**CSS Media Queries ,Animations and JS Introduction**

**Media Queries**

**What is Media Query?**

* Media Queries are **CSS rules that apply styles based on device characteristics** such as **screen width, height, resolution, orientation**.
* Purpose: To make websites **responsive** so they look good on **mobile, tablet, and desktop**.

**Example Definition:**

@media (max-width: 768px) {

body {

background-color: lightgreen;

}

}

* If screen width ≤ 768px → body background becomes light green.

**When Do We Use Media Queries?**

* When a website needs **responsive design**.
* To **adjust layout, font-size, spacing, or visibility** based on screen size.
* To **improve user experience on mobile devices**.
* To **show/hide elements** for different devices.

**Example:**

/\* Desktop styles \*/

nav {

display: flex;

}

/\* Mobile styles \*/

@media (max-width: 600px) {

nav {

display: block;

}

}

**Where / How We Use Media Queries?**

* Inside **CSS file** (internal or external).
* Can use **min-width** or **max-width** conditions.
* Can combine multiple conditions using **and, or, not**.

**Example – Combined Media Query:**

@media (min-width: 600px) and (max-width: 1024px) {

body {

font-size: 16px;

}

}

* Works only for **tablet devices** between 600px and 1024px width.

**Key Media Features**

| **Feature** | **Description** |
| --- | --- |
| max-width | Applies if device width ≤ value |
| min-width | Applies if device width ≥ value |
| orientation | portrait / landscape |

**Example – Responsive Layout**

/\* Desktop \*/

.container {

width: 80%;

margin: 0 auto;

}

/\* Tablet \*/

@media (max-width: 1024px) {

.container {

width: 90%;

}

}

/\* Mobile \*/

@media (max-width: 600px) {

.container {

width: 100%;

padding: 0 10px;

}

}

**Explanation:** Container width adjusts based on screen size → responsive layout.

**CSS Animations**

**What is CSS Animation?**

* CSS Animations allow **HTML elements to change their appearance smoothly over time**.
* You can animate: **position, color, size, rotation, opacity**.
* Uses **@keyframes** to define **steps of animation**.
* Controlled with **animation properties** like duration, iteration-count, timing-function.

**When Do We Use CSS Animations?**

* To **make website interactive and engaging**.
* For **hover effects, sliding menus, fade-in content**.
* For **attention-grabbing UI elements** (buttons, alerts).
* For **visual storytelling** in web design.

**Where / How We Use CSS Animations?**

* Inside CSS (<style> or external .css file).
* Steps:
  1. Define **keyframes** (@keyframes).
  2. Apply animation to element with **animation properties**.

**Example – Bounce Animation:**

@keyframes bounce {

0% { transform: translateY(0); }

50% { transform: translateY(-20px); }

100% { transform: translateY(0); }

}

.button {

animation-name: bounce;

animation-duration: 2s;

animation-iteration-count: infinite;

}

Button will **bounce continuously**.

**Animation Properties**

| **Property** | **Description** |
| --- | --- |
| animation-name | Name of keyframes |
| animation-duration | Duration of animation |
| animation-iteration-count | Number of repetitions (infinite = loop) |
| animation-timing-function | Speed curve (linear, ease, ease-in, ease-out, ease-in-out) |
| animation-delay | Delay before animation starts |
| animation-direction | normal, reverse, alternate |
| animation-fill-mode | forwards, backwards, both (keeps final state) |

**Example – Fade In Animation**

@keyframes fadeIn {

0% { opacity: 0; }

100% { opacity: 1; }

}

.text {

animation-name: fadeIn;

animation-duration: 2s;

animation-fill-mode: forwards;

}

Text will **fade in smoothly** over 2 seconds.

**JS INTRODUCTION**

**1. What is JavaScript (JS)?**

* JavaScript is a **high-level, interpreted programming language** used to make web pages **dynamic and interactive**.
* It is an **object-oriented, client-side scripting language**.
* JS can **run directly in browsers** without compilation.
* Along with **HTML** (structure) and **CSS** (style), JS is a **core frontend technology** of the web.

**Example:**

alert("Hello World!");

This code will show a popup in the browser.

**2. Why We Use JS**

* To **add interactivity** to web pages: e.g., buttons, forms, sliders.
* To **validate forms** before submission.
* To **manipulate HTML elements dynamically** (DOM manipulation).
* To **create animations and effects**.
* To **handle events** like click, hover, scroll.
* To **communicate with servers** (AJAX) without reloading the page.

**Example Uses:**

* Drop-down menus
* Image sliders
* Pop-up modals
* Interactive calculators

**3. When Do We Use JS?**

* When you want **dynamic behavior** on a webpage.
* When a page needs to **react to user input**.
* When a website requires **animations or effects**.
* When you want **fetch data from servers** without reloading the page.

**4. Where Do We Use JS?**

* **Web browsers**: Chrome, Firefox, Safari, Edge.
* **Servers** (Node.js).
* **Web apps** (React.js, Angular, Vue.js).
* **Mobile apps** (React Native).
* **Desktop apps** (Electron).

**5. Types of Using JS**

There are mainly **3 ways to include JS in web pages**:

**1. Inline JS**

* Directly in HTML element using the onclick, onmouseover, etc.

<button onclick="alert('Hello')">Click Me</button>

**2. Internal JS**

* Inside <script> tag in HTML page.

<script>

function greet() {

alert("Hello World!");

}

</script>

<button onclick="greet()">Click Me</button>

**3. External JS**

* Using .js file and linking with <script src="file.js"></script>.

<script src="script.js"></script>

**Advantages:**

* Keeps code organized
* Can reuse in multiple pages

**6. What Happens on a Webpage When JS is Used**

* JS **can modify HTML content** dynamically (DOM manipulation).
* JS **handles user events** like clicks, typing, mouse movements.
* JS **changes styles** dynamically (CSS manipulation).
* JS **validates forms** before sending data to server.
* JS **performs calculations** without page reload.
* JS can **communicate with server asynchronously** (AJAX).

**Example: DOM Manipulation**

<p id="demo">Hello</p>

<button onclick="document.getElementById('demo').innerHTML='Hi!'">Click Me</button>

* On click, JS changes the text of <p> dynamically.