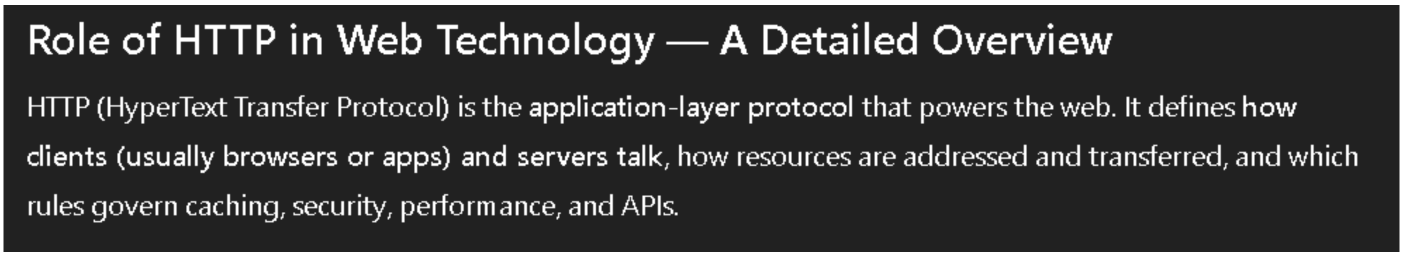
WT

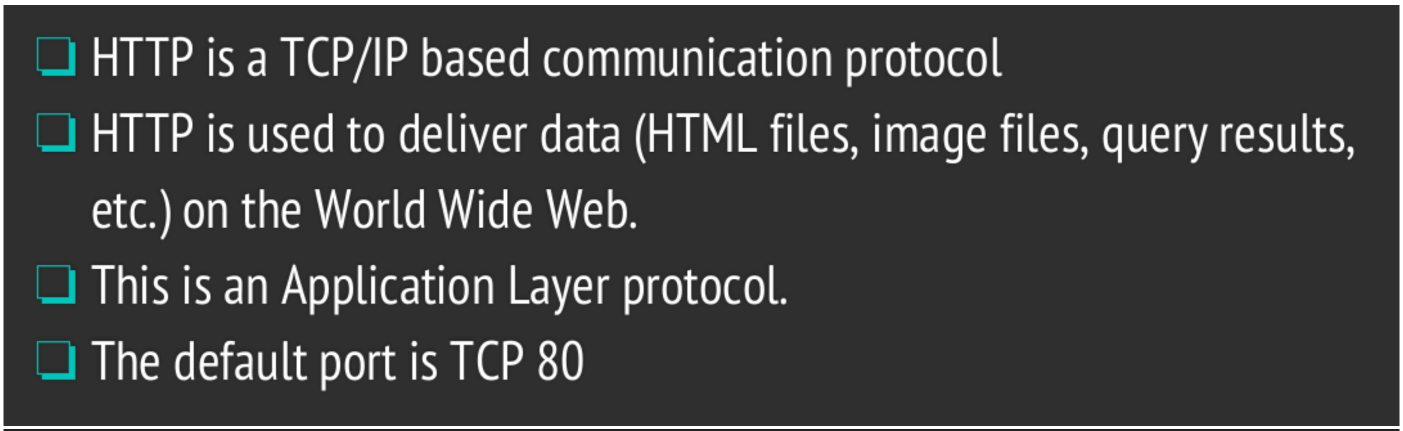
UNIT 1

1) Explain in detail role of HTTP protocol in web technology.

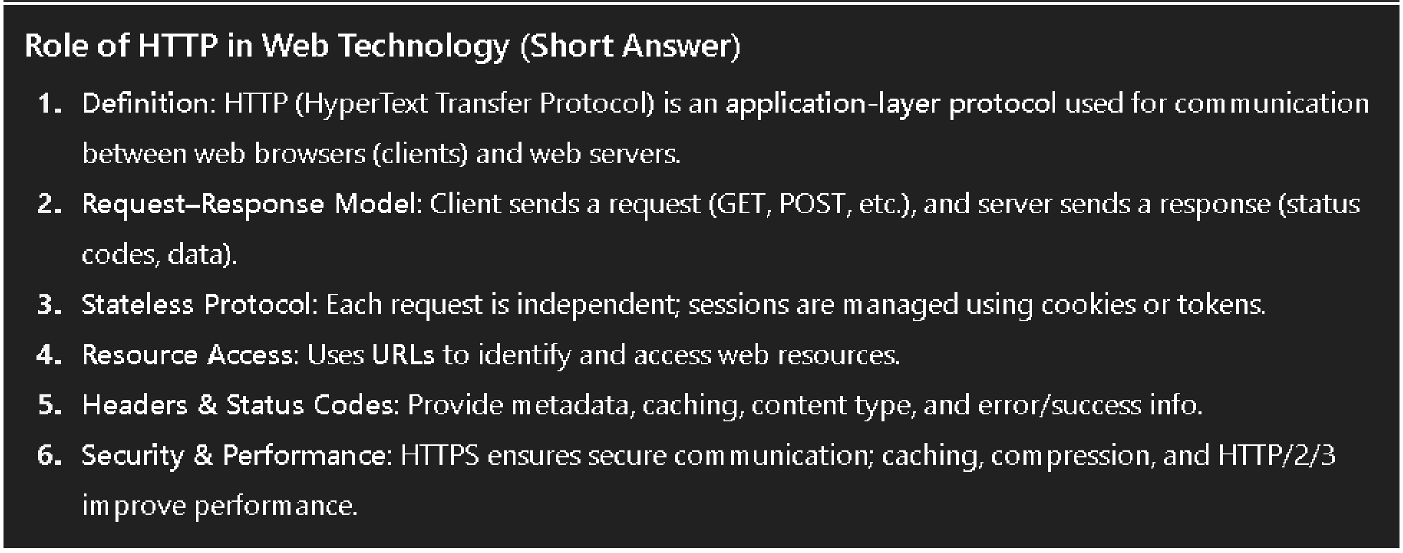


Role of HTTP in Web Technology — A Detailed Overview

HTTP (HyperText Transfer Protocol) is the **application-layer protocol** that powers the web. It defines **how clients (usually browsers or apps) and servers talk**, how resources are addressed and transferred, and which rules govern caching, security, performance, and APIs.

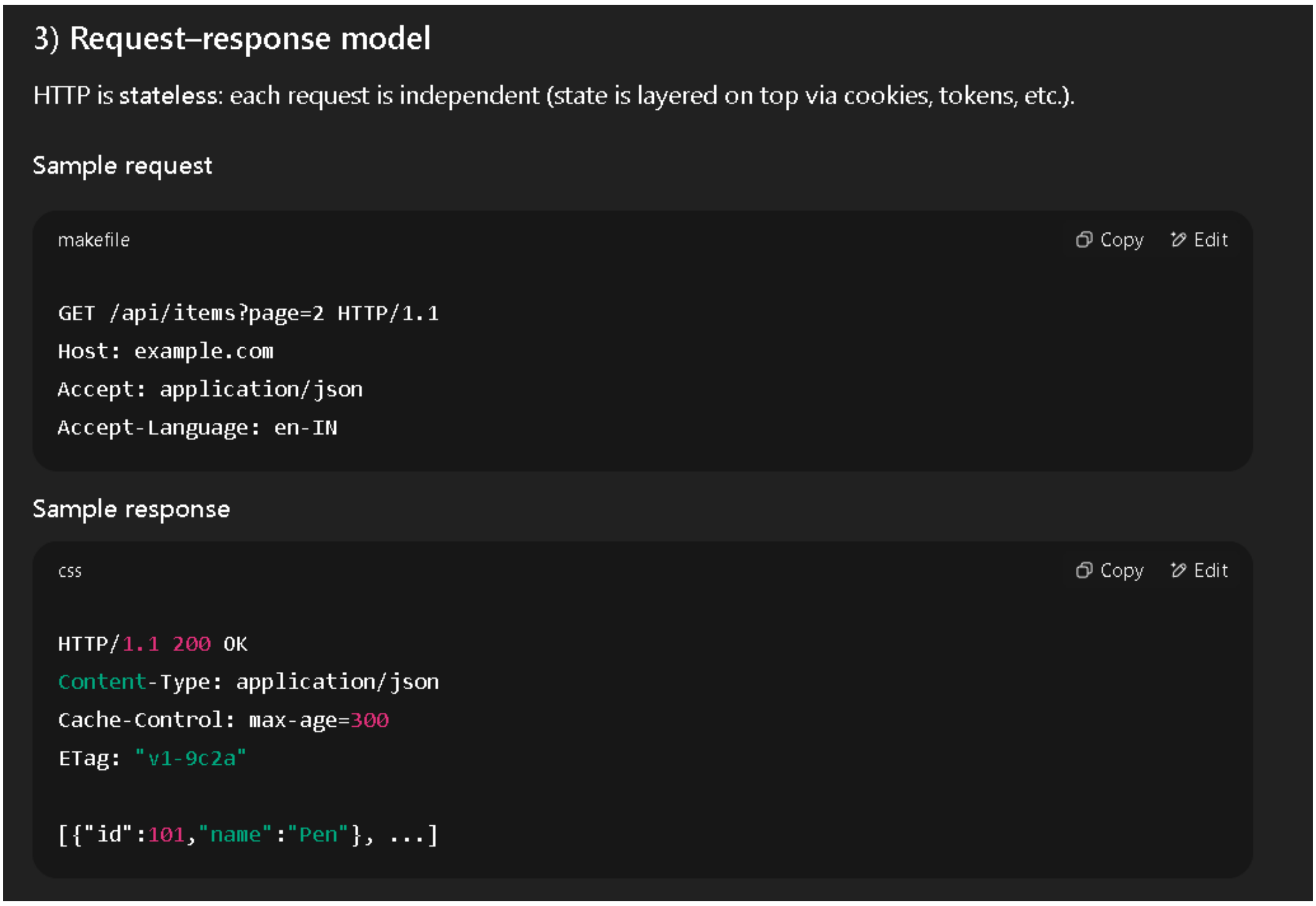


- ❑ HTTP is a TCP/IP based communication protocol
- ❑ HTTP is used to deliver data (HTML files, image files, query results, etc.) on the World Wide Web.
- ❑ This is an Application Layer protocol.
- ❑ The default port is TCP 80



Role of HTTP in Web Technology (Short Answer)

1. **Definition:** HTTP (HyperText Transfer Protocol) is an **application-layer protocol** used for communication between web browsers (clients) and web servers.
2. **Request–Response Model:** Client sends a request (GET, POST, etc.), and server sends a response (status codes, data).
3. **Stateless Protocol:** Each request is independent; sessions are managed using cookies or tokens.
4. **Resource Access:** Uses **URLs** to identify and access web resources.
5. **Headers & Status Codes:** Provide metadata, caching, content type, and error/success info.
6. **Security & Performance:** HTTPS ensures secure communication; caching, compression, and HTTP/2/3 improve performance.



3) Request–response model

HTTP is **stateless**: each request is independent (state is layered on top via cookies, tokens, etc.).

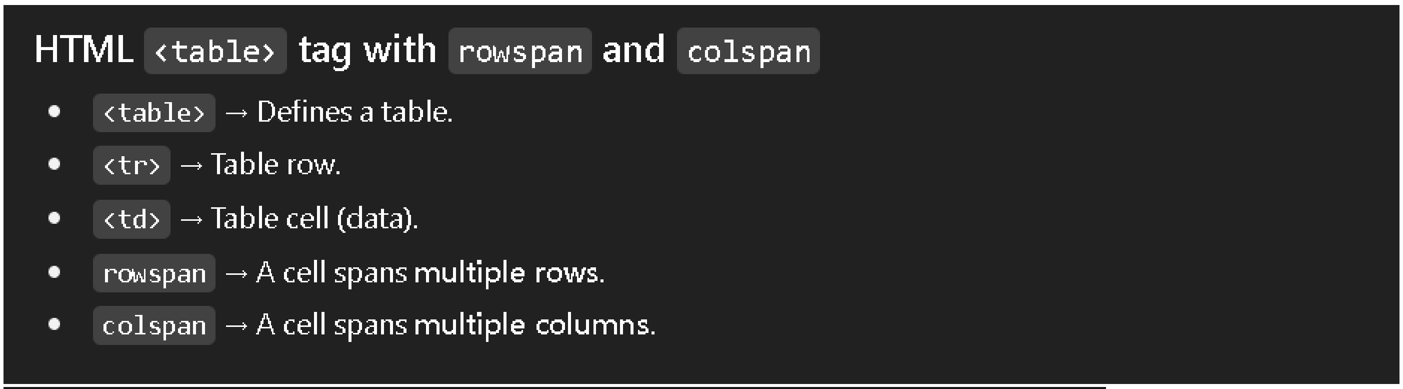
Sample request

```makefile
GET /api/items?page=2 HTTP/1.1
Host: example.com
Accept: application/json
Accept-Language: en-IN
```

Sample response

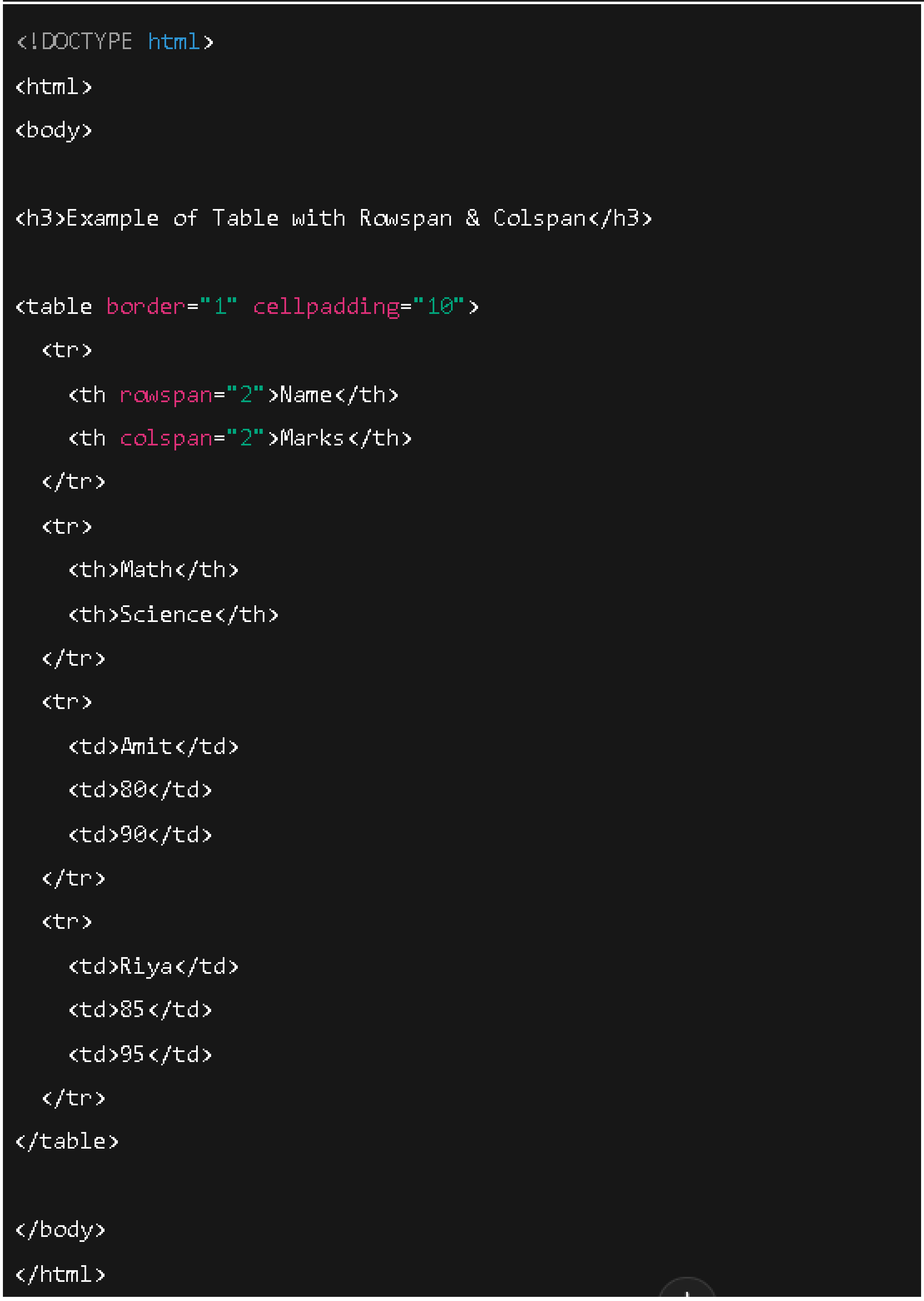
```css
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: max-age=300
ETag: "v1-9c2a"

[{"id":101,"name":"Pen"}, ...]
```

2) Explain example of table tag with concept of rowspan and colspan



HTML `<table>` tag with `rowspan` and `colspan`

- `<table>` → Defines a table.
- `<tr>` → Table row.
- `<td>` → Table cell (data).
- `rowspan` → A cell spans multiple rows.
- `colspan` → A cell spans multiple columns.



```html
<!DOCTYPE html>
<html>
<body>

<h3>Example of Table with Rowspan & Colspan</h3>

<table border="1" cellpadding="10">
  <tr>
    <th rowspan="2">Name</th>
    <th colspan="2">Marks</th>
  </tr>
  <tr>
    <th>Math</th>
    <th>Science</th>
  </tr>
  <tr>
    <td>Amit</td>
    <td>80</td>
    <td>90</td>
  </tr>
  <tr>
    <td>Riya</td>
    <td>85</td>
    <td>95</td>
  </tr>
</table>

</body>
</html>
```

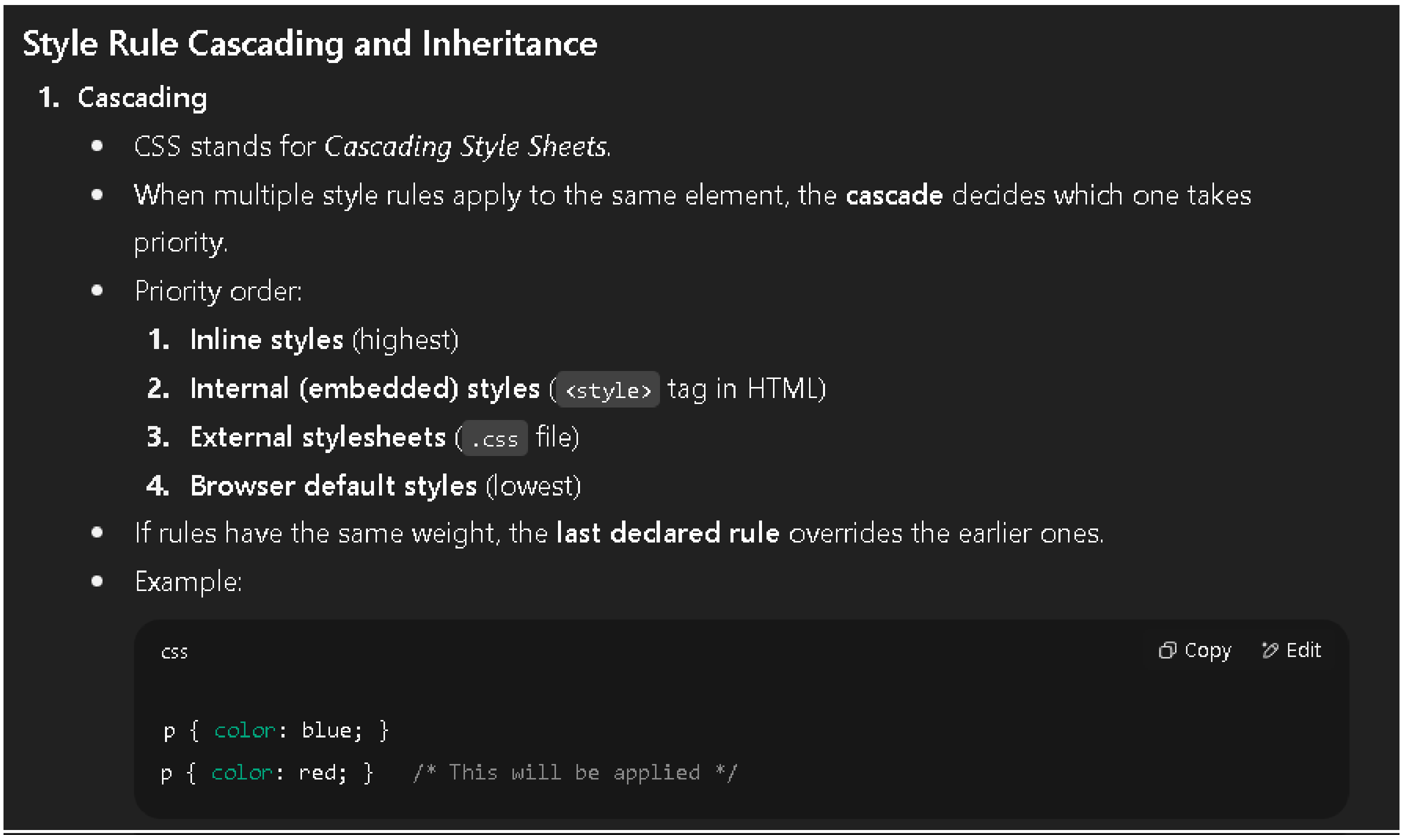
Explanation

1. `rowspan="2"` → "Name" cell covers **two rows vertically**.
2. `colspan="2"` → "Marks" cell covers **two columns horizontally** (Math & Science).

Example of Table with Rowspan & Colspan

Name	Marks	
	Math	Science
Amit	80	90
Riya	85	95

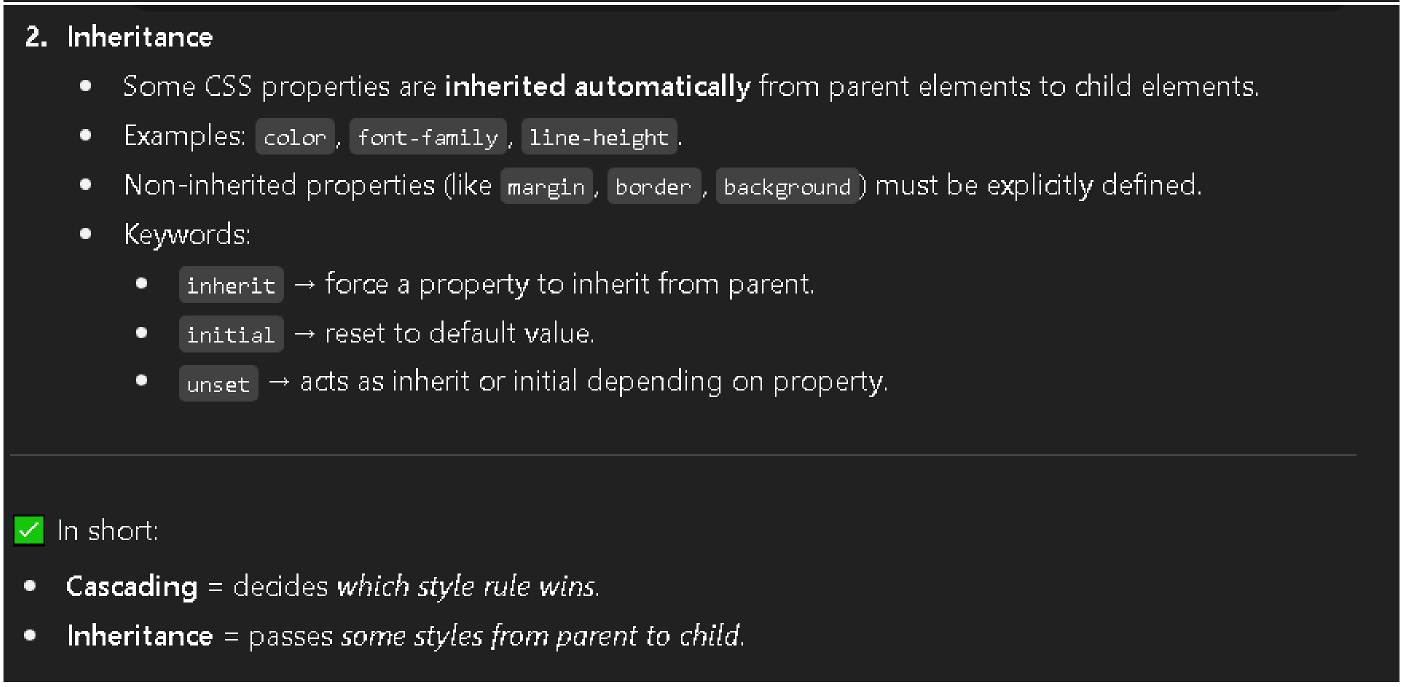
3) Write a short note on Style rule cascading and inheritance.



Style Rule Cascading and Inheritance

1. **Cascading**
   - CSS stands for *Cascading Style Sheets.*
   - When multiple style rules apply to the same element, the **cascade** decides which one takes priority.
   - Priority order:
     1. **Inline styles** (highest)
     2. **Internal (embedded) styles** (`<style>` tag in HTML)
     3. **External stylesheets** (`.css` file)
     4. **Browser default styles** (lowest)
   - If rules have the same weight, the **last declared rule** overrides the earlier ones.
   - Example:

```css
p { color: blue; }
p { color: red; }   /* This will be applied */
```

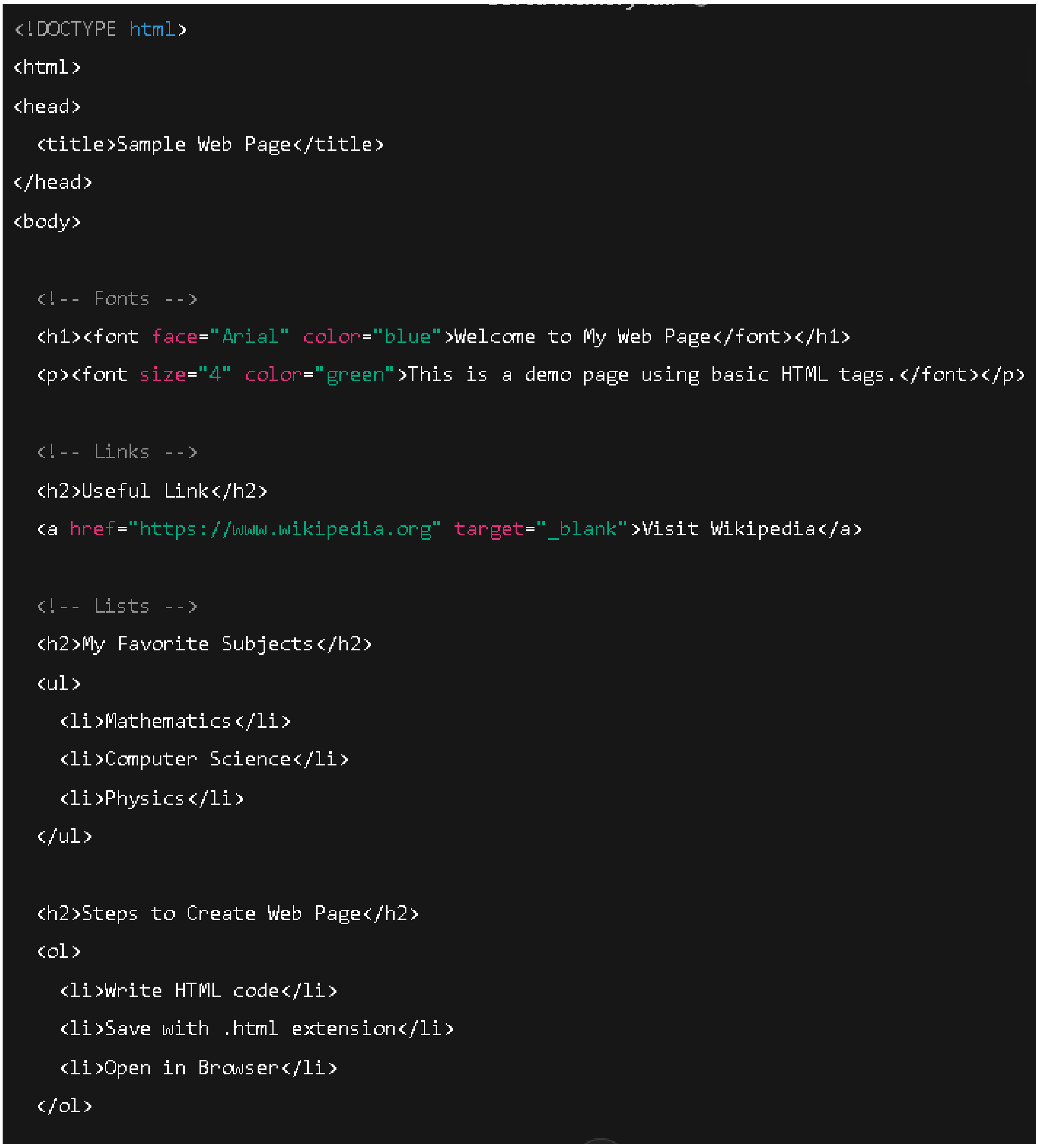


2. **Inheritance**
   - Some CSS properties are **inherited automatically** from parent elements to child elements.
   - Examples: `color`, `font-family`, `line-height`.
   - Non-inherited properties (like `margin`, `border`, `background`) must be explicitly defined.
   - Keywords:
     - `inherit` → force a property to inherit from parent.
     - `initial` → reset to default value.
     - `unset` → acts as inherit or initial depending on property.

✅ In short:

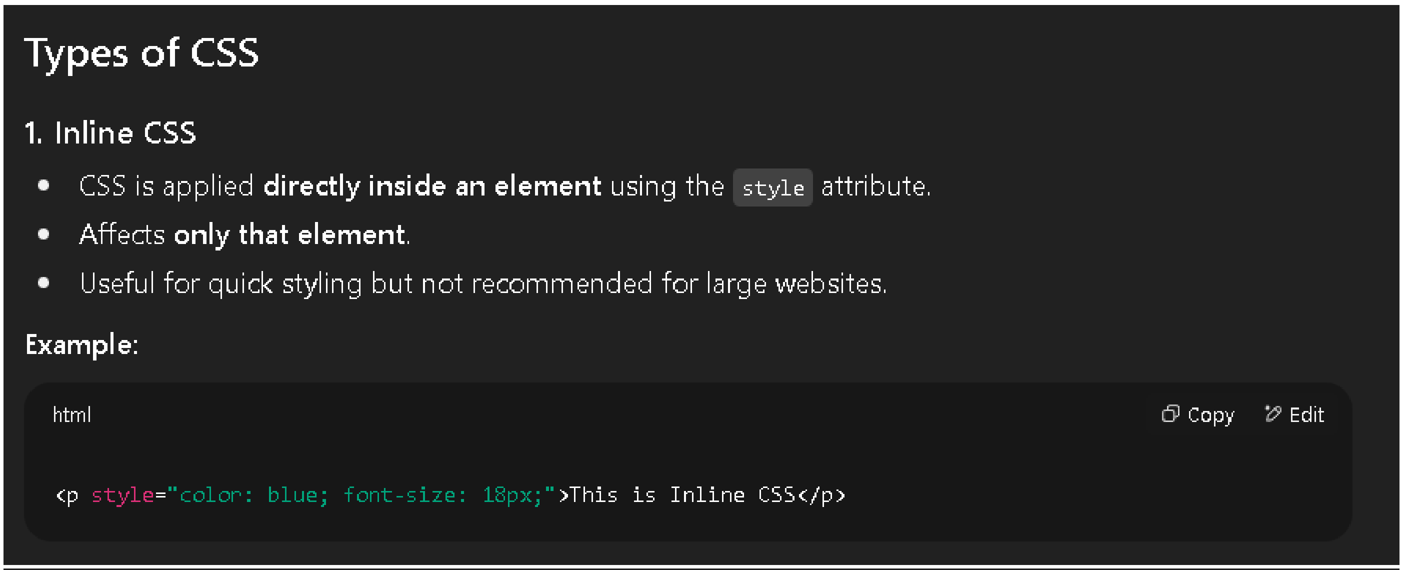
- **Cascading** = decides *which style rule wins.*
- **Inheritance** = passes *some styles from parent to child.*

4) Design any web page by using following tag. fonts, links, frames, lists, tables



```html
<!DOCTYPE html>
<html>
<head>
  <title>Sample Web Page</title>
</head>
<body>

  <!-- Fonts -->
  <h1><font face="Arial" color="blue">Welcome to My Web Page</font></h1>
  <p><font size="4" color="green">This is a demo page using basic HTML tags.</font></p>

  <!-- Links -->
  <h2>Useful Link</h2>
  <a href="https://www.wikipedia.org" target="_blank">Visit Wikipedia</a>

  <!-- Lists -->
  <h2>My Favorite Subjects</h2>
  <ul>
    <li>Mathematics</li>
    <li>Computer Science</li>
    <li>Physics</li>
  </ul>

  <h2>Steps to Create Web Page</h2>
  <ol>
    <li>Write HTML code</li>
    <li>Save with .html extension</li>
    <li>Open in Browser</li>
  </ol>
  <!-- Table -->
  <h2>Student Marks</h2>
  <table border="1" cellpadding="8">
    <tr>
      <th>Name</th>
      <th>Math</th>
      <th>Science</th>
    </tr>
    <tr>
      <td>Amit</td>
      <td>85</td>
      <td>90</td>
    </tr>
    <tr>
      <td>Riya</td>
      <td>88</td>
      <td>92</td>
    </tr>
  </table>

  <!-- Frames -->
  <h2>Frames Example</h2>
  <frameset cols="50%,50%">
    <frame src="https://www.example.com">
    <frame src="https://www.wikipedia.org">
  </frameset>

</body>
</html>
```

5) Explain with suitable example inline, internal and external CSS.



