120 CSS & CSS3 Interview Questions with Answers

1. What is CSS and what are its benefits?

Answer: CSS (Cascading Style Sheets) is a styling language used to describe the presentation of HTML documents. Benefits include separation of content from presentation, consistent styling across multiple pages, improved page loading times, and responsive design capabilities. **Frequency:** Very High (90%) **Companies:** All tech companies including Google, Microsoft, Amazon, Facebook

2. Explain the difference between classes and IDs in CSS.

Answer: Classes can be used multiple times on a page and are defined with a period (.class-name). IDs must be unique to a single element on a page and are defined with a hash symbol (#id-name). IDs have higher specificity than classes. **Frequency:** Very High (85%) **Companies:** Google, Amazon, Facebook, Twitter

3. What is the box model in CSS?

Answer: The CSS box model describes the rectangular boxes generated for elements in the document tree. The standard box model calculates width and height based on the content only, while the alternative box-sizing model (border-box) includes padding and border in the width/height calculation. The CSS Box Model is a fundamental concept in web design and layout. Every HTML element is considered a box, which consists of the following components (from innermost to outermost):

Content: The actual content (text, image, etc.) Padding: Space between the content and the border Border: The edge surrounding the padding (and content) Margin: Space between the element's border and surrounding elements

BOX MODEL

+---------------------------------+ | Margin | | +-------------------------+ | | | Border | | | | +------------------+ | | | | | Padding | | | | | | +------------+ | | | | | | | Content | | | | | | | | +------------+ | | | | | | +------------------+ | | | | +-------------------------+ | | +---------------------------------+

🧠 Explanation with Example:

```
<div class="box">Hello</div>
```

```
.box {
width: 200px;
padding: 10px;
border: 5px solid black;
margin: 20px;
}
```

Total width = 200 (width) + 10*2 (padding) + 5*2 (border) = 230px

Total height = Same calculation applies for height (if specified similarly).

Use box-sizing: border-box to include padding and border within the width/height.

```
* {
    box-sizing: border-box;
}
```

Frequency: Very High (90%) **Companies:** Microsoft, Amazon, Apple, Adobe

4. What is the difference between inline, block, and inline-block display values?

Answer: - Block elements start on a new line and take full width available - Inline elements don't start on a new line and only take up as much width as necessary - Inline-block elements are placed inline but behave like block elements regarding width, height, and margin properties **Frequency:** High (80%) **Companies:** Facebook, Uber, Airbnb, Twitter

5. Explain CSS specificity and how it works.

Answer: Specificity is how browsers decide which CSS property values are most relevant to an element. Specificity hierarchy (highest to lowest): Inline styles, IDs, Classes/attributes/pseudo-classes, Elements/pseudo-elements. When specificity is equal, the last rule defined wins. **Frequency:** High (75%) **Companies:** Google, Microsoft, Spotify, LinkedIn

6. What is the difference between "px", "em", "rem", "vh", and "vw" units?

Answer: - px: Fixed-size unit - em: Relative to the font-size of the parent element - rem: Relative to the font-size of the root element - vh: Relative to 1% of the viewport height - vw: Relative to 1% of the viewport width **Frequency:** High (70%) **Companies:** Facebook, Apple, Netflix, Salesforce

7. Explain CSS positioning (static, relative, absolute, fixed, sticky).

Answer: - static: Default positioning; not affected by top/right/bottom/left properties - relative: Positioned relative to its normal position - absolute: Positioned relative to the nearest positioned ancestor - fixed: Positioned relative to the viewport - sticky: Positioned based on the user's scroll position (hybrid of relative and fixed) **Frequency:** Very High (85%) **Companies:** Amazon, Microsoft, Uber, Shopify

8. What are pseudo-classes and pseudo-elements?

Answer: - Pseudo-classes select elements based on a state or relation (:hover, :first-child) - Pseudo-elements target specific parts of an element (::before, ::after, ::first-line) **Frequency:** High (75%) **Companies:** LinkedIn, Twitter, Dropbox, Airbnb

9. What is the difference between padding and margin?

Answer: Padding is the space between an element's content and its border. Margin is the space outside the border, between the element and other elements. Padding has a background color (the element's background), while margins are transparent. **Frequency:** High (80%) **Companies:** Google, Facebook, Adobe, PayPal

10. What is a CSS preprocessor and what are its benefits?

Answer: CSS preprocessors (e.g., SASS, LESS) extend CSS with variables, nesting, mixins, functions, and more. Benefits include cleaner code organization, reusability, modularity, and mathematical operations. **Frequency:** Medium (65%) **Companies:** Twitter, Airbnb, Slack, Shopify

11. What is CSS Flexbox?

Answer: Flexbox is a one-dimensional layout model designed for arranging items in rows or columns. It provides an efficient way to distribute space and align items even when their sizes are unknown or dynamic. **Frequency:** Very High (85%) **Companies:** Facebook, Amazon, Netflix, Google

12. What is CSS Grid?

Answer: CSS Grid is a two-dimensional layout system for the web, allowing developers to create complex grid-based layouts with rows and columns. Unlike Flexbox, it works in both dimensions simultaneously. **Frequency:** High (70%) **Companies:** Microsoft, Twitter, Adobe, Shopify

13. Explain the "z-index" property.

Answer: The z-index property specifies the stack order of positioned elements. Elements with higher z-index values are placed in front of elements with lower values. It only works on positioned elements (not static). **Frequency:** High (75%) **Companies:** Google, Uber, Airbnb, Pinterest

14. What are media queries and how are they used?

Answer: Media queries allow CSS to apply different styles based on characteristics of the device (screen size, resolution, orientation). They're fundamental for responsive design, enabling different layouts for different devices. **Frequency:** Very High (85%) **Companies:** Microsoft, Facebook, Twitter, Netflix

15. What is the "!important" rule and when should it be used?

Answer: The !important rule increases a declaration's specificity, overriding normal rules of the cascade. It should be used sparingly, typically for utility classes or to override third-party CSS that cannot be modified directly. **Frequency:** Medium (60%) **Companies:** Amazon, Adobe, Spotify, PayPal

16. How do you center an element horizontally and vertically in CSS?

Answer: Multiple methods exist, but the most modern approach uses Flexbox:

```
css .parent { display: flex; justify-content: center; /* Horizontal center */ align-items: center; /* Vertical center */ height: 1
```

Frequency: Very High (85%) **Companies:** Facebook, Google, Uber, LinkedIn

17. Explain the difference between visibility:hidden and display:none.

Answer: - visibility:hidden hides the element but keeps its space in the layout - display:none removes the element completely from the document flow, taking up no space **Frequency:** High (70%) **Companies:** Microsoft, Amazon, Adobe, PayPal

18. What is a CSS sprite and why would you use it?

Answer: A CSS sprite combines multiple images into one larger image, using background-position to display specific parts. Benefits include fewer HTTP requests, faster page loading, and reduced bandwidth usage. **Frequency:** Medium (50%) **Companies:** Google, Facebook, Pinterest, Shopify

19. What are the different ways to apply CSS to HTML?

Answer: 1. External stylesheet (link tag) 2. Internal stylesheet (style tag in document head) 3. Inline styles (style attribute on HTML elements) **Frequency:** High (70%) **Companies:** Microsoft, Twitter, Airbnb, Netflix

