CSS LECTURE PDF

🟩 1. Introduction to CSS

CSS (Cascading Style Sheets) is a language used to style and format web pages. It controls the layout, colors, fonts, and other visual aspects of a webpage.

Example:

```
<!DOCTYPE html>
<html>
<head>
    <style>
        body {
            background-color: lightblue;
        }
        h1 {
            color: navy;
        }
    </style>
</head>
<body>
    <h1>Hello, CSS!</h1>
</body>
</html>
```

Explanation:

- `background-color` changes the page's background.
- `color` changes the text color.

🟩 2. Types of CSS

There are **three** ways to apply CSS:

1. **Inline CSS** (Directly within HTML elements)

```
<h1 style="color: red;">Inline CSS Example</h1>
```

2. **Internal CSS** (Inside `<style>` tag in HTML head section)

```
<head>
    <style>
        h1 { color: green; }
    </style>
</head>
```

3. **External CSS** (In a separate `.css` file)

```
/* style.css */
h1 {
    color: purple;
}
```

HTML file:

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

---

3. CSS Selectors

CSS Selectors are used to target HTML elements to apply styles.

1. **Element Selector**

```
p {
    color: blue;
}
```

2. **Class Selector (`.`):**

```
.myClass {
    font-size: 20px;
}
```

HTML:

```
<p class="myClass">Class Selector Example</p>
```

3. **ID Selector (#):**

```
#uniqueID {
    text-align: center;
}
```

4. **Group Selector (`,`):** Selects multiple elements-
   Ex-
   h1, p { color: purple; }

5. **Advanced Selectors**

- **Child (>), Descendant ( ), Adjacent (+), General Sibling (~)**

```
div > p { color: red; }  /* Direct child */
div p { color: blue; }  /* Any descendant */
```

HTML:

```
<h2 id="uniqueID">ID Selector Example</h2>
```

🟩 4. CSS Box Model

The **Box Model** describes how elements are structured in CSS:

1. **Content** – The main content (text, image).
2. **Padding** – Space between content and border.
3. **Border** – Surrounds padding and content.
4. **Margin** – Space outside the border.

Example:

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

HTML:

```
<div class="box">This is a box model.</div>
```

🟩 5. CSS Colors

You can set colors using various formats:

1. **Color Name**

```
h1 {
    color: red;
}
```

2. **HEX Code**

```
h1 {
    color: #ff5733;
}
```

3. **RGB Values**

```
h1 {
    color: rgb(255, 0, 0);
}
```

4. **RGBA (With Transparency)**

```
h1 {
    color: rgba(0, 0, 255, 0.5);
}
```

Background Properties-
Ex-

```
body {
  background-color: lightgray;
  background-image: url('image.jpg');
  background-size: cover;
  background-position: center;
}
```

🟩 6. CSS Fonts and Text Styling

1. **Font Family**

```
p {
    font-family: Arial, sans-serif;
}
```

Font Size

```
h1 {
    font-size: 32px;
}
```

Unit	Description	Example
px	Pixels (Fixed size)	`font-size: 20px;`
em	Relative to the parent element	`font-size: 1.5em;`
rem	Relative to the root (`html`) element	`font-size: 2rem;`
%	Percentage of parent's font size	`font-size: 120%;`
vw/vh	Relative to viewport width/height	`font-size: 5vw;`

2. **Font Weight**

```
p {
    font-weight: bold;
}
```

Common Values:

Value	Description
normal	Default weight (400)
bold	Bold text (700)
lighter	Lighter than normal
bolder	Bolder than normal
100–900	Numeric scale (thin to extra-bold)

3. **Text Alignment**

```
p {
    text-align: center;
}
```

Value	Description
left	Aligns text to the left
right	Aligns text to the right
center	Centers the text
justify	Spreads text to align both sides

4. Text Decoration

Adds lines and effects to the text (like **underline**).

Syntax:

```
css
text-decoration: value;
```

Values:

Value	Description
none	No decoration
underline	Underlines the text
overline	Line above the text
line-through	Strikethrough (crossed-out text)
blink	Blinking text (not supported in many browsers)

Example:

```
a {
    text-decoration: none; /* Removes
underline */
}

h2 {
    text-decoration: underline;
}
```

5. Text Transform

Controls the capitalization of text.

Syntax:

```
text-transform: value;
```

Values:

Value	Description
none	Default (no transformation)
capitalize	Capitalizes first letter
uppercase	All letters in uppercase
lowercase	All letters in lowercase

Example:

```
h1 {
    text-transform: uppercase;
}

p {
    text-transform: capitalize;
}
```

---

6. Line Height

Controls the **vertical spacing** between lines of text.

