PWA

Monday, September 1, 2025 1:57 PM

5 MARKS (May be)

1. Describe the structure of a basic HTML document. Include a code example.

An HTML document is like the skeleton of a webpage. It provides structure so browsers know how to render content. Every HTML file starts with a **doctype declaration** which tells the browser the version of HTML being used. The entire page is wrapped inside the <html> tag. Inside it, we have two main sections:

- **Head (<head>)** → Contains metadata like the page title, character set, stylesheets, and scripts. This section is *not visible* to the user but is essential for the browser.
- **Body (<body>)** → This is where all the visible content such as headings, paragraphs, images, and links go.

👉 Without this structure, a browser won't properly understand the document.

Example:

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8">
 <title>My First Page</title>
</head>
<body>
 <h1>Welcome to My Page</h1>
 <p>This is the content that users will see.</p>
</body>
</html>
```

Q2. Differentiate between div and span tags with examples.

☑ **DIV ( )**

1. The tag is a **block-level element**, which means it always starts on a new line.
2. It is used to **group larger sections of content** or layout elements on a page.
3. By default, it **takes the full width available** in the browser.

Example:

```
<div style="background: lightblue; padding: 10px;">
 <h2>This section is inside a div</h2>
 <p>Div is useful for layouts.</p>
</div>
```

☑ **SPAN ()**

1. The tag is an **inline element**, meaning it does not start on a new line.
2. It is used to **style or highlight a small part of text** inside a paragraph or heading.
3. By default, it **only takes as much space as its content needs**.

Example:

```
<p>This is a <span style="color: red;">highlighted word</span> in a sentence.</p>
```

👉 In short: **div is for blocks (layout)**, while **span is for inline text styling**.

Q3. Explain five different types of CSS selectors with syntax and describe how each one is used.

☑ **Definition:**

A **CSS selector** is a pattern used to target specific HTML elements so that styles can be applied to them. Selectors allow developers to control the look and feel of a webpage by connecting CSS rules to HTML elements.

☑ **1. Element Selector**

- It selects elements by their tag name.
- Applies the style to **all elements of that type**.
- Example:

```
p { color: blue; }  /* All paragraphs will be blue */
```

☑ **2. Class Selector**

- It selects elements by their class attribute.
- Classes are reusable across multiple elements.
- Example:

```
.note { background: yellow; }  /* Any element with class="note" */
```

☑ **3. ID Selector**

- It selects an element by its unique ID.
- An ID must be used only once in a page.
- Example:

```
#header { font-size: 24px; }  /* Element with id="header" */
```

☑ **4. Group Selector**

- It allows applying the same style to multiple selectors at once.
- Saves time and avoids repetition in CSS code.
- Example:

```
h1, h2, p { font-family: Arial; }  /* Applies font to all three */
```

☑ **5. Universal Selector**

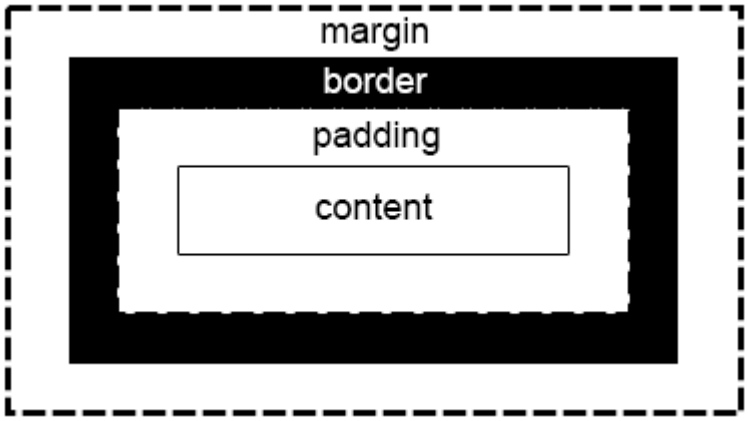
- It selects **all elements** on the page.
- Useful for resetting margins and paddings globally.
- Example:

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

Q4. List and briefly describe the components of the CSS Box Model.

☑ **Definition:**

The **CSS Box Model** is a way of describing how every HTML element is treated as a rectangular box. It explains how content, padding, border, and margin work together to define the space an element occupies on a webpage.



☑ **1. Content**

- This is the innermost part of the box that holds text, images, or other elements.
- It represents the actual data users see inside the element.
- Example: In a <p> tag, the written text is the content.

☑ **2. Padding**

- Padding is the space between the content and the border.
- It increases the clickable/visible area without changing the content.
- Example: padding: 10px; adds extra space inside the element's border.

☑ **3. Border**

- The border surrounds both the content and padding.
- It can be styled with thickness, color, and patterns.
- Example: border: 2px solid black; adds a black line around the element.

☑ **4. Margin**

- Margin is the space outside the border, separating the element from others.
- It controls the distance between two elements.
- Example: margin: 20px; pushes the element away from nearby elements.

☑ **Example (Visualization):**