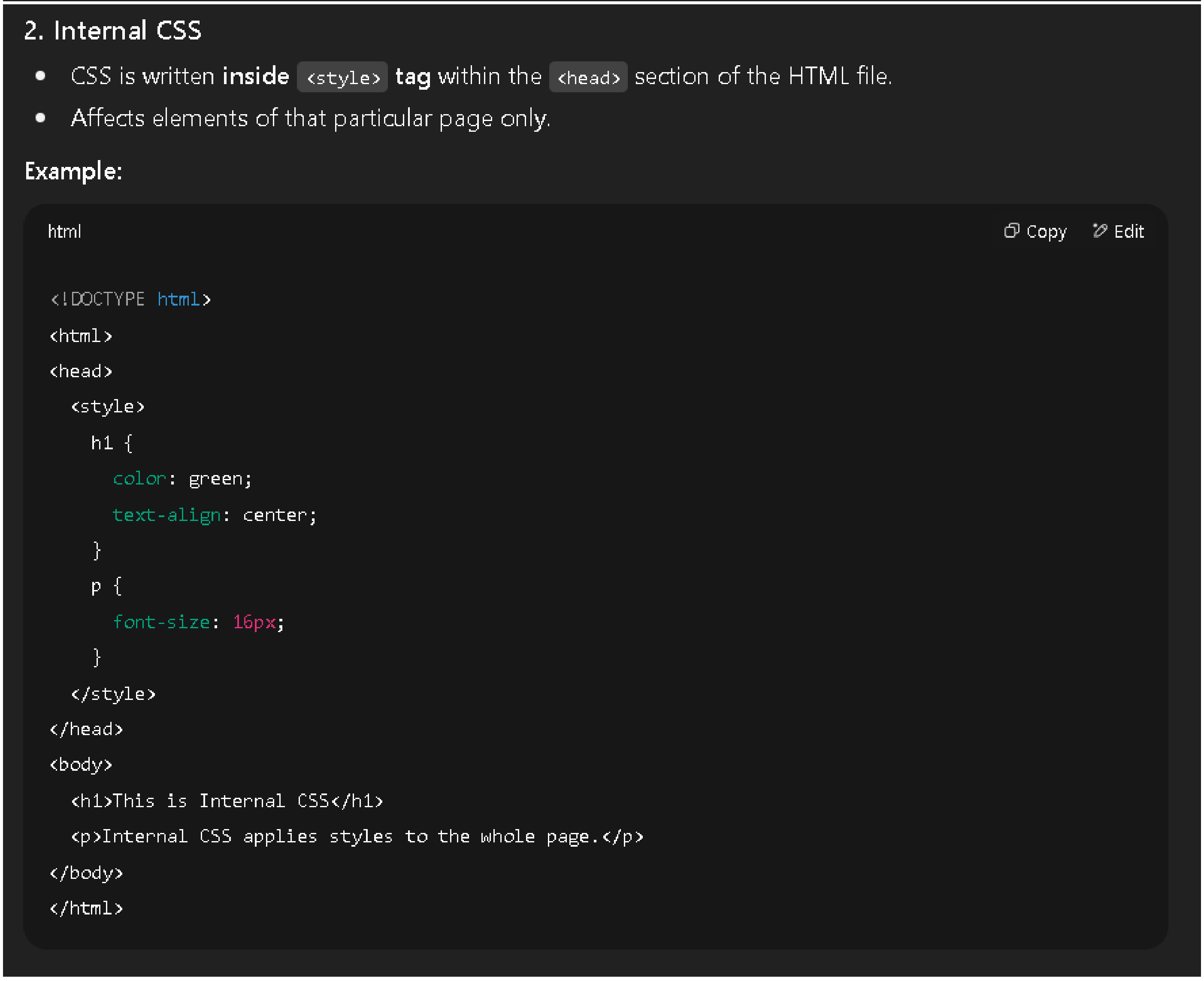
Types of CSS

1. Inline CSS

- CSS is applied **directly inside an element** using the `style` attribute.
- Affects **only that element**.
- Useful for quick styling but not recommended for large websites.

Example:

```html
<p style="color: blue; font-size: 18px;">This is Inline CSS</p>
```



2. Internal CSS

- CSS is written **inside** `<style>` **tag** within the `<head>` section of the HTML file.
- Affects elements of that particular page only.

Example:

```html
<!DOCTYPE html>
<html>
<head>
  <style>
    h1 {
      color: green;
      text-align: center;
    }
    p {
      font-size: 16px;
    }
  </style>
</head>
<body>
  <h1>This is Internal CSS</h1>
  <p>Internal CSS applies styles to the whole page.</p>
</body>
</html>
```

3. External CSS

- CSS rules are written in a **separate .css file**.
- Linked to HTML using `<link>` tag inside `<head>`.
- Useful for **large websites** (same style applied to multiple pages).

Example:

`style.css`

css Copy Edit

```css
h1 {
  color: red;
}
p {
  font-size: 14px;
  color: navy;
}
```

`index.html`

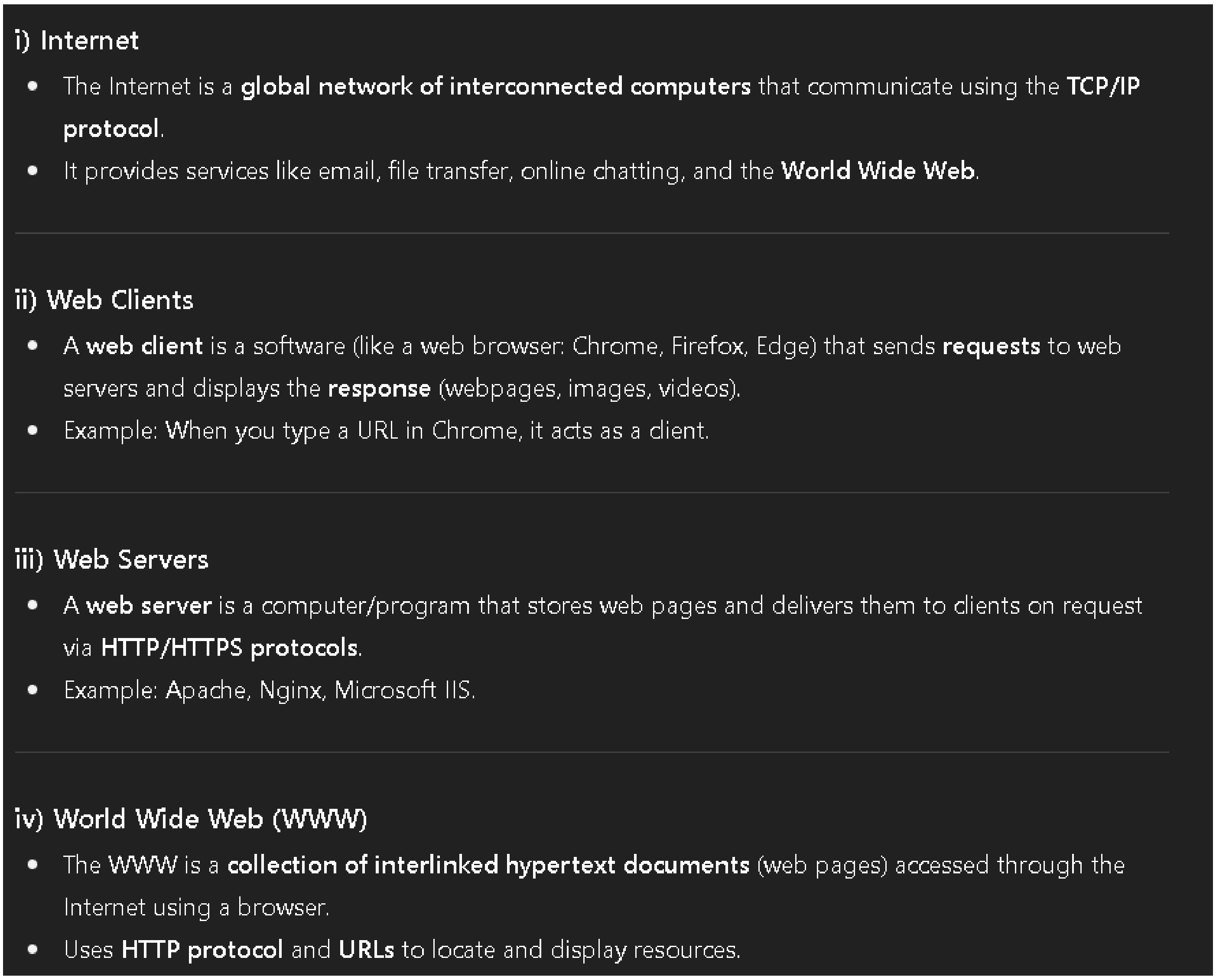
html Copy Edit

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <h1>This is External CSS</h1>
  <p>External CSS keeps design separate from content.</p>
</body>
</html>
```

✅ In short (exam note):

- **Inline CSS** → inside `style` attribute (element-specific).
- **Internal CSS** → inside `<style>` tag (page-specific).
- **External CSS** → in a `.css` file linked via `<link>` (site-wide, reusable).

6) Define following terms i) Internet ii) Web clients iii) Web Servers iv) World Wide Web



i) Internet

- The Internet is a **global network of interconnected computers** that communicate using the **TCP/IP protocol**.
- It provides services like email, file transfer, online chatting, and the **World Wide Web**.

---

ii) Web Clients

- A **web client** is a software (like a web browser: Chrome, Firefox, Edge) that sends **requests** to web servers and displays the **response** (webpages, images, videos).
- Example: When you type a URL in Chrome, it acts as a client.

---

iii) Web Servers

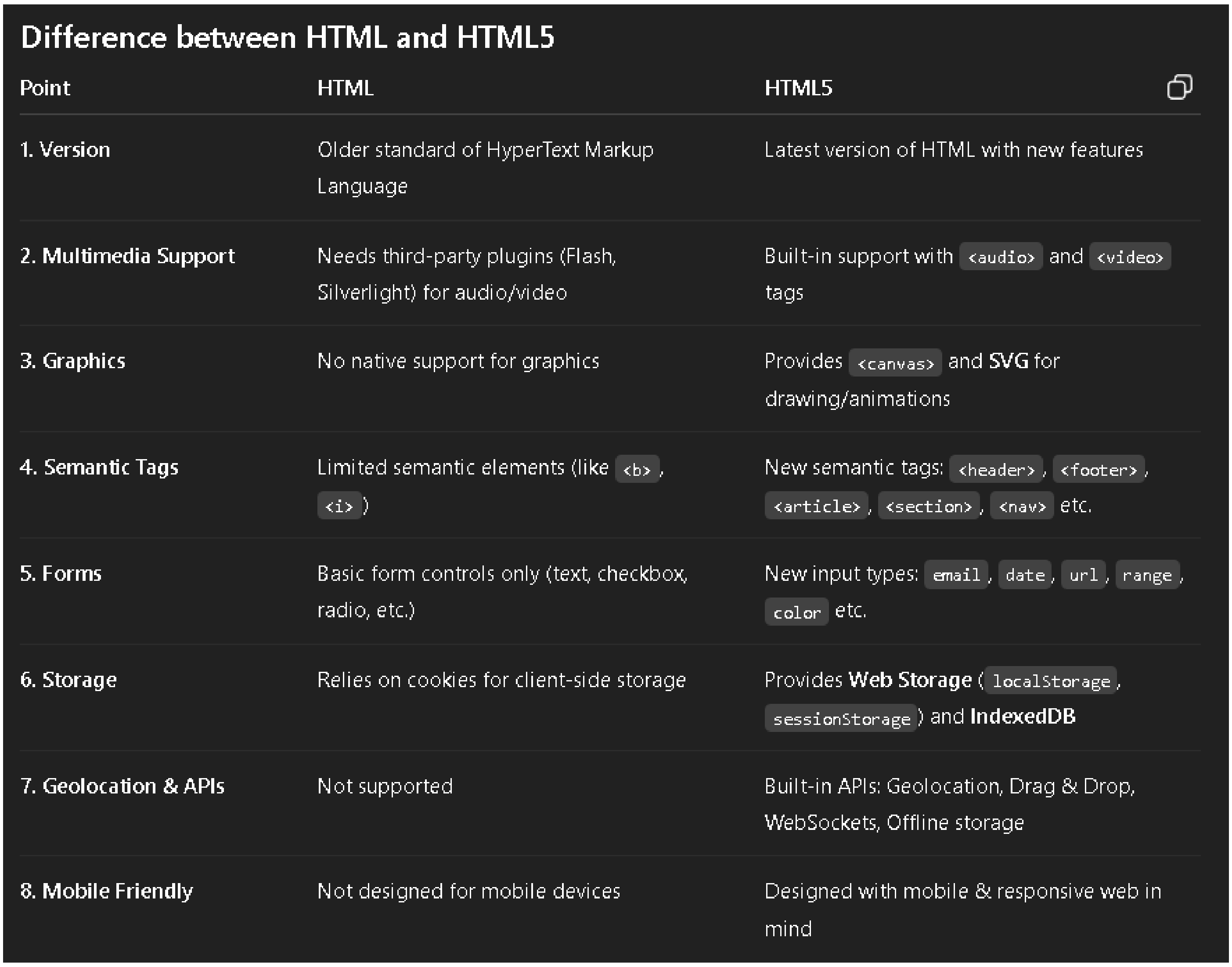
- A **web server** is a computer/program that stores web pages and delivers them to clients on request via **HTTP/HTTPS protocols**.
- Example: Apache, Nginx, Microsoft IIS.

---

iv) World Wide Web (WWW)

- The WWW is a **collection of interlinked hypertext documents** (web pages) accessed through the Internet using a browser.
- Uses **HTTP protocol** and **URLs** to locate and display resources.

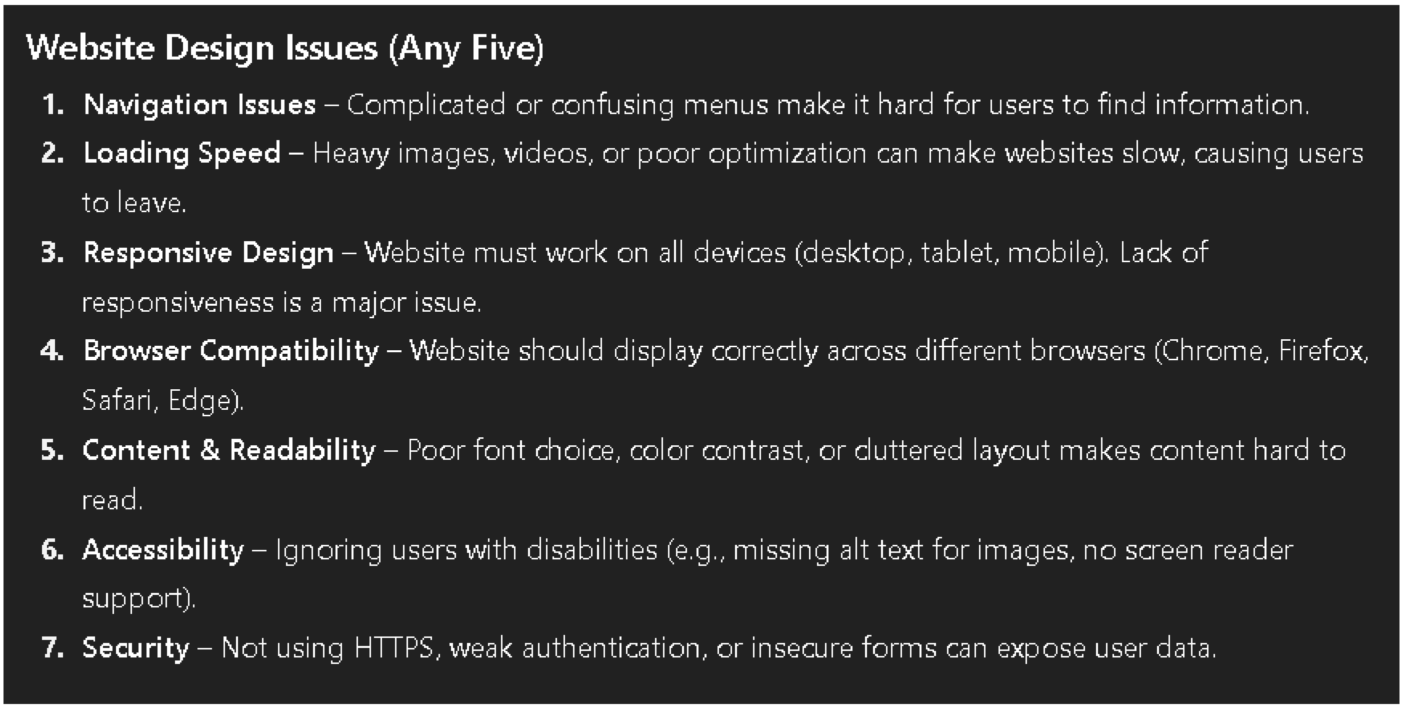
7) Write difference between HTML and HTML5



Difference between HTML and HTML5

Point	HTML	HTML5
1. Version	Older standard of HyperText Markup Language	Latest version of HTML with new features
2. Multimedia Support	Needs third-party plugins (Flash, Silverlight) for audio/video	Built-in support with `<audio>` and `<video>` tags
3. Graphics	No native support for graphics	Provides `<canvas>` and **SVG** for drawing/animations
4. Semantic Tags	Limited semantic elements (like `<b>`, `<i>`)	New semantic tags: `<header>`, `<footer>`, `<article>`, `<section>`, `<nav>` etc.
5. Forms	Basic form controls only (text, checkbox, radio, etc.)	New input types: `email`, `date`, `url`, `range`, `color` etc.
6. Storage	Relies on cookies for client-side storage	Provides **Web Storage** (`localStorage`, `sessionStorage`) and **IndexedDB**
7. Geolocation & APIs	Not supported	Built-in APIs: Geolocation, Drag & Drop, WebSockets, Offline storage
8. Mobile Friendly	Not designed for mobile devices	Designed with mobile & responsive web in mind

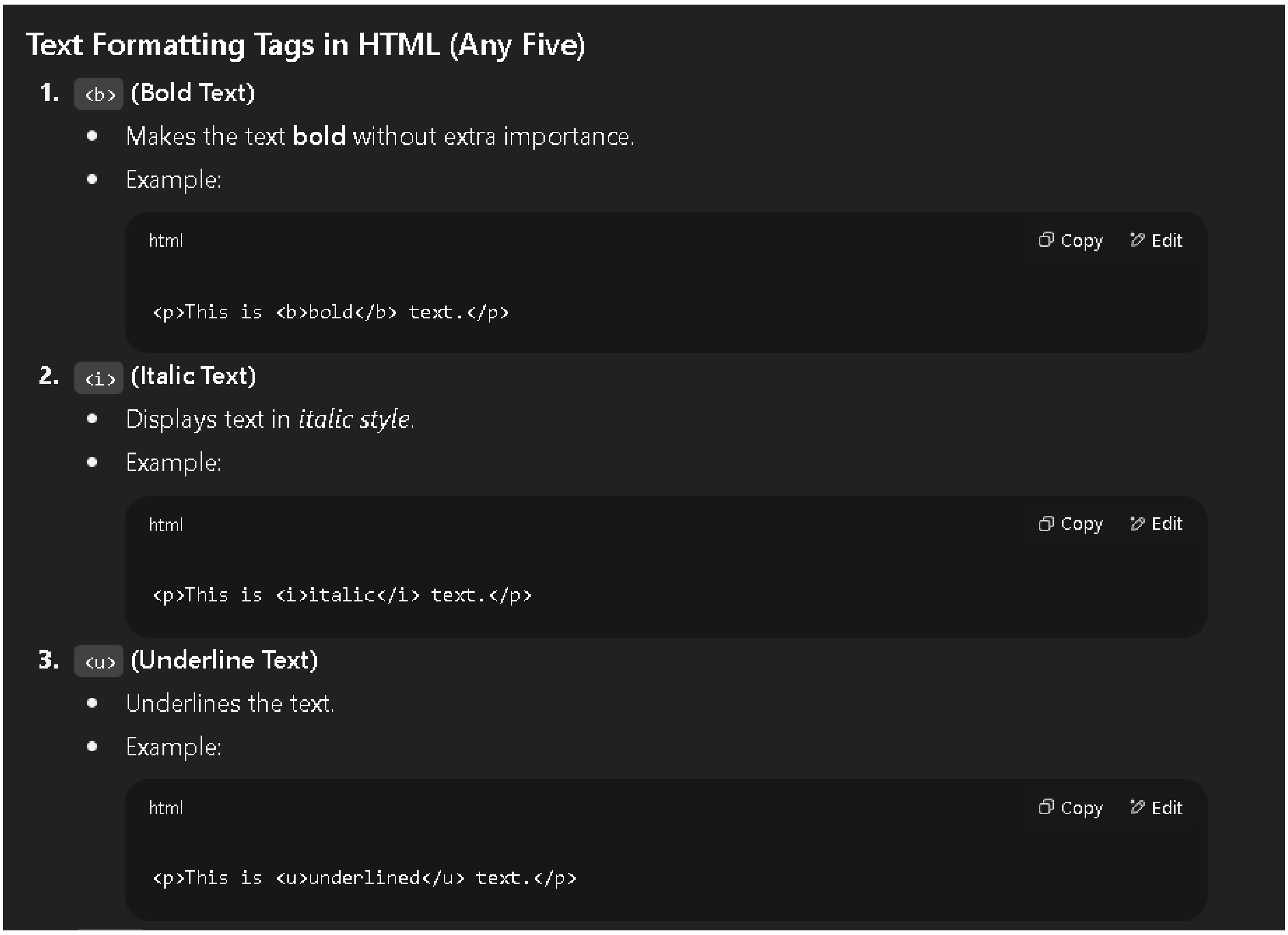
8) What are the different website design issues? (Any five)



Website Design Issues (Any Five)

1. **Navigation Issues** – Complicated or confusing menus make it hard for users to find information.
2. **Loading Speed** – Heavy images, videos, or poor optimization can make websites slow, causing users to leave.
3. **Responsive Design** – Website must work on all devices (desktop, tablet, mobile). Lack of responsiveness is a major issue.
4. **Browser Compatibility** – Website should display correctly across different browsers (Chrome, Firefox, Safari, Edge).
5. **Content & Readability** – Poor font choice, color contrast, or cluttered layout makes content hard to read.
6. **Accessibility** – Ignoring users with disabilities (e.g., missing alt text for images, no screen reader support).
7. **Security** – Not using HTTPS, weak authentication, or insecure forms can expose user data.

9) Write and explain text formatting tags in HTML. (Any five)



Text Formatting Tags in HTML (Any Five)

1. **`<b>` (Bold Text)**
   - Makes the text **bold** without extra importance.
   - Example:

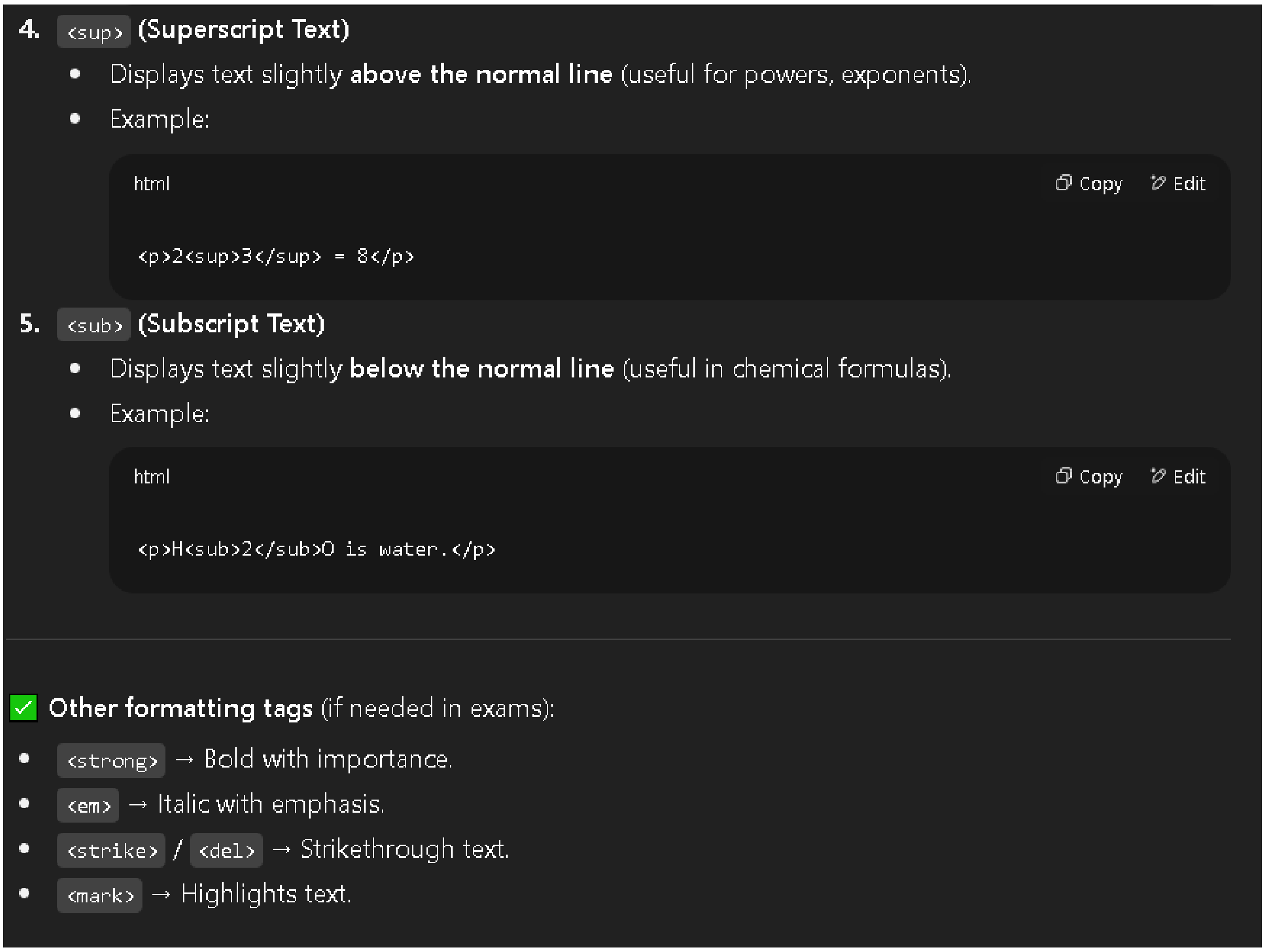
```html
<p>This is <b>bold</b> text.</p>
```

2. **`<i>` (Italic Text)**
   - Displays text in *italic style*.
   - Example:

```html
<p>This is <i>italic</i> text.</p>
```

3. **`<u>` (Underline Text)**
   - Underlines the text.
   - Example:

```html
<p>This is <u>underlined</u> text.</p>
```



4. `<sup>` **(Superscript Text)**
   - Displays text slightly **above the normal line** (useful for powers, exponents).
   - Example:

```html
<p>2<sup>3</sup> = 8</p>
```

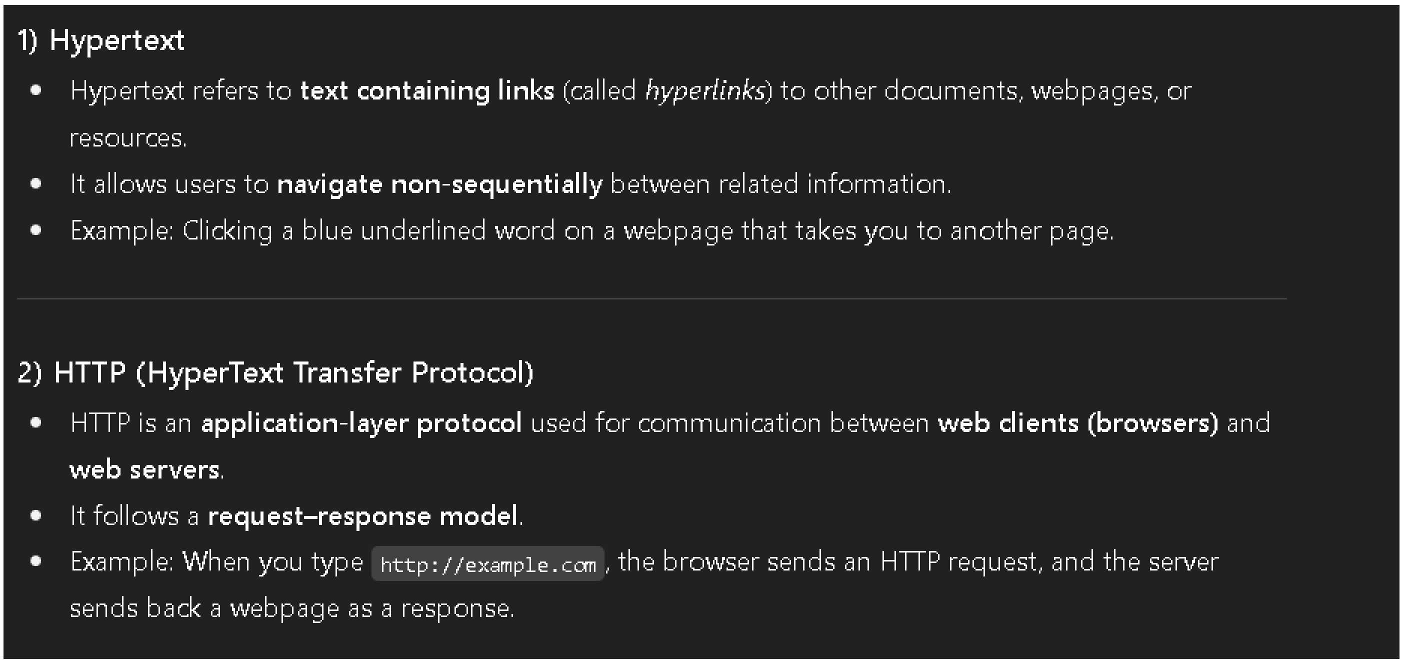
5. `<sub>` **(Subscript Text)**
   - Displays text slightly **below the normal line** (useful in chemical formulas).
   - Example:

