150 HTML and CSS Interview Questions and Answers for UI Developers

HTML Fundamentals & Semantics

1. **What is HTML?**
   Answer: HTML (HyperText Markup Language) is the standard markup language for structuring content on the web. It defines elements, attributes, and how content is organized (headings, paragraphs, lists, forms, media, etc.).

2. **Whats the difference between HTML and HTML5?**
   Answer: HTML5 is the latest version of HTML. It introduced new semantic elements (e.g. `<header>`, `<footer>`, `<article>`, `<section>`), multimedia tags (`<video>`, `<audio>`), canvas, local storage, and APIs (geolocation, drag & drop, etc.).

3. **What is the basic structure of an HTML document?**
   Answer:
   ```
   <!DOCTYPE html>
   <html lang="en">
   <head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Page Title</title>
   <!- CSS, meta tags, etc.  ->
   </head>
   <body>
   <!- Page content ->
   </body>
   </html>
   ```

4. **What is a semantic element? Give examples.**
   Answer: Semantic elements clearly describe their meaning in a human- and machine-readable way. Examples: `<header>`, `<footer>`, `<nav>`, `<article>`, `<section>`, `<aside>`, `<main>`. Using semantics helps accessibility, SEO, and code clarity.

5. **What is the difference between a tag, an element, and an attribute?**
   Answer:
   Tag is the syntax you write, e.g. `<p>`, `</p>`.
   Element is the full thing: `<p>some text</p>`.
   Attribute is extra information inside a tag, e.g. `<img src="img.jpg" alt="desc">`, where `src` and `alt` are attributes.

6. **What are void elements (self-closing elements) in HTML?**
   Answer: Void (or empty) elements dont have a closing tag and cannot have content. Examples: `<img>`, `<br>`, `<hr>`, `<meta>`, `<link>`, `<input>`.

7. **Difference between `<div>` and `<span>`.**
   Answer: `<div>` is a block-level element that starts on a new line, occupies full width by default; `<span>` is inline, does not break the flow, only takes as much width as its content.

8. **Explain block-level vs inline elements.**
   Answer:
   Block-level elements (like `<div>`, `<p>`, `<h1>`) start on a new line and expand to the available width.
   Inline elements (like `<span>`, `<a>`, `<img>`) do not start a new line; they only take as much width as needed.

9. **What are semantic vs non-semantic tags?**
   Answer: Semantic tags (e.g. `<header>`, `<section>`) carry meaning about their content. Non-semantic tags (e.g. `<div>`, `<span>`) do not carry any meaning theyre generic containers.

10. **What is the `<!DOCTYPE html>` declaration? Why is it needed?**
    Answer: It tells the browser to render the document in standards mode (HTML5). Without it, browsers may go into quirks mode (backward-compatibility behaviors).

11. **Whats the difference between `<b>` and `<strong>` (or `<i>` vs `<em>`)?**
    Answer: `<b>` and `<i>` are presentational (bold/italic without meaning). `<strong>` and `<em>` convey meaning (strong importance, emphasis) and are semantic; screen readers may treat them differently.

12. **What are meta tags? Name a few common meta tags.**
    Answer: Meta tags provide metadata about the HTML document (not displayed directly). Examples:
    ```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="keywords" content="">
    ```

13. **What is the `<head>` element for?**
    Answer: The `<head>` contains metadata, title, links to CSS/JS, scripts (that need to be loaded earlier), meta tags, etc. It does not contain visible page content.

14. **What is the `<body>` element?**
    Answer: Contains all the content displayed on the page: text, images, video, forms, etc.

15. **Whats the difference between absolute and relative URLs?**
    Answer:
    Absolute URL: full path including protocol and domain (e.g. `https://www.example.com/page.ht`
    Relative URL: path relative to current page (e.g. `images/pic.jpg`, `../styles/style.css`).

16. **What is the `<link>` tag? When is it used?**
    Answer: `<link>` is used to link external resources (commonly CSS). E.g.:

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

17. **Whats the `<meta viewport>` tag for responsive design?**
Answer: It sets how a page should scale on mobile devices. A common one is:
`<meta name="viewport" content="width=device-width, initial-scale=1.0">`

18. **What is ARIA and when would you use it?**
Answer: ARIA (Accessible Rich Internet Applications) attributes (e.g. `aria-label`, `aria-hidden`, `role`) help make custom or dynamic HTML more accessible to users of assistive technologies (screen readers etc.).