Syntax:

```
line-height: value;
```

Values:

- **Normal** – Default line height (usually 1.2).
- **Number** – Multiplier of the font size.
- **Length (px, em, etc.)** – Fixed value.

Example:

```
p {
    line-height: 1.5; /* 1.5 times the font size */
}
```

---

7. Letter Spacing

Adjusts the **space between letters**.

Syntax:

```
letter-spacing: value;
```

Example:

```
h1 {
    letter-spacing: 2px; /* Adds 2px gap between letters */
}
```

---

8. Word Spacing

Adjusts the **space between words**.

Syntax:

```
word-spacing: value;
```

Example:

```
p {
    word-spacing: 5px; /* Increases space between words */
}
```

---

9. Text Shadow

Adds shadow effects to text.

Syntax:

```
text-shadow: h-offset v-offset blur-radius color;
```

Example:

```
h1 {
    text-shadow: 2px 2px 5px gray;
}
```

10. Text Border

🟩 7. CSS Layouts

1. Types of CSS Layout Models

1. **Normal Flow** – Default behavior of HTML elements.
2. **Flexbox Layout** – One-dimensional layout (Row or Column).
3. **Grid Layout** – Two-dimensional layout (Rows and Columns).

1. Normal Document Flow

By default, HTML elements are displayed in the following flow:

1. **Block Elements** – Take the **full width** (e.g., `<div>`, `<p>`, `<h1>`).
2. **Inline Elements** – Take **only necessary width** (e.g., `<span>`, `<a>`).

<!DOCTYPE html>

<html lang="en">

<head>

<title>Normal Flow Example</title>

<style>

```
        div {
            background: lightblue;
            padding: 10px; margin:
            10px;
        }
        span {
            background: lightgreen; padding:
            5px;
        }
    </style>
</head>
<body>
    <div>This is a block element.</div>
    <span>This is an inline element.</span>
    <span>Another inline element.</span>
</body>
</html>
```

2. CSS Flexbox (For flexible layouts)

Flexbox is a layout model for creating flexible and responsive designs.

Key Flexbox Properties:

Property	Description
`display: flex;`	Activates flexbox on a container.
`flex-direction`	Sets the main axis (`row`, `column`).
`justify-content`	Aligns items horizontally.
`align-items`	Aligns items vertically.
`flex-wrap`	Allows items to wrap onto new lines.
`gap`	Adds space between items.

Ex-

```
<!DOCTYPE html>
```

```html
<html lang="en">
<head>
    <title>Flexbox Layout</title>
    <style>
        .flex-container {
            display: flex;
            flex-direction: row; /* Horizontal */
            justify-content: space-between; /* Spread items */
            align-items: center; /* Center vertically */
            gap: 10px; /* Space between items */
        }
        .flex-item {
            background: lightcoral;
            padding: 20px;
        }
    </style>
</head>
<body>
    <div class="flex-container">
        <div class="flex-item">Item 1</div>
        <div class="flex-item">Item 2</div>
        <div class="flex-item">Item 3</div>
    </div>
</body>
</html>
```

3. **CSS Grid** (For grid-based layouts)
   CSS Grid is a **two-dimensional** system for creating complex page layouts.

Key Grid Properties:

Property	Description
`display: grid;`	Activates CSS Grid on a container.
`grid-template-columns`	Defines column structure.
`grid-template-rows`	Defines row structure.
`gap`	Adds space between grid items.
`grid-column`	Controls an item's column position.
`grid-row`	Controls an item's row position.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Grid Layout</title>
  <style>
    .grid-container {
      display: grid;
      grid-template-columns: repeat(3, 1fr); /* 3 Equal Columns */
      gap: 20px; /* Space between items */
    }
    .grid-item {
      background: lightseagreen;
      padding: 20px;
      text-align: center;
    }
    .wide {
      grid-column: span 2; /* Span 2 columns */
    }
  </style>
</head>
<body>
  <div class="grid-container">
    <div class="grid-item">Item 1</div>
    <div class="grid-item">Item 2</div>
    <div class="grid-item wide">Item 3 (Wide)</div>
    <div class="grid-item">Item 4</div>
  </div>
</body>
</html>
```

🟩 8. CSS Positioning-:

Allows precise manual control over element positioning.

📌 Position Values:

Value	Description
`static`	Default (follows normal flow).
`relative`	Positioned relative to its normal position.
`absolute`	Positioned relative to the nearest positioned ancestor.
`fixed`	Fixed to the viewport (does not scroll).
`sticky`	Toggles between relative and fixed.

1. **Static (Default Position)**