```html
<p>H<sub>2</sub>O is water.</p>
```

✅ **Other formatting tags** (if needed in exams):

- `<strong>` → Bold with importance.
- `<em>` → Italic with emphasis.
- `<strike>` / `<del>` → Strikethrough text.
- `<mark>` → Highlights text.

10) Explain the terms , Hypertext, HTTP, URL.



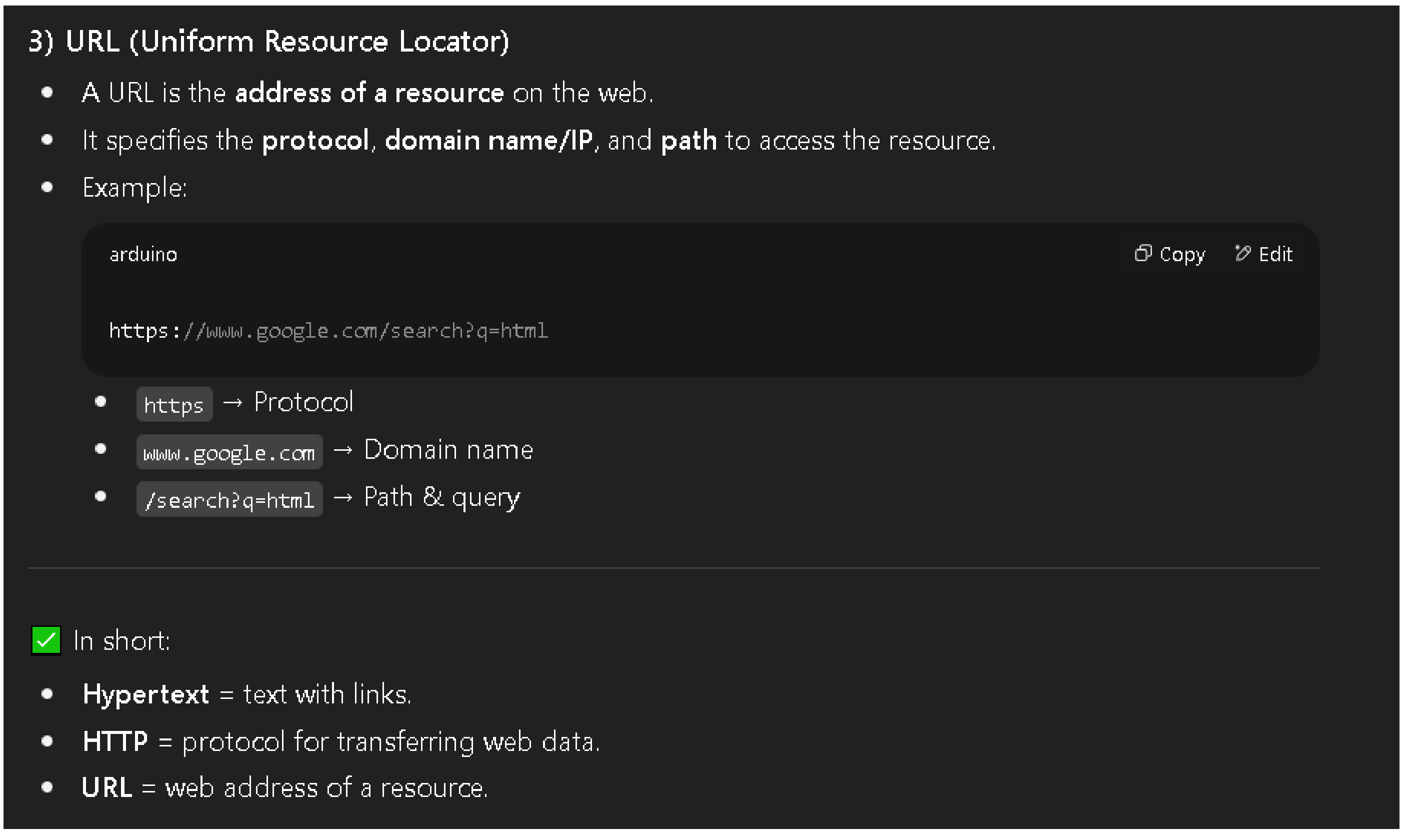
1) Hypertext

- Hypertext refers to **text containing links** (called *hyperlinks*) to other documents, webpages, or resources.
- It allows users to **navigate non-sequentially** between related information.
- Example: Clicking a blue underlined word on a webpage that takes you to another page.

---

2) HTTP (HyperText Transfer Protocol)

- HTTP is an **application-layer protocol** used for communication between **web clients (browsers)** and **web servers**.
- It follows a **request–response model**.
- Example: When you type `http://example.com`, the browser sends an HTTP request, and the server sends back a webpage as a response.



3) URL (Uniform Resource Locator)

- A URL is the **address of a resource** on the web.
- It specifies the **protocol**, **domain name/IP**, and **path** to access the resource.
- Example:

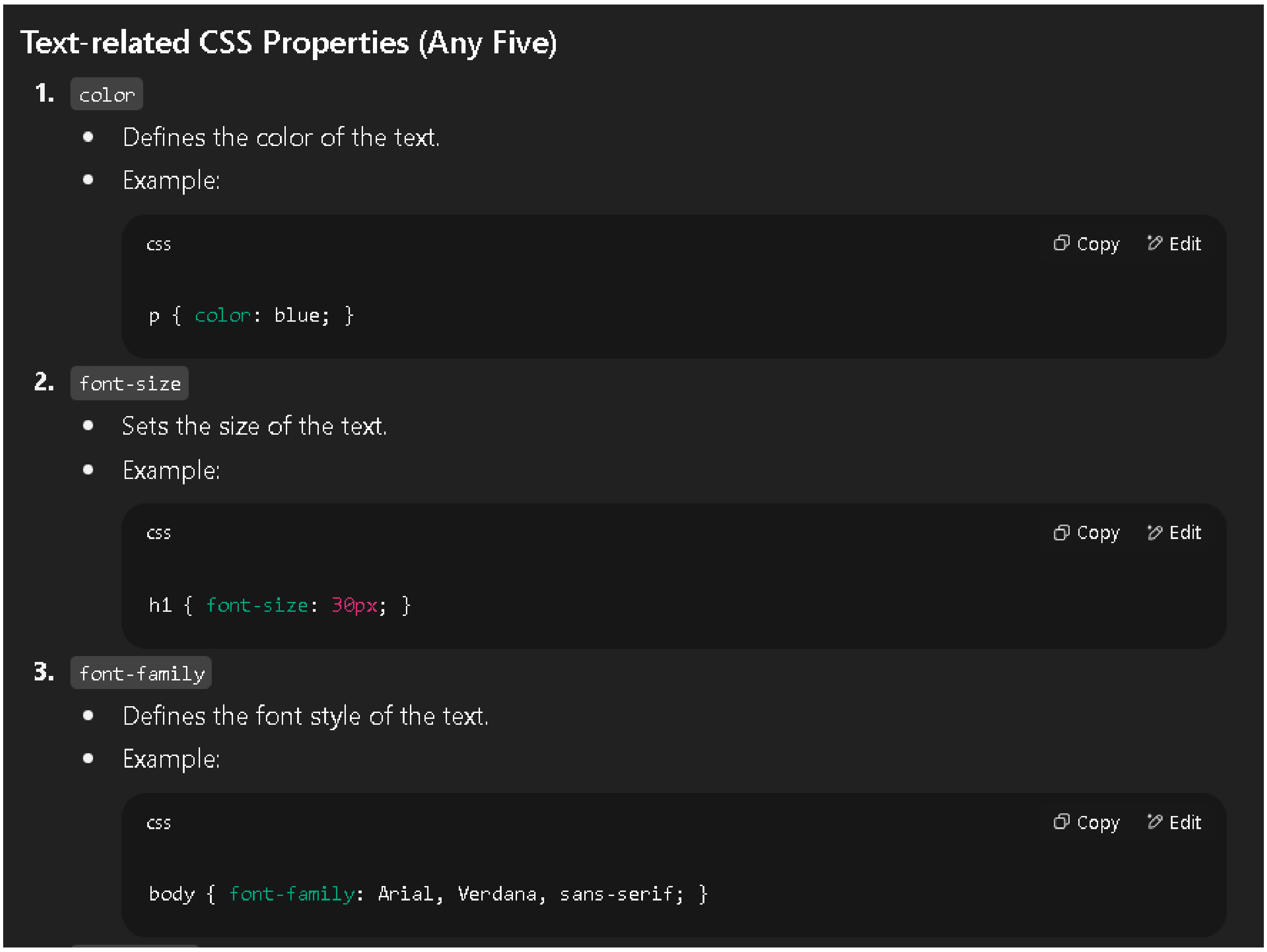
```arduino
https://www.google.com/search?q=html
```

  - `https` → Protocol
  - `www.google.com` → Domain name
  - `/search?q=html` → Path & query

✅ In short:

- **Hypertext** = text with links.
- **HTTP** = protocol for transferring web data.
- **URL** = web address of a resource.

11) Write and explain text related CSS properties. (Any five)



Text-related CSS Properties (Any Five)

1. `color`
   - Defines the color of the text.
   - Example:

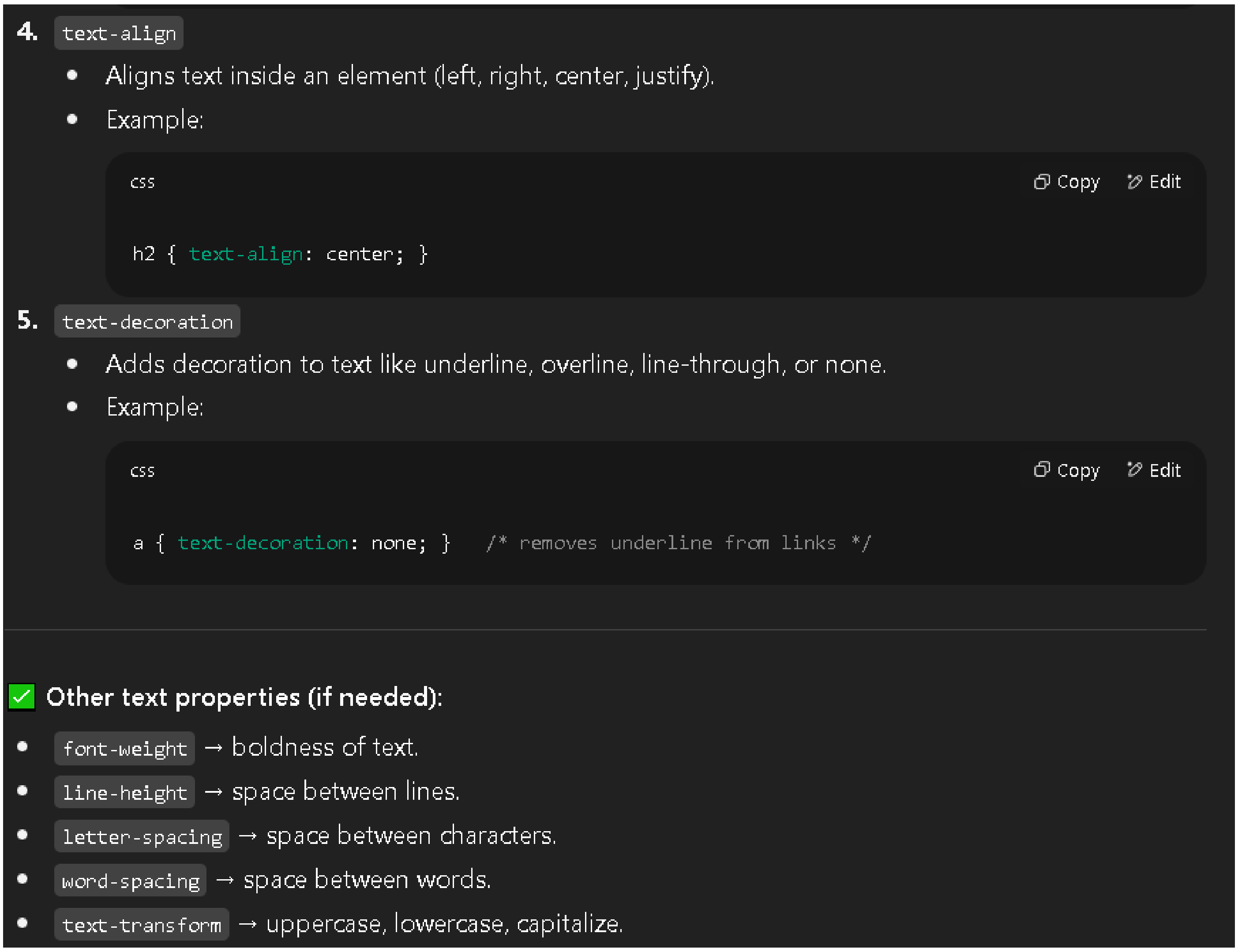
```css
p { color: blue; }
```

2. `font-size`
   - Sets the size of the text.
   - Example:

```css
h1 { font-size: 30px; }
```

3. `font-family`
   - Defines the font style of the text.
   - Example:

```css
body { font-family: Arial, Verdana, sans-serif; }
```



4. `text-align`
   - Aligns text inside an element (left, right, center, justify).
   - Example:

```css
h2 { text-align: center; }
```

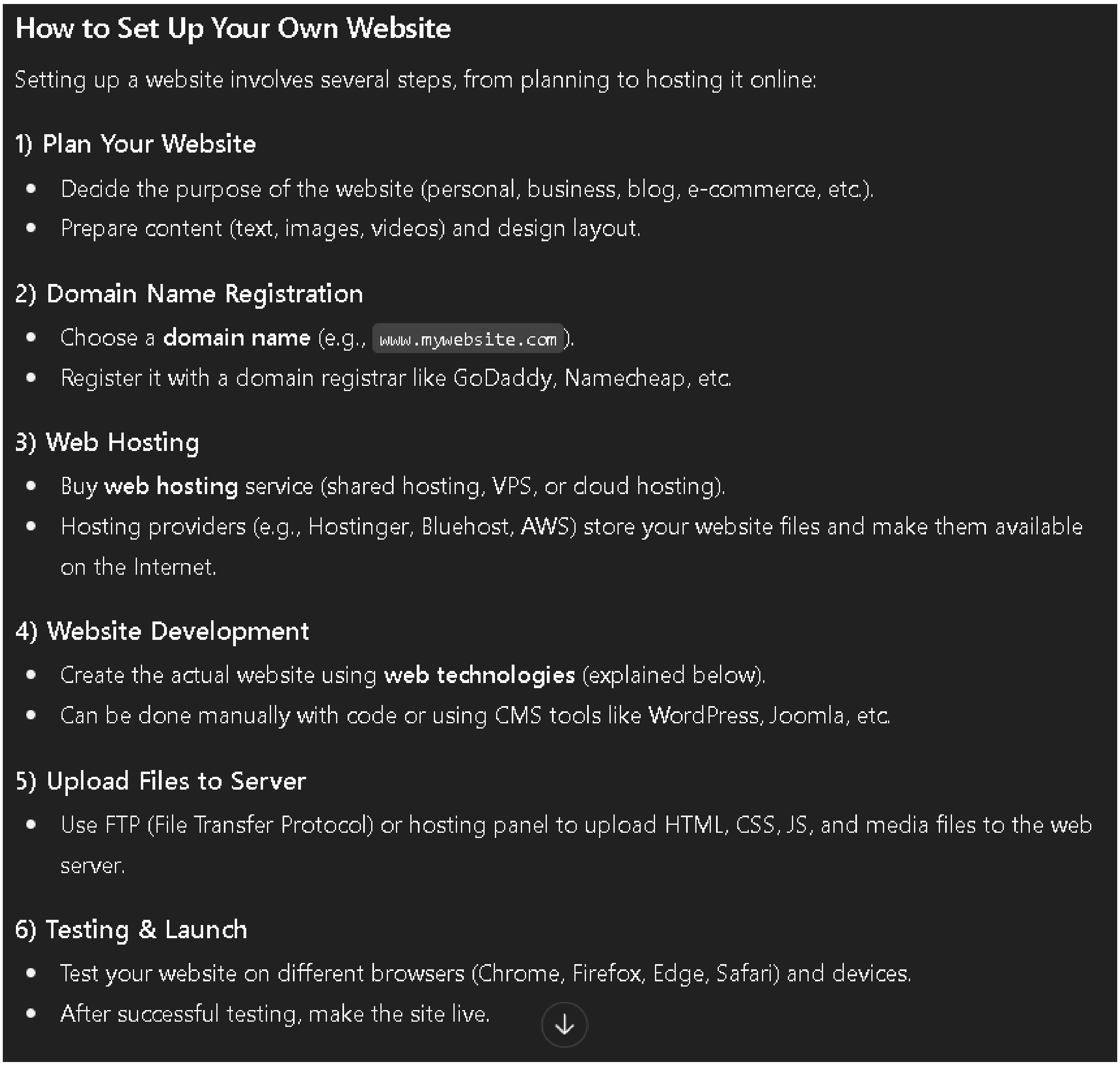
5. `text-decoration`
   - Adds decoration to text like underline, overline, line-through, or none.
   - Example:

```css
a { text-decoration: none; }   /* removes underline from links */
```

✅ Other text properties (if needed):

- `font-weight` → boldness of text.
- `line-height` → space between lines.
- `letter-spacing` → space between characters.
- `word-spacing` → space between words.
- `text-transform` → uppercase, lowercase, capitalize.

12) Elaborate how you can set up your own website. What are the web technologies are required for the same?



How to Set Up Your Own Website

Setting up a website involves several steps, from planning to hosting it online:

1) Plan Your Website

- Decide the purpose of the website (personal, business, blog, e-commerce, etc.).
- Prepare content (text, images, videos) and design layout.

2) Domain Name Registration

- Choose a **domain name** (e.g., `www.mywebsite.com`).
- Register it with a domain registrar like GoDaddy, Namecheap, etc.

3) Web Hosting

- Buy **web hosting** service (shared hosting, VPS, or cloud hosting).
- Hosting providers (e.g., Hostinger, Bluehost, AWS) store your website files and make them available on the Internet.

4) Website Development

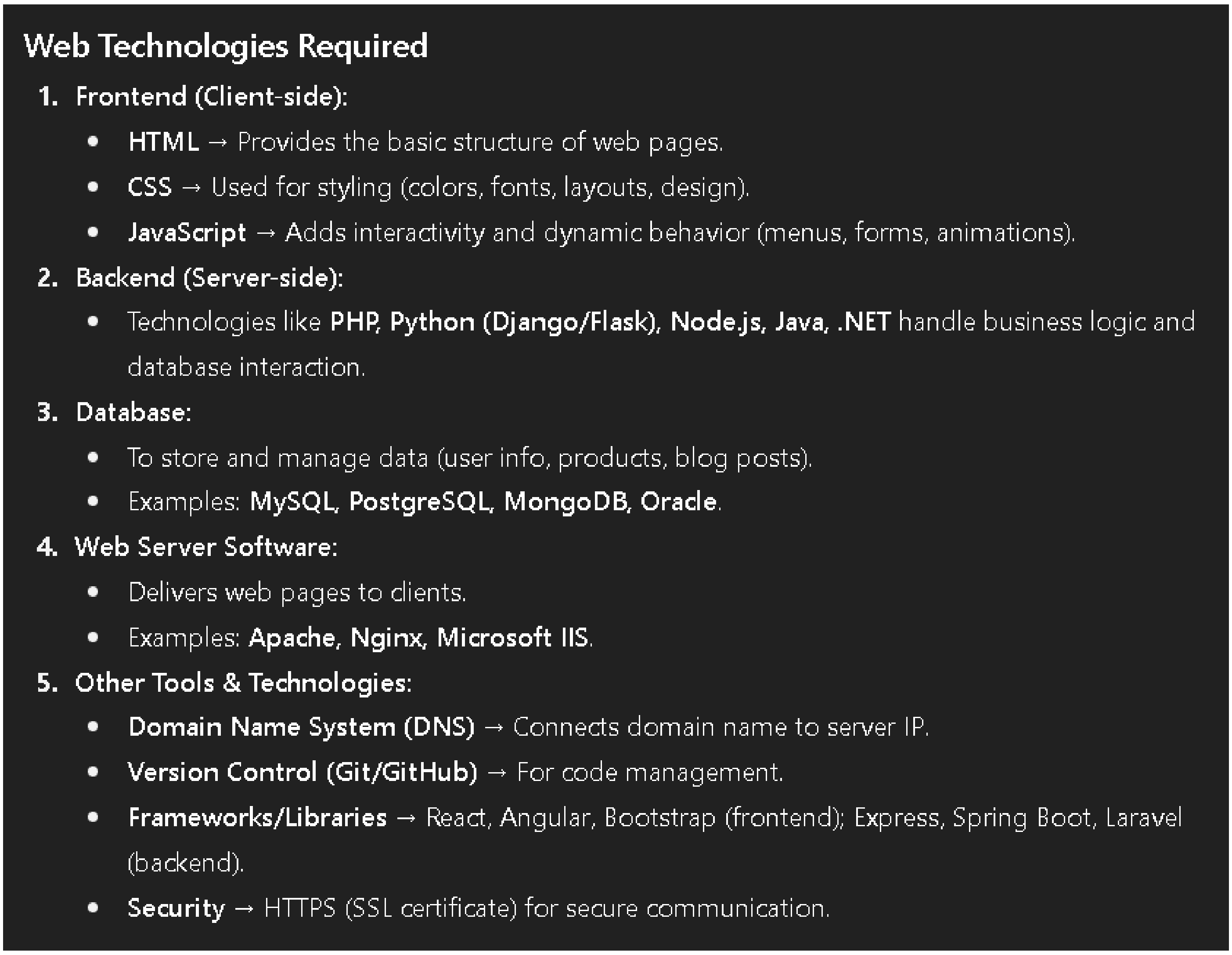
- Create the actual website using **web technologies** (explained below).
- Can be done manually with code or using CMS tools like WordPress, Joomla, etc.

5) Upload Files to Server

- Use FTP (File Transfer Protocol) or hosting panel to upload HTML, CSS, JS, and media files to the web server.

6) Testing & Launch

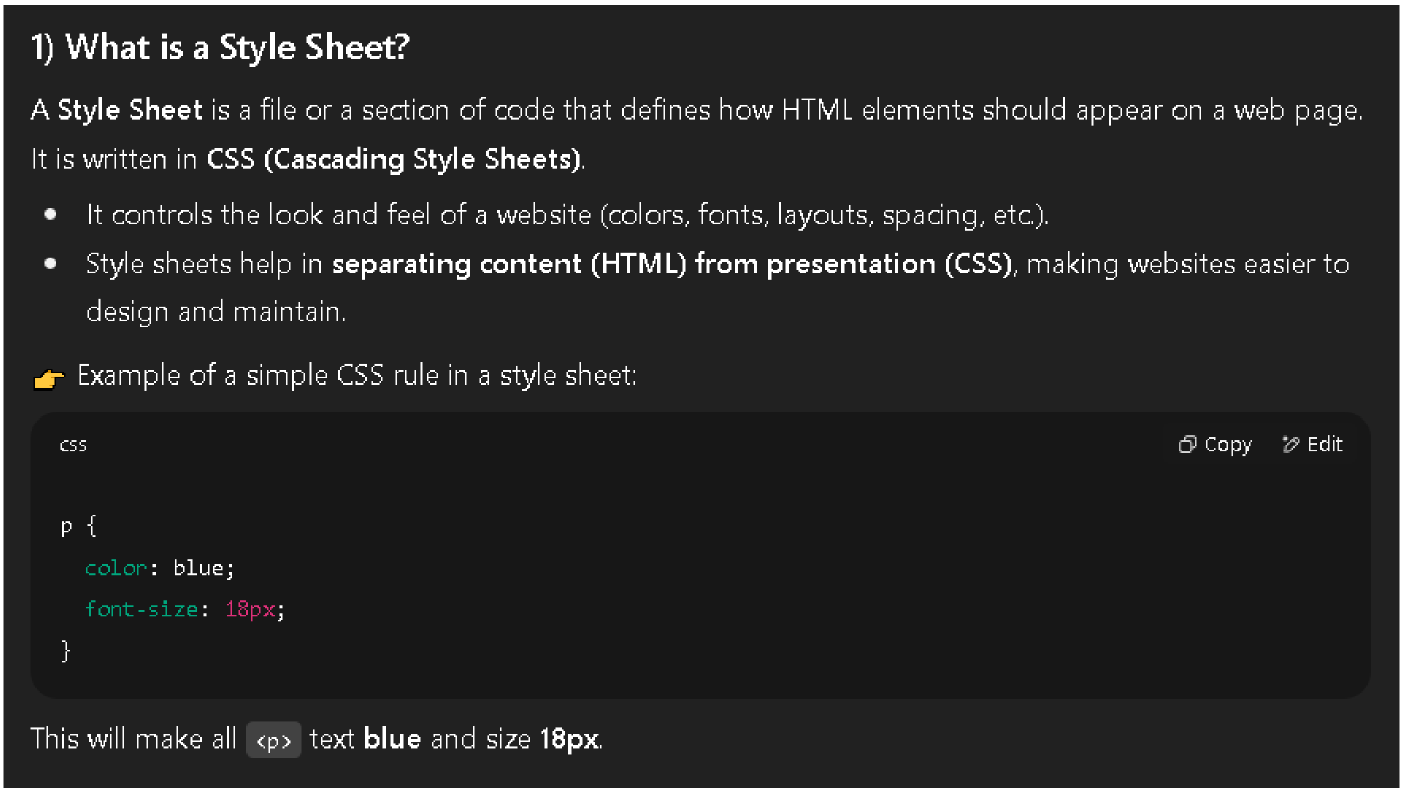
- Test your website on different browsers (Chrome, Firefox, Edge, Safari) and devices.
- After successful testing, make the site live.



Web Technologies Required

1. **Frontend (Client-side):**
   - **HTML** → Provides the basic structure of web pages.
   - **CSS** → Used for styling (colors, fonts, layouts, design).
   - **JavaScript** → Adds interactivity and dynamic behavior (menus, forms, animations).
2. **Backend (Server-side):**
   - Technologies like **PHP, Python (Django/Flask), Node.js, Java, .NET** handle business logic and database interaction.
3. **Database:**
   - To store and manage data (user info, products, blog posts).
   - Examples: **MySQL, PostgreSQL, MongoDB, Oracle**.
4. **Web Server Software:**
   - Delivers web pages to clients.
   - Examples: **Apache, Nginx, Microsoft IIS**.
5. **Other Tools & Technologies:**
   - **Domain Name System (DNS)** → Connects domain name to server IP.
   - **Version Control (Git/GitHub)** → For code management.
   - **Frameworks/Libraries** → React, Angular, Bootstrap (frontend); Express, Spring Boot, Laravel (backend).
   - **Security** → HTTPS (SSL certificate) for secure communication.

13) What is a style sheet? How to create a multi colour text on a web page using CSS?



1) What is a Style Sheet?

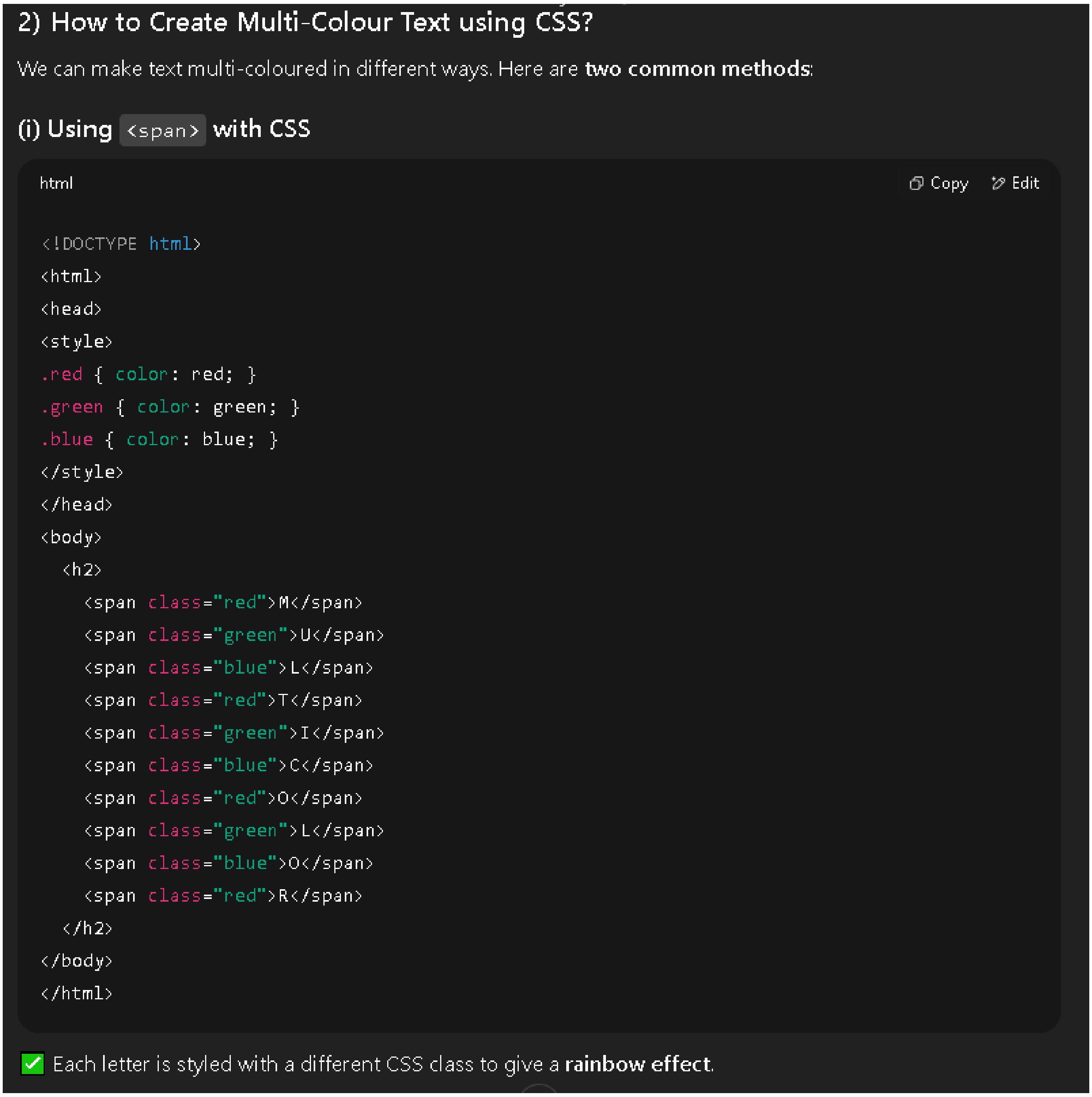
A **Style Sheet** is a file or a section of code that defines how HTML elements should appear on a web page. It is written in **CSS (Cascading Style Sheets)**.

- It controls the look and feel of a website (colors, fonts, layouts, spacing, etc.).
- Style sheets help in **separating content (HTML) from presentation (CSS)**, making websites easier to design and maintain.

👉 Example of a simple CSS rule in a style sheet:

```css
p {
  color: blue;
  font-size: 18px;
}
```

This will make all `<p>` text **blue** and size **18px**.



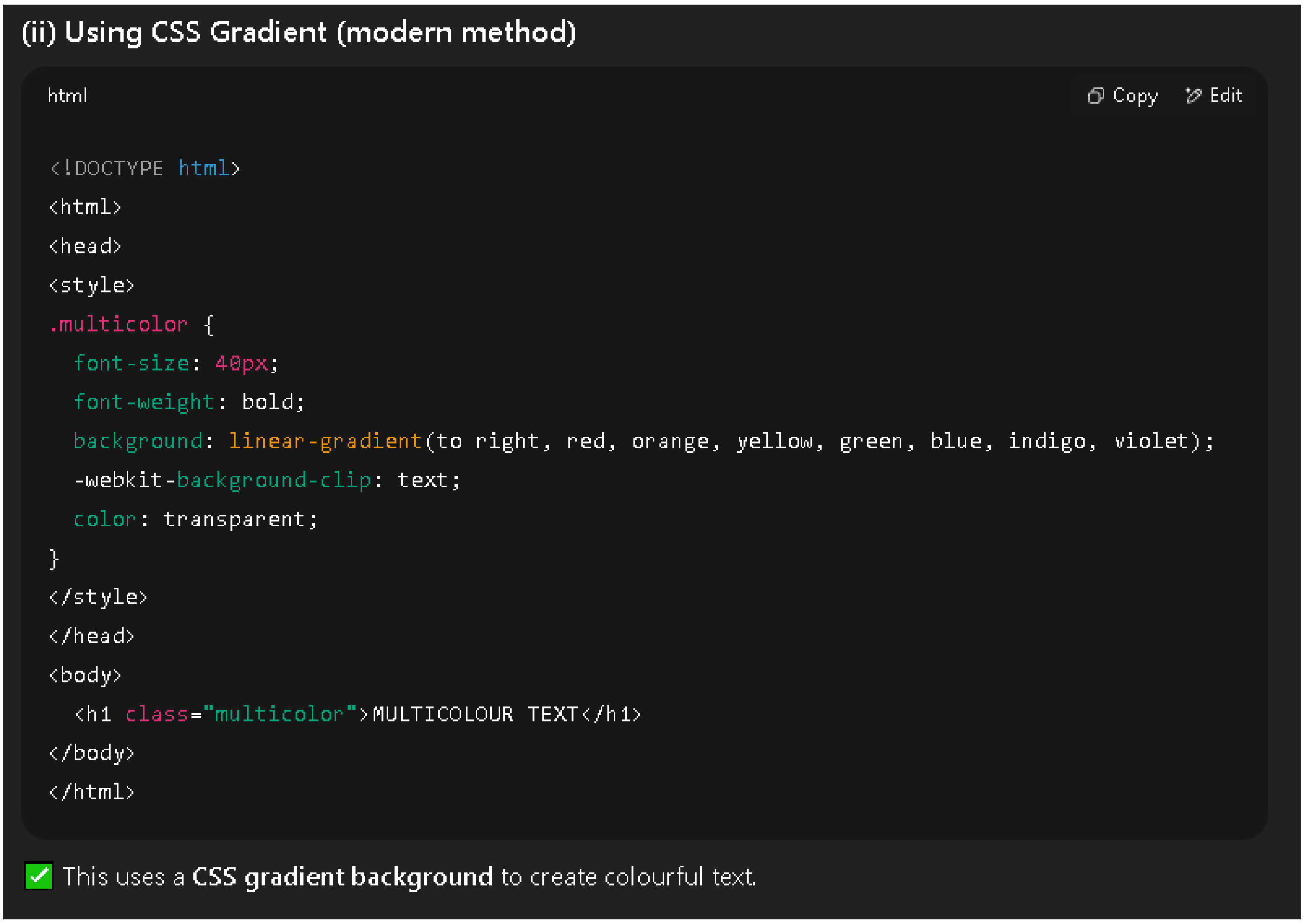
2) How to Create Multi-Colour Text using CSS?

We can make text multi-coloured in different ways. Here are **two common methods**:

(i) Using `<span>` with CSS