19. **Explain the `alt` attribute in `<img>`. Why is it important?**
Answer: The `alt` attribute gives alternative text when the image cannot load or for screen readers. It's crucial for accessibility and SEO.

20. **What is the `<form>` element? How do `method` and `action` attributes work?**
Answer: `<form>` defines a form for user input.
`method` can be `GET` or `POST` (how data is sent).
`action` is the URL endpoint to which form data is sent.

21. **What is `GET` vs `POST` in forms?**
Answer:
`GET`: parameters are appended to URL (visible, limited length), used for idempotent requests.
`POST`: sends data in request body (not visible in URL), used for data submission.

22. **What are `<thead>`, `<tbody>`, `<tfoot>` in tables?**
Answer: They help structure a table into header, body, and footer sections, improving readability and semantics.

23. **What is the `<iframe>` tag? What are the risks?**
Answer: `<iframe>` embeds another page/document into the current page. Risks include security (clickjacking), performance issues, cross-domain issues.

24. **What are HTML comments? Syntax?**
Answer: Comments are ignored by browser:
`<!- This is a comment ->`

25. **What is the difference between `id` and `class` attributes?**
Answer:
`id` should be unique on the page; selects single element.
`class` can be reused; selects groups of elements.

26. **What is the `lang` attribute on `<html>`?**
Answer: It specifies the language of the document (e.g. `lang="en"`). Important for accessibility, SEO, and proper rendering of characters.

27. **What is the correct way to include JavaScript in HTML?**
Answer: You can use `<script>` tags either inline or with `src`. Better: place script tags at end of `<body>` or use `defer`/`async` to avoid blocking rendering.

28. **What are `async` and `defer` when loading scripts?**
Answer:
`async`: script is fetched asynchronously and executed immediately when ready (may

block DOM parsing).
`defer`: fetched asynchronously, but execution is deferred until after HTML parsing finishes.

29. **What are data attributes (`data-*`)? Give use case.**
Answer: Custom attributes starting with `data-` (e.g. `data-id="123"`) to store custom data in HTML. JavaScript can read them via `element.dataset.*`.

30. **What is the difference between `innerHTML` and `textContent`?**
Answer:
`innerHTML`: returns HTML inside the element (including tags).
`textContent`: returns only text nodes (no HTML tags).

31. **What is the difference between element and node in DOM?**
Answer: A node is any object in the DOM tree (element node, text node, comment node, etc.). An element is a type of node representing HTML tags.

32. **Explain the concept of the DOM (Document Object Model).**
Answer: The DOM is a programming interface for HTML and XML documents. It represents the page so that scripts can change document structure, style, content, etc.

33. **What is browser parsing and rendering process of HTML?**
Answer: The browser downloads HTML, parses it to construct the DOM tree, downloads CSS and builds CSSOM, merges into Render Tree, then layout, painting, and compositing.

34. **What is canonical URL / why is `rel="canonical"` used?**
Answer: The `rel="canonical"` link in the `<head>` indicates the preferred URL of a page to avoid duplicate content issues for SEO.

35. **What is the `<picture>` element and `srcset` / `sizes` attributes?**
Answer: `<picture>` allows responsive images with multiple sources (e.g. different formats, sizes). `srcset` provides a list of image candidates, `sizes` gives hints for which image to pick at various viewport widths.

36. **When to use `<section>` vs `<div>` vs `<article>`?**
Answer:
`<section>`: thematically grouped content with a heading.
`<article>`: self-contained content that can be independently distributed (e.g. blog post).
`<div>`: for grouping where semantics arent needed.

37. **What is the difference between HTML and XHTML?**
Answer: XHTML is stricter, XML-based, requiring well-formed tags (lowercase, closed tags, etc.). HTML5 is more forgiving. XHTML is less common nowadays.

38. **What is the `role` attribute?**
Answer: `role` gives accessibility information (ARIA roles) to non-semantic elements to help screen readers, e.g. `role="button"`.

39. **What is the `download` attribute for `<a>` tag?**
Answer: If `<a href="file.pdf" download>`, clicking will prompt a download in-

stead of navigating to it.

40. **What is the `hidden` attribute in HTML?**
Answer: It is a Boolean attribute that indicates the element is not relevant and is hidden. Browsers do not render it (as if `display: none`).

