CSS

CSS Syntax

Definition:
CSS (Cascading Style Sheets) defines how HTML elements should be **displayed on screen, paper, or other media.**

Why / Purpose

- To **separate content and design**
- Makes web pages **stylish & responsive**
- Easy to **maintain and reuse**

Syntax Pattern

```
selector {
  property: value;
  property: value;
}
```

- **Selector** → HTML element to style
- **Property** → style attribute (color, font, etc.)
- **Value** → assigned value

Example

```
p {
  color: blue;
  font-size: 16px;
}
```

Real-Life Example
CSS is like **choosing paint colors and fonts for a house.**

Interview Notes

- CSS is **case-insensitive**
- Semi-colon ; is mandatory for multiple properties

Ways to Assign CSS

Method	Syntax	Scope	Example
Inline	style attribute	Single element	Text
Internal	<style> tag inside <head>	Whole page	<style>p{color:blue;}</style>
External	.css file linked with <link>	Multiple pages	<link rel="stylesheet" href="style.css">

Real-Life Analogy

- Inline → Painting **one wall**
- Internal → Decorating **one room**
- External → Decorating **entire building consistently**

Interview Notes

- External CSS is **best practice**
- Inline CSS is **not recommended** for maintainability

CSS Selectors

Definition:
Selectors are patterns used to **select HTML elements** to apply styles.

Types of CSS Selectors

Selector	Syntax	Purpose
Universal	*	Selects all elements
Type / Element	p	Selects all <p> tags
Class	.classname	Selects elements with class
ID	#idname	Selects element with specific ID
Group	h1, h2, p	Selects multiple elements
Descendant	div p	Selects <p> inside
Child	div > p	Selects direct child <p> of
Attribute	[type="text"]	Selects elements with attribute

Example

```
/* Type selector */
p { color: green; }

/* Class selector */
.text { font-size: 14px; }

/* ID selector */
#main { background-color: yellow; }

/* Attribute selector */
input[type="text"] { border: 1px solid black; }
```

Real-Life Example
Selectors are like **address labels** telling CSS **where to apply styles.**

Interview Notes

- Class selector . can be used **multiple times**
- ID selector # should be **unique per page**
- Use **group selectors** for efficiency

Quick Interview Summary

- CSS syntax → selector { property: value; }
- Ways to assign → Inline, Internal, External
- Selectors → ID #, Class ., Element p, Attribute [attr=value]

Styling Text & Fonts

Definition:
Text styling in CSS is used to **control the appearance of text** such as color, alignment, spacing, and decoration.

Purpose

- Improve **readability**

- Make content **visually attractive**
- Maintain **UI consistency**

Common Text Properties

Property	Description	Example
color	Text color	color: red;
text-align	Alignment	center, left, right
text-decoration	Underline, none	underline, none
text-transform	Case change	uppercase
letter-spacing	Space between letters	2px
word-spacing	Space between words	5px
line-height	Line spacing	1.5

Example

```
p {
  color: blue;
  text-align: justify;
  text-transform: capitalize;
}
```

Real-Life Example

Text styling is like **formatting a Word document.**

Interview Notes

- text-decoration: none removes underline from links
- line-height improves readability

Font Properties

Definition:

Font properties control **typeface, size, style, and weight** of text.

Common Font Properties

Property	Purpose	Example
font-family	Font type	Arial, sans-serif
font-size	Text size	16px, 1em
font-style	Italic/normal	italic
font-weight	Thickness	bold, 400
font-variant	Small caps	small-caps
font	Shorthand	Combines all

Example

```
h1 {
  font-family: 'Arial', sans-serif;
  font-size: 24px;
  font-weight: bold;
}
```

Real-Life Example

Choosing fonts is like **selecting handwriting style.**

Interview Notes

- Always give **fallback fonts**
- px is fixed, em and rem are responsive

CSS Box Model (Basics)

Definition:
The CSS Box Model defines how **every HTML element is treated as a box.**

Box Model Components

```
| Margin |
| Border |
| Padding |
| Content |
```

Explanation

Part	Purpose
Content	Actual data (text/image)
Padding	Space inside border
Border	Edge around padding
Margin	Space outside border