20. Explain the concept of CSS inheritance.

Answer: Inheritance is the mechanism by which certain CSS properties (typically those related to text) are passed from parent to child elements. Not all properties are inherited (e.g., margin, padding, border aren't). **Frequency:** Medium (60%) **Companies:** Google, Adobe, Spotify, PayPal

21. What is the purpose of the "float" property?

Answer: The float property places an element to the left or right of its container, allowing text and inline elements to wrap around it. Originally designed for image placement in text, it was previously used for layouts but has largely been replaced by Flexbox and Grid. **Frequency:** Medium (55%) **Companies:** Amazon, Facebook, LinkedIn, Adobe

22. What is the "clearfix" hack?

Answer: The clearfix hack is a technique used to ensure that a container element properly contains its floated children. Modern version: `css .clearfix::after { content: ""; display: table; clear: both; }` **Frequency:** Low (40%) **Companies:** Facebook, Twitter, PayPal, Shopify

23. Explain CSS transitions.

Answer: CSS transitions allow property changes to occur smoothly over a specified duration. They require: a property to change, a duration, a timing function (optional), and a delay (optional). **Frequency:** High (70%) **Companies:** Google, Netflix, Airbnb, Spotify

24. What are CSS animations and how do they differ from transitions?

Answer: CSS animations use @keyframes to define multiple states during an animation sequence. Unlike transitions (which move from state A to B only), animations can have multiple intermediate states and can loop indefinitely. **Frequency:** Medium (65%) **Companies:** Apple, Amazon, Pinterest, Uber

25. What is the difference between RGB, RGBA, HEX, and HSL color values?

Answer: - RGB: Red, Green, Blue values (rgb(255, 0, 0)) - RGBA: RGB with Alpha transparency (rgba(255, 0, 0, 0.5)) - HEX: Hexadecimal color notation (#FF0000) - HSL: Hue, Saturation, Lightness (hsl(0, 100%, 50%)) **Frequency:** Medium (60%) **Companies:** Adobe, Apple, Pinterest, Twitter

26. What are CSS variables (custom properties)?

Answer: CSS variables are custom properties defined with -- prefix that store specific values to be reused throughout a document. They can be changed in media queries or with JavaScript, enabling dynamic styling. `css :root { --main-color: #0077cc; } .element { color: var(--main-color); }` **Frequency:** High (70%) **Companies:** Google, Microsoft, Netflix, Shopify

27. What is the difference between em and rem units?

Answer: - em is relative to the font-size of its direct or nearest parent - rem is relative to the root (html) element's font-size This makes rem more predictable for responsive design. **Frequency:** High (75%) **Companies:** Facebook, Amazon, Adobe, LinkedIn

28. How does CSS cascade work?

Answer: The cascade determines which CSS rules apply when multiple rules target the same element. It's resolved through: 1. Importance (!important) 2. Specificity 3. Source order (last defined wins) **Frequency:** High (70%) **Companies:** Microsoft, Google, Twitter, Spotify

29. What is a CSS reset and why would you use one?

Answer: A CSS reset removes or normalizes browser default styles to provide a consistent baseline across different browsers. Popular examples include Normalize.css and Reset CSS. It helps avoid cross-browser inconsistencies. **Frequency:** Medium (60%) **Companies:** Facebook, Airbnb, PayPal, Adobe

30. Explain CSS vendor prefixes and why they are used.

Answer: Vendor prefixes (-webkit-, -moz-, -ms-, -o-) allow browsers to implement experimental or non-standard CSS features. They're used to ensure compatibility during the standardization process of new CSS features. **Frequency:** Medium (55%) **Companies:** Google, Apple, Shopify, Pinterest

31. What is responsive design?

Answer: Responsive design is an approach to web design that makes web pages render well on various devices and window/screen sizes. Key components include fluid grids, flexible images, and media queries. **Frequency:** Very High (85%) **Companies:** Microsoft, Amazon, Netflix, Uber

32. What are CSS frameworks and what are their advantages/disadvantages?

Answer: CSS frameworks (like Bootstrap, Tailwind, Foundation) provide pre-written CSS for common components and layouts. Advantages: Speed development, cross-browser compatibility, responsive design out of the box Disadvantages: Bloated code, similar look across sites, learning curve **Frequency:** Medium (65%) **Companies:** Twitter, Airbnb, LinkedIn, PayPal

33. Describe the difference between adaptive and responsive design.

Answer: - Responsive design uses fluid grids and flexible images that continuously adapt to any screen size - Adaptive design uses distinct layouts for specific screen sizes with no fluidity between them **Frequency:** Medium (55%) **Companies:** Google, Microsoft, Shopify, Adobe

34. What is mobile-first design?

Answer: Mobile-first design starts the design process from the mobile end of the responsive spectrum. CSS is written for mobile devices first, then enhanced with media queries for larger screens. It focuses on core content and functionality. **Frequency:** High (70%) **Companies:** Facebook, Twitter, Netflix, Pinterest

35. What are CSS combinators?

Answer: Combinators explain the relationship between selectors: - Descendant (space): Matches all descendants - Child (>): Matches direct children only - Adjacent sibling (+): Matches the adjacent sibling - General sibling (~): Matches all following siblings **Frequency:** Medium (60%) **Companies:** Amazon, Apple, Adobe, LinkedIn

36. Explain the concept of CSS Specificity Hierarchy.

Answer: From highest to lowest: 1. Inline styles (1000) 2. IDs (100) 3. Classes, attributes, pseudo-classes (10) 4. Elements, pseudo-elements (1) A more specific selector overrides less specific ones. **Frequency:** High (70%) **Companies:** Google, Microsoft, Spotify, Uber

37. What is the purpose of the @import rule in CSS?

Answer: The @import rule allows importing one CSS file into another. It must appear before any other CSS rule except @charset. While convenient, it can impact performance as it blocks parallel downloads. **Frequency:** Low (40%) **Companies:** Facebook, Adobe, PayPal, Shopify

38. How do you implement a CSS triangle?

Answer: CSS triangles are made using border properties with transparent borders:

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

Frequency: Medium (50%) **Companies:** Google, Pinterest, Airbnb, Twitter

39. What is a stacking context in CSS?

Answer: A stacking context is a three-dimensional conceptualization of HTML elements along an imaginary z-axis. It's formed by elements with certain properties (z-index when position is not static, opacity < 1, transform, filter, etc.). Z-index only works within the same stacking context. **Frequency:** Medium (55%) **Companies:** Microsoft, Amazon, Netflix, Adobe

40. What is the "currentColor" keyword in CSS?

Answer: currentColor represents the current value of the color property of an element. It's useful for making borders, shadows, or backgrounds match the text color without repeating color values. **Frequency:** Low (35%) **Companies:** Apple, Spotify, Twitter, Shopify

41. What is the "initial" value in CSS?

Answer: The initial keyword sets a CSS property to its default value as defined in the CSS specification, regardless of inherited values or user agent stylesheet.