CSS Basics & Fundamentals

41. **What is CSS?**
Answer: Cascading Style Sheets (CSS) describes how HTML elements are to be displayed (colors, layout, fonts, spacing). It separates presentation from content.

42. **What are the different ways to apply CSS to HTML?**
Answer:
Inline: `<div style="color: red;">`
Internal / Embedded: `<style>` in document head
External: separate `.css` file linked via `<link>`

43. **What is the CSS cascade?**
Answer: The cascade is how CSS rules are applied when multiple rules match the same element. It uses specificity, source order, and importance (`!important`) to decide which rule wins.

44. **What is specificity and how is it calculated?**
Answer: Specificity determines which CSS rule applies. It can be thought of as a score:
Inline style: highest
IDs count more
Classes, attributes, pseudo-classes
Element selectors, pseudo-elements
Universal selector, combinators, inherited styles have lowest

45. **What is the CSS Box Model?**
Answer: Every element is a box comprising: content padding border margin. CSS layout and sizing depend on this model.

46. **What is `box-sizing` and different values?**
Answer: `box-sizing` changes how width and height are calculated.
`content-box` (default): width/height apply to content; padding/border are added outside.
`border-box`: width/height include content + padding + border.

47. **What are CSS selectors? Name types.**
Answer: Selectors pick elements to style. Types:
Element selectors (`div`)
Class selectors (`.my-class`)
ID selectors (`#my-id`)
Attribute selectors (`input[type="text"]`)
Pseudo-class (`:hover`, `:nth-child`)
Pseudo-element (`::before`, `::after`)
Descendant, child, sibling combinators

48. **What are pseudo-classes and pseudo-elements?**
Answer:
Pseudo-classes target elements in special states (`:hover`, `:focus`, `:nth-child(2)`).
Pseudo-elements target parts of an element (`::before`, `::after`, `::first-line`).

49. **What is a CSS preprocessor (Sass, Less, Stylus)?**
Answer: A preprocessor extends CSS with features like variables, nesting, mixins, functions, and then compiles to standard CSS.

50. **What is flexbox?**
Answer: A CSS module (Flexible Box Layout) for arranging items in a one-dimensional layout (row or column), enabling alignment, spacing and reordering with ease.

51. **What is CSS Grid?**
Answer: A two-dimensional layout system (rows and columns) that allows complex layouts without floats or positioning, using `grid-template-rows/columns`, `grid-area`, etc.

52. **Difference between flexbox vs grid?**
Answer:
Flexbox is one-dimensional (row or column) layout, good for smaller layout control along one axis.
Grid is two-dimensional (both axes), powerful for overall page layouts.

53. **What is `float`, and how/when is it used?**
Answer: `float` moves an element left or right, letting inline content wrap around it. Historically used for layouts (columns), but modern layouts prefer flex or grid.

54. **What is `clear` property in CSS?**
Answer: Specifies which sides of an element where floating elements are not allowed. E.g. `clear: both` ensures the element moves below floats.

55. **What is `position` in CSS? Types and how they work?**
Answer: `position` determines how an element is placed. Types: `static` (default, normal flow), `relative` (offset from normal), `absolute` (relative to nearest positioned ancestor), `fixed` (relative to viewport), `sticky` (acts as relative until certain scroll, then fixed).

56. **How to center a div both horizontally and vertically?**
Answer: Many ways, e.g.:
```
.parent {
display: flex;
justify-content: center;
align-items: center;
}
```
Or with grid:
```
.parent {
display: grid;
place-items: center;
}
```
Or using absolute + transform:

```
.child {
position: absolute;
top: 50%; left: 50%;
transform: translate(-50%, -50%);
}
```

57. **What are CSS media queries?**
Answer: They allow applying different CSS rules based on device/viewport characteristics (width, height, orientation, resolution). Essential for responsive design.

58. **What is responsive design vs adaptive design?**
Answer:
Responsive: layout adjusts fluidly using CSS (flex, grid, media queries).
Adaptive: predefined static layouts for certain breakpoints; switches based on device size.

59. **What is a CSS reset / `normalize.css`? Why use it?**
Answer: Browsers have default styles for elements. A CSS reset or `normalize.css` resets or normalizes these defaults to a consistent baseline cross-browser.

