از ساختن معادل custom شروع نکن.

فصل 33

فصل ۳۳ - نحو و مدل پردازش HTML در عمق

در این فصل از سطح «نوشتن تگ» عبور می‌کنیم و می‌بینیم مرورگر چگونه متن HTML را parse می‌کند، DOM چگونه ساخته می‌شود و چرا بعضی خطاهای ظاهری از خود parser می‌آیند.

doctype در سطح دقیق‌تر

DOCTYPE نسخهٔ خاصی از HTML را مثل XHTML قدیمی تعیین نمی‌کند؛ مهم‌ترین نقش آن در HTML مدرن انتخاب حالت استاندارد پردازش مرورگر است.

```
<!doctype html>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

تعادل تگ‌های باز و بسته

بیشتر عناصر معمولی به تگ پایان نیاز دارند، اما بعضی عناصر void هستند و اصلاً end tag ندارند. بستن نادرست تگ‌ها می‌تواند DOM متفاوتی از چیزی که تصور می‌کنی بسازد.

```
<p>درست</p>
<img src="a.jpg" alt="">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

عناصر void

عناصری مثل link، meta، hr، br، img و input محتوای فرزند ندارند و تگ بستهٔ جداگانه برایشان نوشته نمی‌شود.

```
<img src="logo.svg" alt="لوگو">
<br>
<input type="email">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

کامنت و متن خام

کامنت‌ها برای انسان‌اند و در DOM به‌عنوان Comment Node وجود دارند. از کامنت برای نگه‌داری اطلاعات محرمانه استفاده نکن چون کاربر می‌تواند source را ببیند.

```
<!-- یادداشت توسعه‌دهنده: اینجا فرم ثبت‌نام است -->
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

White space و فاصله‌ها

فاصله، tab و line break در HTML همیشه بی‌اثر نیستند. نحوهٔ نمایش آن‌ها تحت تأثیر CSS و نوع عنصر است و هنگام کار با inline content باید این تفاوت را بشناسی.

```
<span>سلام</span> <span>دنیا</span>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

Content model چیست؟

هر عنصر قواعدی دربارهٔ اینکه چه نوع محتوایی می‌تواند داخلش قرار بگیرد دارد. فهم content model جلوی بسیاری از تودرتو کردن‌های نامعتبر را می‌گیرد.

```
<button type="button">
  <span>خرید</span>
</button>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

DOM در برابر متن HTML

ممکن است رشتهٔ HTML و DOM نهایی یکسان نباشند؛ مرورگر هنگام parse بعضی تگ‌ها را جابه‌جا یا ضمنی اضافه می‌کند. بنابراین DevTools را برای دیدن DOM واقعی بررسی کن.

```
<table><tr><td>سلول</td></tr></table>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

Quirks mode و استاندارد

DOCTYPE قدیمی یا ناقص می‌تواند مرورگر را وارد حالت‌های سازگاری کند. برای پروژه‌های مدرن همیشه doctype استاندارد را در اولین خط قرار بده.

```
<!doctype html>
<html lang="fa">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 34

فصل ۳۴ - مرجع کامل عناصر متنی HTML

این فصل عناصر متنی مهمی را که در پروژه‌های واقعی برای معنا، ویرایش، نقل‌قول، تاریخ، داده و نمایش متن استفاده می‌شوند یک‌جا جمع می‌کند.

abbr و title

برای مخفف‌ها از abbr استفاده کن و در صورت نیاز شکل کامل را با title ارائه بده. title را جایگزین متن قابل مشاهده نکن.

```
<p><abbr title="HyperText Markup Language">HTML</abbr> یک زبان نشانه‌گذاری است.</p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

address

address برای اطلاعات تماس مرتبط با نزدیک‌ترین article یا body است؛ برای هر متن تصادفیِ آدرس فیزیکی از آن استفاده نکن.

```
<address>تماس: hello@example.com</address>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

q در برابر blockquote

q برای نقل‌قول کوتاه درون‌خطی و blockquote برای نقل‌قول مستقل و طولانی‌تر مناسب است. cite می‌تواند منبع را معرفی کند.

```
<p>او گفت: <q>یادگیری با تمرین کامل می‌شود.</q></p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

del و ins

برای نشان دادن محتوای حذف‌شده و اضافه‌شده در ویرایش‌ها از del و ins استفاده می‌شود؛ این عناصر معنای تغییر را منتقل می‌کنند.

```
<p>قیمت قبلی: <del>500</del> قیمت جدید: <ins>420</ins> هزار تومان</p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

s در برابر del

s یعنی اطلاعات دیگر دقیق یا مرتبط نیست، اما لزوماً یک ویرایش تاریخی نیست؛ برای نشان دادن تغییر سند، del مناسب‌تر است.

```
<p><s>پیشنهاد تا جمعه معتبر است.</s></p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

b در برابر strong

b متن را برجسته می‌کند بدون اینکه لزوماً اهمیت بیشتری بدهد؛ strong اهمیت یا جدیت بیشتری را منتقل می‌کند. این تفاوت را با ظاهر اشتباه نکن.

```
<p><b>کلیدواژه:</b> HTML</p>
<p><strong>مهم:</strong> رمز خود را منتشر نکن.</p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

i در برابر em

i برای متن با صدای متفاوت، اصطلاح یا نام‌گذاری مناسب در جاهای مشخص است؛ em برای stress معنایی است.

```
<p><em>واقعاً این موضوع مهم است</em>.</p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

u و متن غیرزبانی

u برای حاشیه‌گذاری متنی مثل نام خاص یا annotation مناسب است؛ برای زیرخط تزئینی معمولاً CSS انتخاب بهتری است.

```
<p><u>نام خاص</u> در این مثال مشخص شده است.</p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

dfn

dfn واژه‌ای را که در همان متن در حال تعریف شدن است مشخص می‌کند و برای مستندات، آموزش و واژه‌نامه‌ها مفید است.

```
<p><dfn>DOM</dfn> نمایش درختی سند در مرورگر است.</p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

var، samp و output

var برای متغیر، samp برای خروجی نمونهٔ یک برنامه و output برای نتیجه‌ای که از یک محاسبه یا تعامل فرم به‌دست آمده مناسب‌اند.

```
<p><var>x</var> = <samp>12</samp></p>
<output>24</output>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

wbr

wbr محل مجاز برای شکستن کلمه در شرایط محدودیت عرض است و برای URLها یا شناسه‌های طولانی می‌تواند مفید باشد.

```
<p>super<wbr>long<wbr>identifier</p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 35

فصل ۳۵ - لینک‌ها، URLها و ناوبری پیشرفته

لینک فقط یک تگ a نیست؛ ساخت URL، fragment، query string، base، target و سیاست ارجاع روی رفتار و امنیت اثر دارند.

URL: scheme و origin

URL معمولاً شامل scheme، host، port، path، query و fragment است. origin به scheme/host/port مربوط است و در امنیت مرورگر مفهوم مهمی است.

```
https://example.com:443/products?id=42#reviews
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

query string

پارامترهای query برای فیلتر، جست‌وجو و وضعیت URL مناسب‌اند اما دادهٔ حساس را نباید بدون ملاحظه در URL قرار داد چون ممکن است در history و logها باقی بماند.

```
<a href="/search?q=html&sort=new">جست‌وجوی HTML</a>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

fragment و id

بخش # در URL معمولاً برای جابه‌جایی داخل همان سند استفاده می‌شود و با id عنصر هدف کار می‌کند.

