📚 Complete HTML & CSS Revision Guide

Detailed answers with examples • Perfect for interviews & quick revision • Fully responsive design

All Topics HTML CSS Advanced

📁 HTML Fundamentals

❓ What is HTML and how does it differ from CSS and JavaScript?

HTML **HTML (HyperText Markup Language)** is the standard markup language for creating web pages. It provides the structure and content of a webpage.

> 💡 Think of HTML as the skeleton of a webpage.

Key Differences:

- **HTML:** Structure and content (headings, paragraphs, images)
- **CSS:** Presentation and styling (colors, layout, fonts)
- **JavaScript:** Behavior and interactivity (animations, form validation)

```
<!-- HTML -->
<h1>Welcome</h1>
```

```
<!-- CSS makes it blue -->
<!-- JavaScript makes it interactive -->
```

❓ What are HTML5 Semantic Elements and why are they important?

HTML5 **Semantic elements** clearly describe their meaning to both browser and developer. **Common Semantic Elements:**

- `<header>` - Introductory content
- `<nav>` - Navigation links
- `<main>` - Main content area
- `<article>` - Self-contained content
- `<section>` - Thematic grouping
- `<aside>` - Sidebar content
- `<footer>` - Footer content

Benefits:

- Better SEO (search engines understand content)
- Improved accessibility (screen readers)
- Cleaner, more readable code

❓ What is the purpose of the DOCTYPE declaration?

DOCTYPE The **DOCTYPE declaration** tells the browser which version of HTML the page is written in and triggers standards mode rendering.

```
<!DOCTYPE html> <!-- HTML5 DOCTYPE -->
```

Why it's essential:

- Prevents **quirks mode** (old, inconsistent rendering)
- Ensures consistent rendering across browsers
- Required for valid HTML documents

> 💡 Without DOCTYPE, browsers may render pages in unexpected ways!

❓ What's the difference between ID and Class attributes?

Attributes

ID Attribute:

- Unique identifier for a single element
- CSS selector: `#id-name`
- Used for: JavaScript targeting, page anchors
- High CSS specificity

Class Attribute:

- Reusable identifier for multiple elements
- CSS selector: `.class-name`
- Used for: Styling groups of elements
- Medium CSS specificity

```
<div id="header" class="container primary">Content</div>
```

❓ What's the difference between and elements?

Elements

(Division):

- **Block-level** element
- Starts on a new line, takes full width
- Used for layout and grouping block elements
- Accepts width, height, margins, padding

:

- **Inline-level** element
- Flows with text, takes only necessary width
- Used for styling parts of text
- Ignores width and height properties

```
<div>This is a block element</div>
This text has a <span style="color: red;">red span</span>
inside.
```

❓ How do HTML Forms work and what are common input types?

Forms

HTML Forms collect user input and send it to a server for processing.

Common Input Types:

- `text` - Single-line text input
- `password` - Masked text input
- `email` - Email validation
- `number` - Numeric input
- `date` - Date picker
- `radio` - Single selection
- `checkbox` - Multiple selection

- `submit` - Form submission button

```
<form action="/submit" method="POST">
  <input type="text" name="username" required>
  <input type="submit" value="Send">
</form>
```

📁 CSS Fundamentals

❓ What is the CSS Box Model and how does it work?

Box Model

The **CSS Box Model** describes how every element is represented as a rectangular box with four areas:

- **Content:** Actual text/images (innermost)
- **Padding:** Space around content (inside border)
- **Border:** Line surrounding padding
- **Margin:** Space between elements (outside border)

Box-Sizing Property:

- `content-box` (default) - Width/height apply to content only
- `border-box` - Width/height include content + padding + border

> 💡 Always use `box-sizing: border-box` for predictable layouts!

❓ What's the difference between display: inline and display: block?

Display **display: inline**

- Flows within text content
- Ignores width and height properties
- No line breaks before/after
- Examples: `<span>`, `<a>`, `<strong>`

display: block

- Creates rectangular blocks
- Accepts width and height properties
- Starts on a new line
- Examples: `<div>`, `<p>`, `<h1>`

display: inline-block - Hybrid: flows like inline but accepts dimensions like block.