60. **What is `z-index` and how does stacking context work?**
Answer: `z-index` controls overlapping of positioned elements. Stacking context is an environment where elements are ordered by `z-index`; new stacking contexts can form via `position`, `opacity`, `transform`, etc.

61. **What is `!important` and when to use it?**
Answer: `!important` forces a rule to override any other (except another `!important` with higher specificity). Use sparingly, generally avoided in maintainable CSS.

62. **What is CSS `transitions`? How to use them?**
Answer: Simplest way to animate CSS property changes. Example:
```
.box {
transition: background-color 0.3s ease;
}
.box:hover {
background-color: red;
}
```

63. **What are CSS keyframe animations?**
Answer: Define animations in stages (percentages). Example:
```
@keyframes slide {
from { transform: translateX(0); }
to { transform: translateX(100px); }
}
.box {
animation: slide 2s infinite alternate;
}
```

64. **What are CSS variables (custom properties)?**
Answer: Variables declared with `-my-var: value;` inside selectors (often `:root`). Use with `var(-my-var)`. They support cascading and runtime changes.

65. **What is CSS specificity of custom properties?**
Answer: Custom properties themselves dont have specificity; they follow regular CSS evaluation. But using them in declarations doesnt affect specificity of the rule using them.

66. **What is the universal selector `*`? Use case and drawbacks.**
Answer: `*` matches all elements. Used rarely (e.g. `* { box-sizing: border-box; }`). Drawback: performance hit, may cause unintended styling and inheritance.

67. **What is `inherit`, `initial`, `unset` CSS keywords?**
Answer:
`inherit`: explicitly inherit from parent.
`initial`: use the initial value defined in CSS spec.
`unset`: acts like `inherit` if property is inherited, else `initial`.

68. **What are CSS combinators?**
Answer:
Descendant:
Child: `>`
Adjacent sibling: `+`
General sibling: `~`

69. **What is `opacity` vs `visibility: hidden` vs `display: none`?**
Answer:
`opacity: 0`: makes element transparent but it still occupies space and responds to clicks.
`visibility: hidden`: element is invisible but space is preserved, no click events.
`display: none`: element is removed from layout, no space taken.

70. **How to make a CSS triangle (without any images)?**
Answer: Use border trick:

```
.triangle {
width: 0;
height: 0;
border-left: 50px solid transparent;
border-right: 50px solid transparent;
border-bottom: 100px solid red;
}
```

Intermediate / Advanced HTML & CSS Questions

71. **What is the purpose of `srcset` in `<img>`?**
Answer: It allows specifying multiple image versions for different screen density and widths so the browser can pick the best one. E.g. `srcset="small.jpg 1x, large.jpg 2x"`.

72. **What is `<picture>` with `<source>` element?**
Answer: Allows selecting different images (e.g. format, resolution) per media condition. E.g.:
`<picture>`

```
<source srcset="img.webp" type="image/webp">
<source srcset="img.jpg" type="image/jpeg">
<img src="img.jpg" alt="">
</picture>
```

73. **How to lazy-load images in HTML?**
Answer: Use the `loading="lazy"` attribute on `<img>` or use Intersection Observer in JavaScript to load when element enters viewport.

74. **What is progressive rendering / critical CSS?**
Answer: Load above-the-fold CSS inline (critical CSS) to allow quick first render, and defer non-critical CSS to speed up page load and perceived performance.

75. **What are media features / queries beyond width (e.g. `device-pixel-ratio`)?**
Answer: You can query `prefers-color-scheme`, `orientation`, `resolution` (for retina), `hover`, `pointer`, etc.

76. **What is CSS `object-fit` and `object-position`?**
Answer: Used on replaced elements (images, videos) to control how they fill their container. `object-fit: cover|contain` etc; `object-position` aligns inside the container.

77. **What is `clamp()`, `min()`, `max()` in CSS?**
Answer: CSS functions for responsive sizing. E.g. `width: clamp(200px, 50%, 400px)` sets width between 200px and 400px, ideally 50%.

78. **What is `calc()` in CSS?**
Answer: Allows mathematical expressions in CSS, e.g. `width: calc(100% - 40px)`.

79. **What are CSS logical properties?**
Answer: Logical (writing-mode aware) properties like `margin-inline-start`, `padding-block-end` etc., instead of physical `margin-left`, etc., useful for internationalization.