```
div {
    position: static;
}
```

2. **Relative (Relative to Normal Position)**

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

3. **Absolute (Relative to Nearest Positioned Ancestor)**

```
div {
    position: absolute;
    top: 50px;
    left: 30px;
}
```

4. **Fixed (Fixed on the Screen)**

```
div {
    position: fixed;
    top: 0;
    right: 0;
}
```

---

8. CSS Pseudo-classes & Pseudo-elements-

1. Pseudo-classes

Ex-

a:hover { color: red; }

p:first-child { font-weight: bold; }

2. Pseudo-elements-

Ex-

p::first-letter { font-size: 30px; }

🟩 9. CSS Transitions

Transitions allow smooth animations between styles.

1. What is a CSS Transition?

A **CSS Transition** is a way to animate changes to a CSS property over a specified **duration**.

⚡ Basic Syntax of CSS Transition:

```
css
transition: property duration timing-function delay;
```

Property	Description	Example
`property`	The CSS property you want to animate.	`background-color`
`duration`	How long the transition takes (s/ms).	`1s`, `500ms`
`timing-function`	Controls the animation speed curve.	`ease`, `linear`
`delay` *(optional)*	Wait before starting the animation.	`0.5s`, `200ms`

---

2. Basic Example of a CSS Transition

🎨 Hover to Change Background Color

```
html
CopyEdit
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Basic CSS Transition</title>
    <style>
        .box {
            width: 150px;
            height: 150px;
            background-color: lightblue;
            transition: background-color 0.5s ease;
        }
```

```
        .box:hover {
            background-color: coral; /* Changes on hover */
        }
    </style>
</head>
<body>
    <div class="box"></div>
</body>
</html>
```

📌 Explanation:

- **Initial State:** Light blue box.
- **On Hover:** Smoothly transitions to coral over **0.5 seconds**.

---

3. CSS Transition Properties

⚡ Commonly Animated Properties

You can animate almost any **numerical** CSS property:

Animatable Property	Examples
Colors	`color`, `background-color`, `border-color`
Dimensions	`width`, `height`, `max-height`, `min-width`
Position	`top`, `left`, `right`, `bottom`
Transparency	`opacity`
Borders	`border`, `border-radius`
Transforms	`transform: scale(), rotate(), translate()`

---

4. Advanced CSS Transition Examples

🔥Example 1: Multiple Properties Transition

```
html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Multiple Transitions</title>
    <style>
        .box {
            width: 150px;
            height: 150px;
            background-color: pink;
            border-radius: 0px;
            transition: background-color 0.5s ease, border-radius 1s
linear;
        }

        .box:hover {
            background-color: purple; /* Color change */
            border-radius: 50%;      /* Circular shape */
```

```
        }
    </style>
</head>
<body>
    <div class="box"></div>
</body>
</html>
```

📌 Explanation:

- **background-color** transitions in **0.5s**.
- **border-radius** transitions in **1s**.

🎨 Example 2: Delayed Transition

```
html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Delayed Transition</title>
    <style>
        .box {
            width: 200px;
            height: 100px;
            background: lightgreen;
            transition: width 1s ease-in-out 0.5s; /* 0.5s delay */
        }

        .box:hover {
            width: 400px; /* Expand on hover */
        }
        </style>
</head>
<body>
    <div class="box"></div>
</body>
</html>
```

📌 Explanation:

- The box **waits 0.5 seconds** before expanding.
- Smooth transition occurs over **1 second**.

🎨 Example 3: Transition with Transform

```
html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Transform Transition</title>
    <style>
        .box {
            width: 150px;
            height: 150px;
            background: lightcoral;
            transition: transform 0.7s ease-out;
        }
```

```
        .box:hover {
            transform: rotate(45deg) scale(1.2);
        }
    </style>
</head>
<body>
    <div class="box"></div>
</body>
</html>
```

🔍 Explanation:

- The box rotates by **45 degrees** and **scales up** on hover.

---

5. CSS Transition Timing Functions

The **timing function** controls the speed of the animation.

Timing Function	Behavior
`linear`	Constant speed throughout.
`ease` *(default)*	Starts slow, accelerates, then slows.
`ease-in`	Starts slow and speeds up.
`ease-out`	Starts fast and slows down.
`ease-in-out`	Starts slow, speeds up, slows down.
`cubic-bezier()`	Custom speed curve using control points.

🧪 Example: Timing Functions in Action

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Timing Function Example</title>
    <style>
        .box {
            width: 150px;
            height: 150px;
            background: lightblue;
            margin: 10px;
        }

        .linear { transition: width 2s linear; }
        .ease { transition: width 2s ease; }
        .ease-in { transition: width 2s ease-in; }
        .ease-out { transition: width 2s ease-out; }
        .ease-in-out { transition: width 2s ease-in-out; }

        .box:hover { width: 300px; }
    </style>
</head>
<body>
    <div class="box linear">Linear</div>
    <div class="box ease">Ease</div>
    <div class="box ease-in">Ease-in</div>
```

```
    <div class="box ease-out">Ease-out</div>
    <div class="box ease-in-out">Ease-in-out</div>
</body>
</html>
```