Frequency: Low (40%) **Companies:** Google, LinkedIn, PayPal, Uber

42. What is the "inherit" keyword in CSS?

Answer: The inherit keyword forces a property to inherit its value from its parent element, even for properties that don't inherit by default (like width or border). **Frequency:** Medium (50%) **Companies:** Facebook, Adobe, Pinterest, Airbnb

43. What is CSS specificity specificity and how is it calculated?

Answer: CSS specificity is calculated as: - Inline styles: 1,0,0,0 - IDs: 0,1,0,0 - Classes, attributes, pseudo-classes: 0,0,1,0 - Elements, pseudo-elements: 0,0,0,1 Higher numbers win. If equal, the last rule declared wins. **Frequency:** High (75%) **Companies:** Microsoft, Amazon, Twitter, Netflix

44. Explain the "calc()" function in CSS.

Answer: calc() allows mathematical expressions with different units. For example, calc(100% - 20px) combines percentages and pixels. Useful for responsive layouts that need precise adjustments. **Frequency:** Medium (60%) **Companies:** Google, Apple, Spotify, PayPal

45. What are CSS counters?

Answer: CSS counters are variables maintained by CSS to track how many times they're used. They work with counter-reset, counter-increment, and content properties to create automatic numbering. **Frequency:** Low (30%) **Companies:** Adobe, LinkedIn, Shopify, Pinterest

46. What is the purpose of normalize.css?

Answer: Normalize.css is a modern alternative to CSS resets that preserves useful browser defaults while normalizing styles across browsers. Rather than eliminating all styles, it makes elements render consistently while matching modern standards. **Frequency:** Medium (55%) **Companies:** Facebook, Twitter, Airbnb, Netflix

47. What is the scroll-snap property and how is it used?

Answer: CSS Scroll Snap provides a way to create smooth, controlled scroll experiences by defining "snap points" as users scroll through content. The main properties are:

scroll-snap-type: Sets the axis and strictness of snapping scroll-snap-align: Determines where elements snap within the container scroll-snap-stop: Controls whether the scroll can skip past certain elements

Example: `css .container { scroll-snap-type: x mandatory; overflow-x: scroll; } .item { scroll-snap-align: center; }` **Frequency:** Medium (45%) **Companies:** Apple, Google, Pinterest, Airbnb

48. Explain CSS custom properties (variables).

Answer: CSS variables use the --name syntax and var() function to store and reuse values. They can be changed dynamically with JavaScript or in different contexts like media queries. They follow the cascade and can be scoped to selectors. **Frequency:** High (70%) **Companies:** Google, Apple, Spotify, PayPal

49. What is the "all" property in CSS?

Answer: The "all" property resets all properties to their initial, inherited, or unset state (except unicode-bidi and direction). It's useful for creating isolated styling contexts. **Frequency:** Low (25%) **Companies:** Facebook, LinkedIn, Shopify, Pinterest

50. What is the "unset" value in CSS?

Answer: The "unset" keyword resets a property to its inherited value if it inherits naturally, or to its initial value if it doesn't. It's a combination of "inherit" and "initial". **Frequency:** Low (30%) **Companies:** Microsoft, Adobe, Twitter, Airbnb

51. What are attribute selectors in CSS?

Answer: Attribute selectors target elements based on their attributes and values: - [attr]: Elements with the attribute - [attr="value"]: Exact match - [attr^="value"]: Starts with - [attr$="value"]: Ends with - [attr*="value"]: Contains **Frequency:** Medium (60%) **Companies:** Google, Amazon, Netflix, PayPal

52. Explain CSS text-overflow property.

Answer: text-overflow controls how overflowed content is signaled to users. Values include: - clip: Default, cuts off text - ellipsis: Shows "..." for overflow - string: Custom string (limited support) Requires overflow: hidden and white-space: nowrap to work. **Frequency:** Medium (55%) **Companies:** Apple, Spotify, Pinterest,

LinkedIn

53. What is the difference between nth-child and nth-of-type?

Answer: - :nth-child(n) selects elements that are the nth child of their parent, regardless of type - :nth-of-type(n) selects elements that are the nth instance of their specific type among siblings **Frequency:** Medium (55%) **Companies:** Facebook, Adobe, Uber, Twitter

54. What is the "content" property used for?

Answer: The content property is used with ::before and ::after pseudo-elements to insert generated content. It can contain text, images, quotes, counters, or nothing (with empty string ""). **Frequency:** Medium (60%) **Companies:** Microsoft, Airbnb, Shopify, PayPal

55. How do you create a multi-column layout in CSS?

Answer: CSS multi-column layout is created using:
`css .container { column-count: 3; /* or column-width: 200px */ column-gap: 20px; column-rule: 1px solid #ccc; }`
Frequency: Medium (50%) **Companies:** Google, Netflix, Pinterest, Adobe

56. What are CSS Shapes and how are they used?

Answer: CSS Shapes allow content to flow around custom shapes defined with shape-outside property. Shapes can be basic (circle, ellipse, polygon) or defined by reference to images with alpha channels. **Frequency:** Low (30%) **Companies:** Apple, Facebook, Adobe, LinkedIn

57. What is CSS containment?

Answer: CSS containment (contain property) improves performance by isolating parts of the page from the rest of the document, preventing certain elements from affecting others. Values include: size, layout, style, paint, or combinations. **Frequency:** Low (25%) **Companies:** Google, Microsoft, Twitter, Netflix

58. Explain CSS custom cursors.

Answer: The cursor property can set custom cursors using the url() function:
`css .custom-cursor { cursor: url('path/to/cursor.png'), auto; }` The second value is a fallback if the custom cursor fails to load. **Frequency:** Low (30%) **Companies:** Amazon, Spotify, Pinterest, Uber

59. What are CSS filters?

Answer: CSS filters apply visual effects to elements using the filter property. Examples include blur, brightness, contrast, grayscale, hue-rotate, invert, opacity, saturate, sepia, and drop-shadow. They're often used for image effects. **Frequency:** Medium (55%) **Companies:** Adobe, Facebook, Apple, Airbnb

60. What is the difference between "em" and "strong" tags?

Answer: While technically HTML tags, their styling is important: - em adds emphasis (usually italic) - strong indicates importance (usually bold) Both have semantic meaning beyond their default styles. **Frequency:** Low (40%) **Companies:** Microsoft, Google, Twitter, Shopify

61. What is a CSS pre-processor?

Answer: CSS pre-processors extend CSS with programming features like variables, nesting, mixins, functions, and operations. Popular pre-processors include Sass, LESS, and Stylus. They compile into standard CSS for browsers. **Frequency:** High (70%) **Companies:** LinkedIn, Netflix, PayPal, Adobe

62. What is the CSS "object-fit" property?

Answer: object-fit specifies how a replaced element's content (like an image or video) should be resized to fit its container. Values include: fill, contain, cover, none, scale-down. **Frequency:** Medium (60%) **Companies:** Pinterest, Amazon, Spotify, Airbnb

63. What is the difference between absolute and fixed positioning?

Answer: - absolute positioning is relative to the nearest positioned ancestor - fixed positioning is relative to the viewport and stays in place during scrolling **Frequency:** High (75%) **Companies:** Google, Microsoft, Uber, Facebook