80. **What is CSS scroll snapping?**
Answer: Use `scroll-snap-type`, `scroll-snap-align` to snap scroll positions to children elements, creating nice scroll experiences.

81. **What is CSS containment (`contain` property)?**
Answer: Helps performance by isolating parts of DOM for layout, paint, style. Eg. `contain: layout paint;`.

82. **What is CSS `will-change`?**
Answer: Hint to browser which CSS properties on an element are expected to change (e.g. `transform`, `opacity`), to optimize against expensive repaints/composites.

83. **What are `@supports` rules?**
Answer: Feature queries: check browser support for CSS properties. E.g.
```
@supports (display: grid) {
.container { display: grid; }
}
```

84. **What is CSS `isolation` / `isolation: isolate`?**
Answer: Creates a new stacking context for children, so `z-index` stacking doesnt

leak outside.

85. **What are CSS masks / `clip-path`?**
Answer: Use `clip-path` or masking to shape or hide parts of elements (e.g. circular, polygon shapes).

86. **What is `filter` in CSS?**
Answer: Apply image effects (blurring, brightness, contrast, etc.) via CSS, e.g. `filter: blur(5px)`.

87. **What is CSS `backdrop-filter`?**
Answer: Applies filter effects to backdrop behind an element (e.g. background-blur behind translucent overlay).

88. **Explain reflow vs repaint in browser.**
Answer:
Reflow (layout): browser recalculates positions/sizes of elements.
Repaint (paint): browser draws pixels. Reflow is more expensive.

89. **How to avoid layout thrashing?**
Answer: Batch DOM reads and writes, avoid repeatedly querying layout properties, use `requestAnimationFrame`, minimize forced synchronous layouts.

90. **What is chaining of transformations and order?**
Answer: Transforms (`translate`, `rotate`, `scale`) are applied in order; order matters. Also transform origin matters.

91. **What is `position: sticky`?**
Answer: It behaves like `position: relative` until a threshold, then becomes `fixed` relative to its container.

92. **What is CSS subpixel rendering / fractional pixel layout?**
Answer: Browsers can position elements at sub-pixel values (e.g. 0.5px) for smoother layouts, especially at high DPI.

93. **What are progressive web features like `prefetch`, `preload`, `preconnect`?**
Answer: Resource hints to control loading, e.g. `<link rel="preload" href="style.css">`, `<link rel="prefetch">`, `<link rel="preconnect" href="https://fonts.googleapis.com">`

94. **What is the difference between `@import` and `<link>` in CSS? Which is better?**
Answer: `@import` inside CSS imports another stylesheet (blocks rendering until fetch) slower. `<link>` is preferred for performance.

95. **What is CSS critical rendering path?**
Answer: The sequence of steps (DOM, CSSOM, render tree, layout, paint) needed to render page. Optimizing critical path improves page load.

96. **What is CSS ordering of shorthand vs longhand properties?**
Answer: When using shorthand (e.g. `margin: 10px 5px;`), it resets all sub-properties. Order matters when combining with longhand.

97. **What are vendor prefixes (e.g. `-webkit-`, `-moz-`)? When are they needed?**
Answer: Prefixes used to support CSS features in browsers with experimental sup-

port. Use only when necessary.

98. **What is CSS paint worklet (Houdini)?**
Answer: CSS painting API (part of CSS Houdini) enables custom drawing of elements (via JS) integrated into CSS rendering.

99. **What is the difference between `rem`, `em`, `px`, `%` units?**
Answer:
`px`: fixed pixel unit.
`em`: relative to font-size of element (or parent).
`rem`: relative to root font-size (`html`).
`%`: relative to parent property (width, height, etc.).

100. **What is `ch` unit?**
Answer: `ch` is width of the 0 (zero) glyph in the elements font, useful for sizing relative to character widths.

Advanced HTML5 Topics

101. **What are new form input types introduced in HTML5?**
Answer: HTML5 introduced several input types to improve validation and UX: `email`, `url`, `number`, `range`, `date`, `time`, `color`, `tel`, `search`, `datetime-local`, etc.

102. **How does HTML5 improve form validation?**
Answer: HTML5 provides built-in validation attributes like `required`, `pattern`, `min`, `max`, `step`, `maxlength`, etc., removing the need for extra JavaScript for basic checks.