🟩 10. CSS Animations

CSS **Animations** allow you to create **smooth, multi-step animations** without using JavaScript. You can animate **colors, sizes, positions**, and much more!

---

1. What is a CSS Animation?

CSS animations let you **gradually change** CSS properties from one value to another.

⚡ CSS Animation – Basic Syntax

```
selector {
    animation: name duration timing-function delay iteration-count
direction;
}
```

Property	Description	Example
`animation-name`	Name of the animation (from `@keyframes`).	`slide`
`animation-duration`	Time the animation takes (in seconds or milliseconds).	`2s`, `500ms`
`animation-timing-function`	Speed curve (e.g., ease, linear).	`ease-in-out`
`animation-delay`	Delay before the animation starts.	`1s`, `0.5s`
`animation-iteration-count`	Number of times the animation runs.	`infinite`, `3`
`animation-direction`	Direction of animation (normal, reverse).	`alternate`, `reverse`

---

2. How CSS Animation Works

1. **Define** the animation using `@keyframes`.
2. **Apply** the animation using the `animation` property.

---

3. Simple CSS Animation Example

#️⃣ Example 1: Moving a Box Horizontally

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Simple CSS Animation</title>
    <style>
        .box {
```

```
            width: 100px;
            height: 100px;
            background-color: coral;
            position: relative;
            animation: moveRight 3s ease-in-out infinite;
        }

        @keyframes moveRight {
            0% {
                left: 0;
            }

            50% {
                left: 300px;
            }

            100% {
                left: 0;
            }

        }
    </style>
</head>

<body>
    <div class="box"></div>
</body>

</html>
```

🍒 Explanation:

- **`@keyframes moveRight`** defines the motion.
- The box moves **300px** to the right and comes back.
- The animation loops **infinitely**.

---

4. CSS Animation Properties Explained

1. `animation-name`

Defines the **name** of the animation. This must match the `@keyframes` name.

2. `animation-duration`

Sets how **long** the animation takes to complete.

```
animation-duration: 2s; /* 2 seconds */
```

3. `animation-timing-function`

Controls the **speed** of the animation over time.

Value	Description
`ease`	Default – Slow at start and end.
`linear`	Constant speed.
`ease-in`	Slow start.
`ease-out`	Slow end.

`ease-in-out`	Slow start and end.
`cubic-bezier()`	Custom easing curve.

ϟ Example – Timing Functions

```
animation-timing-function: ease-in-out;
```

---

5. Advanced CSS Animation Examples

Example 2: Bouncing Ball Animation

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Bouncing Ball</title>
    <style>
        .ball {
            width: 50px;
            height: 50px;
            background-color: red;
            border-radius: 50%;
            position: relative;
            animation: bounce 2s ease-in-out infinite;
        }

        @keyframes bounce {
            0% {
                top: 0;
            }

            50% {
                top: 300px;
            }

            100% {
                top: 0;
            }
        }
    </style>
</head>

<body>
    <div class="ball"></div>
</body>

</html>
```

Example 3: Color Changing Animation

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Color Change</title>
    <style>
        .box {
            width: 150px;
```

```
            height: 150px;
            animation: colorChange 3s linear infinite;
        }

        @keyframes colorChange {
            0% {
                background: red;
            }

            50% {
                background: yellow;
            }

            100% {
                background: blue;
            }
        }
    </style>
</head>

<body>
    <div class="box"></div>
</body>

</html>
```

---

6. CSS Animation Directions

Value	Description
`normal`	Plays animation from start to end (default).
`reverse`	Plays animation backward.
`alternate`	Plays forward first, then backward.
`alternate-reverse`	Plays backward first, then forward.

Example 4: Alternate Direction Animation

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Alternate Animation</title>
    <style>
        .box {
            width: 100px;
            height: 100px;
            background: lightcoral;
            position: relative;
            animation: slide 3s ease alternate infinite;
        }

        @keyframes slide {
            from {
                left: 0;
            }

            to {
                left: 300px;
            }
        }
```

```
    </style>
</head>
<body>
    <div class="box"></div>
</body>

</html>
```

---

7. Animation Iteration Count

Defines how many times an animation will repeat.