```
<a href="#pricing">قیمت</a>
<section id="pricing">...</section>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

base

base می‌تواند URL پایه و target پیش‌فرض لینک‌ها را تعیین کند. چون روی همهٔ URLهای نسبی اثر می‌گذارد، باید با احتیاط استفاده شود.

```
<head>
  <base href="https://example.com/docs/">
</head>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

relها در لینک

rel فقط برای noopener نیست؛ مقادیری مثل author، external، ugc، sponsored، nofollow و prev/next کاربردهای مشخص دارند. انتخاب rel باید معنای واقعی ارتباط را منعکس کند.

```
<a href="https://example.com" rel="external noopener noreferrer">منبع</a>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

download و محدودیت‌های آن

download می‌تواند مرورگر را به دانلود یک منبع راهنمایی کند، اما رفتار دقیق به origin و پاسخ سرور وابسته است.

```
<a href="/files/guide.pdf" download>دانلود راهنما</a>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

target و contextهای مرور

target تعیین می‌کند پاسخ کجا باز شود. _self پیش‌فرض است؛ _blank context جدیدی باز می‌کند و برای آن rel مناسب را در نظر بگیر.

```
<a href="/docs" target="_self">همین صفحه</a>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 36

فصل ۳۶ - تصاویر پیشرفته و Responsive Images

از یک img ساده تا art direction و اولویت بارگذاری، مدیریت تصویر در سایت‌های واقعی یکی از مهم‌ترین بخش‌های HTML است.

srcset با width descriptor

srcset امکان ارائهٔ چند فایل با عرض‌های مختلف را می‌دهد تا مرورگر بهترین گزینه را با توجه به شرایط انتخاب کند.

```
<img src="hero-800.jpg" srcset="hero-400.jpg 400w, hero-800.jpg 800w,
hero-1600.jpg 1600w" sizes="100vw" alt="...">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

sizes

sizes به مرورگر کمک می‌کند بداند تصویر تقریباً چه عرضی در layout خواهد داشت؛ مقدار اشتباه می‌تواند باعث انتخاب فایل بیش از حد بزرگ شود.

```
<img src="photo.jpg" srcset="photo-480.jpg 480w, photo-960.jpg 960w"
sizes="(max-width: 700px) 92vw, 680px" alt="...">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

Art direction با picture

وقتی در موبایل و دسکتاپ ترکیب تصویر باید متفاوت باشد، picture و source می‌تواند نسخهٔ متفاوتی ارائه کند؛ این با صرفاً تغییر اندازهٔ همان تصویر فرق دارد.

```
<picture>
  <source media="(max-width: 700px)" srcset="hero-mobile.webp">
  <img src="hero-desktop.webp" alt="نمای محصول">
</picture>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

type در source

type به مرورگر اجازه می‌دهد قبل از دانلود بفهمد منبع چه نوع رسانه‌ای است و در صورت پشتیبانی همان گزینه را انتخاب کند.

```
<picture>
  <source srcset="hero.avif" type="image/avif">
  <source srcset="hero.webp" type="image/webp">
  <img src="hero.jpg" alt="نمای محصول">
</picture>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

loading و decoding

loading=lazy برای منابع خارج از محدودهٔ اولیه و decoding=async می‌تواند در سناریوهای مشخص مفید باشد؛ اما برای تصویر اصلیِ بالای fold بدون تحلیل از lazy استفاده نکن.

```
<img src="hero.webp" alt="تصویر اصلی" width="1200" height="700"
fetchpriority="high">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

fetchpriority

این ویژگی hintی برای اولویت منبع است و باید فقط روی چند منبع واقعاً مهم اعمال شود. استفادهٔ گسترده از آن می‌تواند اولویت‌بندی مرورگر را خراب کند.

```
<img src="hero.webp" alt="تصویر اصلی" fetchpriority="high">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

usemap و area

تصویرهای clickable با image map هنوز بخشی از HTML هستند و برای بعضی رابط‌های خاص کاربرد دارند، اما برای UI عمومی معمولاً گزینه‌های قابل‌دسترسی‌تری وجود دارد.

```
<img src="plan.png" usemap="#map" alt="نقشهٔ طبقات">
<map name="map"><area href="/floor/1" shape="rect" coords="0,0,100,100"
alt="طبقه اول"></map>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 37

فصل ۳۷ - صوت، ویدئو و زیرنویس در سطح حرفه‌ای

در این فصل کنترل کامل‌تری روی media elementها، منابع چندگانه، track و ویژگی‌های پخش بررسی می‌شود.

poster

poster تصویر اولیهٔ ویدئو تا قبل از پخش را تعیین می‌کند و برای تجربهٔ کاربر مهم است.

```
<video controls poster="course-cover.webp"><source src="lesson.mp4"
type="video/mp4"></video>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

preload

none، metadata و auto به مرورگر دربارهٔ میزان پیش‌بارگذاری hint می‌دهند. برای بسیاری از صفحات metadata انتخاب متعادل‌تری از auto است.

```
<video controls preload="metadata"><source src="lesson.mp4" type="video/
mp4"></video>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

muted و autoplay

autoplay معمولاً بدون صدا قابل‌پیش‌بینی‌تر است، اما سیاست‌های مرورگر متفاوت است. برای تجربهٔ کاربر autoplay رسانهٔ صوتی را بدون رضایت روشن نکن.

```
<video autoplay muted playsinline loop><source src="preview.mp4" type="video/mp4"></video>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

playsinline

روی دستگاه‌های موبایل می‌تواند پخش درون صفحه را ممکن کند، به‌ویژه برای previewها و ویدئوهای پس‌زمینه.

```
<video controls playsinline><source src="clip.mp4" type="video/mp4"></video>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

track و WebVTT

track برای descriptions، captions، subtitles و metadata استفاده می‌شود و فایل VTT ساختار مشخصی دارد.

```
<track src="fa.vtt" kind="subtitles" srclang="fa" label="فارسی" default>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

poster، transcript و دسترس‌پذیری

برای ویدئوهای آموزشی فقط subtitle کافی نیست؛ transcript متنی نیز می‌تواند فهم محتوا و جست‌وجوپذیری را بهتر کند.

```
<details><summary>متن درس</summary><p>این متن، transcript است.</p></details>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 38

فصل ۳۸ - فهرست‌ها و جدول‌های پیچیده

در پروژه‌های واقعی، list و table زمانی ارزشمند می‌شوند که ساختار داده و رابطهٔ معنایی درست انتخاب شود.

start و reversed در ol

می‌توانی شمارهٔ آغازین یا جهت شمارش را برای رتبه‌بندی و دستورالعمل‌های خاص تغییر دهی.

```
<ol start="5"><li>پنجم</li><li>ششم</li></ol>
<ol reversed><li>آخر</li><li>قبل‌تر</li></ol>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

value در li

در فهرست مرتب می‌توان برای یک مورد شمارهٔ مشخص تعیین کرد؛ برای نمایش مراحل خاص یا شماره‌گذاری غیرپیوسته مفید است.

```
<ol><li>اول</li><li value="5">پنجم</li></ol>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

لیست‌های تو در تو

قرار دادن ul یا ol داخل li روش درست برای ساخت زیرگروه‌های یک فهرست است.

```
<ul><li>Frontend<ul><li>HTML</li><li>CSS</li></ul></li></ul>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

colgroup و col

برای تعریف گروهی از ستون‌ها و اعمال ویژگی‌های مشترک استفاده می‌شوند. در طراحی جدول‌های بزرگ می‌توانند خوانایی markup را بهتر کنند.

```
<table><colgroup><col><col></colgroup>...</table>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

headers و id در جدول پیچیده

در جدول‌های چندسطحی که scope کافی نیست، headers و id می‌توانند ارتباط دقیق سلول داده با چند سرستون را مشخص کنند.