64. How do you implement a pure CSS dropdown menu?

Answer: Using the :hover pseudo-class with nested elements:

```
css .dropdown { position: relative; } .dropdown-content { display: none; position: absolute; } .dropdown:hover .dropdown-content {
```
Frequency: Medium (60%) **Companies:** Apple, Twitter, Netflix, PayPal

65. What is CSS masking?

Answer: CSS masking partially or fully hides portions of an element using another image/element as a mask. The mask-image property defines the masking layer which determines the alpha channel (transparency). **Frequency:** Low (35%) **Companies:** Adobe, Pinterest, LinkedIn, Shopify

66. What is CSS clip-path?

Answer: clip-path creates a clipping region that determines which parts of an element are visible. Shapes can be basic (circle, ellipse, polygon) or complex SVG paths. Unlike masking, it's binary - pixels are either visible or invisible. **Frequency:** Medium (50%) **Companies:** Facebook, Apple, Airbnb, Spotify

67. What is CSS Grid auto-placement algorithm?

Answer: The Grid auto-placement algorithm determines how grid items are automatically placed into the grid when they're not explicitly positioned. It considers the grid-auto-flow property (row, column, or dense) to optimize the placement. **Frequency:** Medium (45%) **Companies:** Microsoft, Google, Adobe, Uber

68. Explain the difference between "position: relative" and "position: absolute".

Answer: - relative positioning offsets an element from its normal position, keeping its space in the document flow - absolute positioning removes the element from normal flow and positions it relative to its nearest positioned ancestor **Frequency:** High (80%) **Companies:** Amazon, Twitter, Netflix, PayPal

69. What are CSS Houdini APIs?

Answer: CSS Houdini is a set of APIs that expose parts of the CSS engine, giving developers direct access to the CSS Object Model. This allows creation of custom CSS features without waiting for browser implementation. **Frequency:** Low (20%) **Companies:** Google, Facebook, Adobe, Microsoft

70. What is the difference between :root and html selector?

Answer: - :root targets the highest-level parent (usually in HTML documents) - html targets only the HTML element :root has higher specificity as it's a pseudo-class. In XML documents, :root might target a different element than html. **Frequency:** Low (35%) **Companies:** LinkedIn, Spotify, PayPal, Pinterest

71. Explain the mix-blend-mode property.

Answer: mix-blend-mode determines how an element's content blends with its background and the content of its parent. Values include multiply, screen, overlay, darken, lighten, etc. Similar to layer blending in graphic design software. **Frequency:** Low (30%) **Companies:** Adobe, Apple, Twitter, Airbnb

72. What is willchange property in CSS?

Answer: will-change hints browsers about elements that will animate or change, allowing optimization. This can improve performance by preparing for changes, but should be used sparingly to avoid excessive resource consumption. **Frequency:** Low (25%) **Companies:** Google, Netflix, Uber, Facebook

73. What is CSS feature detection?

Answer: CSS feature detection determines if a browser supports specific CSS features. The @supports rule enables conditional application of styles based on feature support: `css @supports (display: grid) { /* Grid-specific styles */ }` **Frequency:** Medium (45%) **Companies:** Microsoft, Amazon, Adobe, Shopify

74. What are CSS custom scrollbars?

Answer: Custom scrollbars are created using the scrollbar-* properties or the older ::-webkit-scrollbar pseudo-element:
```
css /* Modern (Firefox) */ .custom-scroll { scrollbar-width: thin; scrollbar-color: slateblue #f1f1f1; } /* WebKit browsers */ .cus
```
Frequency: Medium (50%) **Companies:** Apple, Pinterest, LinkedIn, PayPal

75. What is CSS content-visibility?

Answer: content-visibility improves page load performance by skipping rendering for off-screen content. The 'auto' value tells browsers they can delay rendering offscreen content until needed. **Frequency:** Low (30%) **Companies:** Google, Facebook, Twitter, Netflix

76. What is a CSS ruleset?

Answer: A CSS ruleset consists of a selector and a declaration block: `css selector { property1: value1; property2: value2; }` The selector targets HTML elements; the declaration block contains property-value pairs. **Frequency:** Medium (60%) **Companies:** Microsoft, Adobe, Spotify, Airbnb

77. What is the CSS aspect-ratio property?

Answer: aspect-ratio specifies a preferred aspect ratio for the box, which will be used to calculate auto sizes and other layout functions: `css .element { aspect-ratio: 16 / 9; }` **Frequency:** Medium (45%) **Companies:** Amazon, Apple, Pinterest, Uber

78. What is the CSS overscroll-behavior property?

Answer: overscroll-behavior controls what happens when a user scrolls beyond the boundaries of an element. Values include: - auto: Default, allows scroll chaining - contain: Prevents scroll chaining but maintains bounce effects - none: Prevents both chaining and overscroll effects **Frequency:** Low (30%) **Companies:** Google, Facebook, LinkedIn, PayPal

79. How do you handle browser-specific styling issues?

Answer: Techniques include: - Feature detection with @supports - Vendor prefixes (-webkit-, -moz-, etc.) - Reset or normalize CSS - Polyfills for missing features - Browser-specific hacks (less recommended) **Frequency:** Medium (60%) **Companies:** Microsoft, Twitter, Netflix, Shopify

80. What is the purpose of "white-space" property?

Answer: white-space controls how whitespace within an element is handled. Values include: - normal: Collapses whitespace, wraps text - nowrap: Collapses whitespace, prevents wrapping - pre: Preserves whitespace, no wrapping - pre-wrap: Preserves whitespace, wraps text - pre-line: Collapses whitespace but preserves line breaks **Frequency:** Medium (55%) **Companies:** Adobe, Apple, PayPal, Airbnb

81. What are logical properties in CSS?

Answer: Logical properties are writing-mode-relative alternatives to physical properties. Examples include: - margin-inline vs margin-left/right - padding-block vs padding-top/bottom - inset-inline-start vs left They automatically adjust based on writing direction. **Frequency:** Medium (45%) **Companies:** Google, Amazon, Spotify, Pinterest

82. Explain CSS element states (:hover, :active, :focus, etc.).

Answer: CSS provides pseudo-classes for various element states: - :hover: When mouse is over the element - :active: When element is being activated (clicked) - :focus: When element has keyboard focus - :visited: For visited links - :focus-within: When element or descendant has focus **Frequency:** High (75%) **Companies:** Facebook, Microsoft, Uber, Twitter

83. What is a CSS reset?

Answer: A CSS reset is a stylesheet that removes or normalizes default browser styling to provide a consistent baseline across browsers. Popular examples include Eric Meyer's Reset CSS and Normalize.css. **Frequency:** Medium (60%) **Companies:** Netflix, LinkedIn, Adobe, PayPal

84. What is the difference between block-level and inline elements?

Answer: - Block-level elements start on new lines and take full width available (div, p, h1-h6) - Inline elements don't start new lines and only take necessary width (span, a, img) Block elements can contain other blocks and inlines; inline elements generally contain only data and other inline elements. **Frequency:** High (75%) **Companies:** Apple, Google, Shopify, Pinterest

85. What is viewport units (vh, vw) and how do they differ from percentages?

Answer: - vh/vw are relative to viewport size (1vh = 1% of viewport height) - Percentages are relative to parent container size Viewport units are useful for full-screen layouts and are unaffected by scrolling. **Frequency:** High (70%) **Companies:** Microsoft, Amazon, Airbnb, Spotify