Value	Description
`infinite`	Repeats forever.
`1, 2, 3...`	Repeats a specific number.

🎉 Example 5: Limited Iterations

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Limited Animation</title>
    <style>
        .box {
            width: 150px;
            height: 150px;
            background: lightgreen;
            animation: grow 2s ease 3; /* Runs 3 times */
        }

        @keyframes grow {
            from {
                transform: scale(1);
            }

            to {
                transform: scale(1.5);
            }
        }
    </style>
</head>

<body>
    <div class="box"></div>
</body>

</html>
```

---

9. Pausing and Playing Animations

Property	Description
`animation-play-state`	Controls play/pause.

🚀 Example 6: Pause on Hover

```
.box:hover {
    animation-play-state: paused;
}
```

---

10. Summary – CSS Animation Cheat Sheet

Property	Example
`animation`	`move 3s ease infinite;`
`animation-name`	`move`
`animation-duration`	`2s`, `500ms`
`animation-timing-function`	`ease`, `linear`
`animation-delay`	`1s`, `0.5s`
`animation-iteration-count`	`infinite`, `3`
`animation-direction`	`alternate`, `reverse`

Would you like to explore **CSS Transforms** or **Responsive Animations** next? 🎯

🟩 11. Responsive Design

Responsive CSS makes websites adaptable to different screen sizes.

CSS Media Queries for Tablet, Laptop, Mobile, and DesktopCSS **media queries** allow you to apply different styles depending on a device's screen size, resolution, or other properties. They are a key part of **responsive web design**.

---

1. Media Query Syntax

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

Parameter	Description
`media-type`	`all`, `screen`, `print`, etc.
`condition`	`max-width`, `min-width`, `orientation`, etc.
`and`	Combines multiple conditions.

---

2. Media Query Breakpoints for Devices

Here are **standard breakpoints** to target different devices:

Device Type	Breakpoint (Width)
📱 Mobile Phones	`max-width: 480px`
📲 Tablets (Portrait)	`min-width: 481px` and `max-width: 768px`
💻 Laptops (Small)	`min-width: 769px` and `max-width: 1024px`
🖥️ Desktops (Large)	`min-width: 1025px` and up

---

3. Full Example – Responsive Design for All Devices

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Responsive Media Queries</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 0;
            padding: 0;
        }

        .container {
            padding: 20px;
        }

        /* 🟩 Default Style for Large Screens (Desktops) */
        .container {
            background-color: lightblue;
        }

        /* 📱 Mobile Devices (max-width: 480px) */
        @media (max-width: 480px) {
            .container {
                background-color: lightcoral;
            }
        }

        /* 📲 Tablets (481px - 768px) */
        @media (min-width: 481px) and (max-width: 768px) {
            .container {
                background-color: lightgreen;
            }
        }

        /* 💻 Laptops (769px - 1024px) */
        @media (min-width: 769px) and (max-width: 1024px) {
```

```
            .container {
                background-color: lightgoldenrodyellow;
            }
        }

        /* 🖥 Desktops (1025px and up) */
        @media (min-width: 1025px) {
            .container {
                background-color: lightblue;
            }
        }
    </style>
</head>

<body>
    <div class="container">
        <h1>Responsive Media Queries</h1>
        <p>Resize your browser window to see the background color
change.</p>
    </div>
</body>

</html>
```

---

4. Explanation

1. **Default Style** – Light blue for large screens (desktop).
2. **Mobile** – Light coral for screens **480px or less**.
3. **Tablet** – Light green for screens **481px to 768px**.
4. **Laptop** – Light goldenrod for screens **769px to 1024px**.
5. **Desktop** – Light blue for screens **1025px and larger**.

---

5. Advanced Media Query Features

1. Orientation-Based Queries

```
/* Portrait Mode */
@media (orientation: portrait) {
    body {
        background: pink;
    }
}

/* Landscape Mode */
@media (orientation: landscape) {
    body {
        background: lightblue;
    }
}
```

2. High-Resolution Displays (Retina)

```
/* For screens with at least 2x pixel density */
@media only screen and (min-resolution: 192dpi) {
    body {
        background: url('high-res-image.jpg');
    }
}
```

🟩 12. CSS Variables

CSS Variables help you reuse values across your stylesheets.

Example:

```
:root {
    --main-color: #4CAF50;
}

h1 {
    color: var(--main-color);
}
```