Example

```
div {
  width: 200px;
  padding: 10px;
  border: 2px solid black;
  margin: 15px;
}
```

Real-Life Example
Box Model is like a **gift box**:

- Gift → content
- Cushion → padding
- Box → border
- Space around → margin

Interview Notes

- Default box sizing adds padding & border to width
- box-sizing: border-box; keeps width fixed

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

Quick Interview Summary

- Text styling → alignment, spacing, decoration
- Font properties → family, size, weight, style
- Box Model → Content → Padding → Border → Margin

CSS Box Model

Definition:
The **CSS Box Model** explains how every HTML element is structured and spaced on a web page.

Content
Definition:
The **content** is the actual data inside the element (text, image, video).
Example

```
div {
  width: 200px;
  height: 100px;
}
```

Padding
Definition
Padding is the **space between content and border**.
Example

```
div {
  padding: 10px;
}
```

Border
Definition
The **border** surrounds the padding and content.
Example

```
div {
  border: 2px solid black;
}
```

Margin
Definition
Margin is the **space outside the border**, separating elements.
Example

```
div {
  margin: 20px;
}
```

Order (Very Important for Interview)
Content → Padding → Border → Margin

Visual Structure

```
|  Margin  |
|  Border  |
| Padding  |
| Content  |
```

Interview One-Line Answer
The CSS Box Model consists of **content, padding, border, and margin**, which define the layout and spacing of elements.

CSS Colors, Backgrounds, Borders & Effects

CSS Colors
Definition
CSS **colors** are used to style **text, borders, backgrounds, and other elements.**

Ways to Define Colors
```
color: red;                 /* Color name */
color: #ff0000;             /* Hex */
color: rgb(255,0,0);        /* RGB */
color: rgba(255,0,0,0.5);   /* RGBA */
```

Interview Point
RGBA allows **transparency**, RGB does not.

Backgrounds
Definition
Background properties control the **background appearance** of an element.

Common Background Properties
```
background-color: lightblue;
background-image: url("img.jpg");
background-repeat: no-repeat;
background-position: center;
background-size: cover;
```
Short Form
```
background: lightblue url("img.jpg") no-repeat center;
```

CSS Borders
Definition
Borders define the **outline** of an element.

Border Properties
```
border-width: 2px;
border-style: solid;
border-color: black;
```
Shorthand
```
border: 2px solid black;
```

Border Types
- solid
- dashed
- dotted
- double
- groove

Interview Tip
Border occupies space in the **box model.**

CSS Effects
Definition
CSS **effects** improve UI appearance using shadows, transparency, and transitions.

Box Shadow
```
box-shadow: 2px 2px 10px gray;
```
Text Shadow
```
text-shadow: 1px 1px 3px black;
```
Opacity
```
opacity: 0.6;
```

Border Radius
border-radius: 10px;
Hover Effect

```
button:hover {
  background-color: green;
}
```

One-Line Interview Answers

- **Color** → Adds visual style
- **Background** → Controls element background
- **Border** → Creates outline
- **Effects** → Enhance UI experience

Quick Revision Table

Feature	Purpose
color	Text color
background	Background styling
border	Element outline
shadow	Depth effect
opacity	Transparency

Fresher Interview Tip
CSS effects are mostly used to **improve user experience,** not business logic.

CSS Layout Basics

Definition:
CSS Layout controls **how elements are arranged and aligned** on a web page.

---

display Property
Definition
The display property defines **how an element is displayed** in the layout.

---

block

- Takes full width
- Starts on a new line
  Examples: , <p>

```
display: block;
```

---

inline

- Takes only content width
- No new line
  Examples: , <a>

```
display: inline;
```

---

inline-block

- Inline + can set width/height

```
display: inline-block;
```

---

none

- Hides element completely

```
display: none;
```

Interview Point

display:none removes the element from layout.

---

position Property
Definition
position specifies **how an element is positioned** on the page.