86. How do CSS Grid and Flexbox complement each other?

Answer: CSS Grid excels at two-dimensional layouts (rows and columns simultaneously), while Flexbox is optimal for one-dimensional layouts (either row OR column). They can be used together - Grid for page-level layout and Flexbox for component-level design. **Frequency:** High (65%) **Companies:** Facebook, Twitter, Netflix, Uber

87. What is the difference between "grid-template" and "grid" shorthand properties?

Answer: - grid-template is a shorthand for grid-template-rows, grid-template-columns, and grid-template-areas - grid is a complete shorthand that also includes grid-auto-rows, grid-auto-columns, and grid-auto-flow **Frequency:** Medium (45%) **Companies:** Google, Adobe, LinkedIn, PayPal

88. What are CSS transforms?

Answer: CSS transforms allow elements to be visually transformed in 2D or 3D space. Functions include: - translate(): Move elements - rotate(): Rotate elements - scale(): Resize elements - skew(): Distort elements **Frequency:** High (70%) **Companies:** Apple, Microsoft, Pinterest, AirBnB

89. What is the CSS backface-visibility property?

Answer: backface-visibility determines if the back face of an element is visible when facing the user. It's commonly used with 3D transforms. Values are visible (default) or hidden. **Frequency:** Low (30%) **Companies:** Facebook, Netflix, Adobe, Shopify

90. Explain CSS reflow and repaint.

Answer: - Reflow (layout): Recalculating positions and dimensions of elements - Repaint: Updating the visual presentation without changing layout Reflows are more expensive performance-wise. Properties like transform and opacity can avoid reflows. **Frequency:** Medium (50%) **Companies:** Google, Amazon, Uber, PayPal

91. What is the difference between "em", "ex", "ch", and "rem" units?

Answer: - em: Relative to font-size of the parent - ex: Relative to x-height of the font (roughly height of 'x') - ch: Relative to width of the "0" character - rem: Relative to font-size of the root element **Frequency:** Medium (50%) **Companies:** Microsoft, Twitter, LinkedIn, Pinterest

92. How do you implement vertical text with CSS?

Answer: Vertical text can be implemented using:
`css .vertical-text { writing-mode: vertical-rl; /* or vertical-lr */ text-orientation: upright; /* for upright characters */ }`
Frequency: Low (25%) **Companies:** Apple, Adobe, Spotify, Airbnb

93. What are CSS counters and how do they work?

Answer: CSS counters are variables maintained by CSS whose values can be incremented by CSS rules to track how many times they're used. Implemented with counter-reset, counter-increment, and counter() function in content property. **Frequency:** Low (35%) **Companies:** Google, Facebook, PayPal, Shopify

94. Explain the CSS @supports rule.

Answer: @supports (feature detection) allows applying styles conditionally based on browser support for CSS features:
`css @supports (display: grid) { .container { display: grid; } }` **Frequency:** Medium (45%) **Companies:** Microsoft, Netflix, Adobe, LinkedIn

95. What is CSS pointer-events property?

Answer: pointer-events controls how an element responds to mouse/touch events. Setting it to "none" makes the element ignore pointer events, passing them to elements behind it. Useful for overlays or temporarily disabling interactions. **Frequency:** Medium (50%) **Companies:** Amazon, Twitter, Pinterest, Uber

96. What are CSS Subgrids?

Answer: Subgrid allows nested grids to inherit track sizes from their parent grid, enabling content alignment across nested grid structures. It's part of CSS Grid Level 2 specification. **Frequency:** Low (30%) **Companies:** Apple, Google, Adobe, Airbnb

97. What is CSS contain property?

Answer: The contain property improves performance by isolating an element's content from the rest of the document. Values include: - layout: Element is independent for layout purposes - paint: Element's descendants don't display outside its bounds - size: Element's size doesn't depend on its children **Frequency:** Low (25%) **Companies:** Facebook, Microsoft, Spotify, PayPal

98. What are CSS Container Queries?

Answer: Container queries allow styling elements based on the size of their container rather than the viewport. This enables truly responsive components regardless of where they're placed in the layout.
`css @container (min-width: 400px) { .card { display: grid; grid-template-columns: 1fr 2fr; } }` **Frequency:** Medium (40%)
Companies: Google, Netflix, Twitter, Pinterest

99. What is the CSS "gap" property?

Answer: The gap property (and its row-gap/column-gap components) defines spacing between grid/flex items without using margins. Originally for Grid layouts but now works with Flexbox too. **Frequency:** Medium (55%) **Companies:** Amazon, Adobe, LinkedIn, Shopify

100. How would you implement a CSS-only accordion?

Answer: A CSS-only accordion can be implemented using the :target pseudo-class or with hidden checkboxes and labels:
`css /* Using checkbox method */ .accordion input[type="checkbox"] { display: none; } .accordion .content { max-height: 0; overflow:`
Frequency: Medium (55%) **Companies:** Microsoft, Apple, Uber, Airbnb

101. Explain BEM (Block Element Modifier) methodology and when you would use it.

Answer: BEM (Block Element Modifier) is a CSS naming convention that creates reusable components through a clear structure. It uses the format `.block__element--modifier`. Blocks are standalone components, elements are parts of blocks (identified with double underscore), and modifiers are variations or states (identified with double dash). BEM is particularly useful for large projects with multiple developers as it creates a clear, strict naming structure that avoids cascade conflicts and improves code readability and maintainability. **Frequency:** High (70%) **Companies:** Airbnb, Shopify, BBC, Yandex, Uber

102. Compare and contrast CSS methodologies: BEM vs SMACSS vs OOCSS.

Answer: - BEM (Block Element Modifier): Uses strict naming conventions with block__element--modifier pattern. Strengths: clear component relationships, good for component-based design. Weakness: long class names. - SMACSS (Scalable and Modular Architecture for CSS): Categorizes CSS rules into Base, Layout, Module, State, and Theme. Strengths: organized structure, good for large projects. Weakness: more abstract, steeper learning curve. - OOCSS (Object Oriented CSS): Focuses on separating structure from skin and container from content. Strengths: promotes reusability, reduces code duplication. Weakness: can lead to many small classes and more HTML markup.