```
<th id="m1">ماه</th>
<td headers="m1">فروردین</td>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

caption و summary ذهنی

caption عنوان قابل مشاهدهٔ جدول است. برای توضیح پیچیدگی جدول بهتر است متن پیرامونی یا توضیح قابل دسترسی اضافه شود تا وابسته به سازوکارهای منسوخ نباشی.

```
<table><caption>فروش بر اساس ماه و منطقه</caption>...</table>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 39

فصل ۳۹ - مرجع کامل فرم‌ها و کنترل‌های ورودی

این فصل تمام ابزارهای اصلی فرم را از inputهای تخصصی تا output، datalist، option و ویژگی‌های کمتر دیده‌شده پوشش می‌دهد.

input type=color

برای انتخاب رنگ مناسب است؛ اگر مقدار رنگ بخشی از مدل کسب‌وکار است، دادهٔ نهایی را در سرور نیز validate کن.

```
<label for="accent">رنگ</label><input id="accent" name="accent" type="color" value="#6c5ce7">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

input type=file

برای انتخاب فایل است. accept فقط hint انتخاب فایل است و امنیت یا validation سرور محسوب نمی‌شود.

```
<input type="file" name="avatar" accept="image/png,image/jpeg" multiple>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

multiple

در inputهای file/email و بعضی کنترل‌ها چند مقدار را فعال می‌کند. برای email باید بدانید که جداکننده‌ها و نحوهٔ پردازش چگونه است.

```
<input type="email" name="emails" multiple>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

list و datalist

list با id یک datalist پیوند می‌خورد و پیشنهادها را برای input نشان می‌دهد؛ برخلاف select، کاربر الزاماً محدود به گزینه‌ها نیست.

```
<input list="cities" name="city"><datalist id="cities"><option value="تهران"><option value="تبریز"></datalist>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

output

output برای نتیجهٔ یک محاسبه یا تغییر تعاملی مفید است و می‌تواند با for و نام فرم ارتباط معنایی داشته باشد.

```
<form oninput="sum.value=Number(a.value)+Number(b.value)"><input id="a" name="a" value="2"><input id="b" name="b" value="3"><output name="sum">5</output></form>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

meter در برابر progress

meter مقدار را در یک بازهٔ شناخته‌شده نشان می‌دهد؛ progress مقدار پیشرفت یک کار را. این دو را به‌جای هم استفاده نکن.

```
<meter min="0" max="100" value="78">78%</meter>
<progress max="100" value="45">45%</progress>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

textarea و maxlength

textarea یک کنترل چندخطی است و ویژگی‌هایی مثل maxlength، minlength، cols، rows و placeholder برای تجربهٔ کاربر مفیدند.

```
<textarea name="message" rows="6" minlength="10" maxlength="500"></textarea>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

select و optgroup

برای انتخاب از مجموعهٔ محدود گزینه‌ها مناسب است و optgroup می‌تواند گزینه‌ها را دسته‌بندی کند.

```
<select name="course"><optgroup label="Frontend"><option>HTML</
option><option>CSS</option></optgroup></select>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

option selected disabled hidden

optionها می‌توانند حالت اولیه، غیرفعال بودن و مخفی بودن در فهرست را کنترل کنند. این ویژگی‌ها در ساخت placeholder واقعی select مهم‌اند.

```
<select name="country"><option value="" selected disabled hidden>کشور را انتخاب کنید</option><option value="ir">ایران</option></select>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

dirname

dirname جهت متن واردشده توسط کاربر را نیز به سمت سرور ارسال می‌کند و برای ورودی‌های دوزبانه یا RTL/LTR می‌تواند مفید باشد.

```
<input type="text" name="comment" dirname="comment.dir">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

form attribute روی کنترل بیرون فرم

گاهی یک input لازم است از نظر ساختاری بیرون form باشد؛ ویژگی form می‌تواند آن را به فرم مشخصی متصل کند.

```
<form id="checkout" action="/checkout" method="post"></form>
<input form="checkout" name="coupon">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 40

فصل ۴۰ - فرم و مرورگر: داده، نام، ارسال و FormData

فرم HTML قرارداد بین کاربر، مرورگر و سرور است. درک name و successful controls برای ساخت فرم‌های واقعی حیاتی است.

name مهم‌تر از id برای ارسال

id برای اتصال label و اسکریپت عالی است، اما نامی که معمولاً در submit به سرور می‌رسد name است. کنترل بدون name معمولاً در payload فرم قرار نمی‌گیرد.

```
<input id="email" name="email" type="email">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

successful controls

disabled controls و بعضی کنترل‌های بدون name در ارسال فرم وارد دادهٔ نهایی نمی‌شوند. این موضوع هنگام دیباگ payload بسیار مهم است.

```
<input name="active" value="yes"><input name="ignored" value="x" disabled>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

checkbox و value

برای checkbox بهتر است value صریح تعیین کنی؛ اگر تیک نخورده باشد معمولاً اصلاً داده‌ای با آن نام ارسال نمی‌شود.

```
<input type="checkbox" name="terms" value="accepted">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

radio و group

radioها با name مشترک در یک گروه قرار می‌گیرند و از آن مجموعه فقط یک گزینه انتخاب می‌شود.

```
<label><input type="radio" name="plan" value="free"> رایگان</label>
<label><input type="radio" name="plan" value="pro"> حرفه‌ای</label>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

formnovalidate

این ویژگی روی یک دکمه می‌تواند مسیر ارسال خاصی را از اعتبارسنجی بومی مستثنی کند؛ مثلاً دکمهٔ ذخیرهٔ موقت.

```
<button type="submit" name="draft" formnovalidate>ذخیرهٔ موقت</button>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

requestSubmit در JavaScript

وقتی JavaScript وارد می‌شود، requestSubmit برای شبیه‌سازی submit واقعی و عبور از validation معمولاً مناسب‌تر از submit() خام است.

```
form.requestSubmit();
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 41

فصل ۴۱ - Head حرفه‌ای: base، script، link و metadata

بخش head فقط title و favicon نیست؛ ترتیب و نوع linkها و scriptها می‌تواند روی عملکرد، SEO و رفتار مرورگر اثر بگذارد.

meta robots

برای کنترل رفتار index/follow در سناریوهای خاص استفاده می‌شود، ولی directives مهم سایت را با معماری کلی SEO هماهنگ کن.

```
<meta name="robots" content="index,follow">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

meta referrer

referrer policy تعیین می‌کند هنگام درخواست‌های خروجی چه مقدار اطلاعات referrer ارسال شود. سیاست باید متناسب با نیاز حریم خصوصی انتخاب شود.

```
<meta name="referrer" content="strict-origin-when-cross-origin">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

theme-color

می‌تواند رنگ UI مرورگر یا محیط نصب‌شده را در بعضی پلتفرم‌ها تحت تأثیر قرار دهد و برای تجربهٔ یکپارچه مفید است.

```
<meta name="theme-color" content="#5f52d9">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

color-scheme

به مرورگر اعلام می‌کند صفحه چه طرح‌های رنگی را پشتیبانی می‌کند و می‌تواند از اختلاف بین کنترل‌های بومی و UI جلوگیری کند.

```
<meta name="color-scheme" content="light dark">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

link rel=preload

preload برای منبعی است که به‌طور قطعی در مسیر critical لازم است و باید as مناسب داشته باشد.

```
<link rel="preload" href="hero.webp" as="image">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

link rel=modulepreload

برای آماده‌سازی ماژول‌هایی که به‌زودی توسط module script لازم‌اند کاربرد دارد؛ استفادهٔ بدون نیاز می‌تواند ترافیک اضافه ایجاد کند.