Types of Position
static (default)
- Normal document flow

```
position: static;
```

---

relative
- Positioned relative to itself

```
position: relative;
top: 10px;
left: 20px;
```

---

absolute
- Positioned relative to nearest positioned parent

```
position: absolute;
top: 0;
right: 0;
```

---

fixed
- Fixed to viewport (does not scroll)

```
position: fixed;
bottom: 0;
```

---

sticky
- Acts relative until scroll, then fixed

```
position: sticky;
top: 0;
```

Interview Tip
Absolute positioning works relative to the **nearest non-static parent**.

---

float Property
Definition
float is used to **move elements left or right**, allowing text to wrap around.

Values

```
float: left;
float: right;
```

Clear Float

```
clear: both;
```

Interview Point
float is **old layout technique**, replaced by **Flexbox & Grid**.

One-Line Interview Answers
- **display** → Defines how elements appear
- **position** → Controls element placement
- **float** → Aligns elements left/right

Modern Layout Note (Important)
Today, **Flexbox** and **Grid** are preferred over float for layouts.

CSS Flexbox Layout

Definition:
Flexbox (Flexible Box Layout) is a **one-dimensional layout model** used to align and distribute space between items in a container.

Flex Container & Flex Items
Flex Container
The parent element where Flexbox is applied.

```
.container {
  display: flex;
}
```

Flex Items
Direct children of the flex container.

Interview Point
Only **direct child elements** become flex items.

---

flex-direction
Definition
flex-direction defines the **direction in which flex items are placed.**

Values
row (default)
Items arranged left → right

```
flex-direction: row;
```

---

row-reverse
Right → left

```
flex-direction: row-reverse;
```

---

column
Top → bottom

```
flex-direction: column;
```

---

column-reverse
Bottom → top

```
flex-direction: column-reverse;
```

Interview Tip
row = horizontal layout
column = vertical layout

---

justify-content
Definition
justify-content aligns flex items **along the main axis.**
(Main axis depends on flex-direction)

Common Values
flex-start (default)
Items start from beginning

```
justify-content: flex-start;
```

---

center
Items centered

```
justify-content: center;
```

---

flex-end
Items at the end

justify-content: flex-end;

---

space-between
Equal space **between items**
justify-content: space-between;

---

space-around
Equal space **around items**
justify-content: space-around;

---

space-evenly
Equal space **everywhere**
justify-content: space-evenly;

Interview Favorite Question
Q: Difference between space-between and space-around?
A: space-around adds space on both ends; space-between does not.

Example

```
.container {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
}
```

One-Line Interview Answers

- **Flexbox** → Flexible one-dimensional layout
- **flex-direction** → Direction of items
- **justify-content** → Alignment on main axis

Important Note
justify-content works on **main axis**,
align-items works on **cross axis**.

CSS Grid Layout

Definition:
CSS Grid Layout is a **two-dimensional layout system** that allows you to design layouts using **rows and columns**.

Grid vs Flexbox (Quick)

- **Grid** → 2D (rows + columns)
- **Flexbox** → 1D (row OR column)

Grid Container & Grid Items
Grid Container
The parent element where grid is applied.

```
.container {
  display: grid;
}
```

Grid Items
Direct children of the grid container.

---

grid-template-columns
Definition
Defines **number of columns** and their **widths**.

Examples
Fixed width columns
grid-template-columns: 100px 200px 100px;

Percentage based
grid-template-columns: 30% 40% 30%;

Using fr (fraction unit)
grid-template-columns: 1fr 2fr 1fr;

- Middle column gets more space

Repeat function
grid-template-columns: repeat(3, 1fr);

Interview Tip
fr divides available space **fractionally.**

---

grid-template-rows
Definition
Defines **number of rows** and their **heights.**

Examples
grid-template-rows: 100px 200px;
grid-template-rows: repeat(2, 1fr);