Each methodology works best for different project requirements - BEM for component libraries, SMACSS for large applications, OOCSS for highly reusable UI elements. **Frequency:** Medium (55%) **Companies:** Facebook, Netflix, Amazon, Microsoft, PayPal

103. How would you structure CSS for a large-scale application to ensure maintainability?

Answer: For large-scale applications, I would implement: 1. A methodology like BEM or SMACSS for consistent naming 2. CSS architecture with clear organization (e.g., atomic design principles) 3. CSS modules or scoped styles to prevent global conflicts 4. Variables for consistent theming (custom properties) 5. Documentation for complex components and usage patterns 6. Separation of concerns (layout vs. components vs. utilities) 7. Performance considerations (code splitting, critical CSS) 8. Style linting for enforcing conventions 9. Source control best practices for CSS files 10. Testing strategy for visual regression

This approach creates maintainable, scalable CSS that multiple teams can work with. **Frequency:** High (65%) **Companies:** Google, LinkedIn, Salesforce, Adobe, Twitter

104. Explain CSS Modules and how they help solve common CSS problems.

Answer: CSS Modules are a way of writing CSS that automatically scopes styles to a specific component by creating unique class names during build time. This solves several common CSS problems: prevents global namespace pollution, eliminates selector specificity wars, avoids unintentional style inheritance, and reduces the need for complex naming conventions. When using CSS Modules, classnames like `.button` get transformed into something like `.Button_button_1gxM4` at build time, ensuring styles remain encapsulated to their component. This approach works particularly well with component-based architectures like React. **Frequency:** High (75%) **Companies:** Facebook, Spotify, Airbnb, Pinterest, Atlassian

105. What CSS properties trigger layout, paint, and composite operations? How can you optimize for performance?

Answer: Properties that trigger: - Layout (most expensive): width, height, margin, padding, display, border, top/right/bottom/left, font-size, position - Paint (medium cost): color, background, box-shadow, visibility, text-shadow, border-radius - Composite (least expensive): opacity, transform, filter, will-change

Performance optimization strategies: 1. Prefer transform/opacity changes for animations (composite-only) 2. Use will-change for elements that will animate 3. Avoid forced synchronous layouts (layout thrashing) 4. Batch DOM read/write operations 5. Reduce paint areas with contain: paint 6. Minimize repaint triggers during scrolling 7. Use hardware acceleration for animations via transform: translateZ(0) 8. Simplify selectors to speed up matching **Frequency:** Medium (60%) **Companies:** Google, Facebook, Netflix, Apple, Uber

106. Explain "critical CSS" and how would you implement it.

Answer: Critical CSS is an optimization technique that extracts and inlines the minimum CSS required to render the above-the-fold content of a page. This improves page load times by allowing visible content to render without waiting for external CSS files to download.

Implementation approaches: 1. Manual extraction: Identify and extract styles for above-the-fold content 2. Automated tools: Use tools like Critical, Penthouse, or CriticalCSS 3. Build process integration: Add critical CSS extraction to webpack/gulp pipeline 4. Server-side detection: Vary critical CSS based on device/viewport 5.

Delivery: Inline critical styles in , then asynchronously load full CSS

This technique is particularly valuable for initial page load performance and Core Web Vitals metrics. **Frequency:** Medium (55%) **Companies:** Amazon, Google, Walmart, Shopify, Airbnb

107. How would you diagnose and fix a CSS performance issue in a production application?

Answer: To diagnose and fix CSS performance issues:

1. Use Chrome DevTools Performance panel to identify bottlenecks:
    - Record during scrolling, animations, or interactions
    - Look for long frames with layout/paint operations
    - Check "Rendering" tab for paint flashing and layout boundaries
2. Common diagnosis methods:
    - Identify frequently triggered layouts/reflows
    - Find paint-heavy areas (large or frequent)
    - Check for non-composite animations
    - Analyze unused CSS with Coverage tool
    - Measure selector matching performance
3. Common fixes:
    - Replace layout/paint properties with transform/opacity for animations
    - Add will-change for elements that animate frequently
    - Reduce selector complexity for hot paths
    - Apply content-visibility: auto for off-screen content
    - Split CSS into critical and non-critical paths
    - Remove unused CSS or implement code splitting
    - Replace expensive animations with simpler alternatives
    - Add contain: strict where appropriate **Frequency:** Medium (60%) **Companies:** Facebook, LinkedIn, Twitter, Airbnb, PayPal

108. Explain CSS custom properties (variables) and provide scenarios where they're better than preprocessor variables.

Answer: CSS custom properties (variables) are native CSS entities defined with -- prefix and accessed with var() function. They follow the cascading rules, inherit through the DOM, and can be modified via JavaScript.

Scenarios where they're better than preprocessor variables: 1. Runtime changes: Can be updated dynamically with JavaScript without rebuilding 2. Contextual changes: Can be scoped to media queries, states, or themes 3. DOM inheritance: Values can inherit and be overridden down the DOM tree 4. Interaction with JavaScript: Can react to user interactions or application state 5. Responsive adaptations: Can change values at different breakpoints without duplication 6. Theme switching: Can update global theming without page reload 7. Progressive enhancement: Can provide fallbacks for older browsers

Example:
`css :root { --primary-color: blue; } @media (prefers-color-scheme: dark) { :root { --primary-color: lightblue; } } .button { backgr`
Frequency: High (75%) **Companies:** Google, Microsoft, Adobe, Spotify, Shopify

109. How would you implement dark mode using CSS variables?

Answer: To implement dark mode with CSS variables:

1. Define theme variables in :root:
   `css :root { --text-color: #333; --background-color: #fff; --heading-color: #0077cc; /* other theme variables */ }`
2. Create a dark mode alternative:
   `css @media (prefers-color-scheme: dark) { :root { --text-color: #eee; --background-color: #121212; --heading-color: #59b0ff; }`
3. For manual toggle, use a class:
   `css .dark-theme { --text-color: #eee; --background-color: #121212; --heading-color: #59b0ff; }`
4. Apply variables throughout your CSS:
   `css body { color: var(--text-color); background-color: var(--background-color); } h1, h2, h3 { color: var(--heading-color); }`
5. Add JavaScript toggle functionality:
   `javascript const toggleButton = document.getElementById('theme-toggle'); toggleButton.addEventListener('click', () => { documen`

This approach enables instant theme switching without page reload and respects user system preferences. **Frequency:** High (70%) **Companies:** Apple, Twitter, Facebook, Reddit, GitHub

110. What are CSS Houdini APIs and how do they extend CSS capabilities?

Answer: CSS Houdini is a set of low-level APIs that give developers direct access to the CSS Object Model (CSSOM), enabling the creation of custom CSS features without waiting for browser implementations. Main APIs include:

1. Paint API: Create custom background patterns and images with JavaScript
2. Layout API: Define custom layout algorithms beyond existing CSS layouts
3. Animation Worklet API: Run animations off the main thread for smoother performance
4. Typed OM: Structured, performance-optimized way to work with CSS values
5. Properties and Values API: Register custom properties with types and inheritance
6. Parser API: Allow custom CSS parsing rules
7. Font Metrics API: Access low-level font metrics information

Benefits include polyfilling future CSS features, creating previously impossible layouts and animations, and improving performance by working closer to the browser's rendering engine.