```
<link rel="modulepreload" href="/assets/app.js">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

preconnect و crossorigin

preconnect اتصال به originی را زودتر برقرار می‌کند؛ crossorigin در بعضی سناریوهای منابع cross-origin باعث تطبیق cache و credential mode می‌شود.

```
<link rel="preconnect" href="https://cdn.example.com" crossorigin>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

dns-prefetch

یک hint سبک‌تر برای resolve کردن DNS است و در کنار preconnect یا به تنهایی در برخی سناریوها استفاده می‌شود.

```
<link rel="dns-prefetch" href="//cdn.example.com">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

script type=module

module scriptها deferred-by-default هستند و scope ماژولی دارند؛ برای پروژه‌های مدرن نقطهٔ شروع مناسبی برای JS ساختاریافته‌اند.

```
<script type="module" src="/assets/app.js"></script>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

script nomodule

در سناریوهای سازگاری قدیمی، nomodule اجازه می‌دهد نسخهٔ fallback برای مرورگرهای بدون module ارائه شود.

```
<script nomodule src="legacy.js"></script>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

integrity و crossorigin در script/link

SRI با hash می‌تواند تمامیت یک منبع خارجی را بررسی کند. برای CDNها باید مقدار integrity و crossorigin با منبع واقعی هماهنگ باشد.

```
<script src="https://cdn.example.com/lib.js" integrity="sha384-..."
crossorigin="anonymous"></script>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 42

فصل ۴۲ - Accessibility پیشرفته و طراحی برای فناوری کمکی

از semantic HTML تا focus management و نام قابل دسترس، این فصل نگاه عمیق‌تری به ساخت رابطی دارد که بدون ماوس و با assistive technology نیز قابل استفاده باشد.

accessible name

عنصر تعاملی باید نام قابل تشخیص برای کاربرِ فناوری کمکی داشته باشد. متن دکمه، label و aria-label از روش‌های مختلف ساخت این نام‌اند.

```
<button type="button">ذخیره</button>
<button type="button" aria-label="بستن">×</button>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

role فقط در صورت نیاز

role می‌تواند معنای عنصر را تغییر دهد و اگر اشتباه استفاده شود دسترس‌پذیری را خراب می‌کند. اول عنصر native را انتخاب کن.

```
<button type="button">باز کردن</button>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

aria-describedby

برای وصل کردن کنترل به توضیح کمکی یا پیام خطا کاربرد دارد و از aria-label متمایز است.

```
<p id="hint">حداکثر 500 نویسه.</p>
<textarea aria-describedby="hint"></textarea>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

aria-invalid و خطا

وقتی یک فیلد واقعاً نامعتبر است می‌توان state مناسب را به فناوری کمکی اعلام کرد، در کنار یک پیام خطای قابل مشاهده.

```
<input aria-invalid="true" aria-describedby="email-error">
<p id="email-error">ایمیل معتبر وارد کنید.</p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

live region

برای اطلاع‌رسانی تغییرات پویا مثل پیام موفقیت یا خطا می‌توان از aria-live استفاده کرد؛ نباید برای هر تغییر کوچک استفاده شود.

```
<div aria-live="polite" id="status"></div>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

focus-visible و keyboard

در CSS، focus-visible به حفظ نشانهٔ فوکوس برای کاربرانی که با صفحه‌کلید حرکت می‌کنند کمک می‌کند. HTML باید مسیر tab منطقی ایجاد کند.

```
<button type="button">ذخیره</button>
<a href="/next">بعدی</a>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

tabindex=0 و tabindex=-1

0 عنصر را در ترتیب طبیعی tab قرار می‌دهد؛ -1 اجازه می‌دهد با JavaScript فوکوس بگیرد بدون اینکه وارد ترتیب tab عادی شود. از اعداد مثبت پرهیز کن.

```
<div id="dialog-title" tabindex="-1">عنوان</div>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

dialog و focus management

باز کردن dialog فقط نمایش box نیست؛ focus باید به هدف مناسب منتقل و پس از بستن نیز مدیریت شود. عنصر dialog بومی نقطهٔ شروع مناسبی است.

```
<dialog id="d"><h2>تأیید</h2><button type="button">بستن</button></dialog>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

reduced motion و content

HTML باید ساختار درستی بدهد و CSS می‌تواند برای کاهش حرکت، تجربه را سازگار کند. ساختار را وابسته به animation نکن.

```
<main><h1>گزارش</h1><p>محتوا باید بدون animation قابل فهم باشد.</p></main>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 43

فصل ۴۳ - SEO عمیق، ساختار محتوا و دادهٔ ساختاریافته

SEO پایدار از ساختار محتوا شروع می‌شود؛ این فصل HTML را در کنار structured data، hreflang، canonical و معماری داخلی بررسی می‌کند.

hreflang

برای نسخه‌های زبانی یا منطقه‌ایِ یک محتوا می‌توان alternate links با hreflang ارائه کرد تا موتور جست‌وجو نسخهٔ مناسب را بهتر تشخیص دهد.

```
<link rel="alternate" hreflang="fa-IR" href="https://example.com/fa/">
<link rel="alternate" hreflang="en" href="https://example.com/en/">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

x-default

x-default می‌تواند نسخهٔ عمومی یا انتخاب زبان را برای کاربرانی که در هیچ نسخهٔ مشخصی قرار نمی‌گیرند معرفی کند.

```
<link rel="alternate" hreflang="x-default" href="https://example.com/">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

canonical صحیح

canonical باید URL استاندارد محتوای فعلی باشد و به صفحات نامرتبط اشاره نکند. canonical یک redirect یا جایگزین validation نیست.

```
<link rel="canonical" href="https://example.com/articles/html">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

JSON-LD و structured data

دادهٔ ساختاریافته معمولاً به شکل JSON-LD در script type=application/ld+json قرار می‌گیرد. schema باید با محتوای قابل مشاهده هماهنگ باشد.

```
<script type="application/ld+json">
{
  "@context":"https://schema.org",
  "@type":"Article",
  "headline":"آموزش HTML"
}
</script>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

breadcrumb

ساختار breadcrumb هم برای کاربر مفید است و هم می‌تواند با structured data تقویت شود.

```
<nav aria-label="مسیر صفحه"><ol><li><a href="/">خانه</a></li><li><a href="/courses">دوره‌ها</a></li><li aria-current="page">HTML</li></ol></nav>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

anchor text برای SEO

متن لینک توصیفی به کاربر و موتور جست‌وجو می‌گوید مقصد چیست. عنوان صفحهٔ مقصد را در صورت منطقی بودن در متن لینک منعکس کن.

```
<a href="/html/forms">آموزش کامل فرم‌های HTML</a>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

محتوای تکراری

صفحات پارامتردار و نسخه‌های مختلف URL می‌توانند محتوای مشابه ایجاد کنند؛ معماری URL و canonical باید آگاهانه طراحی شوند.

```
<link rel="canonical" href="https://example.com/products/42">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 44

فصل ۴۴ - امنیت وب در لایهٔ HTML

امنیت واقعی ترکیبی از HTML، مرورگر، HTTP، سرور و JavaScript است. در این فصل مرز مسئولیت HTML را دقیق می‌کنیم.

CSP و HTML

Content Security Policy عمدتاً یک سیاست HTTP است، اما meta http-equiv نیز در برخی سناریوها وجود دارد و نباید با هدر کامل یکی فرض شود.

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

CSP nonce و script

nonce باید توسط سرور به‌شکل غیرقابل‌پیش‌بینی تولید و در policy و عنصر script هماهنگ شود؛ مقدار ثابت یا قابل حدس راهکار امنیتی نیست.