```html
<!DOCTYPE html>
<html>
<head>
<style>
.red { color: red; }
.green { color: green; }
.blue { color: blue; }
</style>
</head>
<body>
  <h2>
    <span class="red">M</span>
    <span class="green">U</span>
    <span class="blue">L</span>
    <span class="red">T</span>
    <span class="green">I</span>
    <span class="blue">C</span>
    <span class="red">O</span>
    <span class="green">L</span>
    <span class="blue">O</span>
    <span class="red">R</span>
  </h2>
</body>
</html>
```

✅ Each letter is styled with a different CSS class to give a **rainbow effect**.

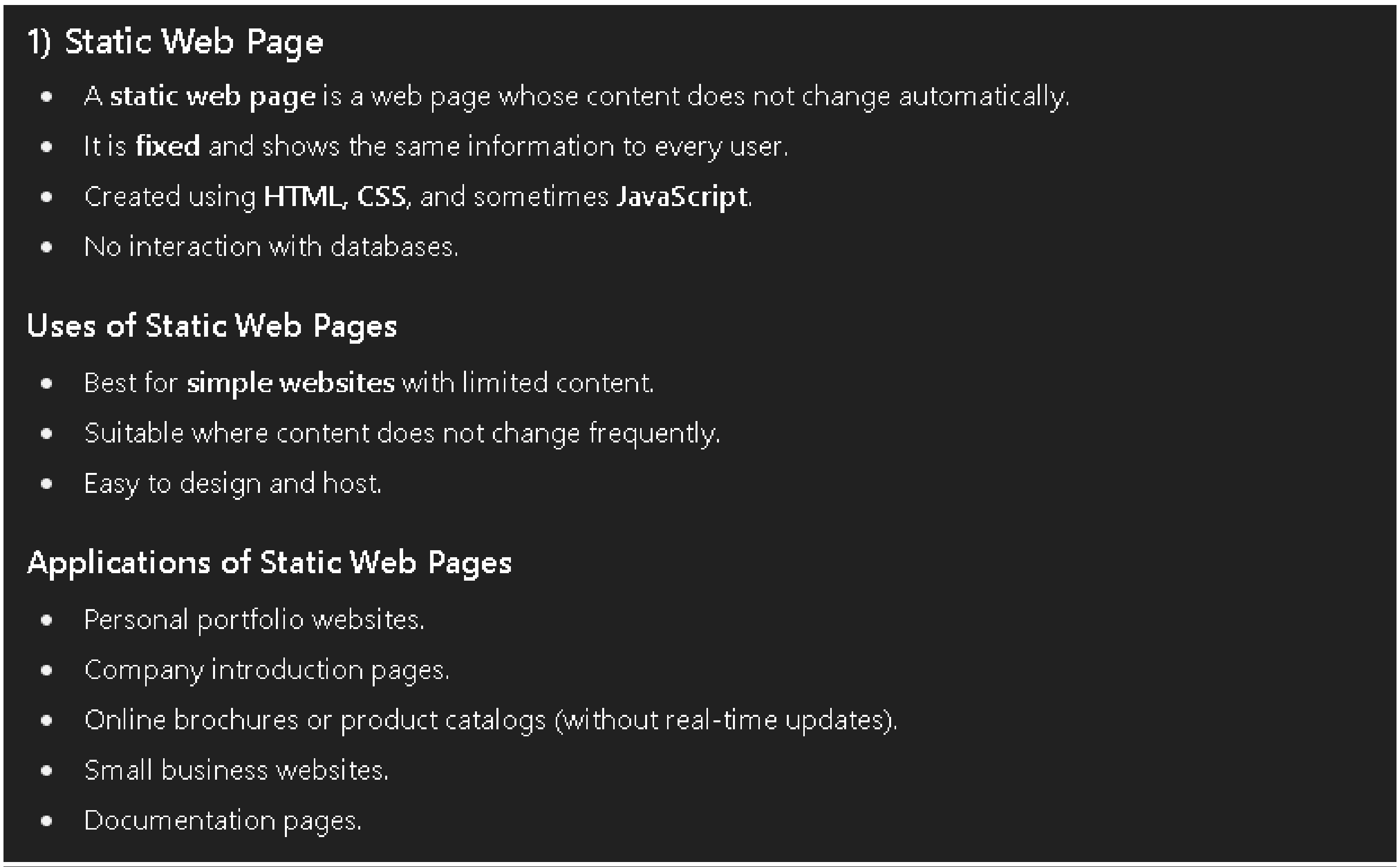


(ii) Using CSS Gradient (modern method)

```html
<!DOCTYPE html>
<html>
<head>
<style>
.multicolor {
  font-size: 40px;
  font-weight: bold;
  background: linear-gradient(to right, red, orange, yellow, green, blue, indigo, violet);
  -webkit-background-clip: text;
  color: transparent;
}
</style>
</head>
<body>
  <h1 class="multicolor">MULTICOLOUR TEXT</h1>
</body>
</html>
```

✅ This uses a **CSS gradient background** to create colourful text.

14) What is use of static & dynamic web page? Enlist applications of static& dynamic web page.



1) Static Web Page

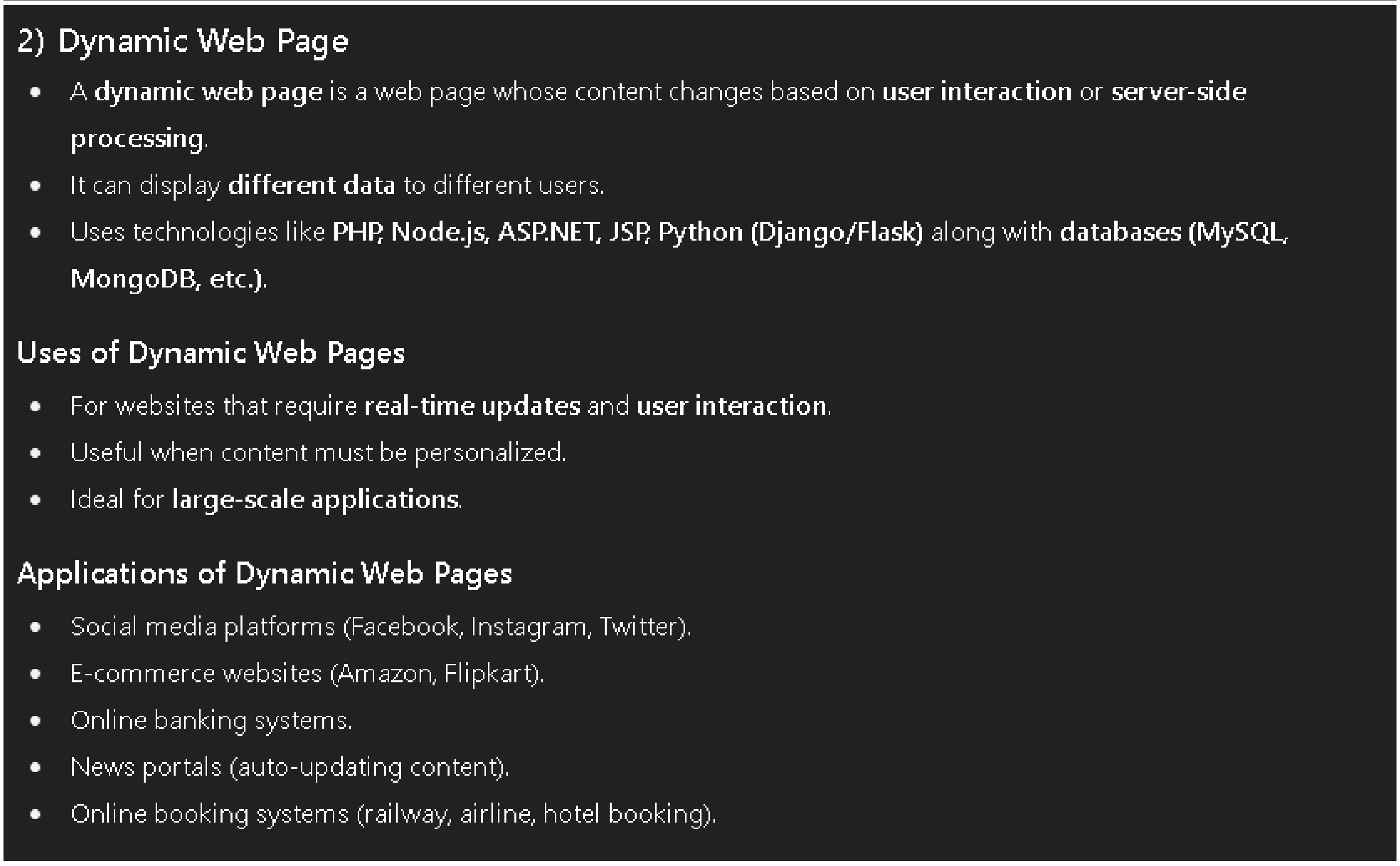
- A **static web page** is a web page whose content does not change automatically.
- It is **fixed** and shows the same information to every user.
- Created using **HTML, CSS**, and sometimes **JavaScript**.
- No interaction with databases.

Uses of Static Web Pages

- Best for **simple websites** with limited content.
- Suitable where content does not change frequently.
- Easy to design and host.

Applications of Static Web Pages

- Personal portfolio websites.
- Company introduction pages.
- Online brochures or product catalogs (without real-time updates).
- Small business websites.
- Documentation pages.



2) Dynamic Web Page

- A **dynamic web page** is a web page whose content changes based on **user interaction** or **server-side processing**.
- It can display **different data** to different users.
- Uses technologies like **PHP, Node.js, ASP.NET, JSP, Python (Django/Flask)** along with **databases (MySQL, MongoDB, etc.)**.

Uses of Dynamic Web Pages

- For websites that require **real-time updates** and **user interaction**.
- Useful when content must be personalized.
- Ideal for **large-scale applications**.

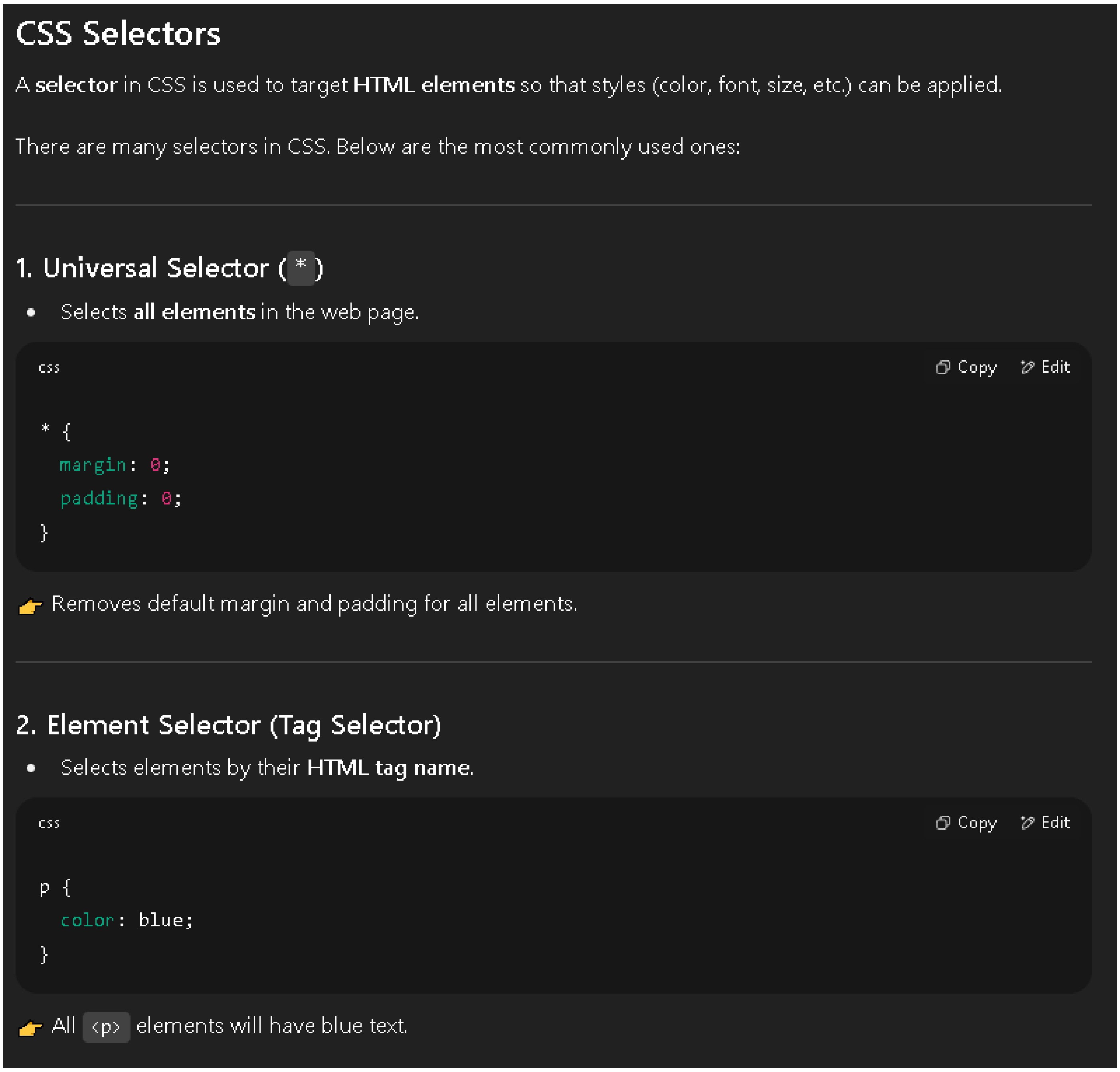
Applications of Dynamic Web Pages

- Social media platforms (Facebook, Instagram, Twitter).
- E-commerce websites (Amazon, Flipkart).
- Online banking systems.
- News portals (auto-updating content).
- Online booking systems (railway, airline, hotel booking).

Quick Comparison (for exam table form):

Aspect	**Static Web Page**	**Dynamic Web Page**
Content	Fixed (same for all)	Changes dynamically
Technology	HTML, CSS, JS only	Server-side languages + DB
Cost	Cheap & simple	Expensive & complex
Examples	Portfolio, brochure	Facebook, Amazon, Gmail

15) Explain different selectors present in CSS.



CSS Selectors

A **selector** in CSS is used to target **HTML elements** so that styles (color, font, size, etc.) can be applied.

There are many selectors in CSS. Below are the most commonly used ones:

---

1. Universal Selector (`*`)

- Selects **all elements** in the web page.

```css
* {
  margin: 0;
  padding: 0;
}
```

👉 Removes default margin and padding for all elements.

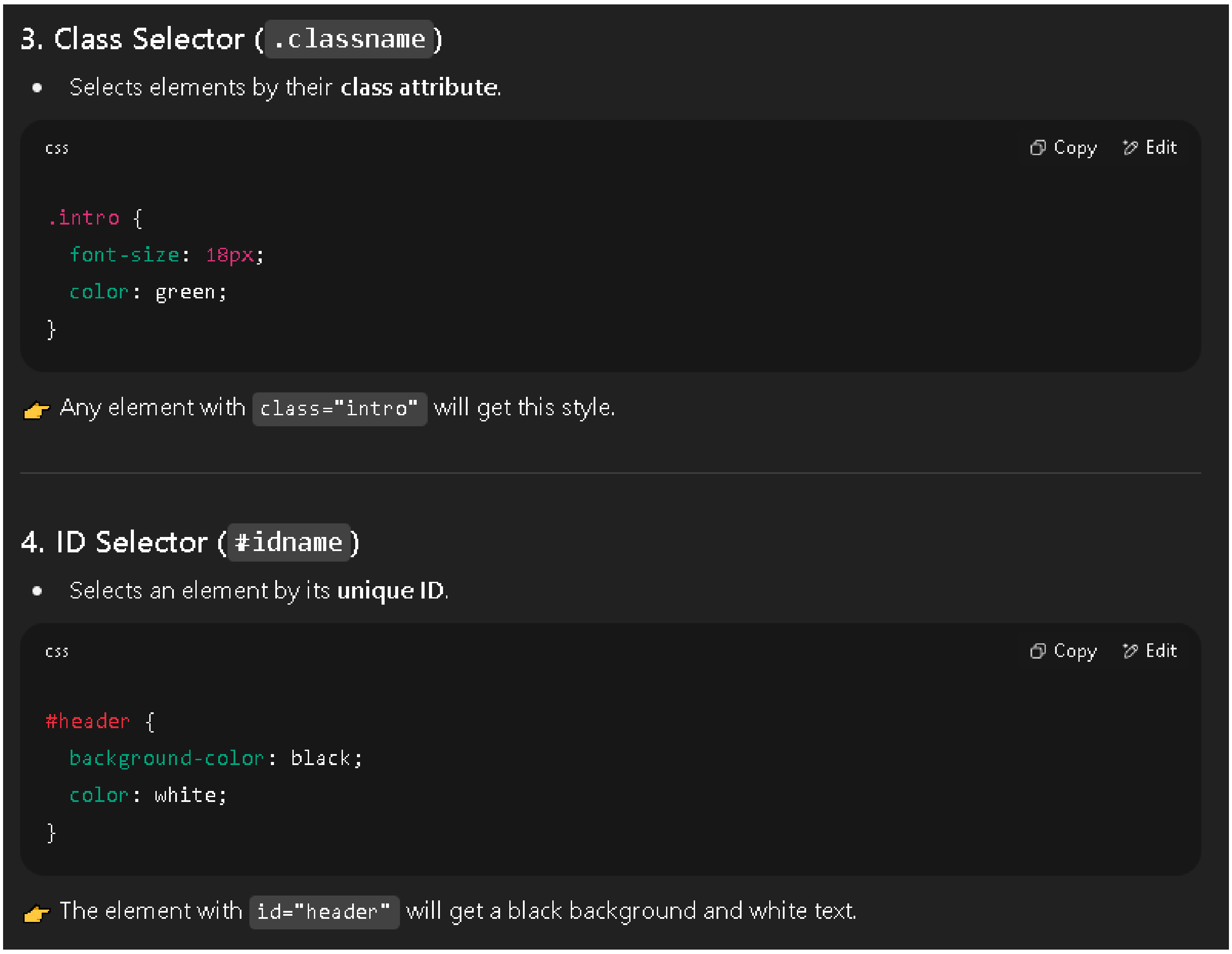
---

2. Element Selector (Tag Selector)

- Selects elements by their **HTML tag name**.

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

👉 All `<p>` elements will have blue text.



3. Class Selector (`.classname`)

- Selects elements by their **class attribute**.

```css
.intro {
  font-size: 18px;
  color: green;
}
```

👉 Any element with `class="intro"` will get this style.

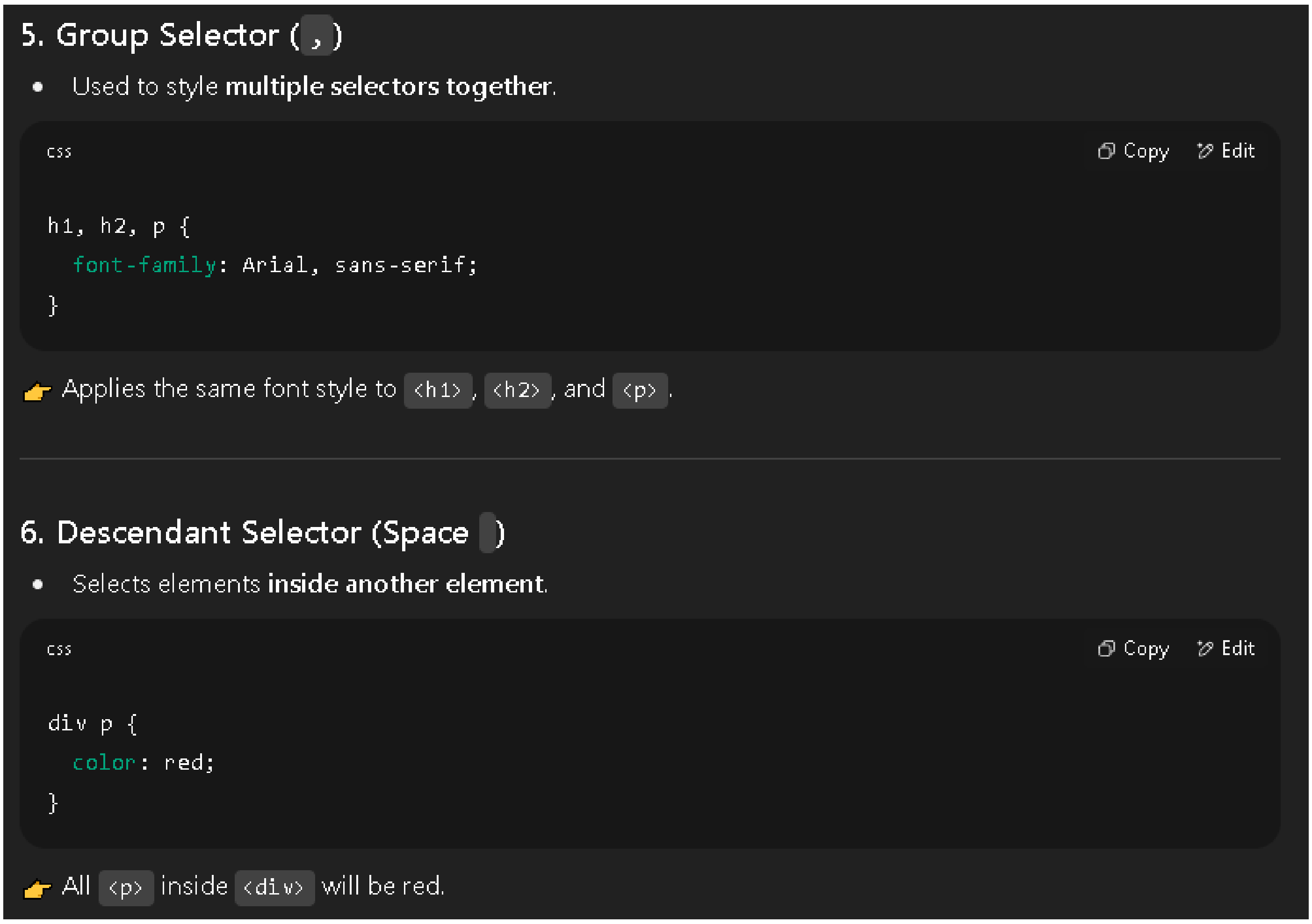
---

4. ID Selector (`#idname`)

- Selects an element by its **unique ID**.

```css
#header {
  background-color: black;
  color: white;
}
```

👉 The element with `id="header"` will get a black background and white text.



5. Group Selector (`,`)

- Used to style **multiple selectors together**.

```css
h1, h2, p {
  font-family: Arial, sans-serif;
}
```

👉 Applies the same font style to `<h1>`, `<h2>`, and `<p>`.

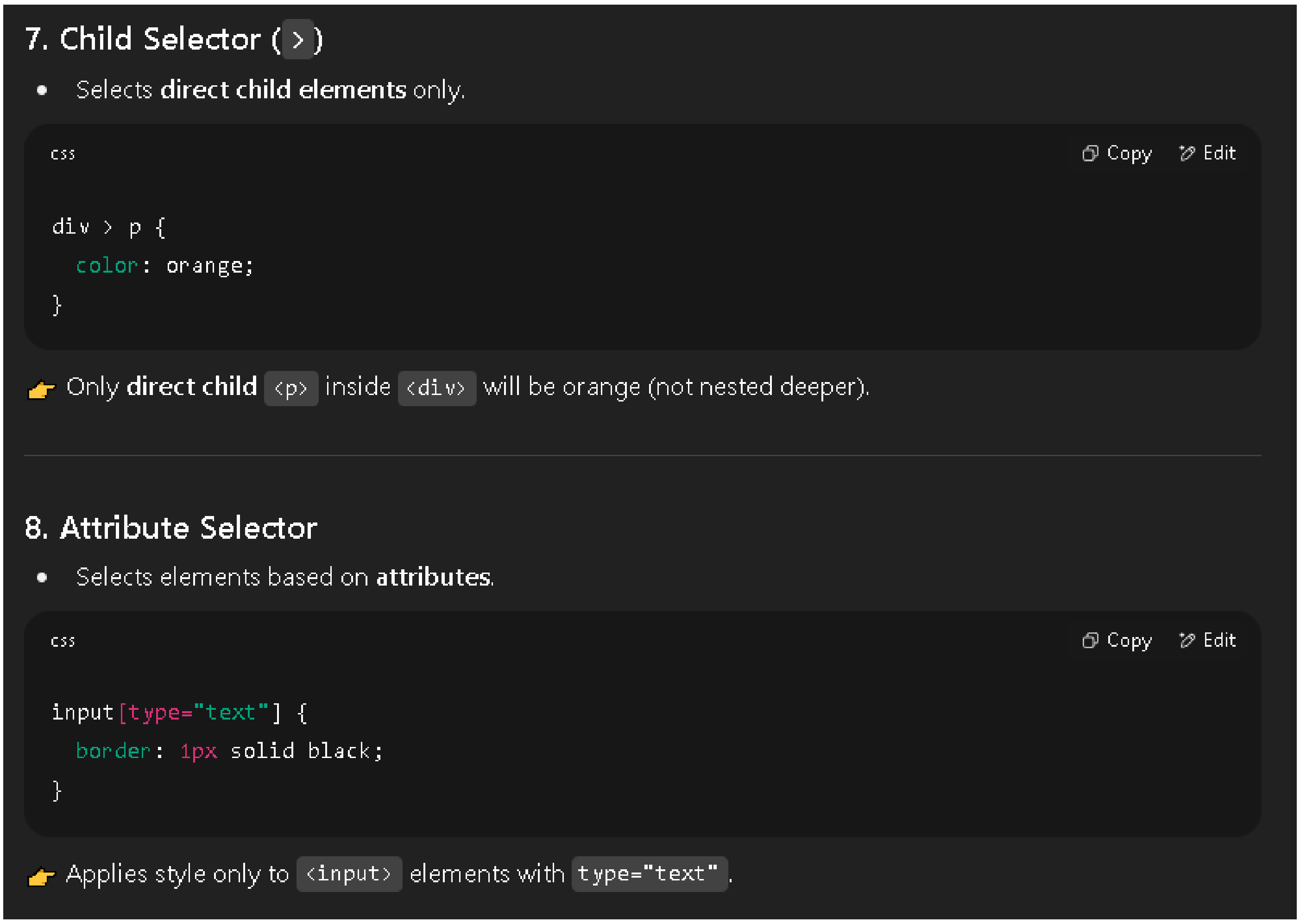
---

6. Descendant Selector (Space ` `)

- Selects elements **inside another element**.

```css
div p {
  color: red;
}
```

👉 All `<p>` inside `<div>` will be red.



7. Child Selector (`>`)

- Selects **direct child elements** only.

```css
div > p {
  color: orange;
}
```

👉 Only **direct child** `<p>` inside `<div>` will be orange (not nested deeper).

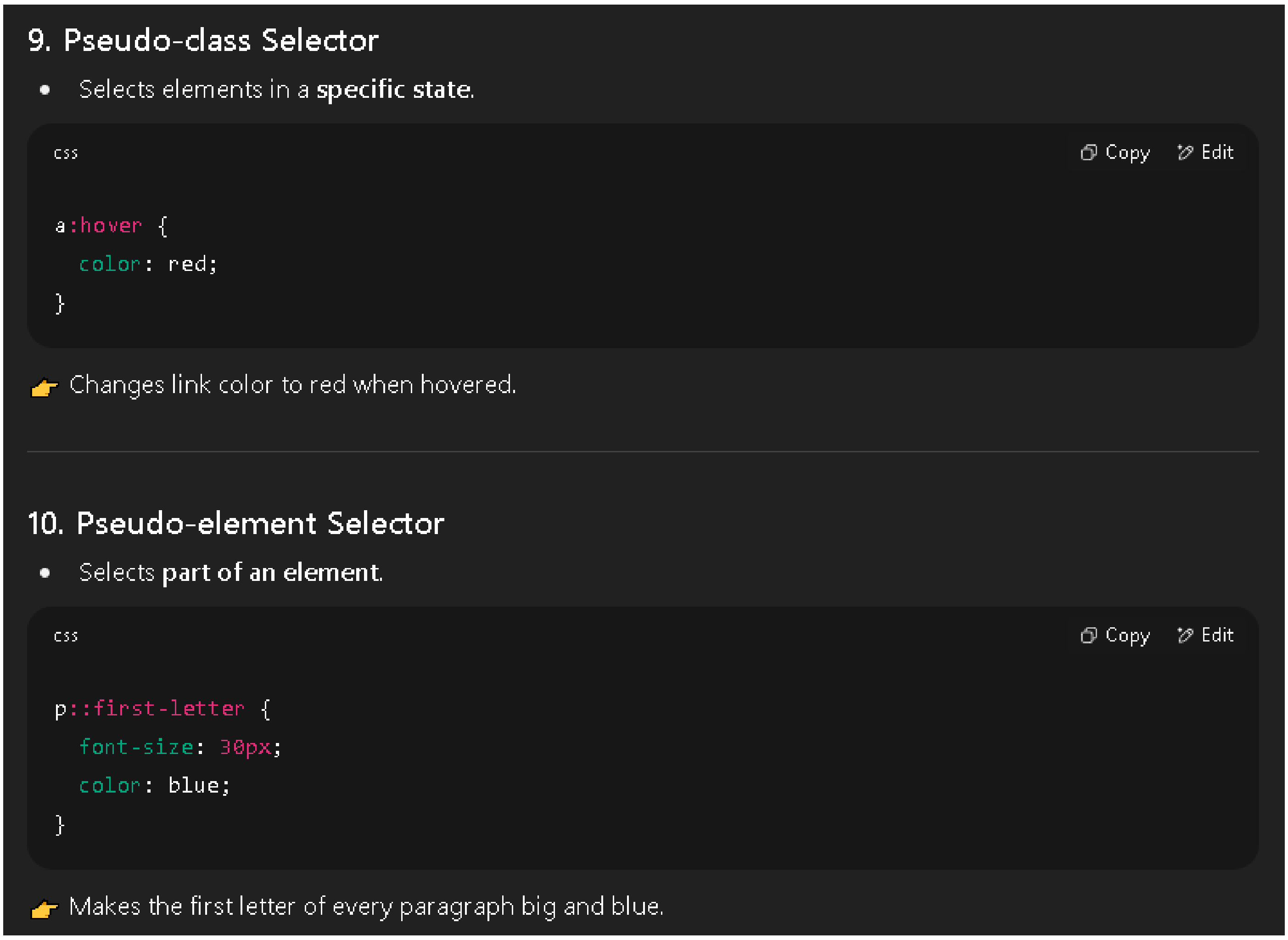
---

8. Attribute Selector

- Selects elements based on **attributes**.

```css
input[type="text"] {
  border: 1px solid black;
}
```

👉 Applies style only to `<input>` elements with `type="text"`.



9. Pseudo-class Selector

- Selects elements in a **specific state**.

```css
a:hover {
  color: red;
}
```

👉 Changes link color to red when hovered.

---

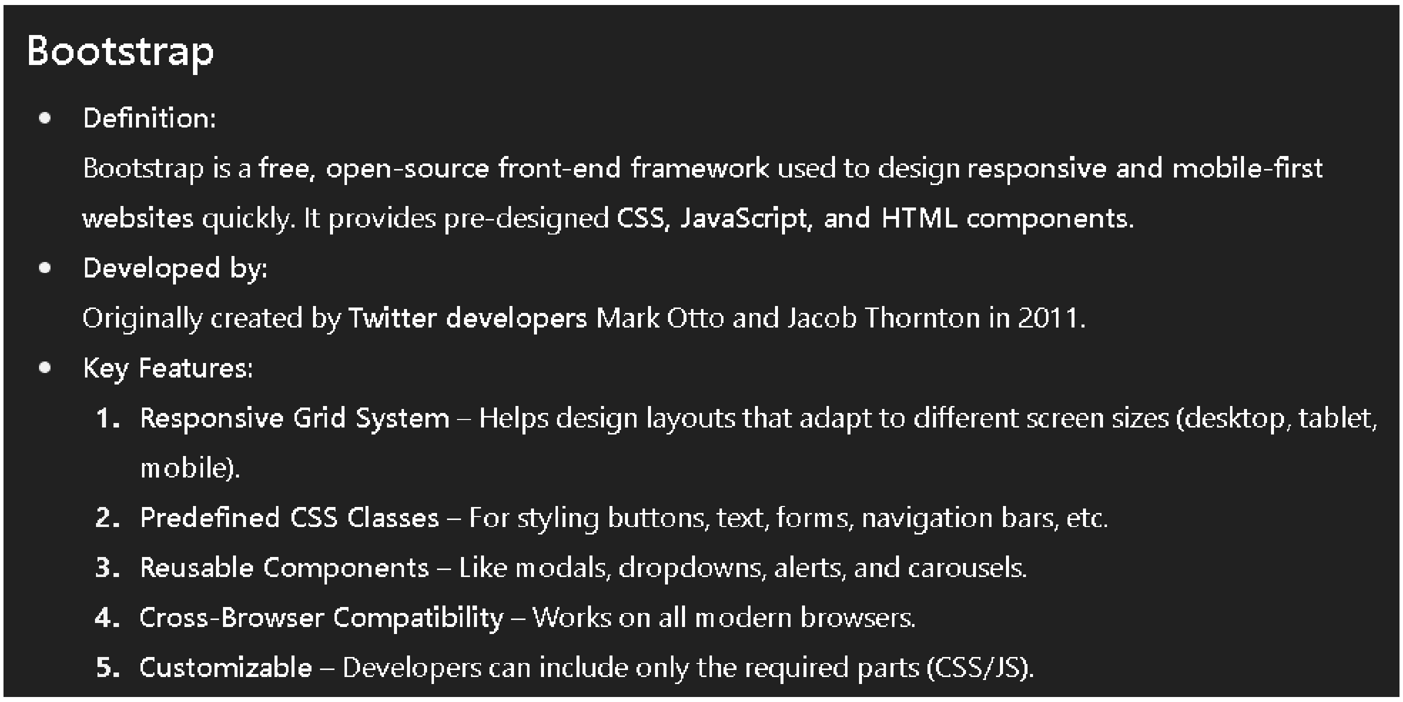
10. Pseudo-element Selector

- Selects **part of an element**.