Complete Example

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: 100px 200px;
  gap: 10px;
}
```

One-Line Interview Answers

- **CSS Grid** → Two-dimensional layout system
- **grid-template-columns** → Defines columns
- **grid-template-rows** → Defines rows

Important Interview Notes

- Grid items auto-place if not specified
- gap controls spacing between rows & columns
- Grid is best for **page-level layouts**

Common Interview Question
Q: When to use Grid over Flexbox?
A: When layout needs **rows and columns together.**

Responsive Web Design (RWD)

Definition:
Responsive Design is the approach of designing websites so they **adapt to different screen sizes and devices** automatically.

Purpose

- Works on **mobile, tablet, laptop, desktop**

- Improves **user experience**
- Mandatory in modern web development

Media Queries
Definition
Media Queries allow CSS to be applied **based on device screen size, resolution, or orientation.**

Basic Syntax

```
@media (condition) {
  /* CSS rules */
}
```

Example

```
@media (max-width: 768px) {
  body {
    background-color: lightblue;
  }
}
```

Applies when screen width is **≤ 768px**

Breakpoints
Definition
Breakpoints are specific **screen widths** at which the layout changes.

Common Breakpoints (Interview-Standard)

Device	Width
Mobile	≤ 576px
Tablet	577px – 768px
Laptop	769px – 992px
Desktop	≥ 993px

---

min-width vs max-width
max-width (Desktop First)

```
@media (max-width: 600px) {
  /* Mobile styles */
}
```

---

min-width (Mobile First – Recommended)

```
@media (min-width: 600px) {
  /* Tablet & above */
}
```

Interview Tip
Mobile-first design is preferred in industry.

Responsive Example

```
.container {
  width: 100%;
}

@media (min-width: 768px) {
  .container {
    width: 70%;
    margin: auto;
  }
}
```

One-Line Interview Answers

- **Responsive Design** → Adapts UI to screen size
- **Media Query** → Applies CSS conditionally
- **Breakpoint** → Screen size where layout changes

Important Interview Notes

- Use %, vw, vh, fr instead of fixed px
- Combine with **Flexbox/Grid**
- Always test on multiple devices

Common Interview Question
Q: Why use media queries?
A: To create device-specific layouts without separate websites.

CSS Transitions, Animations & Transform

Definition:
These CSS features are used to **add motion and visual effects** to web elements, improving **user experience (UI).**

CSS Transitions
Definition
A **transition** smoothly changes a CSS property **from one state to another** (usually on hover).

Properties
transition-property
transition-duration
transition-timing-function
transition-delay

Shorthand

```
transition: background-color 0.5s ease;
```

Example

```
button {
  background: blue;
  transition: background-color 0.3s;
}

button:hover {
  background: green;
}
```

Interview Point
Transitions need a **trigger** (hover, focus, active).

CSS Transform
Definition
transform changes the **shape, size, position, or rotation** of an element.

Common Transform Functions

```
transform: translate(50px, 20px);
transform: rotate(45deg);
transform: scale(1.2);
transform: skew(10deg);
```

Example

```
div:hover {
  transform: scale(1.1);
}
```

Interview Tip

Transforms do **not affect document flow.**

Real-Life Example

Rotating or zooming an image on phone.

CSS Animations

Definition

Animations allow **continuous or multi-step motion** using @keyframes.

Syntax

```
@keyframes moveBox {
  from { left: 0; }
  to { left: 200px; }
}
```

Usage

```
.box {
  position: relative;
  animation: moveBox 2s infinite;
}
```

Animation Properties

animation-name
animation-duration
animation-iteration-count
animation-delay
animation-timing-function

Shorthand

```
animation: moveBox 2s ease infinite;
```

Interview Point

Animations **do not need user interaction.**

Transition vs Animation (Interview Favorite)

Feature	Transition	Animation
Trigger	Required	Not required
Steps	Start → End	Multiple steps
Loop	No	Yes
Control	Limited	Full control

One-Line Interview Answers

- **Transition** → Smooth state change
- **Transform** → Visual change (rotate, scale)
- **Animation** → Continuous motion

Best Practices

- Use animations **sparingly**
- Prefer **transform & opacity** for performance
- Avoid heavy animations on mobile

Fresher Interview Tip
Use **transition for simple effects, animation for complex motion.**