```
<script nonce="SERVER_GENERATED_NONCE">init();</script>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

iframe sandbox

sandbox بدون توکن‌های لازم دسترسی‌های iframe را محدود می‌کند. اضافه کردن مجوزها باید حداقلی و بر اساس نیاز واقعی باشد.

```
<iframe src="/preview.html" sandbox="allow-forms" title="پیش‌نمایش"></iframe>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

allow و Permissions Policy

allow در iframe می‌تواند قابلیت‌هایی مانند fullscreen یا geolocation را کنترل کند؛ فقط دسترسی مورد نیاز را اعطا کن.

```
<iframe src="/map.html" allow="fullscreen" title="نقشه"></iframe>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

referrerpolicy

برای جلوگیری از افشای بیش از حد URL مبدأ می‌توان referrer policy مشخص کرد؛ مقدار مناسب به نیاز حریم خصوصی و تحلیل وابسته است.

```
<a href="https://example.com" referrerpolicy="strict-origin-when-cross-origin">منبع</a>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

autocomplete و دادهٔ حساس

autocomplete برای تجربهٔ کاربر مفید است و لزوماً ابزار امنیتی نیست. تصمیم دربارهٔ اطلاعات حساس باید با طراحی احراز هویت و سیاست‌های سرور هماهنگ باشد.

```
<input type="text" name="username" autocomplete="username">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

HTML escaping context-aware

دادهٔ کاربر نباید مستقیم وارد markup شود. متن، URL، attribute و JavaScript context روش‌های escaping متفاوت دارند.

```
<p id="result">متن امن باید از طریق APIهای DOM وارد شود.</p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 45

فصل ۴۵ - HTML، DOM، Browser APIs و JavaScript

HTML نقطهٔ شروع رابط است و DOM پل بین markup و منطق برنامه. این فصل مرز مسئولیت‌ها و اتصال دقیق آن‌ها را روشن می‌کند.

querySelector و id/class

برای گرفتن عنصر از DOM می‌توان از selectorهای CSS استفاده کرد. انتخاب واضح و idهای پایدار دیباگ را ساده‌تر می‌کند.

```
const form = document.querySelector("#checkout");
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

textContent در برابر innerHTML

textContent برای متن خام امن‌تر و ساده‌تر است؛ innerHTML برای ساخت markup استفاده می‌شود و اگر دادهٔ کاربر را بدون sanitization وارد کنی می‌تواند XSS ایجاد کند.

```
el.textContent = userName;
// نه: el.innerHTML = userName;
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

dataset

attributeهای data-* از طریق dataset به JavaScript قابل دسترسی‌اند و برای داده‌های کوچکِ وابسته به عنصر مناسب‌اند.

```
<button data-product-id="42">افزودن</button>
<script>console.log(document.querySelector("button").dataset.productId)</
script>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

classList

برای افزودن و حذف کلاس‌های CSS از classList استفاده می‌شود؛ تغییر کلاس معمولاً برای stateهای UI ساده و خوانا است.

```
button.classList.toggle("active");
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

createElement و append

وقتی ساختار باید با دادهٔ پویا ساخته شود، createElement و APIهای DOM می‌توانند بدون innerHTML markup امن تولید کنند.

```
const li = document.createElement("li");
li.textContent = "HTML";
list.append(li);
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

FormData

FormData به‌ویژه برای ارسال فرم با fetch و فایل‌های انتخابی مفید است و رفتار موفق کنترل‌ها را تا حد زیادی با submit واقعی هماهنگ می‌کند.

```
const data = new FormData(form);
fetch(form.action, {method: "POST", body: data});
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

Constraint Validation API

checkValidity، reportValidity و validationMessage اجازه می‌دهند JavaScript با validation بومی مرورگر هماهنگ بماند.

```
if (!form.checkValidity()) { form.reportValidity(); }
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 46

فصل ۴۶ - SVG، MathML و Canvas در HTML

HTML مدرن می‌تواند با SVG و MathML برای گرافیک و فرمول و با canvas برای رسم مبتنی بر اسکریپت ترکیب شود.

SVG در HTML

SVG برای آیکون، نمودار، illustration و گرافیک برداری مناسب است و به‌صورت inline در HTML نیز می‌تواند قرار بگیرد.

```
<svg viewBox="0 0 100 100" role="img" aria-label="دایرهٔ نمونه"><circle
cx="50" cy="50" r="40"></circle></svg>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

SVG accessible

وقتی SVG معنای بصری مهمی دارد، نام قابل دسترس و در صورت نیاز title/desc را فراهم کن؛ اگر تزئینی است می‌توان آن را برای فناوری کمکی پنهان کرد.

```
<svg aria-hidden="true" viewBox="0 0 24 24"><path d="..."></path></svg>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

MathML

MathML برای نمایش فرمول ریاضی معنایی است و برای محتوای آموزشی STEM می‌تواند از تصویر بهتر باشد.

```
<math><mrow><mi>x</mi><mo>=</mo><mn>2</mn></mrow></math>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

canvas

canvas یک بوم پیکسلی است و برای گرافیک پویا، بازی و نمودارهای تعاملی کاربرد دارد؛ محتوای canvas مانند عنصرهای داخلیِ مستقل ندارد.

```
<canvas id="chart" width="600" height="300">نمودار در مرورگر شما قابل نمایش نیست.</canvas>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

canvas و دسترس‌پذیری

برای canvas متن جایگزین و در صورت نیاز کنترل‌ها یا دادهٔ معادل ارائه کن؛ روی تصویر پیکسلی تنها حساب نکن.

```
<canvas aria-label="نمودار فروش ماهانه" role="img" width="600"
height="300">داده‌ٔ نمودار در متن توضیح داده شده است.</canvas>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

SVG در برابر CSS background

اگر گرافیک بخشی از محتوای معنادار صفحه است SVG یا img مناسب‌تر است؛ اگر صرفاً تزئینی است CSS background می‌تواند کافی باشد.

```
<img src="logo.svg" alt="لوگوی شرکت">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 47

فصل ۴۷ - Web Components در سطح حرفه‌ای

template و slot شروع کار بودند؛ این فصل custom elements، lifecycle و Shadow DOM را به‌عنوان ابزارهای ساخت اجزای قابل استفادهٔ مجدد بررسی می‌کند.

custom element names

نام custom element باید ساختار مشخص و معمولاً شامل خط تیره داشته باشد تا با عنصرهای native تداخل نکند.

```
class UserCard extends HTMLElement {}
customElements.define("user-card", UserCard);
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

connectedCallback

این lifecycle callback وقتی عنصر به DOM متصل می‌شود اجرا می‌شود و جای مناسبی برای setup اولیه است.

```
class UserCard extends HTMLElement {
  connectedCallback(){ this.textContent = "کاربر"; }
}
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

attributeChangedCallback

برای واکنش به تغییر attributeهای مشخص استفاده می‌شود و با observedAttributes کنترل می‌شود.

```
static get observedAttributes(){ return ["name"]; }
attributeChangedCallback(name, oldValue, newValue) {}
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

Shadow DOM

Shadow DOM کپسوله‌سازی بخشی از DOM و style را ممکن می‌کند و برای componentهای قابل استفادهٔ مجدد مفید است.

```
const root = this.attachShadow({mode:"open"});
root.innerHTML = `<span>سلام</span>`;
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

---

slot با fallback

slot می‌تواند محتوای ورودی کاربر component را در نقاط مشخص قرار دهد و متن fallback برای نبود محتوا مفید است.

```
<slot name="title">عنوان پیش‌فرض</slot>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

تعامل با فرم‌ها در component

برای componentهای فرم‌محور باید به form-associated custom elements و APIهای مرتبط توجه کنی؛ این حوزه از HTML معمولی پیشرفته‌تر است.