```css
p::first-letter {
  font-size: 30px;
  color: blue;
}
```

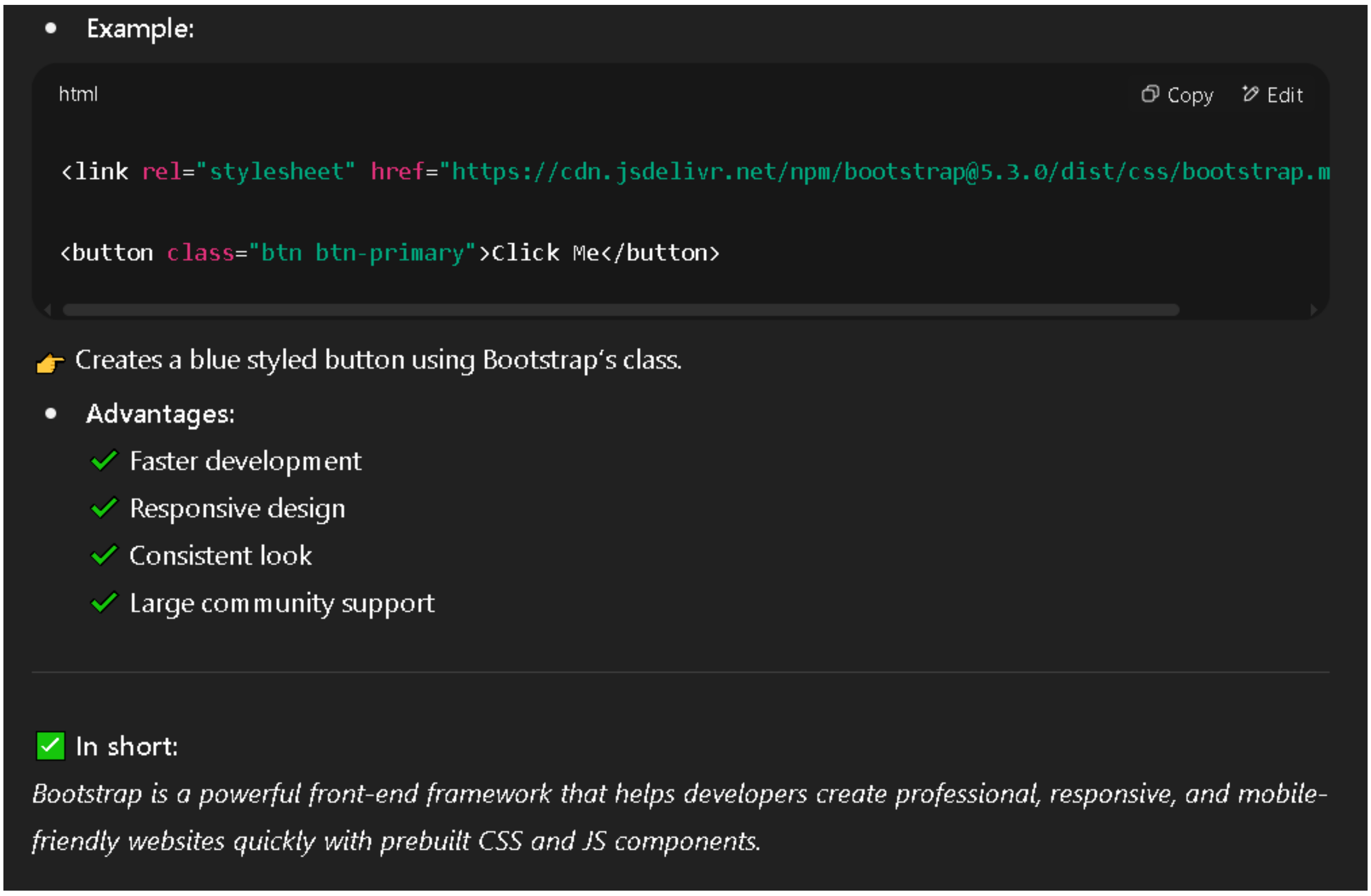
👉 Makes the first letter of every paragraph big and blue.

16) Write a short note on Bootstrap



Bootstrap

- **Definition:**
  Bootstrap is a **free, open-source front-end framework** used to design **responsive and mobile-first websites** quickly. It provides pre-designed **CSS, JavaScript, and HTML components.**
- **Developed by:**
  Originally created by **Twitter developers** Mark Otto and Jacob Thornton in 2011.
- **Key Features:**
  1. **Responsive Grid System** – Helps design layouts that adapt to different screen sizes (desktop, tablet, mobile).
  2. **Predefined CSS Classes** – For styling buttons, text, forms, navigation bars, etc.
  3. **Reusable Components** – Like modals, dropdowns, alerts, and carousels.
  4. **Cross-Browser Compatibility** – Works on all modern browsers.
  5. **Customizable** – Developers can include only the required parts (CSS/JS).



- **Example:**

```html
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.m

<button class="btn btn-primary">Click Me</button>
```

👉 Creates a blue styled button using Bootstrap's class.

- **Advantages:**
  ✅ Faster development
  ✅ Responsive design
  ✅ Consistent look
  ✅ Large community support

✅ **In short:**
Bootstrap is a powerful front-end framework that helps developers create professional, responsive, and mobile-friendly websites quickly with prebuilt CSS and JS components.

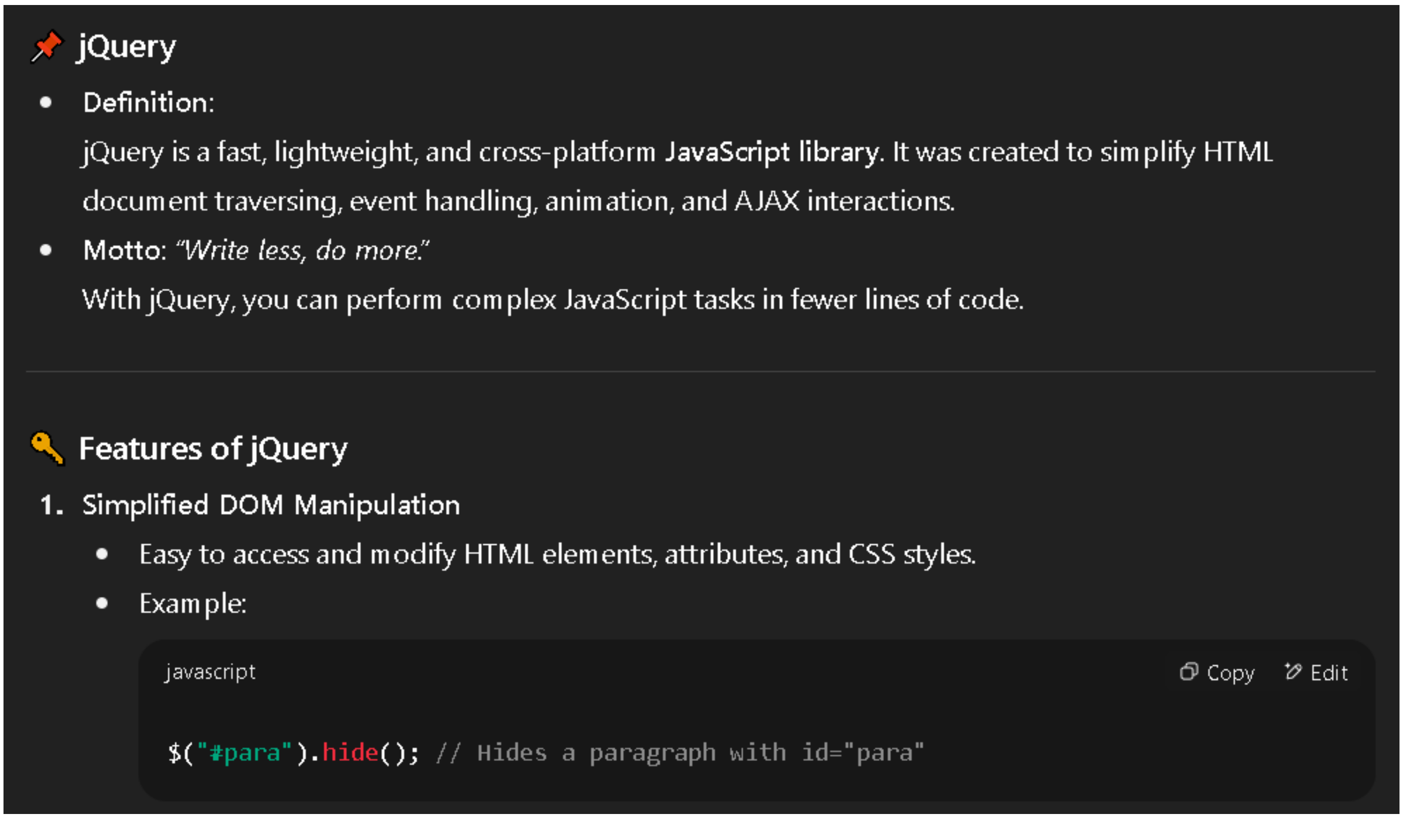
UNIT 2

17) Write an HTML page and also provide a JavaScript for accepting a user ID and password from the user to ensure the input is not empty.

```
<!DOCTYPE html>
<html>
<head>
    <title>Login Form</title>
    <script>
        // JavaScript function to validate form
        function validateForm() {
            var userId = document.getElementById("userid").value;
            var password = document.getElementById("password").value;

            if (userId === "" || password === "") {
                alert("User ID and Password cannot be empty!");
                return false;  // Prevents form submission
            }
            else {
                alert("Login successful!");
                return true;   // Allows form submission
            }
        }
    </script>
</head>
```

```
<body>
    <h2>Login Page</h2>
    <form onsubmit="return validateForm()">
        <label for="userid">User ID:</label>
        <input type="text" id="userid" name="userid"><br><br>

        <label for="password">Password:</label>
        <input type="password" id="password" name="password"><br><br>

        <input type="submit" value="Login">
    </form>
</body>
</html>
```

18) Write a note on jQuery.



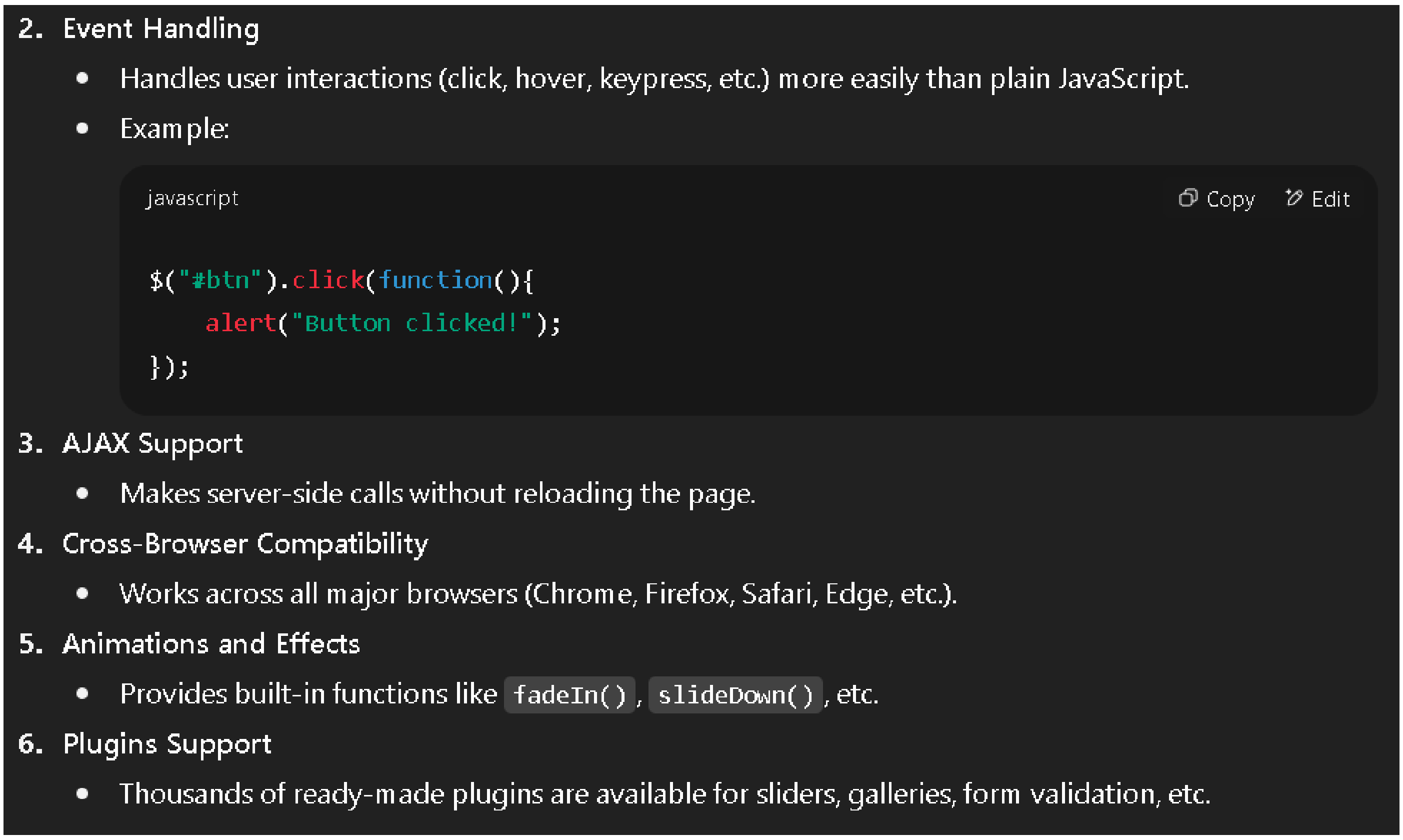
📌 jQuery

- **Definition:**
  jQuery is a fast, lightweight, and cross-platform **JavaScript library**. It was created to simplify HTML document traversing, event handling, animation, and AJAX interactions.
- **Motto:** *"Write less, do more."*
  With jQuery, you can perform complex JavaScript tasks in fewer lines of code.

🔑 Features of jQuery

1. **Simplified DOM Manipulation**
   - Easy to access and modify HTML elements, attributes, and CSS styles.
   - Example:

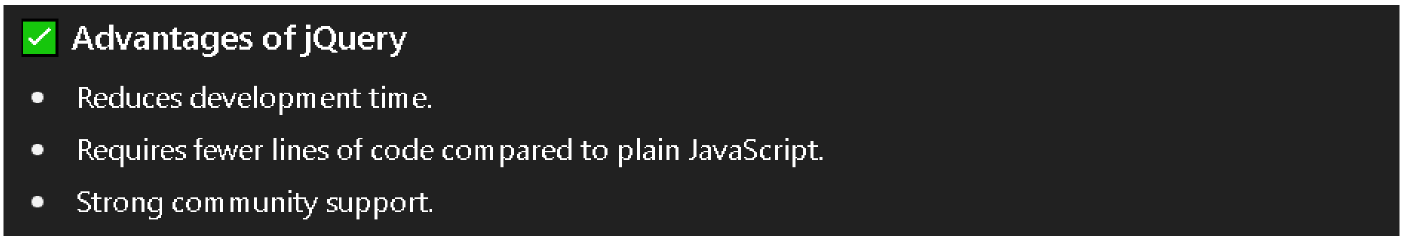
```javascript
$("#para").hide(); // Hides a paragraph with id="para"
```



2. **Event Handling**
   - Handles user interactions (click, hover, keypress, etc.) more easily than plain JavaScript.
   - Example:

```javascript
$("#btn").click(function(){
    alert("Button clicked!");
});
```

3. **AJAX Support**
   - Makes server-side calls without reloading the page.
4. **Cross-Browser Compatibility**
   - Works across all major browsers (Chrome, Firefox, Safari, Edge, etc.).
5. **Animations and Effects**
   - Provides built-in functions like `fadeIn()`, `slideDown()`, etc.
6. **Plugins Support**
   - Thousands of ready-made plugins are available for sliders, galleries, form validation, etc.



✅ Advantages of jQuery

- Reduces development time.
- Requires fewer lines of code compared to plain JavaScript.
- Strong community support.

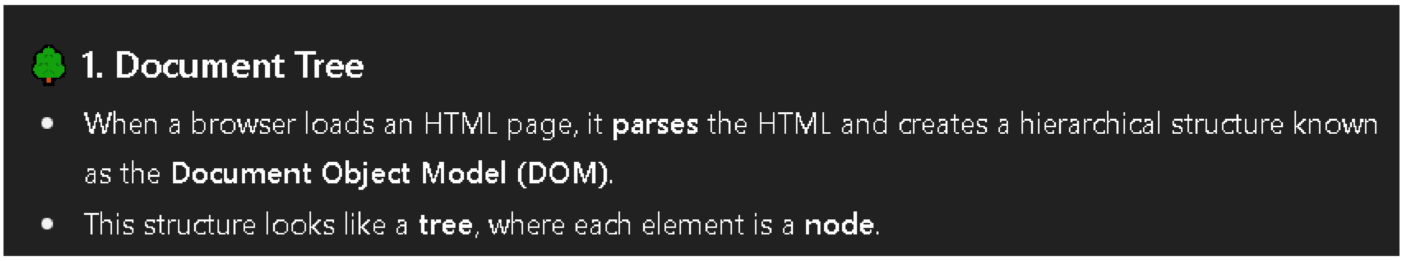
📖 **Example**

html

```html
<!DOCTYPE html>
<html>
<head>
    <title>jQuery Example</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script>
        $(document).ready(function(){
            $("#btn").click(function(){
                $("#msg").text("Hello, jQuery!").css("color","blue");
            });
        });
    </script>
</head>
<body>
    <button id="btn">Click Me</button>
    <p id="msg"></p>
</body>
</html>
```

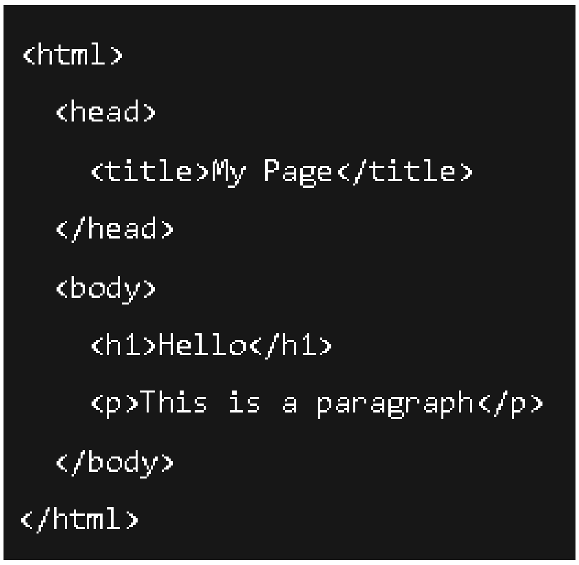
👉 When the button is clicked, the text `"Hello, jQuery!"` will appear in blue.

19) Explain the document tree and DOM event handling.

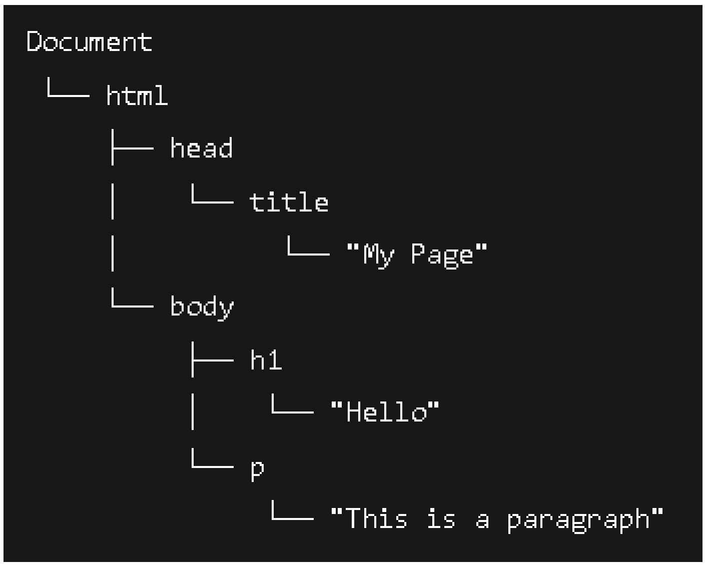


🌳 1. Document Tree

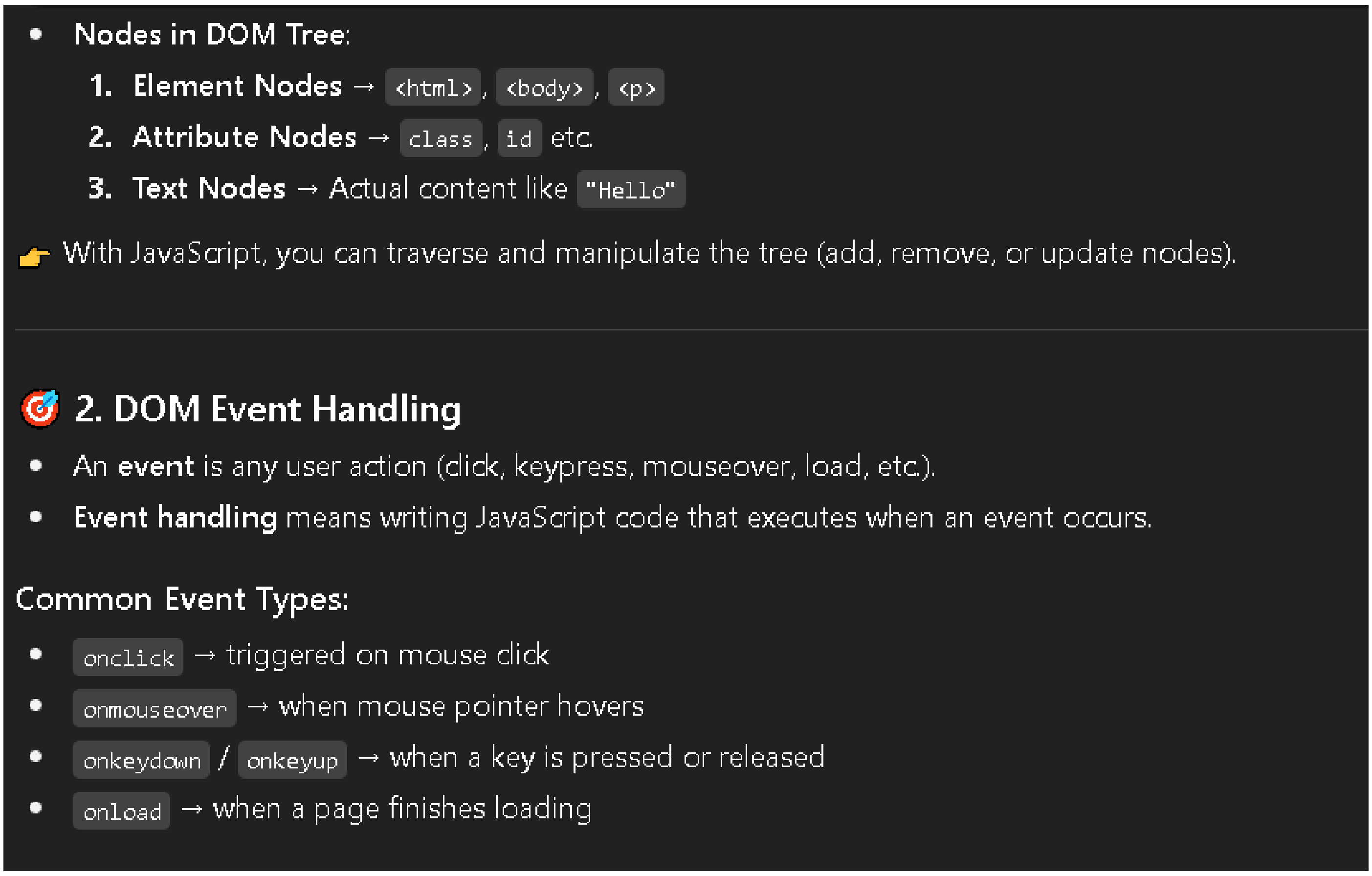
- When a browser loads an HTML page, it **parses** the HTML and creates a hierarchical structure known as the **Document Object Model (DOM)**.
- This structure looks like a **tree**, where each element is a **node**.



```html
<html>
  <head>
    <title>My Page</title>
  </head>
  <body>
    <h1>Hello</h1>
    <p>This is a paragraph</p>
  </body>
</html>
```



```
Document
 └── html
      ├── head
      │    └── title
      │         └── "My Page"
      └── body
           ├── h1
           │    └── "Hello"
           └── p
                └── "This is a paragraph"
```



- Nodes in DOM Tree:
  1. **Element Nodes** → `<html>`, `<body>`, `<p>`
  2. **Attribute Nodes** → `class`, `id` etc.
  3. **Text Nodes** → Actual content like `"Hello"`

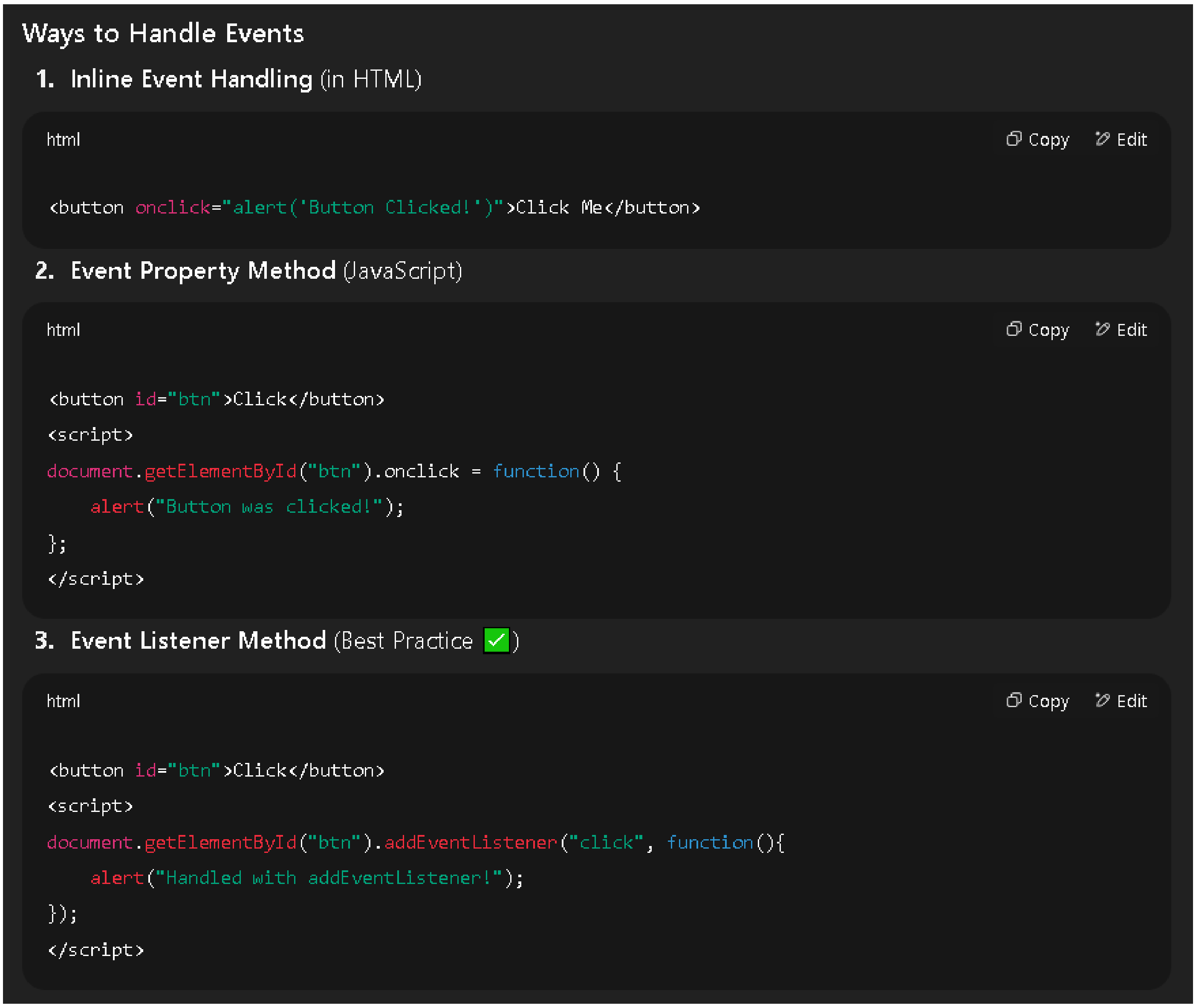
👉 With JavaScript, you can traverse and manipulate the tree (add, remove, or update nodes).

🎯 2. DOM Event Handling

- An **event** is any user action (click, keypress, mouseover, load, etc.).
- **Event handling** means writing JavaScript code that executes when an event occurs.

Common Event Types:

- `onclick` → triggered on mouse click
- `onmouseover` → when mouse pointer hovers
- `onkeydown` / `onkeyup` → when a key is pressed or released
- `onload` → when a page finishes loading



Ways to Handle Events

1. **Inline Event Handling** (in HTML)

html