Example: `js // Registering a custom paint worklet CSS.paintWorklet.addModule('my-paint-worklet.js');` `css .element { background-image: paint(myCustomPaint); }` **Frequency:** Low (35%) **Companies:** Google, Microsoft, Opera, Samsung Internet, Mozilla

111. How do you ensure your CSS supports accessibility requirements?

Answer: To ensure CSS supports accessibility:

1. Color contrast: Maintain WCAG 2.1 contrast ratios (4.5:1 for normal text, 3:1 for large text)
   `css /* Good color contrast */ .text { color: #333; background-color: #fff; }`
2. Focus states: Make focusable elements clearly visible with distinct focus styles
   `css button:focus { outline: 3px solid #4d90fe; outline-offset: 2px; }`
3. Text sizing: Use relative units (em/rem) to respect user font size preferences `css body { font-size: 1rem; } h1 { font-size: 2rem; }`
4. Content visibility: Avoid hiding content users need to access
   `css /* Accessible hiding */ .visually-hidden { position: absolute; height: 1px; width: 1px; overflow: hidden; clip: rect(1px 1p`
5. Hover/focus parity: Ensure hover effects also work for keyboard `css .button:hover, .button:focus { background-color: #0056b3; }`
6. Text spacing: Accommodate custom user text spacing `css p { line-height: 1.5; }`
7. Reduced motion: Support reduced motion preferences
   `css @media (prefers-reduced-motion: reduce) { .animation { animation: none; } }`
8. Print styles: Ensure content remains accessible when printed
   `css @media print { .no-print { display: none; } body { font-size: 12pt; } }` **Frequency:** High (65%) **Companies:** Google, Microsoft, Apple, IBM, Government agencies

112. Explain how to use CSS to support users with visual impairments.

Answer: Supporting users with visual impairments through CSS:

1. Text size and scaling:
   `css html { font-size: 100%; /* Respect browser settings */ } body { font-size: 1rem; line-height: 1.5; }`
2. High contrast mode detection: `css @media (forced-colors: active) { button { forced-color-adjust: none; } }`
3. Focus indicators: `css :focus { outline: 3px solid #4d90fe; outline-offset: 2px; }`
4. Non-color information cues: `css .error { color: #d32f2f; border-left: 4px solid #d32f2f; padding-left: 1rem; }`
5. Font choices for readability: `css body { font-family: -apple-system, BlinkMacSystemFont, sans-serif; letter-spacing: 0.01em; }`
6. Respecting dark mode preferences:
   `css @media (prefers-color-scheme: dark) { :root { --text-color: #eee; --background-color: #121212; } }`
7. Adequate spacing for touch targets: `css button, .clickable { min-height: 44px; min-width: 44px; padding: 0.5rem 1rem; }`
8. Screen reader only content:
   `css .sr-only { position: absolute; width: 1px; height: 1px; padding: 0; margin: -1px; overflow: hidden; clip: rect(0, 0, 0, 0);`
   Frequency: Medium (60%) **Companies:** Microsoft, Apple, Adobe, Government sites, Healthcare

113. What CSS considerations should be made for users who prefer reduced motion?

Answer: For users who prefer reduced motion:

1. Detect the preference: `css @media (prefers-reduced-motion: reduce) { /* Reduced motion styles */ }`
2. Replace or reduce animations:
   `css @media (prefers-reduced-motion: reduce) { * { animation-duration: 0.001ms !important; animation-iteration-count: 1 !importa`
3. Provide alternative static presentations: ```css .card { transition: transform 0.3s ease; } .card:hover { transform: scale(1.05); }

   @media (prefers-reduced-motion: reduce) { .card { transition: none; } .card:hover { transform: none; box-shadow: 0 0 0 2px #4d90fe; /* Alternative hover state */ } }```
4. Consider scroll behavior: ```css html { scroll-behavior: smooth; }

   @media (prefers-reduced-motion: reduce) { html { scroll-behavior: auto; } } ```
5. Disable parallax effects:
   `css @media (prefers-reduced-motion: reduce) { .parallax { background-attachment: scroll !important; } }`
6. Replace video backgrounds:
   `css @media (prefers-reduced-motion: reduce) { .video-bg { display: none; } .static-bg { display: block; } }`

This helps prevent vestibular disorders, motion sickness, and reduces cognitive load while still providing meaningful interaction cues through non-motion alternatives. **Frequency:** Medium (55%) **Companies:** Apple, Microsoft, Twitter, Meta, Shopify

114. Compare CSS-in-JS solutions (like styled-components or Emotion) with traditional CSS approaches.

Answer: CSS-in-JS vs Traditional CSS:

CSS-in-JS Advantages: 1. Component-scoped styles with automatic unique class names 2. No global namespace conflicts 3. Dynamic styling based on props/state 4. Dead code elimination (only used styles are included) 5. Colocation of styles with components 6. JavaScript variables shared with styles 7. Runtime evaluation and theme application 8. TypeScript integration for style props

Traditional CSS Advantages: 1. Separation of concerns (HTML/CSS/JS) 2. Browser caching of CSS files 3. No runtime style generation overhead 4. Works without JavaScript 5. Easier for CSS specialists without JS knowledge 6. Standard tooling (linters, formatters) 7. No framework lock-in 8. Better browser devtools support

Performance Considerations: - CSS-in-JS may have runtime performance cost - Traditional CSS loads and parses separately from JS - CSS-in-JS often creates more granular style rules - Critical CSS extraction can be more complex with CSS-in-JS

Examples: Styled-components:
`jsx const Button = styled.button` background: ${props => props.primary ? '#0077cc' : 'white'}; color: ${props => props.primary ? 'w`

Traditional CSS:
`css .button { padding: 0.5rem 1rem; } .button-primary { background: #0077cc; color: white; } .button-secondary { background: white;`
Frequency: High (75%) **Companies:** Facebook, Airbnb, Pinterest, Atlassian, Uber

115. What are the performance implications of CSS-in-JS?

Answer: Performance implications of CSS-in-JS:

Potential Performance Costs: 1. Runtime overhead for style generation and injection 2. JavaScript bundle size increases 3. Additional CPU work for style processing 4. Memory usage for style objects and runtime processing 5. Style recalculation on prop changes 6. Runtime CSS injection can block rendering 7. Time-to-Interactive may be delayed

Performance Benefits: 1. Critical CSS extraction is often automatic 2. Only used styles are included (tree-shaking) 3. Modern solutions use style extraction at build time 4. Component-based chunking aligns with code-splitting 5. Avoids unused CSS rules in the global scope

Optimization Strategies: 1. Server-side rendering of critical styles 2. Static extraction when possible (zero-runtime solutions) 3. Memoization of styled components 4. Sharing style instances across components 5. Using "zero runtime" CSS-in-JS libraries (e.g., Linaria)

Comparison: - Static CSS: Fastest initial load, no runtime cost - Runtime CSS-in-JS: More flexible, potentially slower initial render - Zero-runtime CSS-in-JS: Balance between both approaches

Modern CSS-in-JS libraries like Emotion, styled-components (v5+), and Stitches have optimized for performance, but still have some overhead compared to static CSS. **Frequency:** Medium (60%) **Companies:** Facebook, Spotify, Airbnb, Pinterest, Vercel

116. How would you implement a theme switching system with CSS?

Answer: To implement a theme switching system:

Method 1: CSS Variables (Modern approach) ```css /* Define themes */ :root { /* Light theme (default) */ --color-background: #ffffff; --color-text: #333333; --color-primary: #0077cc; }

[data-theme="dark"] { --color-background: #121212; --color-text: #f0f0f0; --color-primary: #59b0ff; }

/* Apply variables */ body { background-color: var(--color-background); color: var(--color-text); } ```

```
// Theme switcher
function toggleTheme() {
  const currentTheme = document.documentElement.getAttribute('data-theme') || 'light';
  const newTheme = currentTheme === 'light' ? 'dark' : 'light';

  document.documentElement.setAttribute('data-theme', newTheme);
  localStorage.setItem('theme', newTheme);
}

// Load saved theme
document.addEventListener('DOMContentLoaded', () => {
  const savedTheme = localStorage.getItem('theme');
  if (savedTheme) {
    document.documentElement.setAttribute('data-theme', savedTheme);
  } else if (window.matchMedia('(prefers-color-scheme: dark)').matches) {
    document.documentElement.setAttribute('data-theme', 'dark');
  }
});
```