```
class MyInput extends HTMLElement {
  static formAssociated = true;
}
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 48

فصل ۴۸ - PWA، manifest و قابلیت نصب

HTML بخشی از تجربهٔ اپ وب است و می‌تواند با manifest، icons و service worker به یک PWA تبدیل شود؛ این بخش مرز HTML و APIهای مرورگر را روشن می‌کند.

Web App Manifest

manifest اطلاعاتی مثل نام، آیکون، رنگ و start_url برنامهٔ وب را معرفی می‌کند.

```
<link rel="manifest" href="/manifest.webmanifest">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

icons

آیکون‌های مناسب برای نصب و محیط‌های مختلف باید ابعاد و نوع مناسب داشته باشند.

```
<link rel="icon" href="/favicon.svg" type="image/svg+xml">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

apple-touch-icon

برخی محیط‌های اپل به آیکون جداگانه نیاز دارند؛ باید طبق نیاز پلتفرم و طراحی فعلی بررسی شود.

```
<link rel="apple-touch-icon" href="/icons/icon-180.png">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

service worker و HTML

service worker با JavaScript کنترل cache و درخواست‌ها را برعهده می‌گیرد؛ HTML تنها نقطهٔ ثبت/معرفی asset است.

```
<script type="module" src="/register-sw.js"></script>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

offline fallback

برای تجربهٔ آفلاین معمولاً shell HTML و assetهای ضروری cache می‌شوند و routeهای fallback به طراحی service worker وابسته‌اند.

```
<main><h1>آفلاین هستید</h1><p>نسخهٔ cache این صفحه از داده شده است نمایش داده.</p></main>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 49

فصل ۴۹ - بین‌المللی‌سازی، RTL و چندزبانی حرفه‌ای

در پروژه‌های فارسی، عربی و چندزبانه، lang، dir، bdi/bdo، locale و متن‌های ترکیبی نقش کلیدی دارند.

lang در بخش‌های مختلف

زبان فقط روی html تعیین نمی‌شود؛ می‌توان برای بخشی از محتوا زبان متفاوت را مشخص کرد تا خوانش و پردازش درست‌تر شود.

```
<p lang="en">This is English.</p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

dir=auto

برای محتوای ورودی کاربر که جهت آن مشخص نیست dir=auto می‌تواند اولین نویسهٔ قوی را مبنای جهت قرار دهد.

```
<p dir="auto">123ABC متن کاربر</p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

bdi برای نام کاربر

وقتی رشته‌ای با جهت نامعلوم داخل متن RTL قرار می‌گیرد، bdi از اثرگذاری جهت آن بر متن اطراف جلوگیری می‌کند.

```
<p>کاربر: <bdi>Alex_2026</bdi></p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

bdo

bdo جهت را به‌صورت صریح override می‌کند و برای نمونه‌های آموزشی یا متن‌هایی با نیاز خاص استفاده می‌شود.

```
<p><bdo dir="ltr">ABC-123</bdo></p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

ruby برای تلفظ/annotation

ruby در برخی زبان‌ها برای نمایش راهنماهای تلفظ یا annotation بالای متن کاربرد دارد.

```
<ruby>漢<rt>かん</rt></ruby>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

ترکیب فارسی و کد

برای نمایش کد انگلیسی در صفحهٔ RTL از code و CSS مناسب استفاده کن تا جهت و خوانایی به‌هم نریزد.

```
<p dir="rtl">نسخهٔ <code dir="ltr">HTML</code> را اجرا کن.</p>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

فصل 50

فصل ۵۰ - اعتبارسنجی، تست، دیباگ و استاندارد حرفه‌ای HTML

آخرین مرحلهٔ مهارت HTML فقط نوشتن کد نیست؛ باید بتوانی خطا را پیدا کنی، DOM واقعی را ببینی، اعتبار markup را بررسی کنی و صفحه را در شرایط مختلف تست کنی.

HTML Validator

یک validator خطاهای ساختاری، attributeهای نامعتبر و برخی ناسازگاری‌های استاندارد را آشکار می‌کند. validator جای تست کاربردی را نمی‌گیرد.

```
<!doctype html>
<html lang="fa">
<head><meta charset="utf-8"><title>نمونه</title></head>
<body><h1>سلام</h1></body>
</html>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

DevTools Elements

پنل Elements برای دیدن DOM واقعی، attributeها، stateهای فرم و تغییرات لحظه‌ای حیاتی است. تفاوت view-source و DOM live را بشناس.

```
<!-- view-source؛ متن خام را نشان می‌دهد  Elements DOM فعلی را -->
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

View Source

View Source چیزی نزدیک‌تر به متن HTML دریافت‌شده را نشان می‌دهد، در حالی که JavaScript ممکن است DOM را بعداً تغییر داده باشد.

```
<script>document.body.dataset.debug="yes";</script>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

کنسول و خطاهای resource

خطاهای 404، CORS، MIME type و CSP اغلب در Network/Console مشخص می‌شوند و همیشه ناشی از خود HTML نیستند.

```
<img src="/images/missing.webp" alt="تصویر">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

تست با keyboard

با Tab، Shift+Tab، Enter و Space کل صفحه را بررسی کن و مطمئن شو focus گم نمی‌شود و کنترل‌های تعاملی واقعاً قابل استفاده‌اند.

```
<nav aria-label="ناوبری اصلی"><a href="/">خانه</a><a href="/courses">دوره‌ها</
a></nav>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

تست موبایل

viewport، اندازهٔ کنترل‌ها، media، overflow و طول متن را روی viewportهای کوچک آزمایش کن؛ ظاهر دسکتاپ تضمین‌کنندهٔ تجربهٔ موبایل نیست.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

تست با صفحه‌خوان

خوانش headingها، label، landmarks و statusها را با یک screen reader بررسی کن؛ این کار خطاهایی را آشکار می‌کند که با چشم دیده نمی‌شوند.

```
<main aria-labelledby="page-title"><h1 id="page-title">پروفایل</h1></main>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

خطاهای encoding و BOM

charset، UTF-8 و نام‌گذاری فایل‌ها باید هماهنگ باشند. اگر فارسی به شکل mojibake دیده می‌شود، زنجیرهٔ encoding را بررسی کن.

```
<meta charset="utf-8">
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

HTML برای تولید، نه فقط تمرین

در محیط واقعی باید security headers، cache، minification، build pipeline و template engine را هم در کنار HTML بسنجی؛ HTML تمیز باید بخشی از سیستم باشد.

```
<link rel="stylesheet" href="/assets/app.css">
<script type="module" src="/assets/app.js"></script>
```

کاربرد و نکته: این مبحث را در یک پروژهٔ واقعی اجرا کن، سپس خروجی را در DevTools و یک validator بررسی کن. سعی کن یک نسخهٔ آگاهانهٔ اشتباه هم بسازی تا تفاوت رفتار مرورگر را ببینی.

مرجع نهایی

مرجع نهایی - عناصر، ویژگی‌ها و الگوهای سریع

این بخش برای مرور سریع بعد از مطالعهٔ فصل‌هاست: عناصر استاندارد، عناصر کمتر دیده‌شده، ویژگی‌های مهم و الگوهای آماده در چند گروه فشرده شده‌اند.