```html
<button onclick="alert('Button Clicked!')">Click Me</button>
```

2. **Event Property Method** (JavaScript)

html

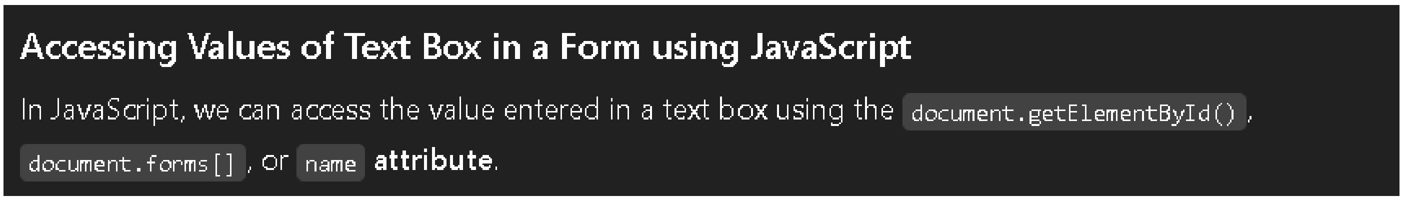
```html
<button id="btn">Click</button>
<script>
document.getElementById("btn").onclick = function() {
    alert("Button was clicked!");
};
</script>
```

3. **Event Listener Method** (Best Practice ✅)

html

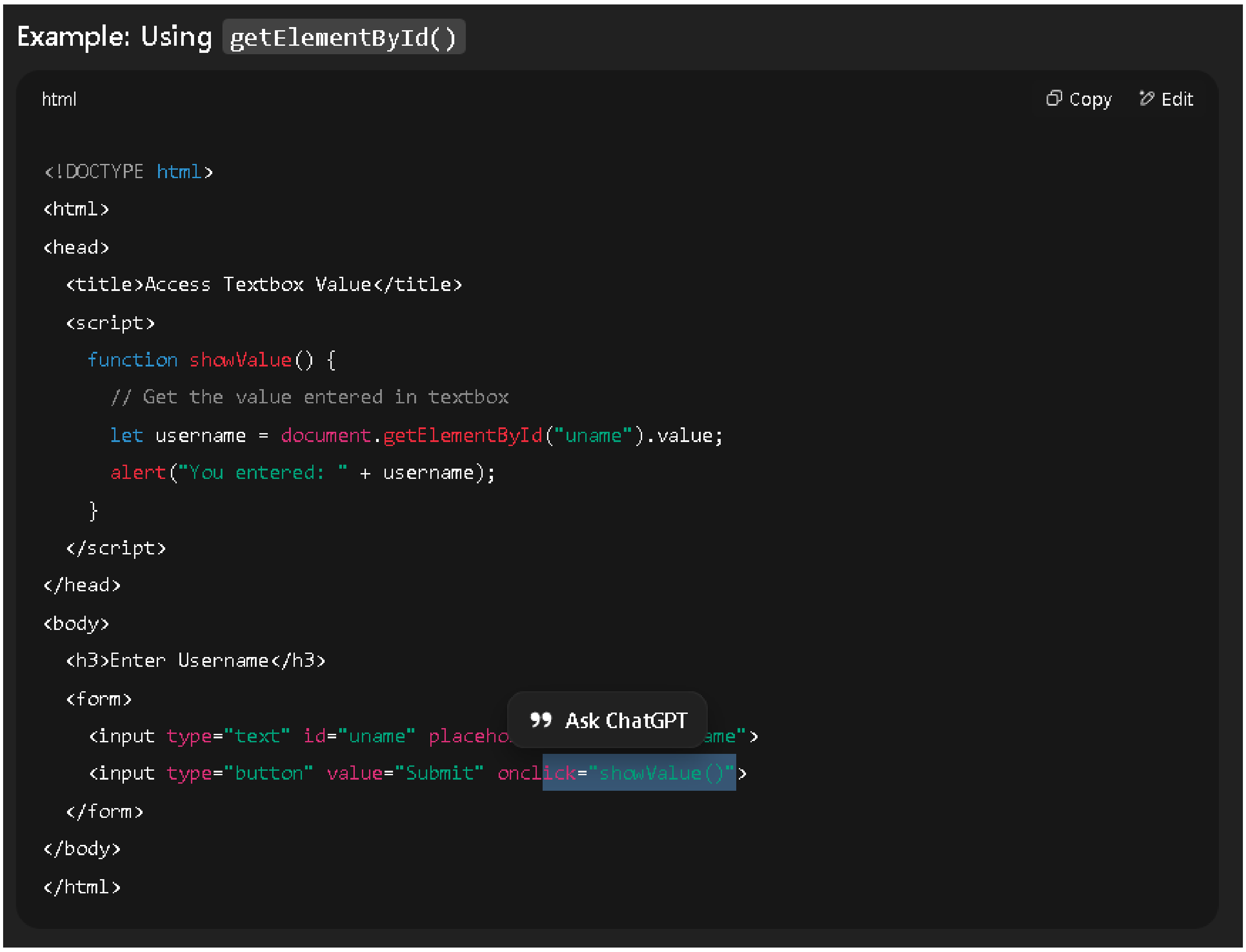
```html
<button id="btn">Click</button>
<script>
document.getElementById("btn").addEventListener("click", function(){
    alert("Handled with addEventListener!");
});
</script>
```

20) Explain with example how to access values of text box of form using JavaScript?



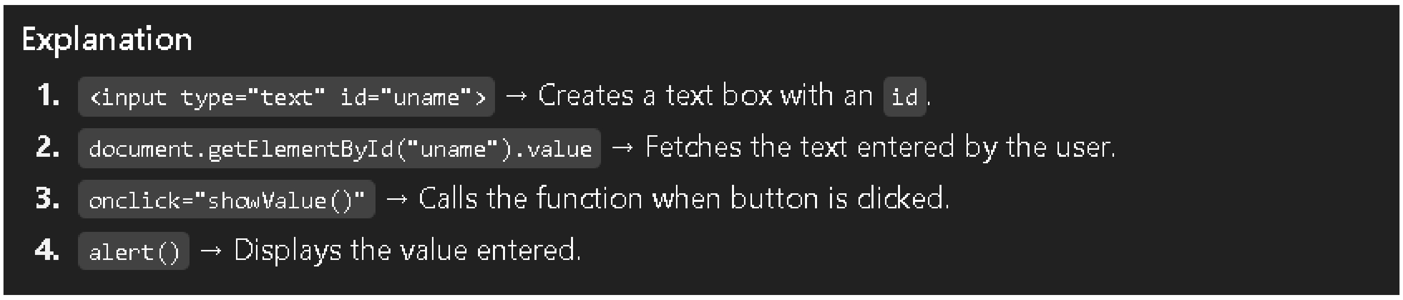
Accessing Values of Text Box in a Form using JavaScript

In JavaScript, we can access the value entered in a text box using the `document.getElementById()`, `document.forms[]`, or `name` **attribute**.



Example: Using `getElementById()`

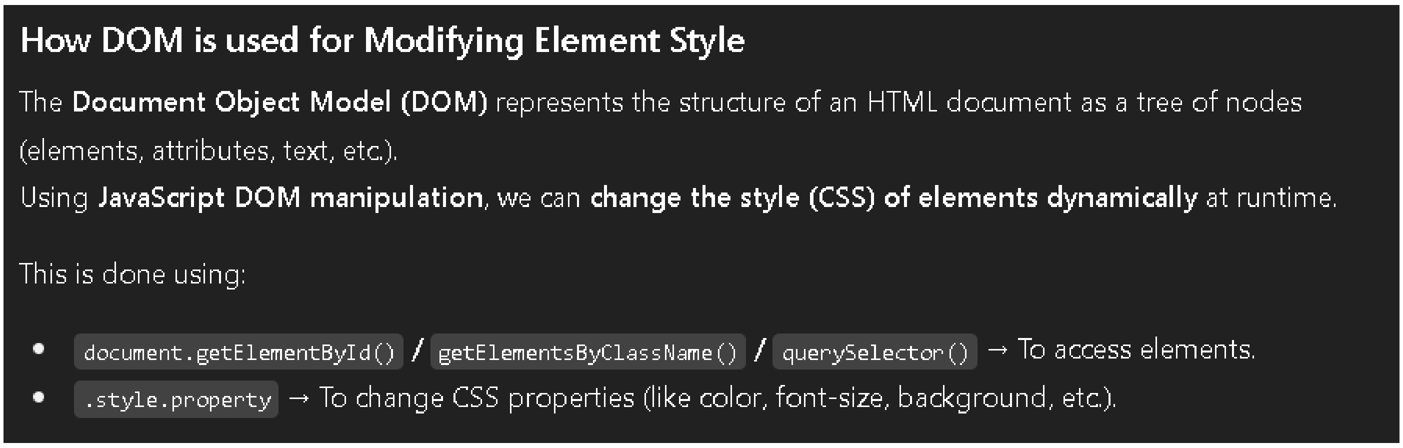
```html
<!DOCTYPE html>
<html>
<head>
  <title>Access Textbox Value</title>
  <script>
    function showValue() {
      // Get the value entered in textbox
      let username = document.getElementById("uname").value;
      alert("You entered: " + username);
    }
  </script>
</head>
<body>
  <h3>Enter Username</h3>
  <form>
    <input type="text" id="uname" placeho          ame">
    <input type="button" value="Submit" onclick="showValue()">
  </form>
</body>
</html>
```



Explanation

1. `<input type="text" id="uname">` → Creates a text box with an `id`.
2. `document.getElementById("uname").value` → Fetches the text entered by the user.
3. `onclick="showValue()"` → Calls the function when button is clicked.
4. `alert()` → Displays the value entered.

21) How DOM is used for modifying element style explain with suitable example.

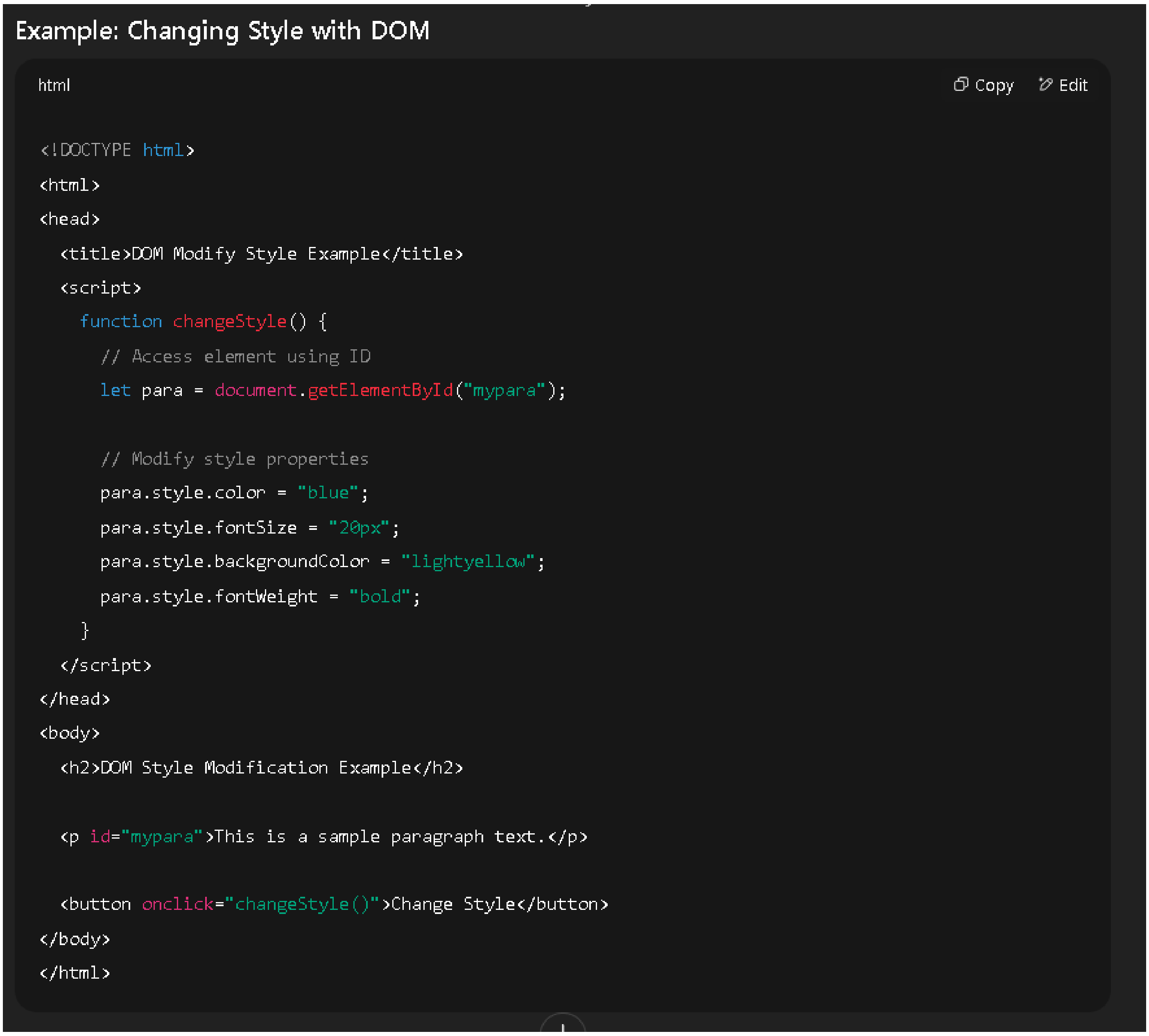


How DOM is used for Modifying Element Style

The **Document Object Model (DOM)** represents the structure of an HTML document as a tree of nodes (elements, attributes, text, etc.).
Using **JavaScript DOM manipulation**, we can **change the style (CSS) of elements dynamically** at runtime.

This is done using:

- `document.getElementById()` / `getElementsByClassName()` / `querySelector()` → To access elements.
- `.style.property` → To change CSS properties (like color, font-size, background, etc.).



Example: Changing Style with DOM

html

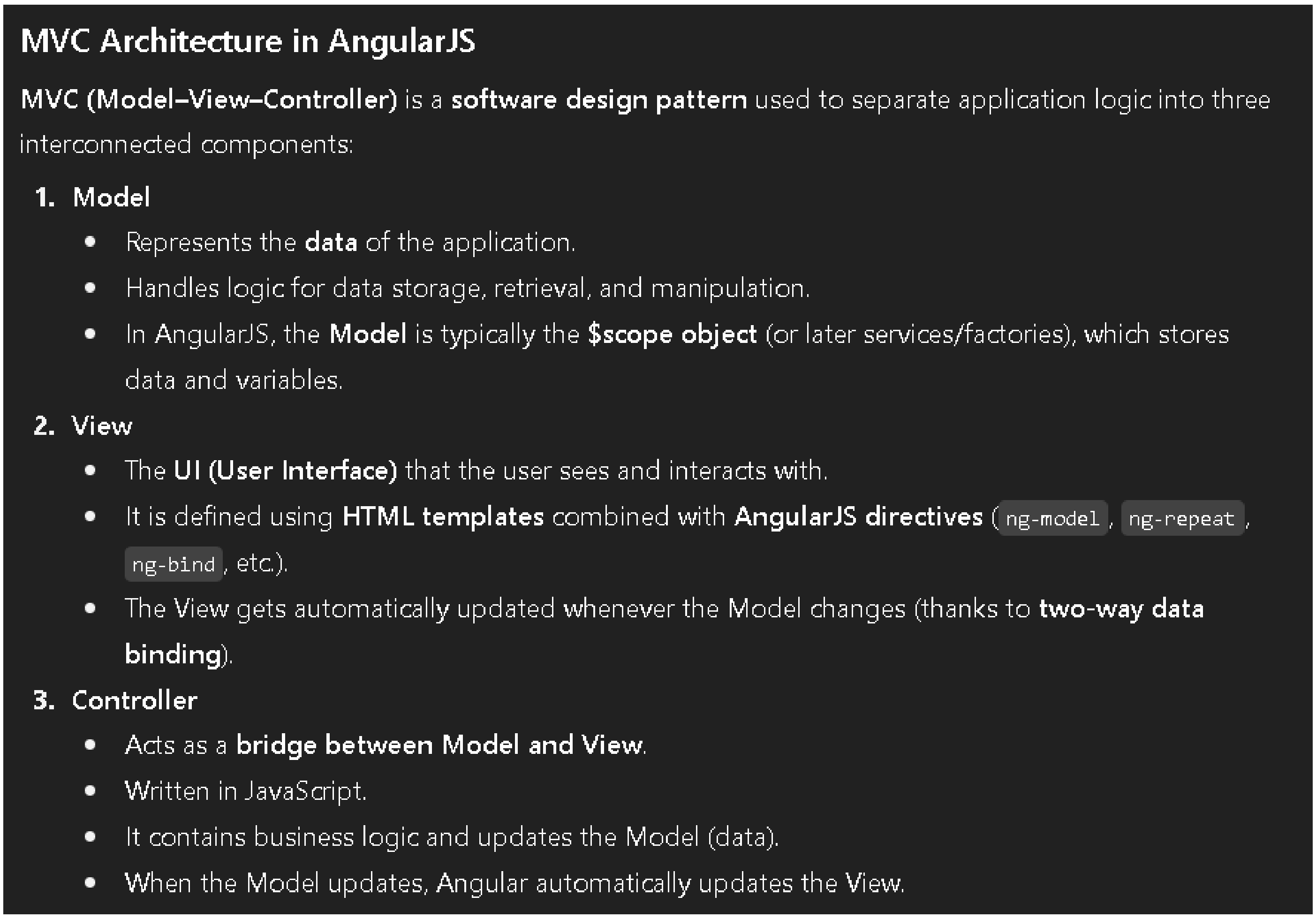
```html
<!DOCTYPE html>
<html>
<head>
  <title>DOM Modify Style Example</title>
  <script>
    function changeStyle() {
      // Access element using ID
      let para = document.getElementById("mypara");

      // Modify style properties
      para.style.color = "blue";
      para.style.fontSize = "20px";
      para.style.backgroundColor = "lightyellow";
      para.style.fontWeight = "bold";
    }
  </script>
</head>
<body>
  <h2>DOM Style Modification Example</h2>

  <p id="mypara">This is a sample paragraph text.</p>

  <button onclick="changeStyle()">Change Style</button>
</body>
</html>
```

Explanation

1. `getElementById("mypara")` → Gets the paragraph element.
2. `para.style.color = "blue";` → Changes text color to blue.
3. `para.style.fontSize = "20px";` → Increases font size.
4. `para.style.backgroundColor = "lightyellow";` → Adds background color.
5. When button is clicked, JavaScript **modifies the element's CSS dynamically**.

✅ **Conclusion**: The DOM allows JavaScript to directly interact with HTML elements and change their style properties dynamically, making web pages **interactive and visually dynamic**.

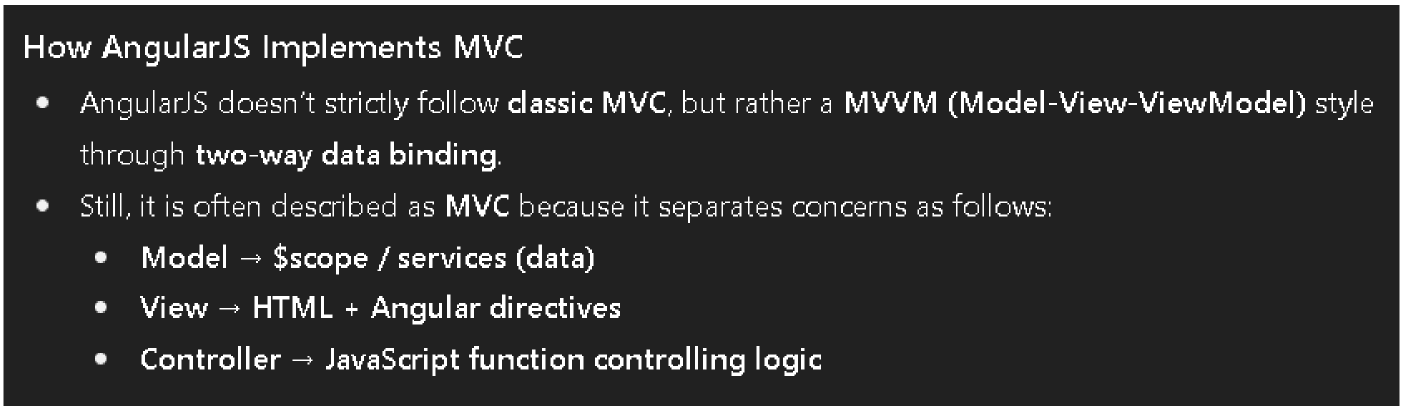
22) What is MVC architecture with respect to Angular IS.



MVC Architecture in AngularJS

MVC (Model–View–Controller) is a **software design pattern** used to separate application logic into three interconnected components:

1. **Model**
   - Represents the **data** of the application.
   - Handles logic for data storage, retrieval, and manipulation.
   - In AngularJS, the **Model** is typically the **$scope object** (or later services/factories), which stores data and variables.
2. **View**
   - The **UI (User Interface)** that the user sees and interacts with.
   - It is defined using **HTML templates** combined with **AngularJS directives** (`ng-model`, `ng-repeat`, `ng-bind`, etc.).
   - The View gets automatically updated whenever the Model changes (thanks to **two-way data binding**).
3. **Controller**
   - Acts as a **bridge between Model and View**.
   - Written in JavaScript.
   - It contains business logic and updates the Model (data).
   - When the Model updates, Angular automatically updates the View.



How AngularJS Implements MVC

- AngularJS doesn't strictly follow **classic MVC**, but rather a **MVVM (Model-View-ViewModel)** style through **two-way data binding**.
- Still, it is often described as **MVC** because it separates concerns as follows:
  - **Model → $scope / services (data)**
  - **View → HTML + Angular directives**
  - **Controller → JavaScript function controlling logic**

html Copy Edit

```html
<!DOCTYPE html>
<html ng-app="myApp">
<head>
  <title>AngularJS MVC Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <script>
    // Controller
    angular.module("myApp", [])
    .controller("MainCtrl", function($scope) {
        // Model
        $scope.name = "Karan";

        $scope.changeName = function() {
            $scope.name = "AngularJS Learner";
        };
    });
  </script>
</head>
<body ng-controller="MainCtrl">
  <!-- View -->
  <h2>Hello, {{name}}</h2>
  <button ng-click="changeName()">Change Name</button>
</body>
</html>
```

Explanation of Example

- **Model** → `$scope.name` stores the data.
- **View** → `{{name}}` in HTML displays the data.
- **Controller** → `MainCtrl` updates the model when the button is clicked.
- When the model (`$scope.name`) changes, the view updates **automatically** (no manual DOM manipulation needed).

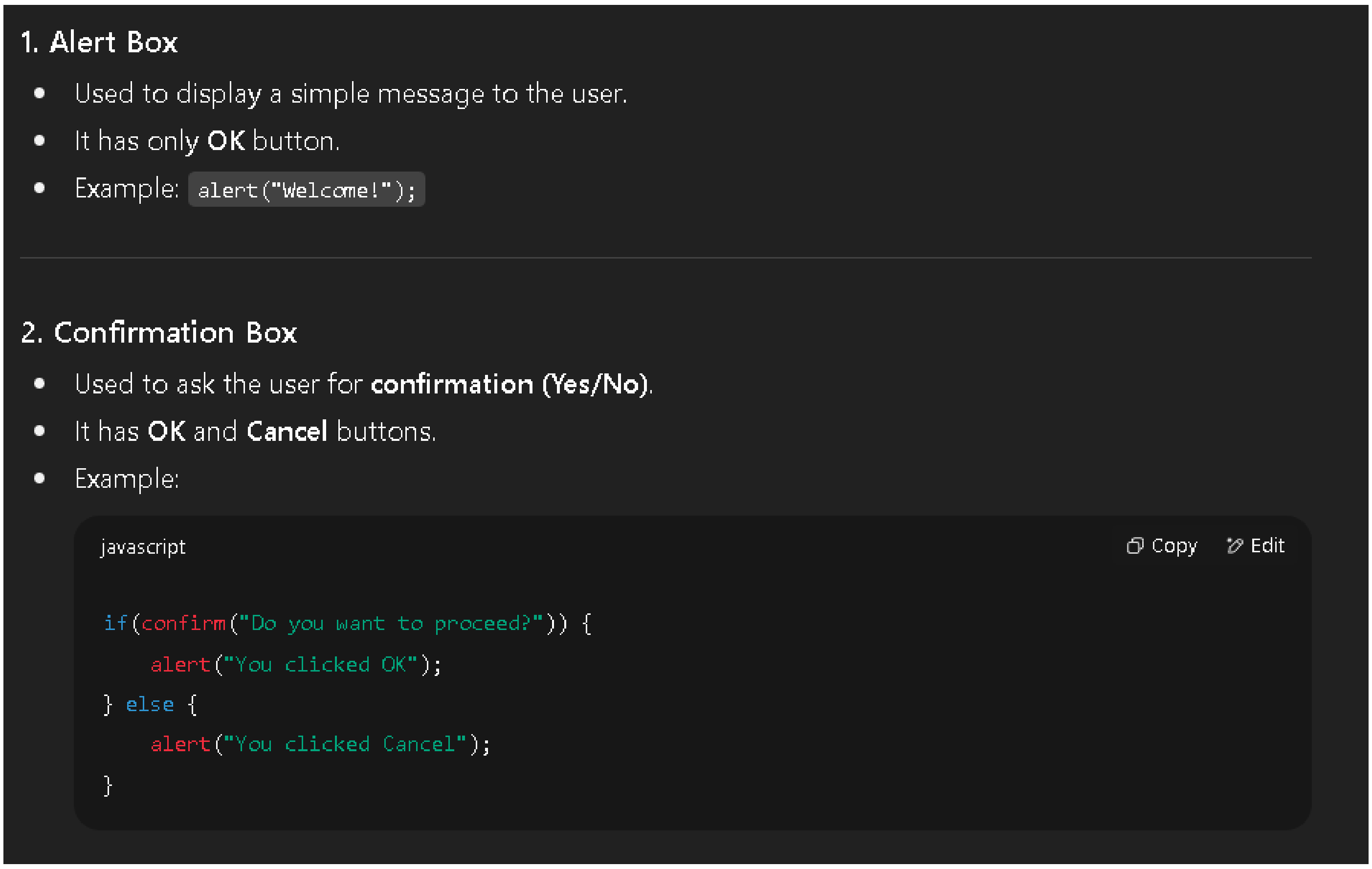
✅ **Conclusion:**

In AngularJS, the MVC architecture ensures **separation of concerns**:

- **Model** → Manages data (`$scope`, services).
- **View** → Displays data (HTML + directives).
- **Controller** → Handles logic and updates the model.

  Angular's **two-way data binding** makes this MVC very smooth.

23) Explain Alert, Confirmation and Prompt box in Javascript. Also write HTML+Javascript code to take two numbers as input and show their sum on alert box.



1. Alert Box

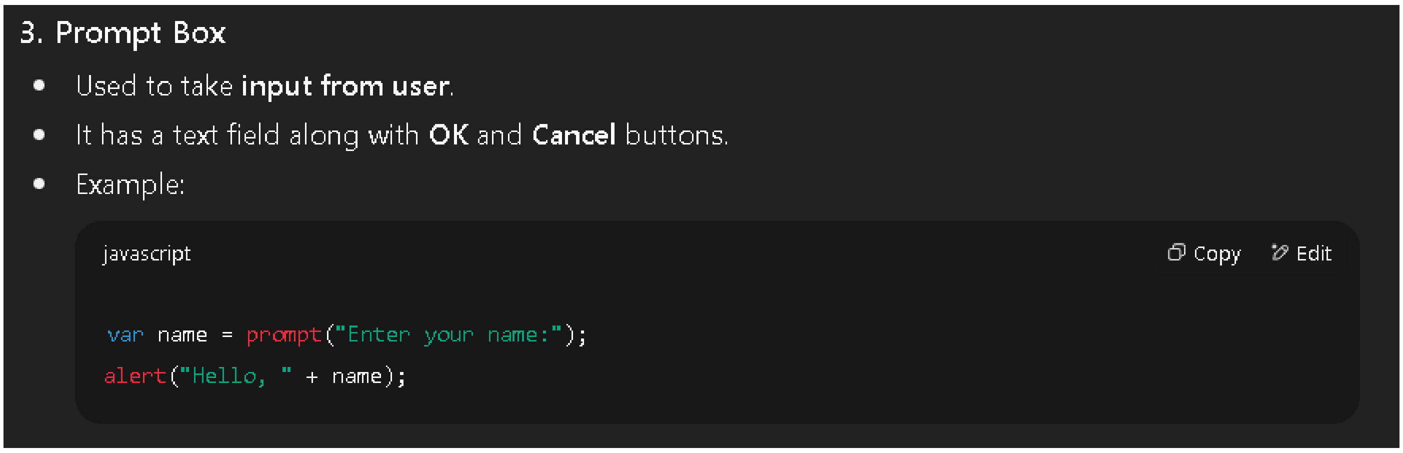
- Used to display a simple message to the user.
- It has only **OK** button.
- Example: `alert("Welcome!");`

---

2. Confirmation Box

- Used to ask the user for **confirmation (Yes/No)**.
- It has **OK** and **Cancel** buttons.
- Example:

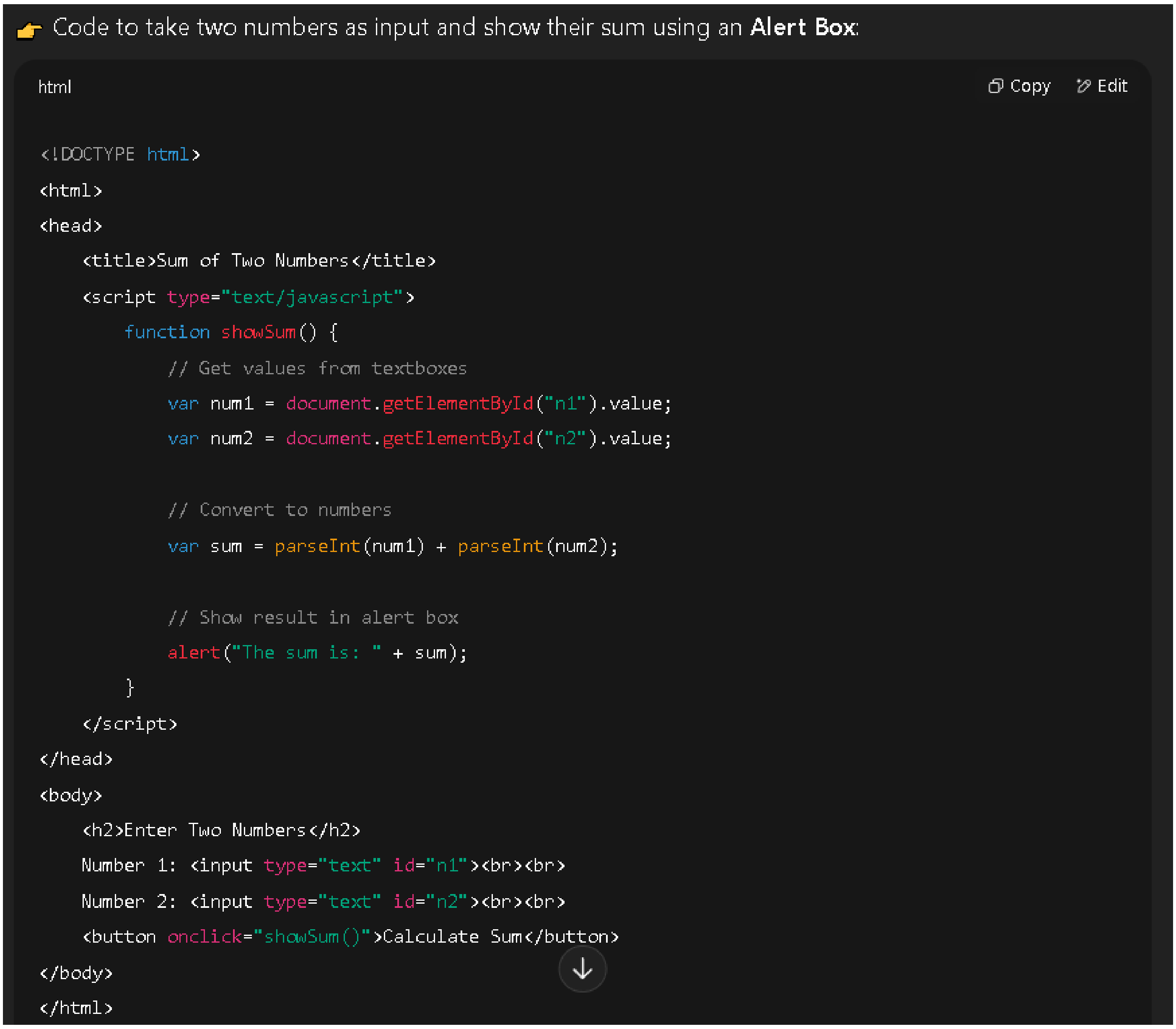
```javascript
if(confirm("Do you want to proceed?")) {
    alert("You clicked OK");
} else {
    alert("You clicked Cancel");
}
```



3. Prompt Box

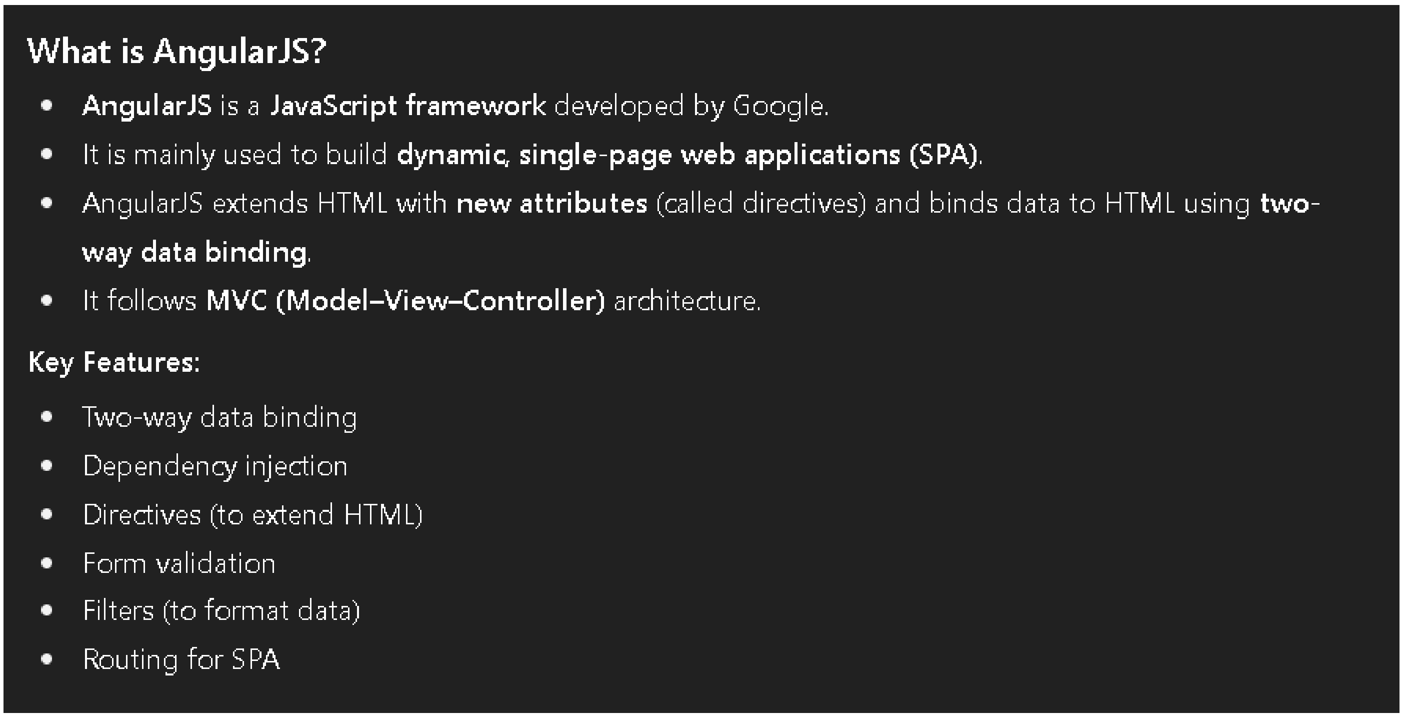
- Used to take **input from user**.
- It has a text field along with **OK** and **Cancel** buttons.
- Example:

```javascript
var name = prompt("Enter your name:");
alert("Hello, " + name);
```


👉 Code to take two numbers as input and show their sum using an **Alert Box**:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Sum of Two Numbers</title>
    <script type="text/javascript">
        function showSum() {
            // Get values from textboxes
            var num1 = document.getElementById("n1").value;
            var num2 = document.getElementById("n2").value;

            // Convert to numbers
            var sum = parseInt(num1) + parseInt(num2);

            // Show result in alert box
            alert("The sum is: " + sum);
        }
    </script>
</head>
<body>
    <h2>Enter Two Numbers</h2>
    Number 1: <input type="text" id="n1"><br><br>
    Number 2: <input type="text" id="n2"><br><br>
    <button onclick="showSum()">Calculate Sum</button>
</body>
</html>
```

24) What is Angular JS? What are the ng-directies used in Angular JS?