103. **What is the purpose of the `autocomplete` attribute?**
Answer: It controls whether browsers should auto-fill input fields. Example:
`<input type="email" autocomplete="off">`

104. **What are HTML5 APIs? Give examples.**
Answer: APIs introduced with HTML5 allow richer web apps:
Geolocation API
Local Storage / Session Storage (Web Storage API)
Canvas API
Web Workers
WebSockets
Drag and Drop API

105. **What is the difference between `localStorage`, `sessionStorage`, and cookies?**
Answer:

Feature	**localStorage**	**sessionStorage**	**Cookies**
Persistence	Until manually cleared	Until tab is closed	Set by expiry
Capacity	~10 MB	~5 MB	~4 KB
Accessible to server	5 No	5 No	1 Yes

106. **What is the `<canvas>` element used for?**

Answer: `<canvas>` provides a 2D drawing surface for graphics, charts, and animations via JavaScript (`getContext('2d')`).

107. **What is the `<svg>` element? How does it differ from `<canvas>`?**
Answer: SVG (Scalable Vector Graphics) defines vector shapes in XML syntax.
Difference:
SVG: resolution-independent, DOM-based, can be styled with CSS.
Canvas: pixel-based, does not scale well, redrawn via JS.

108. **What is the difference between `localStorage` and `IndexedDB`?**
Answer:
`localStorage` stores simple keyvalue pairs (strings).
`IndexedDB` is a full NoSQL database in the browser for complex data (objects, transactions, queries).

109. **What is the Drag and Drop API in HTML5?**
Answer: Allows elements to be draggable using attributes `draggable="true"` and JS events like `ondragstart`, `ondrop`, and `ondragover`.

110. **What is WebSocket in HTML5?**
Answer: A protocol for full-duplex communication between client and server over a single TCP connection useful for real-time applications (chats, live dashboards).

111. **What is the `<template>` tag in HTML5?**
Answer: Defines HTML that isnt rendered until used via JS. Example:
```
<template id="cardTemplate">
<div class="card">...</div>
</template>
```

112. **What is the `<dialog>` element?**
Answer: Introduced in HTML5 used for modal or non-modal dialogs. You can open it using `dialog.showModal()` in JavaScript.

113. **What is the `<details>` and `<summary>` tag?**
Answer: Used to create collapsible sections without JS:
```
<details>
<summary>More Info</summary>
<p>Hidden content</p>
</details>
```

114. **What is `<mark>` element used for?**
Answer: Highlights or marks text as relevant or important (like search results).

115. **What are microdata and `itemprop` attributes?**
Answer: They define structured data (schema.org) for SEO helps search engines understand content. Example:
```
<div itemscope itemtype="https://schema.org/Person">
<span itemprop="name">John Doe</span>
</div>
```

116. **What is the `<time>` element?**
Answer: Represents a specific time/date, optionally machine-readable:
```
<time datetime="2025-10-07">Oct 7, 2025</time>
```

117. **What is Web Storage quota per domain?**
Answer: Typically about 5 MB per domain (varies by browser).

118. **What is the `<main>` element for?**
Answer: Represents the main content unique to that page (only one `<main>` per page).

119. **What is `<output>` element in HTML5?**
Answer: Displays the result of a calculation or user action inside a form.
```
<output name="result"></output>
```

120. **What is the `<meter>` and `<progress>` element?**
Answer: Used to display scalar values or progress. Example:
```
<meter value="0.6">60%</meter>
<progress value="40" max="100">40%</progress>
```

CSS Intermediate to Advanced

121. **What is the difference between `transition` and `animation` in CSS?**
Answer:
`transition`: occurs between two states (hover, active, etc.).
`animation`: can have multiple keyframes and loop continuously.

122. **How to create a fade-in animation in CSS?**
Answer:
```
.fade-in {
animation:  fade 2s ease-in forwards;
}
@keyframes fade {
from { opacity:  0; }
to { opacity:  1; }
}
```

123. **What is the difference between hardware acceleration and software rendering in CSS?**
Answer: GPU-accelerated properties (`transform`, `opacity`) are faster; other properties like `top`, `left` may cause layout thrash and reflow.

124. **What are CSS sprites? Why are they used?**
Answer: A single image containing multiple icons. Used to reduce HTTP requests. You display specific parts using `background-position`.

125. **What are CSS `transitions` that cant be animated?**
Answer: Not all CSS properties are animatable e.g. `display`, `visibility`, `background-image`.