عناصر اصلی سند

عناصر/ویژگی‌ها: html, head, body, title, meta, link, base, style, script, noscript

کجا به کار می‌آید: برای ساخت خود سند و کنترل metadata و منابع.

```
<!-- الگوی ذهنی -->
<!-- سپس، انتخاب کن اول عنصر معنایی را اضافه کنattribute های لازم را. -->
```

ساختار معنایی

عناصر/ویژگی‌ها: main, header, footer, nav, section, article, aside, h1-h6, address

کجا به کار می‌آید: برای معماری معنی‌دار صفحه و landmarks.

```
<!-- الگوی ذهنی -->
<!-- های لازم را اضافه کنattribute اول عنصر معنایی را انتخاب کن، سپس .-->
```

متن

عناصر/ویژگی‌ها: p, div, span, strong, em, b, i, mark, small, s, del, ins, q, blockquote, cite, abbr, dfn, code, pre, kbd, samp, var, time, data, sub, sup, bdi, bdo, ruby, rt, rp, br, wbr, hr

کجا به کار می‌آید: برای نمایش و انتقال معنای متن.

```
<!-- الگوی ذهنی -->
<!-- های لازم را اضافه کنattribute اول عنصر معنایی را انتخاب کن، سپس .-->
```

پیوند و رسانه

عناصر/ویژگی‌ها: a, area, map, img, picture, source, audio, video, track, iframe, object, embed

کجا به کار می‌آید: برای navigation و embedded content.

```
<!-- الگوی ذهنی -->
<!-- های لازم را اضافه کنattribute اول عنصر معنایی را انتخاب کن، سپس .-->
```

لیست و جدول

عناصر/ویژگی‌ها: ul, ol, li, menu, dl, dt, dd, table, caption, colgroup, col, thead, tbody, tfoot, tr, th, td

کجا به کار می‌آید: برای داده‌های ترتیبی و جدولی.

```
<!-- الگوی ذهنی -->
<!-- های لازم را اضافه کنattribute اول عنصر معنایی را انتخاب کن، سپس .-->
```

فرم

عناصر/ویژگی‌ها: form, label, input, button, select, option, optgroup, textarea, datalist, fieldset,

legend, output, progress, meter

کجا به کار می‌آید: برای دریافت و ارائهٔ داده.

```
<!-- الگوی ذهنی -->
<!-- های لازم را اضافه کنattribute اول عنصر معنایی را انتخاب کن، سپس .-->
```

تعامل بومی

عناصر/ویژگی‌ها: details, summary, dialog

کجا به کار می‌آید: برای interactionهای پایه بدون ساخت widget جعلی.

```
<!-- الگوی ذهنی -->
<!-- های لازم را اضافه کنattribute اول عنصر معنایی را انتخاب کن، سپس .-->
```

گرافیک و فرمت‌های تعبیه‌شده

عناصر/ویژگی‌ها: svg, math, canvas

کجا به کار می‌آید: برای vector graphic، فرمول و رسم پویا.

```
<!-- الگوی ذهنی -->
<!-- های لازم را اضافه کنattribute اول عنصر معنایی را انتخاب کن، سپس .-->
```

ویژگی‌های بسیار مهم

عناصر/ویژگی‌ها: id, class, style, title, lang, dir, hidden, inert, contenteditable, spellcheck, translate,

data, *-tabindex

کجا به کار می‌آید: برای کنترل هویت، رفتار، جهت و metadata عنصر.


```
<!-- الگوی ذهنی -->
<!-- های لازم را اضافه کنattribute اول عنصر معنایی را انتخاب کن، سپس .-->
```


فرم و ارسال

عناصر/ویژگی‌ها: action, method, enctype, name, value, required, disabled, readonly, checked,

selected, multiple, autocomplete, accept, min, max, step, pattern, minlength, maxlength,

inputmode, dirname, form, formaction, formmethod, formenctype, formnovalidate

کجا به کار می‌آید: برای رفتار و اعتبارسنجی دقیق فرم.


```
<!-- الگوی ذهنی -->
<!-- های لازم را اضافه کنattribute اول عنصر معنایی را انتخاب کن، سپس .-->
```

منابع و عملکرد

عناصر/ویژگی‌ها: loading, decoding, fetchpriority, rel=preload, preconnect, dns-prefetch,
modulepreload, defer, async, type=module, integrity, crossorigin

کجا به کار می‌آید: برای resource loading و runtime integration.

```
<!-- الگوی ذهنی -->
<!-- سپس، انتخاب کن اول عنصر معنایی را اضافه کنattributeهای لازم را. -->
```

امنیت و حریم خصوصی

عناصر/ویژگی‌ها: sandbox, allow, referrerpolicy, nonce, integrity, crossorigin

کجا به کار می‌آید: برای کاهش سطح دسترسی و کنترل منابع.

```
<!-- الگوی ذهنی -->
<!-- سپس، انتخاب کن اول عنصر معنایی را اضافه کنattributeهای لازم را. -->
```

SEO

عناصر/ویژگی‌ها: title, meta description, canonical, robots, hreflang, alternate, JSON-LD

کجا به کار می‌آید: برای فهم بهتر محتوا و نسخهٔ اصلی صفحات.

```
<!-- الگوی ذهنی -->
<!-- سپس، انتخاب کن اول عنصر معنایی را اضافه کنattributeهای لازم را. -->
```

ضمیمه A

ضمیمه A - Cheat Sheet سریع HTML

در این برگهٔ خلاصه، عناصر پرتکرار را یکجا داری تا هنگام کدنویسی سریع به آن مراجعه کنی.

ساختار

```
&lt;!DOCTYPE html&gt;
&lt;html dir="rtl" lang="fa"&gt;
  &lt;head&gt;...&lt;/head&gt;
  &lt;body&gt;...&lt;/body&gt;
&lt;/html&gt;
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «ساختار» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

متن

```
&lt;h1&gt;...&lt;/h1&gt;
&lt;p&gt;...&lt;/p&gt;
&lt;strong&gt;...&lt;/strong&gt;
&lt;em&gt;...&lt;/em&gt;
&lt;br&gt;
&lt;hr&gt;
```

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

لینک و تصویر

```
&lt;a href="/page"&gt;لینک&lt;/a&gt;
&lt;img src="img.webp" alt="توضیح"&gt;
&lt;picture&gt;...&lt;/picture&gt;
```

کاربرد عملی: در پروژهٔ واقعی فقط نمایش تصویر مهم نیست؛ باید متن جایگزین، اندازهٔ مناسب، نسبت تصویر، بارگذاری تنبل در موارد مناسب و نسخهٔ مناسب برای صفحه‌نمایش‌های مختلف را هم مدیریت کنی.

نکات دقیق و خطاهای رایج: برای تصاویر تزئینی معمولاً alt خالی بهتر از متن ساختگی است؛ برای تصویرِ دارای اطلاعات، alt باید همان اطلاعات مهم را منتقل کند. همچنین width و height از تغییر ناگهانی چیدمان جلوگیری می‌کنند.

```
<img src="product.webp" alt="گوشی مشکی با نمایشگر 6.4 اینچی" width="640"
height="480" loading="lazy">
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فهرست

```
&lt;ul&gt;&lt;li&gt;...&lt;/li&gt;&lt;/ul&gt;
&lt;ol&gt;&lt;li&gt;...&lt;/li&gt;&lt;/ol&gt;
&lt;dl&gt;&lt;dt&gt;...&lt;/dt&gt;&lt;dd&gt;...&lt;/dd&gt;&lt;/dl&gt;
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «فهرست» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

ساختار معنایی

```
&lt;header&gt;...&lt;/header&gt;
&lt;nav&gt;...&lt;/nav&gt;
&lt;main&gt;...&lt;/main&gt;
&lt;section&gt;...&lt;/section&gt;
&lt;article&gt;...&lt;/article&gt;
&lt;aside&gt;...&lt;/aside&gt;
&lt;footer&gt;...&lt;/footer&gt;
```

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «ساختار معنایی» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

فرم

```
&lt;form&gt;
  &lt;label&gt;...&lt;/label&gt;
  &lt;input&gt;
  &lt;select&gt;...&lt;/select&gt;
  &lt;textarea&gt;...&lt;/textarea&gt;
  &lt;button type="submit"&gt;...&lt;/button&gt;
&lt;/form&gt;
```

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.
نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت‌نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

رسانه

```
&lt;audio controls&gt;...&lt;/audio&gt;
&lt;video controls&gt;...&lt;/video&gt;
&lt;iframe title="..."&gt;...&lt;/iframe&gt;
```

کاربرد عملی: رسانهٔ وب یک موضوع فنی چندبخشی است: منبع فایل، فرمت، کنترل پخش، دسترس‌پذیری، زیرنویس، بارگذاری، پوستر و سازگاری مرورگر باید با هم دیده شوند.

نکات دقیق و خطاهای رایج: برای محتوای آموزشی زیرنویس، کنترل‌های واضح و fallback مناسب مهم‌اند. autoplay به‌خصوص برای صدا باید با احتیاط استفاده شود.