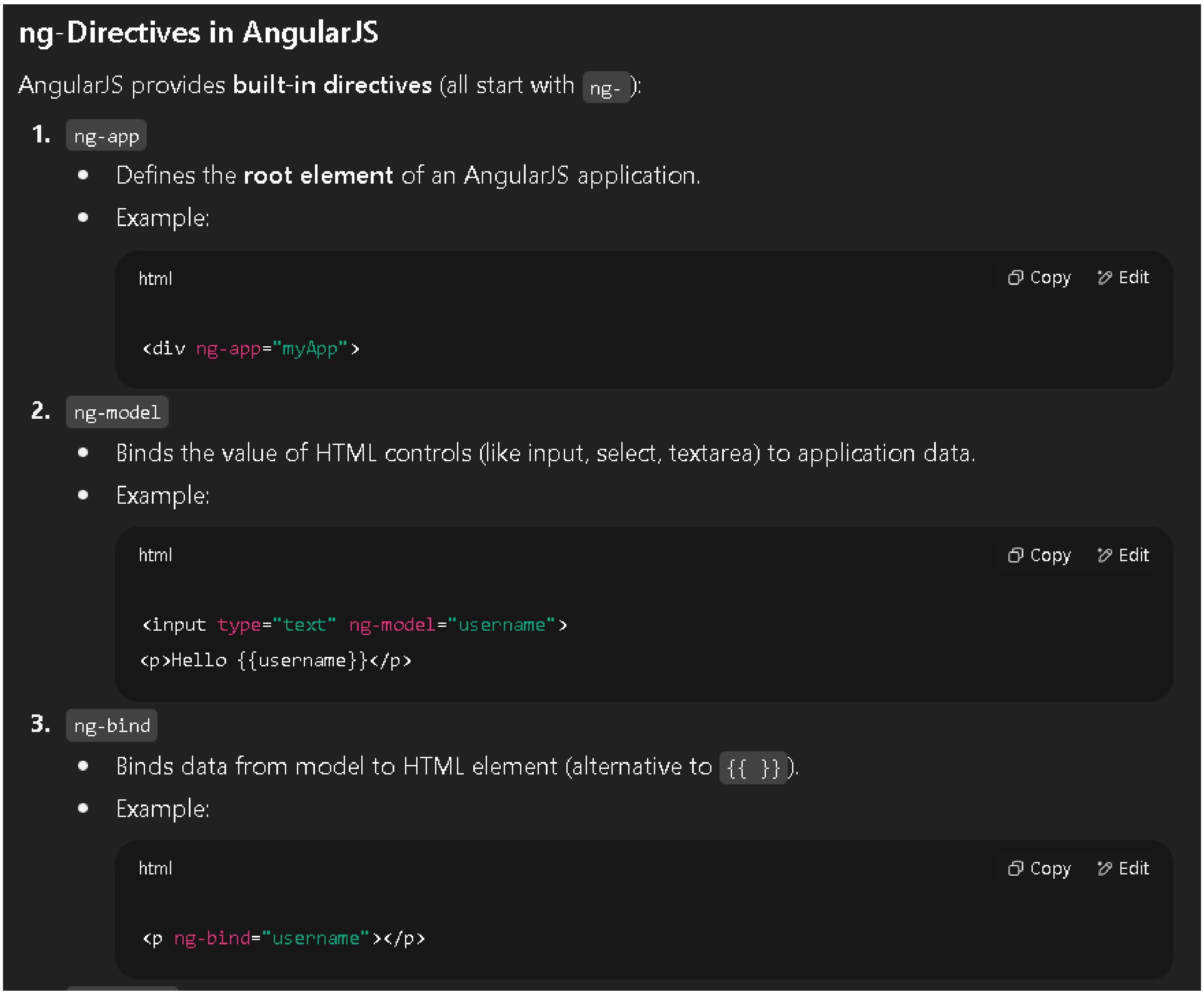


What is AngularJS?

- **AngularJS** is a **JavaScript framework** developed by Google.
- It is mainly used to build **dynamic, single-page web applications (SPA)**.
- AngularJS extends HTML with **new attributes** (called directives) and binds data to HTML using **two-way data binding**.
- It follows **MVC (Model–View–Controller)** architecture.

Key Features:

- Two-way data binding
- Dependency injection
- Directives (to extend HTML)
- Form validation
- Filters (to format data)
- Routing for SPA



ng-Directives in AngularJS

AngularJS provides **built-in directives** (all start with `ng-`):

1. `ng-app`
   - Defines the **root element** of an AngularJS application.
   - Example:

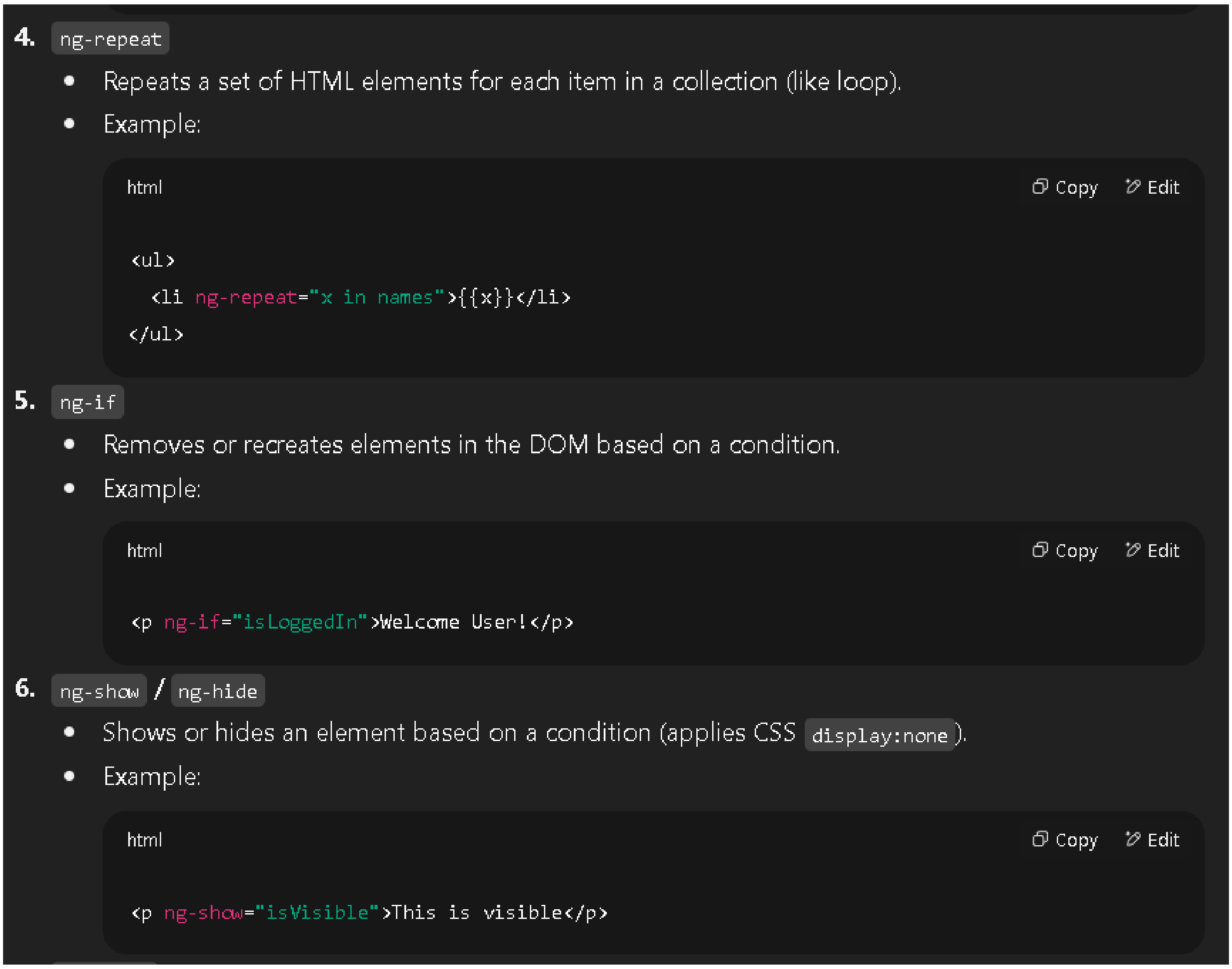
```html
<div ng-app="myApp">
```

2. `ng-model`
   - Binds the value of HTML controls (like input, select, textarea) to application data.
   - Example:

```html
<input type="text" ng-model="username">
<p>Hello {{username}}</p>
```

3. `ng-bind`
   - Binds data from model to HTML element (alternative to `{{ }}`).
   - Example:

```html
<p ng-bind="username"></p>
```



4. `ng-repeat`
   - Repeats a set of HTML elements for each item in a collection (like loop).
   - Example:

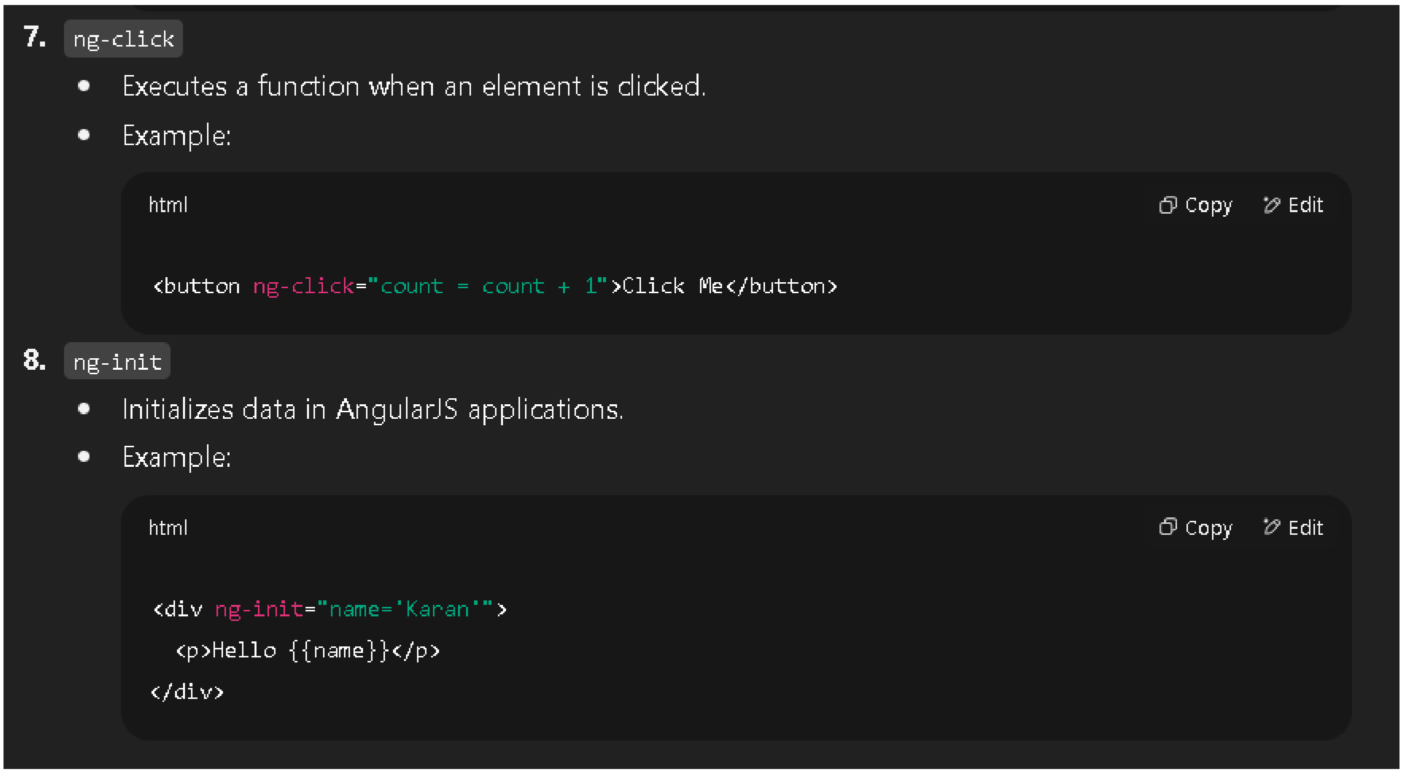
```html
<ul>
  <li ng-repeat="x in names">{{x}}</li>
</ul>
```

5. `ng-if`
   - Removes or recreates elements in the DOM based on a condition.
   - Example:

```html
<p ng-if="isLoggedIn">Welcome User!</p>
```

6. `ng-show` / `ng-hide`
   - Shows or hides an element based on a condition (applies CSS `display:none`).
   - Example:

```html
<p ng-show="isVisible">This is visible</p>
```



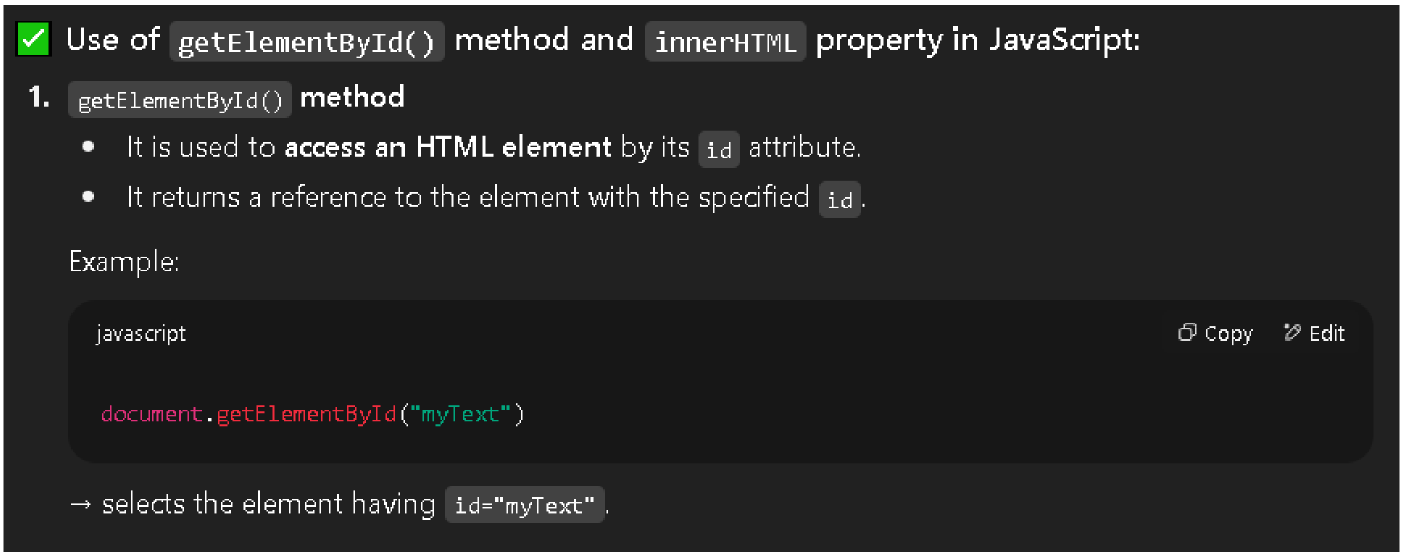
7. `ng-click`
   - Executes a function when an element is clicked.
   - Example:

```html
<button ng-click="count = count + 1">Click Me</button>
```

8. `ng-init`
   - Initializes data in AngularJS applications.
   - Example:

```html
<div ng-init="name='Karan'">
  <p>Hello {{name}}</p>
</div>
```

25) What is the use of getElementById() method and inner HTML property? Also write suitable example.



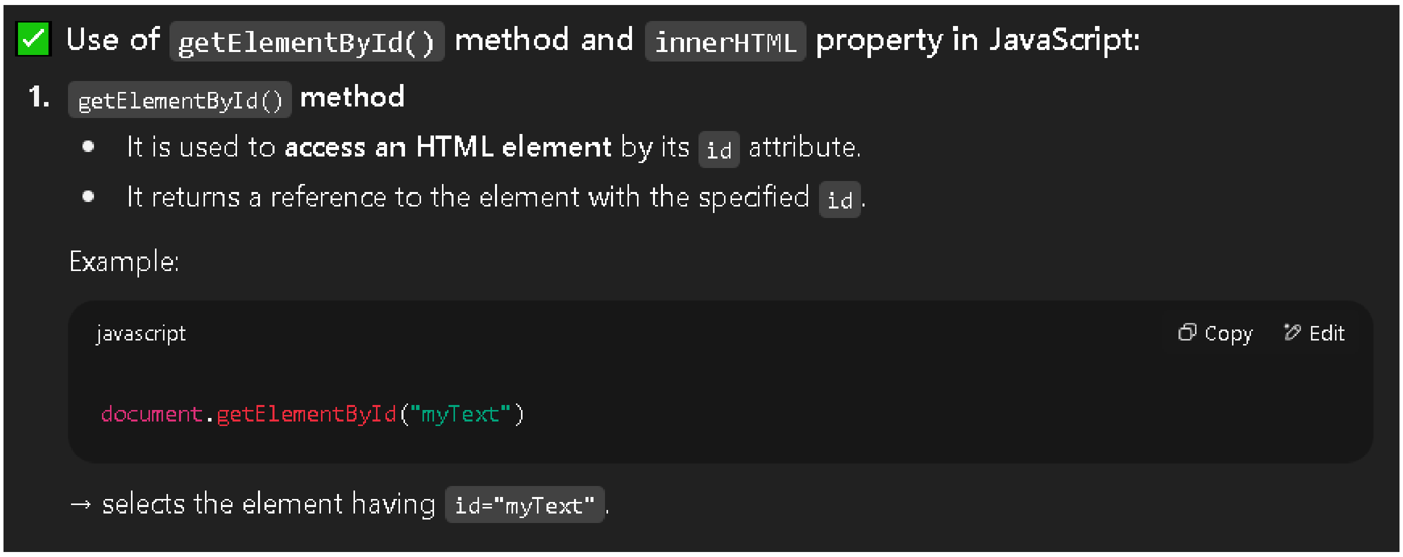
✅ Use of `getElementById()` method and `innerHTML` property in JavaScript:

1. `getElementById()` **method**
   - It is used to **access an HTML element** by its `id` attribute.
   - It returns a reference to the element with the specified `id`.

Example:

```javascript
document.getElementById("myText")
```

→ selects the element having `id="myText"`.



✅ Use of `getElementById()` method and `innerHTML` property in JavaScript:

1. `getElementById()` **method**
   - It is used to **access an HTML element** by its `id` attribute.
   - It returns a reference to the element with the specified `id`.

Example:

```javascript
document.getElementById("myText")
```

→ selects the element having `id="myText"`.

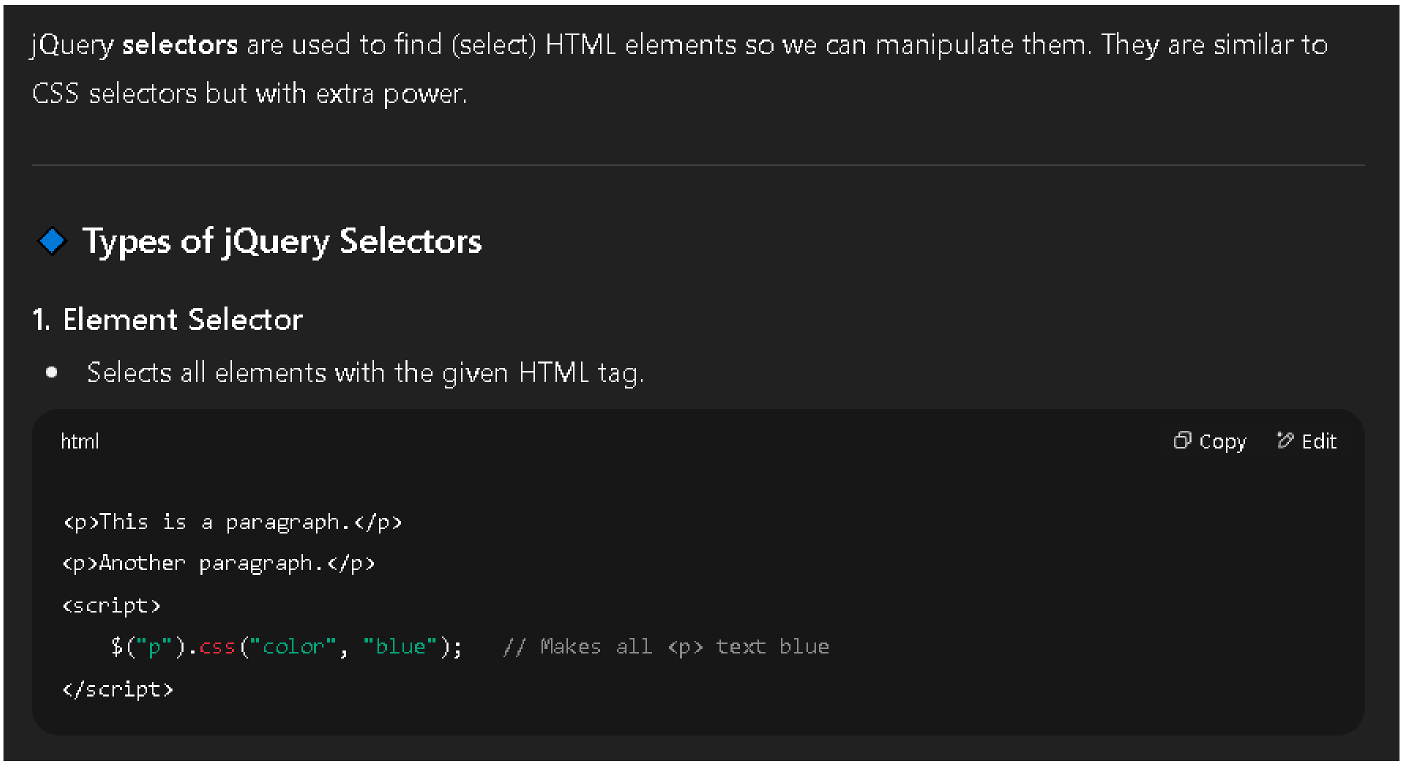
```html
<!DOCTYPE html>
<html>
<head>
    <title>getElementById & innerHTML Example</title>
    <script>
        function showMessage() {
            // Access the value of textbox
            var name = document.getElementById("username").value;

            // Display the message using innerHTML
            document.getElementById("output").innerHTML = "Welcome, " + name + "!";
        }
    </script>
</head>
<body>
    <h2>getElementById & innerHTML Demo</h2>

    Enter your name: <input type="text" id="username">
    <button onclick="showMessage()">Submit</button>

    <p id="output"></p>
</body>
</html>
```

🔷 **Explanation:**

- `getElementById("username")` → fetches the textbox input.
- `.value` → retrieves the value entered by the user.
- `innerHTML` → updates the `<p>` tag content dynamically.

👉 This way we use `getElementById()` **to fetch elements** and `innerHTML` **to modify content**.

26) Explain the different jQuery selector with example for each.



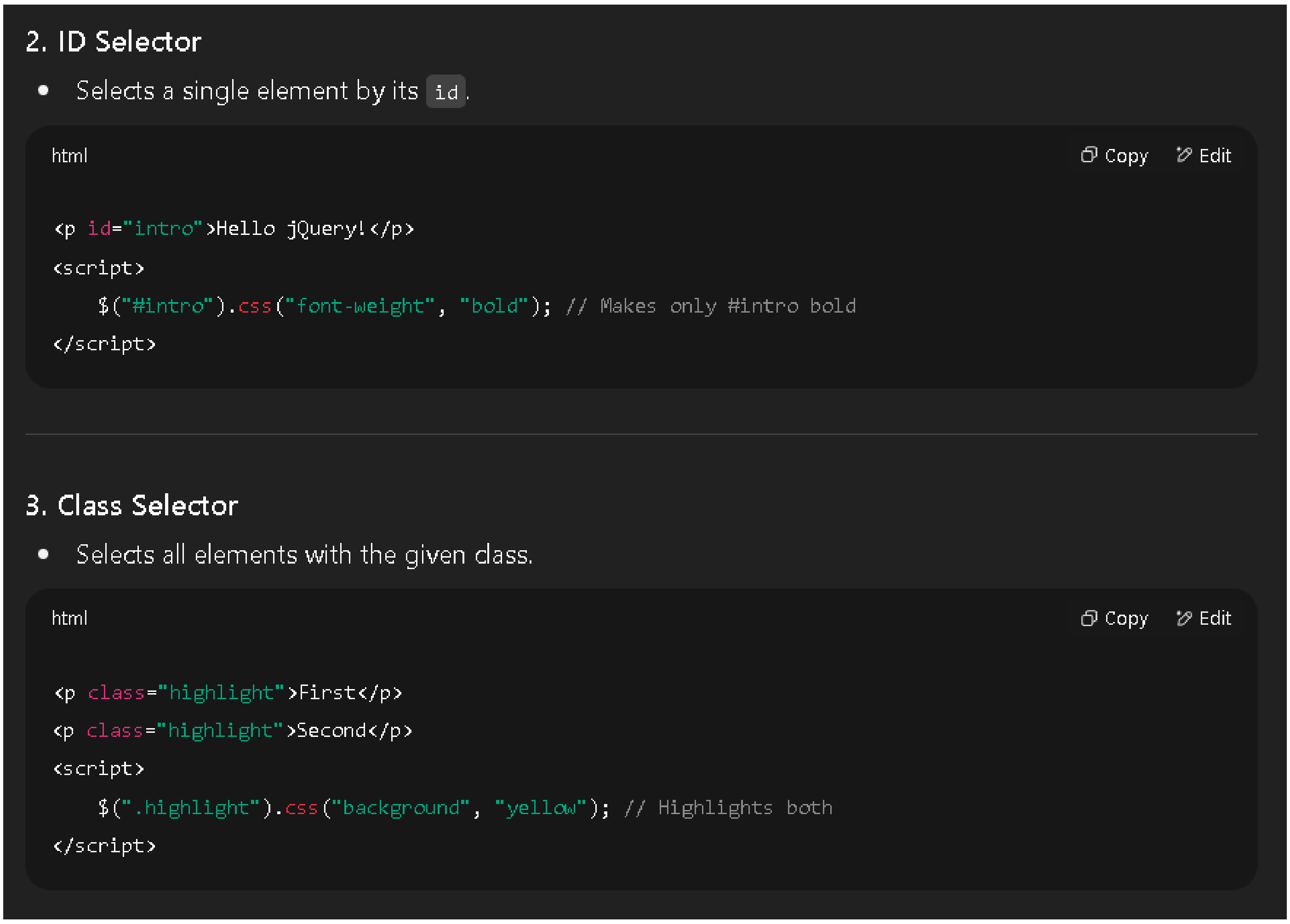
jQuery **selectors** are used to find (select) HTML elements so we can manipulate them. They are similar to CSS selectors but with extra power.

🔷 Types of jQuery Selectors

1. Element Selector

- Selects all elements with the given HTML tag.

```html
<p>This is a paragraph.</p>
<p>Another paragraph.</p>
<script>
    $("p").css("color", "blue");   // Makes all <p> text blue
</script>
```



2. ID Selector

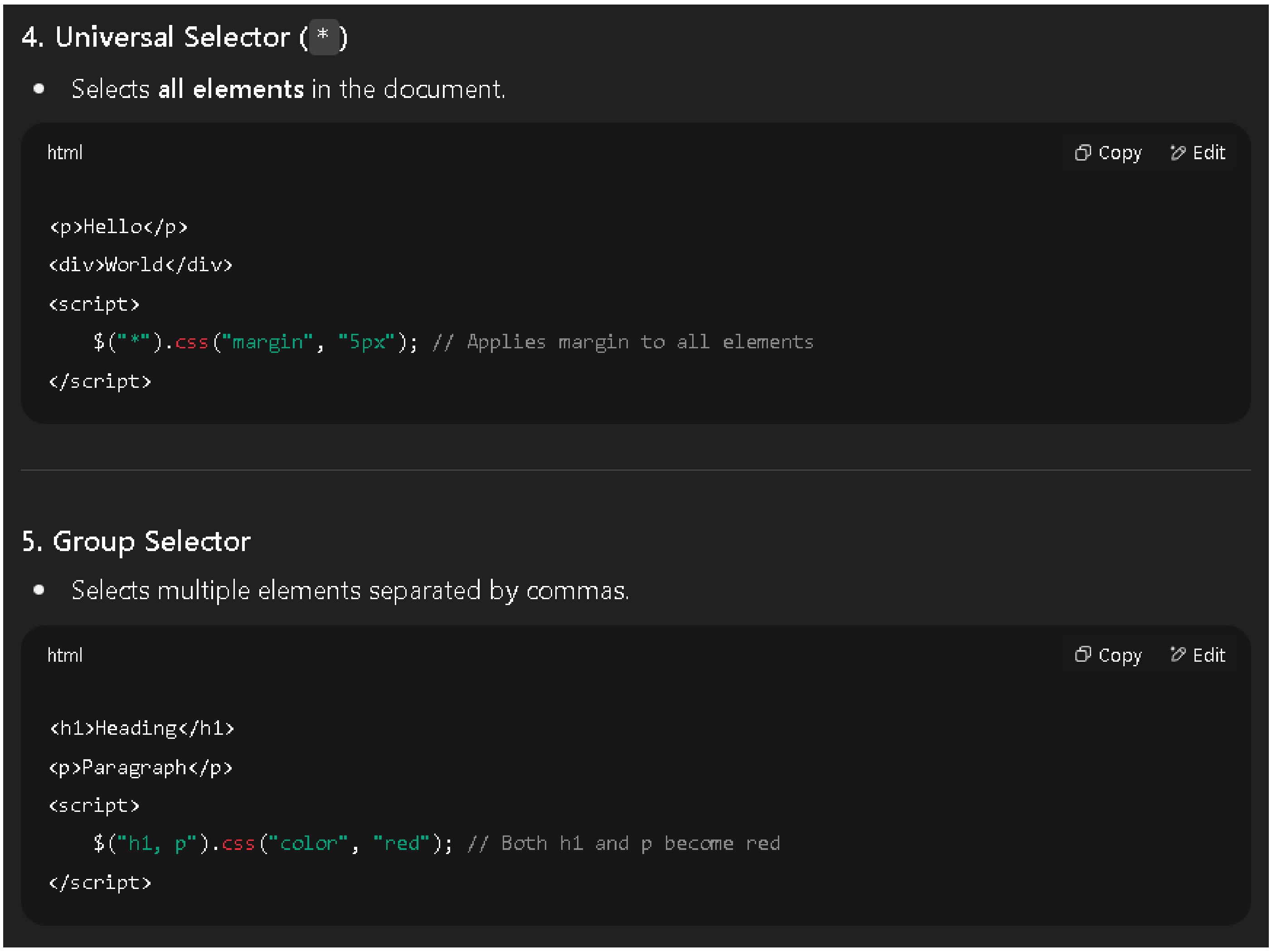
- Selects a single element by its `id`.

```html
<p id="intro">Hello jQuery!</p>
<script>
    $("#intro").css("font-weight", "bold"); // Makes only #intro bold
</script>
```

3. Class Selector

- Selects all elements with the given class.

```html
<p class="highlight">First</p>
<p class="highlight">Second</p>
<script>
    $(".highlight").css("background", "yellow"); // Highlights both
</script>
```



4. Universal Selector (`*`)

- Selects **all elements** in the document.

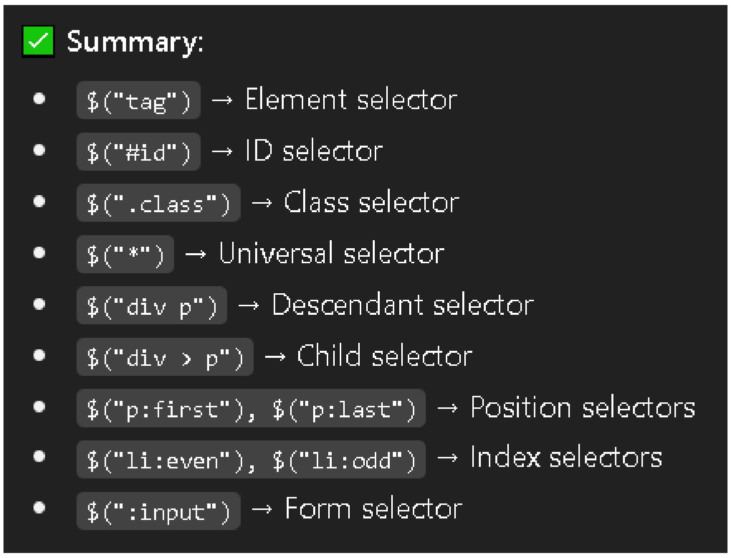
```html
<p>Hello</p>
<div>World</div>
<script>
    $("*").css("margin", "5px"); // Applies margin to all elements
</script>
```

---

5. Group Selector

- Selects multiple elements separated by commas.

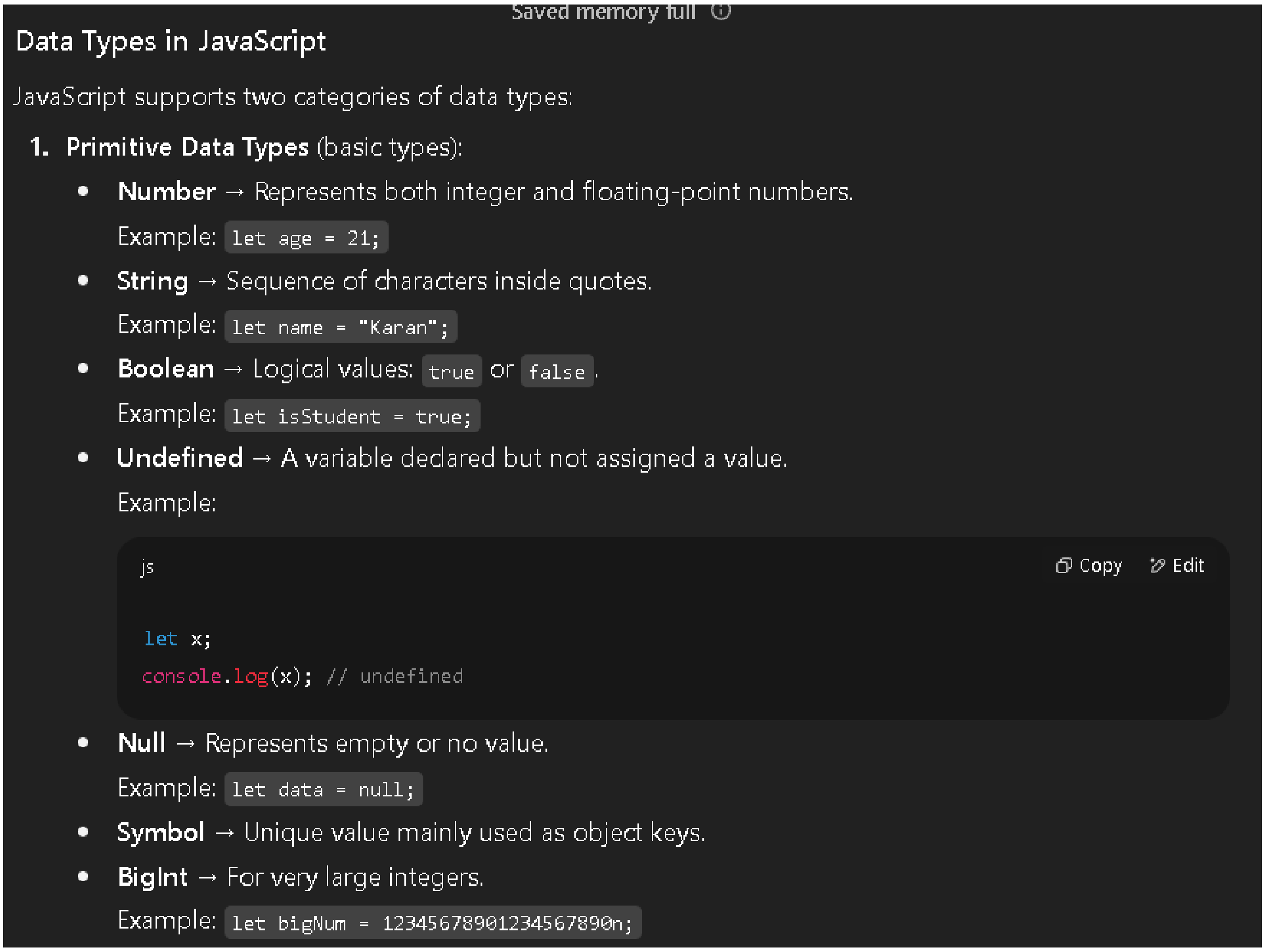
```html
<h1>Heading</h1>
<p>Paragraph</p>
<script>
    $("h1, p").css("color", "red"); // Both h1 and p become red
</script>
```



✅ Summary:

- `$("tag")` → Element selector
- `$("#id")` → ID selector
- `$(".class")` → Class selector
- `$("*")` → Universal selector
- `$("div p")` → Descendant selector
- `$("div > p")` → Child selector
- `$("p:first"), $("p:last")` → Position selectors
- `$("li:even"), $("li:odd")` → Index selectors
- `$(":input")` → Form selector

27) Explain data types & variables in Javascript with suitable example?