Method 2: Class-based (Better compatibility) ```css /* Light theme (default) */ body { background-color: #ffffff; color: #333333; }

/* Dark theme */ body.dark-theme { background-color: #121212; color: #f0f0f0; } ```

Additional Features: 1. Transition between themes: `css body { transition: background-color 0.3s ease, color 0.3s ease; }`

1. Theme-specific images: `css .theme-image { content: var(--theme-image-url); }`
2. System preference detection:
   `css @media (prefers-color-scheme: dark) { :root:not([data-theme="light"]) { --color-background: #121212; --color-text: #f0f0f0;`
   Frequency: High (70%) **Companies:** Twitter, Reddit, GitHub, Google, Medium

117. Explain your approach to debugging complex CSS issues across browsers.

Answer: My approach to cross-browser CSS debugging:

1. Systematic Investigation: - Start with browser developer tools to inspect elements - Compare computed styles between browsers - Isolate the issue by creating a minimal test case - Use feature detection rather than browser detection

2. Common Cross-Browser Issues & Solutions: - Flexbox: Add appropriate prefixes, avoid mix of flex and min/max constraints - Grid: Provide fallbacks for older browsers - Height calculations: Check box-sizing differences - Font rendering: Use web-safe font stacks or font-display - Default margins/padding: Use a CSS reset/normalize

3. Tools & Techniques: - BrowserStack/LambdaTest for testing in multiple environments - Feature detection with @supports - Autoprefixer for vendor prefix management - CSS linting for compatibility warnings - Visual regression testing tools to catch rendering differences

4. Debugging Process: - Add temporary debugging styles (e.g., borders, backgrounds) - Incrementally remove or add CSS to isolate problem - Test with different content lengths/sizes - Check parent element styles that might affect children - Validate HTML structure

5. Documentation: - Document browser-specific fixes with comments - Create a pattern library of cross-browser components - Maintain a knowledge base of workarounds - Add browser support notes to component documentation

6. Edge Cases: - Handle print styles separately - Test with different pixel densities - Check both mouse and touch interactions - Validate RTL language support **Frequency:** High (65%) **Companies:** Microsoft, Mozilla, Adobe, Shopify, Salesforce

118. How would you refactor legacy CSS code for a large application?

Answer: Refactoring legacy CSS approach:

1. Assessment & Planning: - Audit existing CSS (size, specificity, duplication) - Identify unused CSS with coverage tools - Define coding standards and methodology (BEM, SMACSS, etc.) - Create visual inventory of UI components - Define CSS architecture that aligns with application structure

2. Incremental Approach: - Start with global styles (typography, colors, spacing) - Extract variables/custom properties for consistency - Adopt the "strangler pattern" - build new system alongside old - Create isolation for new components to prevent conflicts - Replace components one by one rather than full rewrite

3. Technical Implementation: - Move from global styles to component encapsulation - Reduce specificity by flattening selectors - Remove !important declarations

systematically - Replace hardcoded values with variables - Organize code with logical file structure - Add comments for complex sections

4. Testing & QA: - Implement visual regression testing - Create component documentation - Test across browsers and viewports - Add safeguards against future specificity problems

5. Performance Optimization: - Remove duplicate declarations - Combine similar rules - Split CSS into logical chunks (critical/non-critical) - Reduce selector complexity

6. Knowledge Sharing: - Document patterns and standards - Create living style guide - Train team on new architecture - Set up linting to maintain quality

This approach balances immediate improvements with systematic refactoring while minimizing risk. **Frequency:** Medium (60%) **Companies:** IBM, Oracle, Salesforce, Financial institutions, Government sites

119. What strategies do you use to test CSS across different browsers and devices?

Answer: Strategies for cross-browser/device CSS testing:

1. Testing Infrastructure: - Browser testing services (BrowserStack, LambdaTest, Sauce Labs) - Device labs for physical testing - Virtual machines with different browser versions - Responsive design mode in DevTools - Mobile emulators and simulators

2. Automated Testing: - Visual regression testing (Percy, BackstopJS, Chromatic) - Screenshot comparison across browsers - CSS linting with browser compatibility rules - Automated reports for CSS usage and support

3. Development Practices: - Progressive enhancement approach - Feature detection with @supports - Graceful degradation fallbacks - Mobile-first responsive design - Using Autoprefixer for vendor prefixes

4. Test Coverage Approach: - Define browser support matrix (primary/secondary) - Prioritize testing for high-traffic browsers/devices - Create test scenarios for specific CSS features - Test extremes (old browsers, small screens, large screens) - Test with different font sizes and zoom levels

5. Common Test Cases: - Layout integrity at various viewports - Hover/focus states work consistently - Touch interactions on mobile devices - Font rendering consistency - Animation performance - Print stylesheet testing - RTL language support

6. Workflow Integration: - Incorporate testing in CI/CD pipeline - Pre-commit hooks for CSS validation - Collaborative bug tracking system - Document known issues and workarounds **Frequency:** Medium (60%) **Companies:** Microsoft, Mozilla, Adobe, Shopify, Booking.com

120. How do you handle CSS for print layouts?

Answer: Handling CSS for print layouts:

1. Print-specific Stylesheet: ```css /* Option 1: Separate print stylesheet */

/* Option 2: Media query in main CSS */ @media print { /* Print-specific styles */ } ```

2. Key Print Considerations: ```css @media print { /* Hide non-essential elements */ nav, footer, .ads, .comments, .no-print { display: none !important; }

/* Ensure background colors/images print */ * { -webkit-print-color-adjust: exact !important; print-color-adjust: exact !important; }

/* Optimize text for reading */ body { font-size: 12pt; line-height: 1.5; font-family: Georgia, serif; color: #000; background: #fff; }

/* Ensure links are useful in printed form */ a { color: #000; text-decoration: underline; } a[href]:after { content: " (" attr(href) ")"; font-size: 90%; }

/* Avoid breaking elements across pages */ h1, h2, h3, h4, h5, h6, img { page-break-after: avoid; page-break-inside: avoid; }

/* Control page breaks */ .page-break { page-break-before: always; }

/* Expand content to full width */ .container { width: 100%; max-width: none; margin: 0; padding: 0; }

/* Make tables more readable */ table { border-collapse: collapse; } table, th, td { border: 1px solid #000; } } ```

3. Advanced Techniques: - Control widow/orphan text: `css p { orphans: 3; widows: 3; }`

- Add print-specific content: `css .print-only { display: none; } @media print { .print-only { display: block; } }`
- Page size and orientation: `css @page { size: letter portrait; margin: 2cm; }`

4. Testing Print Styles: - Use browser print preview - Test across browsers (particularly Chrome and Firefox) - Check PDF output - Validate with different paper sizes - Test with and without background colors/images setting enabled **Frequency:** Medium (45%) **Companies:** News sites, Educational institutions, Government agencies, Documentation sites