```
<video controls preload="metadata" poster="poster.jpg" width="720">
  <source src="lesson.mp4" type="video/mp4">
  <track src="fa.vtt" kind="subtitles" srclang="fa" label="فارسی">
</video>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

ضمیمه B

ضمیمه B - مسیر یادگیری سریع از صفر تا تسلط

به جای حفظ همهٔ تگ‌ها، مسیر را مرحله‌ای طی کن و هر مرحله را با پروژه تثبیت کن.

مرحله ۱

HTML پایه: ساختار سند، heading، paragraph، link، image، list.

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.

نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

مرحله ۲

ساختار معنایی: header، nav، main، section، article، footer و جدول‌ها.

کاربرد عملی: جدول برای دادهٔ دوبعدی و رابطهٔ سطر/ستون مناسب است، نه برای چیدمان صفحه. ساختار صحیح جدول شامل عنوان، سرستون‌های واضح و در صورت نیاز ارتباط دقیق headerهاست.

نکات دقیق و خطاهای رایج: هرچه جدول پیچیده‌تر شود، استفاده از scope و در موارد خاص id/headers مهم‌تر می‌شود. برای موبایل نیز رفتار overflow را با CSS حل کن، نه با حذف داده‌ها.

```
<table>
  <caption>فروش ماهانه</caption>
  <thead><tr><th scope="col">ماه</th><th scope="col">فروش</th></tr></thead>
  <tbody><tr><td>فروردین</td><td>120</td></tr></tbody>
</table>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

مرحله ۳

فرم‌ها: input typeها، validation، textarea، select، label و upload.

کاربرد عملی: فرم در HTML فقط مجموعه‌ای از inputها نیست؛ ارتباط label با کنترل، نامِ قابل ارسال، روش ارسال، نوع داده، وضعیت اعتبارسنجی و تجربهٔ کاربر همگی بخشی از طراحی صحیح فرم هستند.

نکات دقیق و خطاهای رایج: مهم‌ترین تست این است که فرم را فقط با صفحه‌کلید، بدون JavaScript و با دادهٔ نامعتبر هم امتحان کنی. اعتبارسنجی سمت مرورگر جای اعتبارسنجی سرور را نمی‌گیرد.

```
<form action="/signup" method="post">
  <label for="email">ایمیل</label>
  <input id="email" name="email" type="email" autocomplete="email" required>
  <button type="submit">ثبت‌نام</button>
</form>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

مرحله ۴

Accessibility و SEO: heading structure، alt، landmark، metadata و لینک داخلی.

کاربرد عملی: کار اصلی این مبحث ایجاد مسیر قابل فهم برای کاربر و ربات است؛ متن لینک باید مقصد را توضیح دهد و URLها باید برای ساختار سایت، جابه‌جایی صفحات و اشتراک‌گذاری قابل اتکا باشند.

نکات دقیق و خطاهای رایج: برای لینک خارجیِ بازشده در تب جدید، rel مناسب را فراموش نکن. برای لینک داخلی از متن‌های مبهم مثل «اینجا کلیک کن» دوری کن.

```
<a href="/courses/html">دورهٔ کامل HTML</a>
<a href="https://example.com" target="_blank" rel="noopener noreferrer">منبع خارجی</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

مرحله ۵

HTML پیشرفته: template/slot، dialog/details، picture/source/track و iframe sandbox.

کاربرد عملی: عناصر متنی زمانی ارزشمندند که اطلاعاتی دربارهٔ «معنای» متن بدهند، نه اینکه صرفاً ظاهر آن را تغییر دهند. این معنا روی دسترس‌پذیری، SEO و ابزارهای پردازش محتوا اثر می‌گذارد.

نکات دقیق و خطاهای رایج: برای بزرگ یا پررنگ کردن متن از HTML معنایی اشتباه استفاده نکن؛ ظاهر باید مسئولیت CSS باشد و HTML مسئولیت معنا را بر عهده بگیرد.

```
<p><strong>هشدار:</strong> این بخش مهم است؛ جزئیات بیشتر در مستندات همین فصل آمده است.</p>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

مرحله ۶

سه پروژه کامل: پروفایل شخصی، صفحه محصول، صفحه مقاله/وبلاگ.

کاربرد عملی: بخش بزرگی از خطاهای HTML در پروژه‌های واقعی به مسیر نادرست فایل‌ها مربوط است. مسیر نسبی نسبت به URL سند فعلی و مسیر مطلق نسبت به ریشهٔ origin تفسیر می‌شود.

نکات دقیق و خطاهای رایج: روی Windows ممکن است تفاوت حروف را نبینی ولی روی بسیاری از سرورها نام فایل case-sensitive است. نام‌گذاری ثابت از خطاهای عجیب جلوگیری می‌کند.

```
<img src="./images/logo.svg" alt="لوگو">
<a href="../index.html">بازگشت</a>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

مرحله ۷

بعد از تسلط بر HTML، CSS را جدی یاد بگیر و سپس JavaScript را برای رفتار و تعامل اضافه کن.

کاربرد عملی: این مبحث بخشی از طراحی صحیح HTML برای «مرحله ۷» است. آن را فقط حفظ نکن؛ در یک صفحهٔ واقعی، نقش معنایی، ورودی‌ها و خروجی آن و رابطه‌اش با مرورگر و سایر عناصر را بررسی کن.

نکات دقیق و خطاهای رایج: نکتهٔ حرفه‌ای: قبل از انتخاب هر عنصر از خودت بپرس آیا عنصر معنایی‌تری وجود دارد؟ بعد صفحه را با keyboard، موبایل و یک validator بررسی کن.

```
<section>
  <h2>نمونهٔ عملی</h2>
  <p>این بخش را در پروژهٔ خودت با دادهٔ واقعی امتحان کن.</p>
</section>
```

تمرین سریع: مثال بالا را در یک فایل HTML اجرا کن و سپس یک تغییر واقعی روی آن بده؛ مثلاً زبان، نام فیلد، مسیر فایل یا ساختار معناشناختی را تغییر بده و نتیجه را بررسی کن.

منابع مرجع: برای تطبیق با استاندارد زندهٔ HTML و مستندات عملی، منابعی مانند HTML Living Standard (WHATWG) و HTML Reference / MDN به‌عنوان منابع مرجع در نظر گرفته شده‌اند.

یادداشت پایانی: HTML را با نوشتن پروژه یاد بگیر، نه با حفظ‌کردن فهرست تگ‌ها. هر مبحث را همان روز با یک صفحهٔ واقعی تمرین کن.

این نسخه برای مطالعه روی موبایل و دسکتاپ طراحی شده و در قالب HTML قابل جست‌وجو و چاپ است.