126. **What is reflow and repaint in CSS performance?**
Answer:
Reflow (layout): recalculates element positions and sizes.
Repaint: redraws elements visually. Reflow is more expensive.

127. **What is critical CSS and how to implement it?**
Answer: Inline only the CSS required for above-the-fold content to render faster,

defer non-critical CSS for later.

128. **How do you optimize CSS for performance?**
Answer:
Minify and combine files
Use `will-change` carefully
Limit complex selectors
Avoid inline styles
Use `content-visibility: auto;`

129. **What is the difference between `overflow: hidden`, `scroll`, and `auto`?**
Answer:
`hidden`: cuts off overflowing content
`scroll`: always shows scrollbars
`auto`: shows scrollbars only when needed

130. **What is `clip-path`?**
Answer: Used to clip part of an element visually. Example:
`clip-path: circle(50% at 50% 50%);`

131. **What is `aspect-ratio` in CSS?**
Answer: Controls width-to-height ratio. Example:
`img { aspect-ratio: 16/9; }`

132. **What is the difference between `min-width` and `max-width` media queries?**
Answer:
`min-width`: styles apply above the specified width (mobile-first).
`max-width`: styles apply below the width (desktop-first).

133. **What are container queries?**
Answer: A new CSS feature that allows styling elements based on the size of their container rather than viewport.

134. **What is the difference between `absolute` and `fixed` positioning?**
Answer:
`absolute`: positioned relative to nearest positioned ancestor.
`fixed`: positioned relative to the viewport.

135. **What is stacking context in CSS?**
Answer: An element that creates its own layer in the stacking order (e.g. via `position`, `opacity < 1`, `z-index`).

136. **What is `mix-blend-mode`?**
Answer: Determines how an elements content blends with the background e.g. `multiply`, `screen`, `overlay`.

137. **What is CSS `isolation` and why use `isolation: isolate`?**
Answer: Prevents blending/stacking contexts from affecting siblings. Helps maintain predictable rendering.

138. **What is the `content-visibility` property?**
Answer: Defers rendering of off-screen content until it becomes visible, improving performance. Example:

```
.card { content-visibility:  auto; }
```

139. **What are CSS logical properties useful for?**
Answer: They make layouts direction-agnostic for internationalization (e.g. `margin-inline-star` works for both LTR/RTL).

140. **What is the difference between `grid-template-areas` and `grid-template-columns/rows`?**
Answer:
`grid-template-areas`: define named areas for layout visually.
`grid-template-columns/rows`: define explicit sizing.

141. **What is CSS subgrid?**
Answer: Allows nested grid items to inherit and align with parent grids rows/columns.

142. **What is the difference between `inline-flex` and `flex`?**
Answer:
`display: flex`: creates block-level flex container.
`display: inline-flex`: creates inline-level flex container.

143. **What is `gap` property in Flexbox and Grid?**
Answer: Defines consistent spacing between items without using margins. Works in both flex and grid layouts.

144. **What is the difference between `translate`, `position`, and `margin` for movement?**
Answer:
`margin`: reflows layout.
`position` (`top/left`): triggers reflow.
`transform: translate`: does not reflow (GPU-accelerated).

145. **What is the shadow DOM in HTML/CSS?**
Answer: Part of Web Components encapsulates DOM and CSS so styles dont leak in or out.

146. **What is the difference between `:host` and `:host-context` selectors in shadow DOM?**
Answer: `::host` styles the shadow host element itself; `:host-context()` applies styles when host matches a context outside.

147. **What are CSS custom media queries?**
Answer: They let you define named media queries for reuse.
```
@custom-media -small (max-width:  600px);
@media (-small) { ...  }
```

148. **What is the difference between `inline`, `inline-block`, and `block` display?**
Answer:
`inline`: cannot set width/height.
`inline-block`: inline element that respects width/height.
`block`: starts on a new line, takes full width.

149. **What is the difference between `nth-child` and `nth-of-type`?**
Answer:
`nth-child`: selects based on overall child index.

`nth-of-type`: selects based on type (tag name).

150. **What are CSS layers (`@layer`)?**
Answer: CSS Layers control the cascade order across stylesheets. Example:
```
@layer reset, base, theme;
@layer reset { * { margin:  0; } }
@layer theme { button { color:  blue; } }
```
They ensure predictable priority regardless of load order.