❓ How does the CSS position property work?

Position **position: static** - Default, normal document flow

position: relative - Positions relative to its normal position

```
.element {
  position: relative;
  top: 20px;
  left: 10px;
}
```

position: absolute - Removed from flow, positioned relative to nearest positioned ancestor

position: fixed - Relative to viewport, stays during scrolling

position: sticky - Hybrid of relative and fixed

❓ What's the difference between margin and padding?

Spacing **Padding:**

- Space **inside** the element, between content and border
- Background color extends into padding
- Does NOT collapse
- Use for: Internal spacing within components

Margin:

- Space **outside** the element, between elements
- Transparent (no background)
- Vertical margins collapse (only larger margin applies)
- Use for: External spacing between components

```
.box {
  padding: 20px; /* Internal space */
  margin: 30px; /* External space */
  border: 2px solid black;
}
```

❓ What is the purpose of the z-index property?

z-index The **z-index** property controls the stacking order of positioned elements along the z-axis (depth). **How it works:**

- Higher z-index = closer to user (in front)
- Lower z-index = further from user (behind)
- Negative values = behind normal content
- Only works on **positioned** elements

```
.modal {
  position: fixed;
  z-index: 1000; /* High = in front */
}
.backdrop {
  position: fixed;
  z-index: 999; /* Lower = behind */
}
```

> 💡 z-index only works within the same stacking context!

❓ What is CSS Specificity and how does it work?

Specificity **CSS Specificity** determines which CSS rules are applied when multiple rules target the same element. **Specificity Hierarchy (high to low):**

1. Inline styles (`style=""`)
2. ID selectors (`#id`)
3. Class/attribute/pseudo-class selectors
4. Element/pseudo-element selectors

Calculation:

- Inline: 1000 points
- ID: 100 points
- Class: 10 points
- Element: 1 point

```
#header .nav li a { } /* 100 + 10 + 1 + 1 = 112 */
.nav li.active a { } /* 10 + 1 + 10 + 1 = 22 */
/* First rule wins! */
```

📁 Advanced Concepts

❓ CSS Grid vs Flexbox - When to use which?

Layout **CSS Grid:**

- **2-dimensional** layout (rows + columns simultaneously)
- Container-based control
- Perfect for: Overall page layout, complex grids
- Use when you need control over both axes

Flexbox:

- **1-dimensional** layout (row OR column)
- Item-based flexibility
- Perfect for: Component layout, content distribution
- Use when layout is in one direction

> 💡 Use Grid for big picture layout, Flexbox for component layout!

❓ What are Media Queries and how are they used for responsive design?

Responsive **Media Queries** apply different CSS styles based on device characteristics like screen width. **Common Breakpoints:**

- Mobile: < 768px
- Tablet: 768px - 1024px
- Desktop: > 1024px

```
/* Mobile First Approach */
.container { padding: 1rem; }

/* Tablet */
@media (min-width: 768px) {
  .container { max-width: 720px; }
}

/* Desktop */
@media (min-width: 1024px) {
  .container { max-width: 960px; }
}
```

❓ What are CSS Preprocessors and why use SASS/LESS?

Preprocessors **CSS Preprocessors** extend CSS with programming features and compile to regular CSS. **Key SASS Features:**

- **Variables:** Store colors, fonts for reuse
- **Nesting:** Hierarchical organization
- **Mixins:** Reusable code blocks
- **Functions:** Mathematical operations

- **Import:** Modular file structure

```
// SASS Variables
$primary-color: #3498db;
$spacing: 1rem;

.button {
  background: $primary-color;
  padding: $spacing;
}
```

🎯 **Perfect for interviews & quick revision** • 📱 **Fully responsive design** • ⚡ **Detailed explanations with examples**

Bookmark this page and revisit before interviews or when you need a quick refresher!