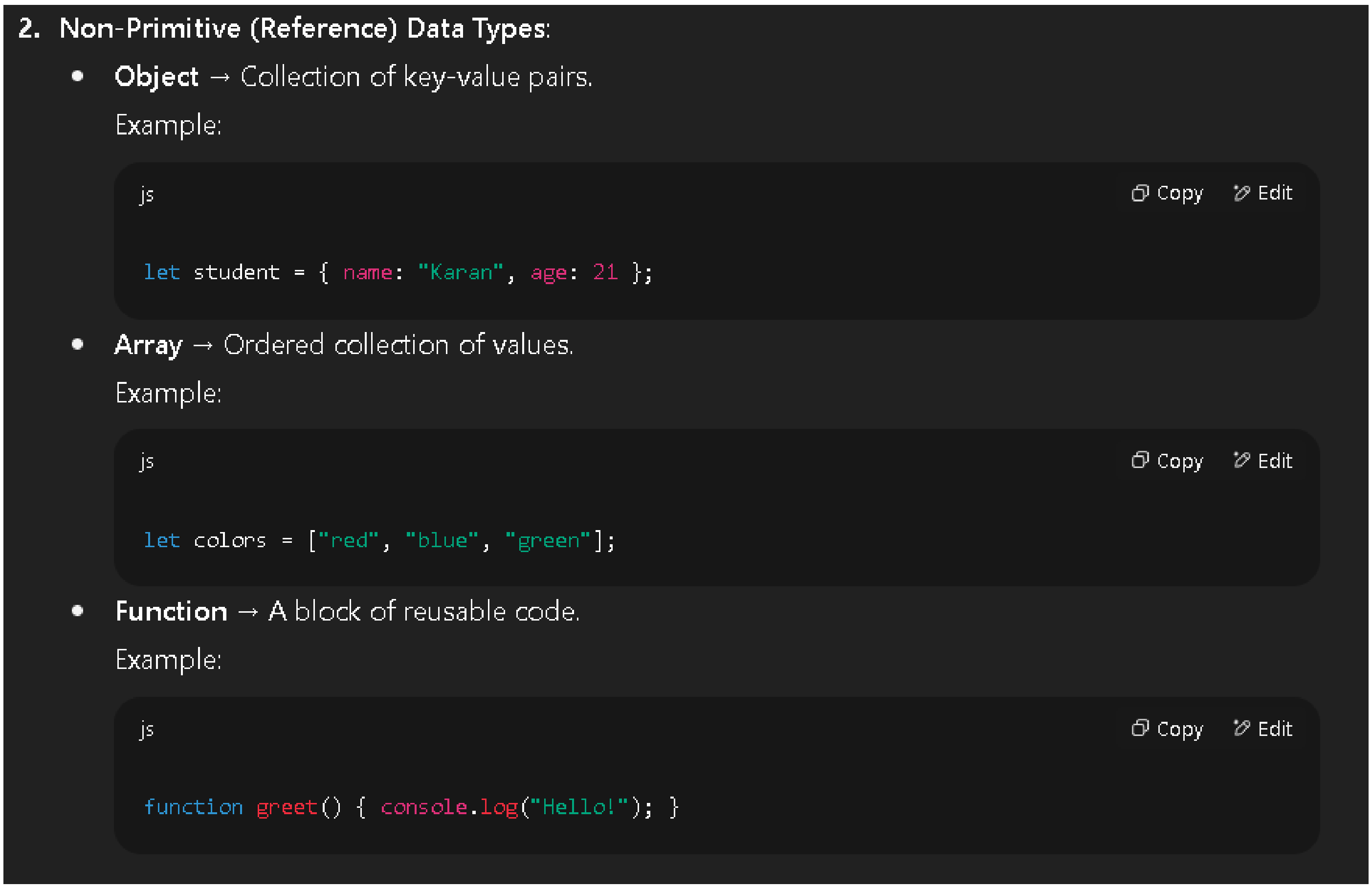
Data Types in JavaScript

JavaScript supports two categories of data types:

1. **Primitive Data Types** (basic types):
   - **Number** → Represents both integer and floating-point numbers.
     Example: `let age = 21;`
   - **String** → Sequence of characters inside quotes.
     Example: `let name = "Karan";`
   - **Boolean** → Logical values: `true` or `false`.
     Example: `let isStudent = true;`
   - **Undefined** → A variable declared but not assigned a value.
     Example:

```js
let x;
console.log(x); // undefined
```

   - **Null** → Represents empty or no value.
     Example: `let data = null;`
   - **Symbol** → Unique value mainly used as object keys.
   - **BigInt** → For very large integers.
     Example: `let bigNum = 1234567890123456789 0n;`



2. **Non-Primitive (Reference) Data Types**:
   - **Object** → Collection of key-value pairs.
     Example:

```js
let student = { name: "Karan", age: 21 };
```

   - **Array** → Ordered collection of values.
     Example:

```js
let colors = ["red", "blue", "green"];
```

   - **Function** → A block of reusable code.
     Example:

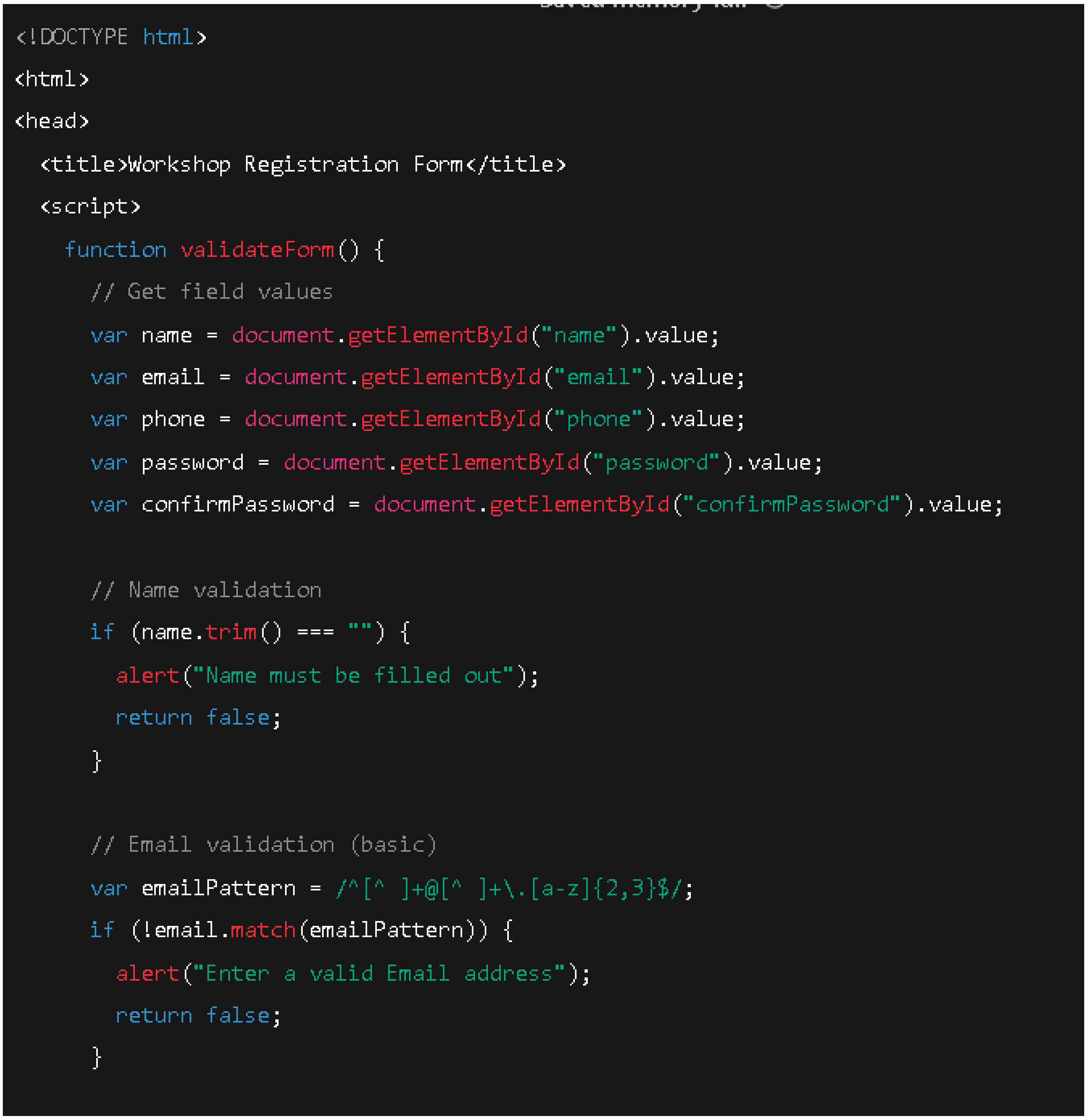
```js
function greet() { console.log("Hello!"); }
```

Variables in JavaScript

- Variables are used to store data values.
- Declared using `var`, `let`, or `const`:
    1. **var** → Function-scoped, old way of declaring.
    2. **let** → Block-scoped, can be reassigned.
    3. **const** → Block-scoped, value cannot be reassigned.

```
<script>
  // Numbers
  let a = 10;
  let b = 20.5;

  // String
  let name = "Karan";

  // Boolean
  let isAdmin = false;

  // Null & Undefined
  let emptyValue = null;
  let notAssigned;

  // Array
  let fruits = ["apple", "banana", "mango"];

  // Object
  let person = { firstName: "John", lastName: "Doe" };

  // Output
  console.log("a =", a, "b =", b);
  console.log("Name:", name);
  console.log("Is Admin?", isAdmin);
  console.log("Null:", emptyValue, "Undefined:", notAssigned);
  console.log("Fruits:", fruits);
  console.log("Person:", person);
</script>
```

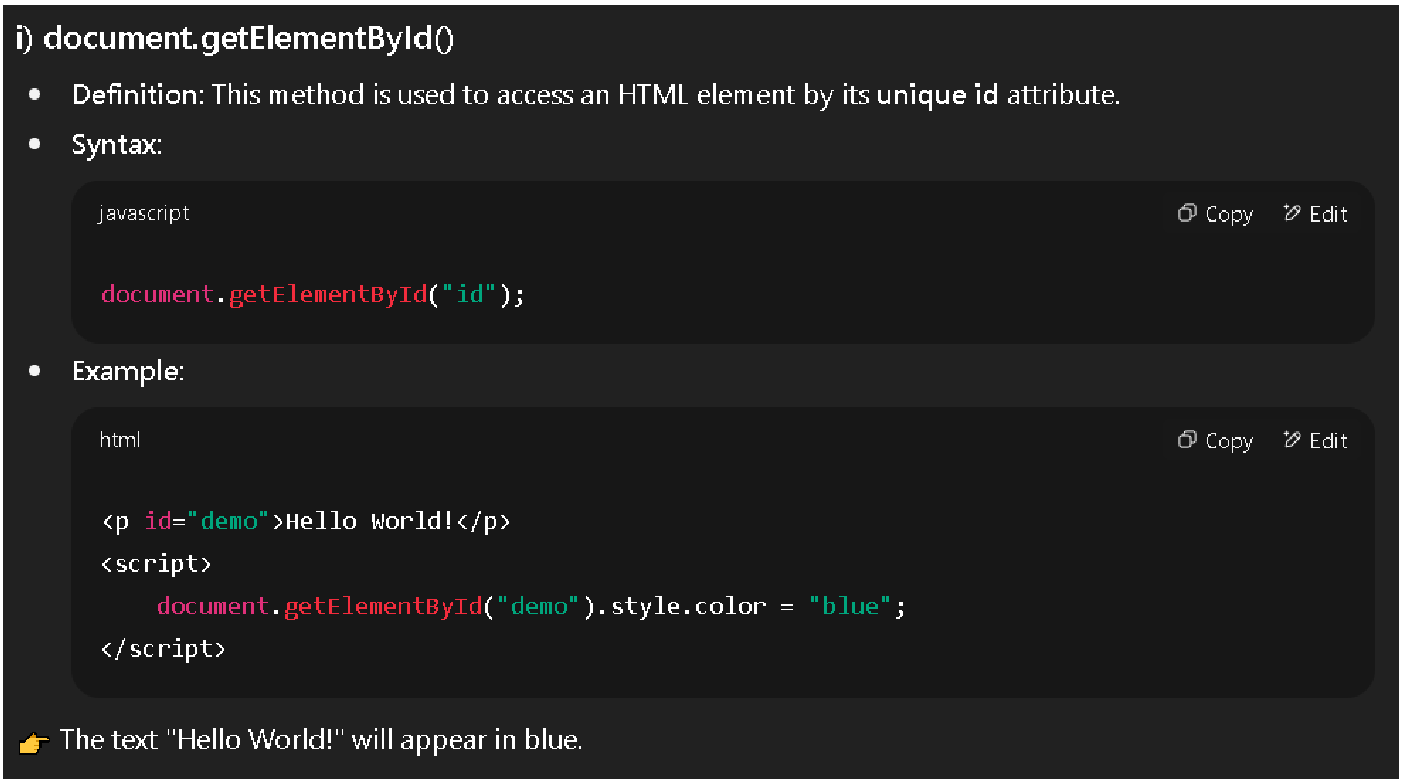
28) Design a form to accept workshop registration details from participants and validate any five fields using Java Script? (Assume suitable structure)



```
<!DOCTYPE html>
<html>
<head>
  <title>Workshop Registration Form</title>
  <script>
    function validateForm() {
      // Get field values
      var name = document.getElementById("name").value;
      var email = document.getElementById("email").value;
      var phone = document.getElementById("phone").value;
      var password = document.getElementById("password").value;
      var confirmPassword = document.getElementById("confirmPassword").value;

      // Name validation
      if (name.trim() === "") {
        alert("Name must be filled out");
        return false;
      }

      // Email validation (basic)
      var emailPattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;
      if (!email.match(emailPattern)) {
        alert("Enter a valid Email address");
        return false;
      }
    // Phone validation (only numbers & 10 digits)
    var phonePattern = /^[0-9]{10}$/;
    if (!phone.match(phonePattern)) {
      alert("Enter a valid 10-digit phone number");
      return false;
    }

    // Password validation
    if (password.length < 6) {
      alert("Password must be at least 6 characters long");
      return false;
    }

    // Confirm password
    if (password !== confirmPassword) {
      alert("Passwords do not match");
      return false;
    }

    alert("Registration Successful!");
    return true;
  }
</script>
```

```html
</head>
<body>
  <h2>Workshop Registration Form</h2>
  <form onsubmit="return validateForm()">
    <label>Name:</label>
    <input type="text" id="name"><br><br>

    <label>Email:</label>
    <input type="text" id="email"><br><br>

    <label>Phone:</label>
    <input type="text" id="phone"><br><br>

    <label>Password:</label>
    <input type="password" id="password"><br><br>

    <label>Confirm Password:</label>
    <input type="password" id="confirmPassword"><br><br>

    <input type="submit" value="Register">
  </form>
</body>
</html>
```

29) Explain the following document object properties with proper syntax? i) document.getElementById() ii) document.getElementByClass() iii) document.getElementByName()



i) document.getElementById()

- Definition: This method is used to access an HTML element by its **unique id** attribute.
- Syntax:

javascript

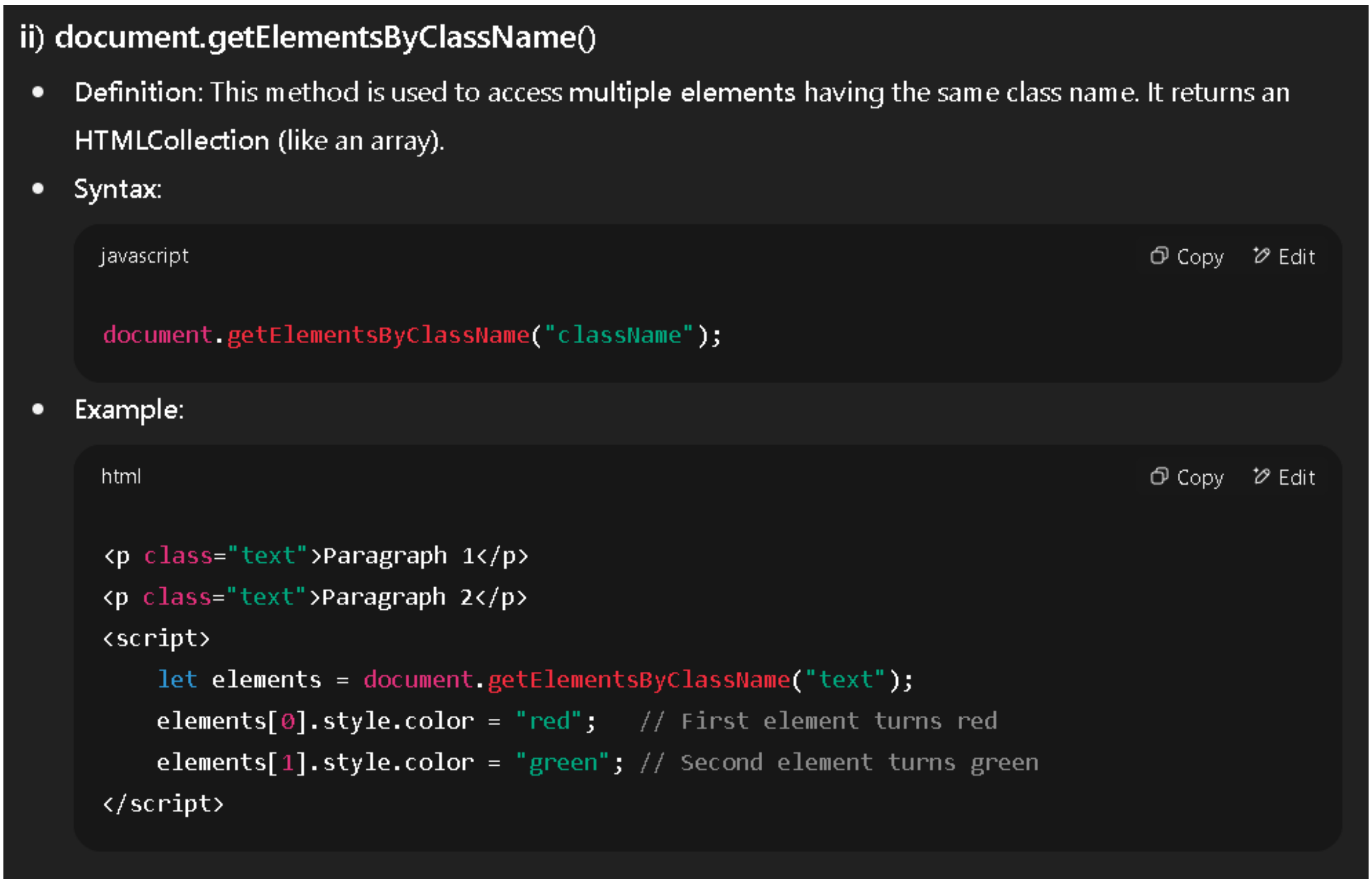
```javascript
document.getElementById("id");
```

- Example:

html

```html
<p id="demo">Hello World!</p>
<script>
    document.getElementById("demo").style.color = "blue";
</script>
```

👉 The text "Hello World!" will appear in blue.



ii) document.getElementsByClassName()

- Definition: This method is used to access **multiple elements** having the same class name. It returns an HTMLCollection (like an array).
- Syntax:

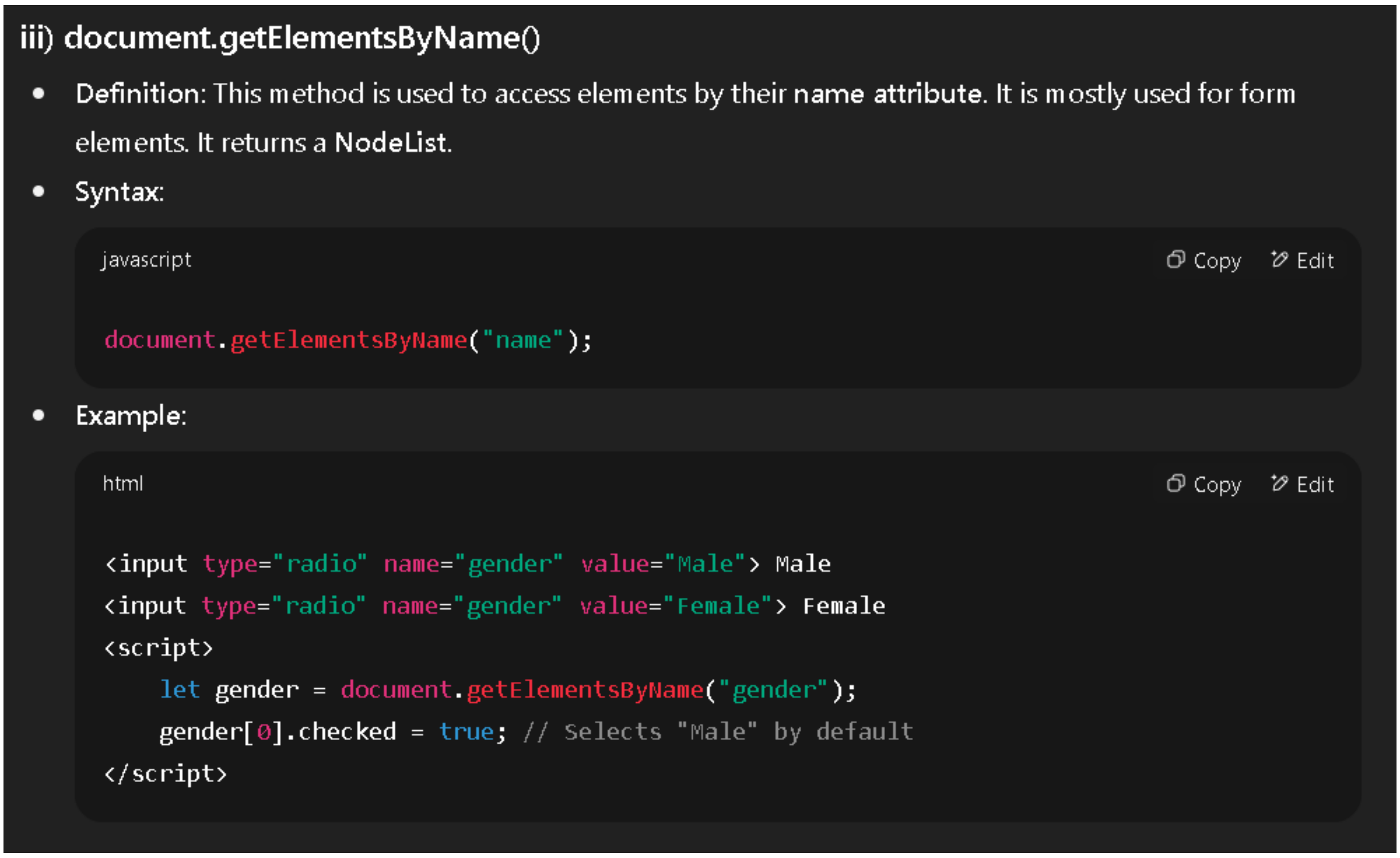
javascript

```javascript
document.getElementsByClassName("className");
```

- Example:

html

```html
<p class="text">Paragraph 1</p>
<p class="text">Paragraph 2</p>
<script>
    let elements = document.getElementsByClassName("text");
    elements[0].style.color = "red";   // First element turns red
    elements[1].style.color = "green"; // Second element turns green
</script>
```



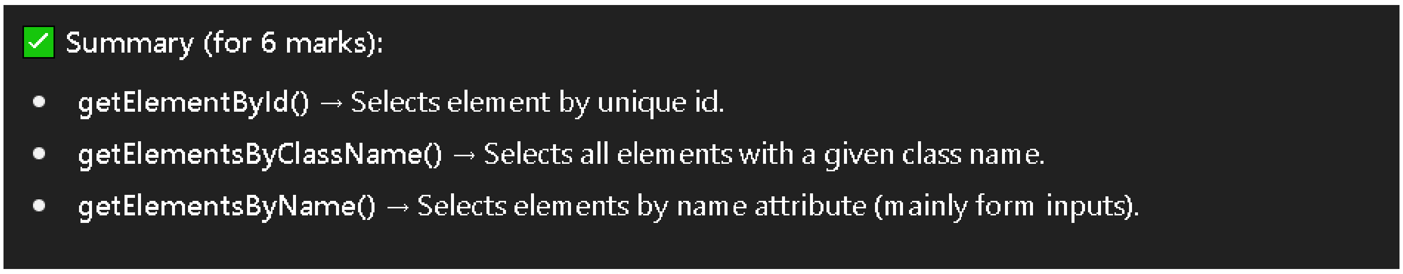
iii) document.getElementsByName()

- Definition: This method is used to access elements by their **name attribute**. It is mostly used for form elements. It returns a **NodeList**.
- Syntax:

```javascript
document.getElementsByName("name");
```

- Example:

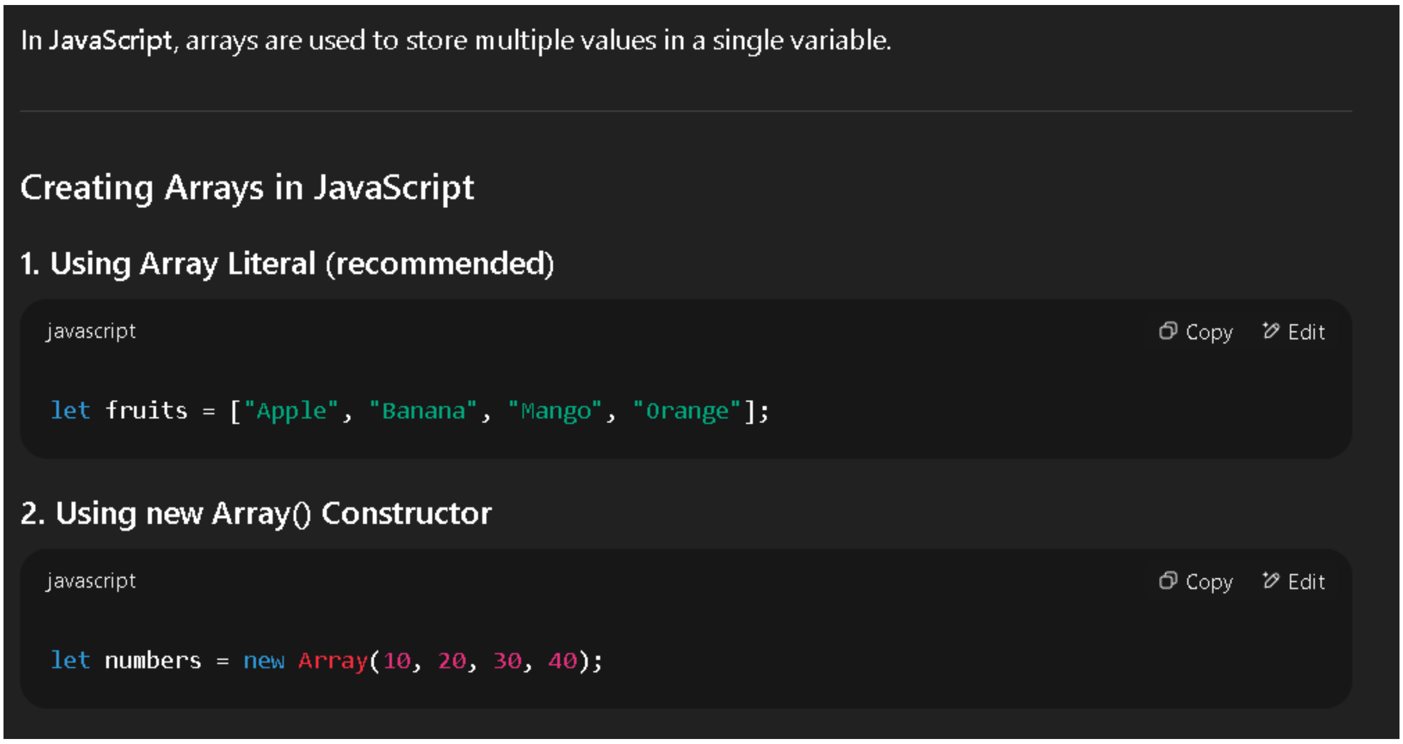
```html
<input type="radio" name="gender" value="Male"> Male
<input type="radio" name="gender" value="Female"> Female
<script>
    let gender = document.getElementsByName("gender");
    gender[0].checked = true; // Selects "Male" by default
</script>
```



✅ Summary (for 6 marks):

- getElementById() → Selects element by unique id.
- getElementsByClassName() → Selects all elements with a given class name.
- getElementsByName() → Selects elements by name attribute (mainly form inputs).

30) How to create array and read elements in Java script.



In **JavaScript**, arrays are used to store multiple values in a single variable.

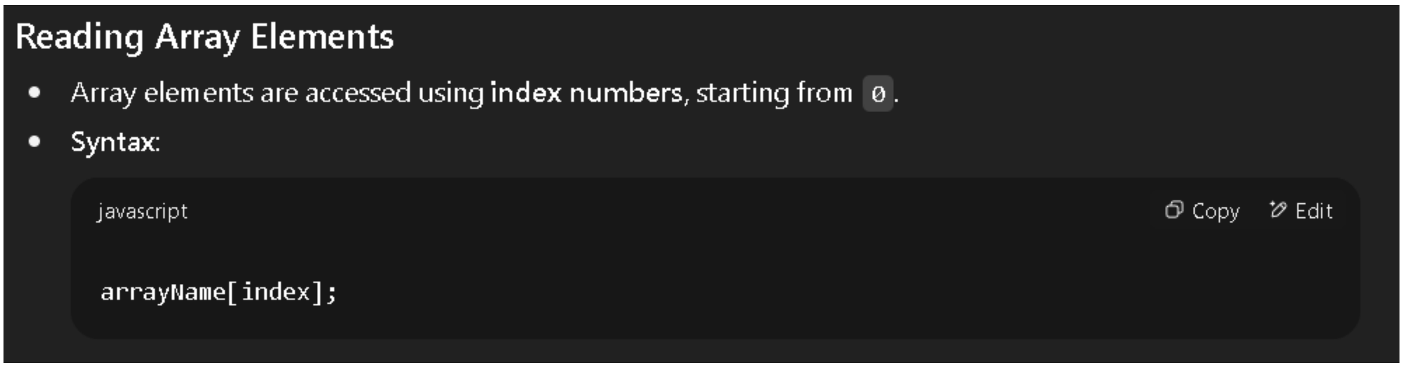
Creating Arrays in JavaScript

1. Using Array Literal (recommended)

```javascript
let fruits = ["Apple", "Banana", "Mango", "Orange"];
```

2. Using new Array() Constructor

```javascript
let numbers = new Array(10, 20, 30, 40);
```



Reading Array Elements

- Array elements are accessed using **index numbers**, starting from `0`.
- **Syntax**:

```javascript
arrayName[index];
```

```html
<!DOCTYPE html>
<html>
<head>
  <title>Array Example</title>
</head>
<body>
  <script>
    // Creating array
    let fruits = ["Apple", "Banana", "Mango", "Orange"];

    // Reading elements
    document.write("First fruit: " + fruits[0] + "<br>");
    document.write("Second fruit: " + fruits[1] + "<br>");

    // Reading all using loop
    document.write("<b>All Fruits:</b><br>");
    for(let i = 0; i < fruits.length; i++) {
        document.write(fruits[i] + "<br>");
    }
  </script>
</body>
</html>
```

Output:

sql Copy Edit

```sql
First fruit: Apple
Second fruit: Banana
All Fruits:
Apple
Banana
Mango
Orange
```

👉 So in short:

- Create array: `let arr = [value1, value2, value3];`
- Read element: `arr[index]`