```
div {
  margin: 20px;     /* Outer space */
  border: 2px solid black;  /* Border line */
  padding: 15px;    /* Inner spacing */
  width: 200px;     /* Content width */
}
```

Q5. What is the purpose of the display property in CSS? List any four common values and explain their function.

☑ **Definition:**

The **display property** in CSS determines how an element is shown on a webpage. It controls whether an element behaves like a block, inline, inline-block, or disappears completely. Choosing the right display type is important for designing page layouts and arranging elements properly.

☑ **1. display: block**

- A block-level element always starts on a new line.
- It takes up the full width of its parent container.
- Commonly used for structural elements like , <p>, and <h1>.

Example:

```
div {
  display: block;
  background: lightblue;
}
```

☑ **2. display: inline**

- An inline element does not start on a new line.
- It only takes as much width as the content needs.
- Commonly used for text-related elements like , <a>, or .

Example:

```
span {
  display: inline;
  color: red;
}
```

☑ **3. display: inline-block**

- Acts like an inline element but allows width and height to be applied.
- Useful for creating buttons or menus in a row.
- Combines the flexibility of inline with the control of block.

Example:

```
button {
  display: inline-block;
  width: 120px;
  height: 40px;
}
```

☑ **4. display: none**

- Completely hides the element from the page.
- The element does not occupy any space in the layout.
- Useful for conditional content (like hiding menus until clicked).

Example:

```
.hidden {
  display: none;
}
```

Q6. What is a CSS transition? List and explain any four transition-related properties with example.

☑ **Definition:**

A **CSS transition** allows changes in CSS properties to happen smoothly over a specified time instead of instantly. For example, when hovering over a button, its color can gradually change rather than switching immediately. Transitions improve user experience by making interactions feel more natural.

☑ **1. transition-property**

- Defines which CSS property will be animated (e.g., color, background-color, width).
- Without this, no animation will occur.
- Example:

```
button {
  transition-property: background-color;
```

}

☑ **2. transition-duration**

- Sets how long the transition should take.
- Accepts values in seconds (s) or milliseconds (ms).
- Example:

```
button {
 transition-duration: 2s;
}
```

☑ **3. transition-timing-function**

- Controls the speed curve of the animation.
- Common values:
  - ease (slow → fast → slow),
  - linear (constant speed),
  - ease-in (slow start),
  - ease-out (slow end).
- Example:

```
button {
 transition-timing-function: ease-in-out;
}
```

☑ **4. transition-delay**

- Sets a delay before the transition starts.
- Useful when you want the effect to wait before playing.
- Example:

```
button {
 transition-delay: 0.5s;
}
```

☑ **Full Example:**

```
button {
 background-color: blue;
 color: white;
 transition-property: background-color, transform;
 transition-duration: 1s;
 transition-timing-function: ease;
 transition-delay: 0s;
}
button:hover {
 background-color: red;
 transform: scale(1.1);
}
```

10 marks:(MAY BE)

Q7. Explain how forms work in HTML. Design a detailed registration form using various input types such as text, password, radio, checkbox, email, and dropdown. Add submit and reset buttons and

describe each part.

☑ **Definition:**

An **HTML form** is used to collect user input. It acts as a container for input fields like text boxes, checkboxes, and buttons. When submitted, the form sends the collected data to a server or script for processing.

☑ **How forms work (Key Points):**

1. Forms are created using the <form> tag, which defines the action (where data goes) and method (how data is sent, e.g., GET/POST).
2. Different <input> types allow users to enter data in specific formats (e.g., text, email, password).
3. Additional elements like <select> (dropdown), <textarea> (multi-line input), and <button> help create interactive, user-friendly forms.

☑ **Example – Registration Form:**

```
<form action="register.php" method="post">
  <h2>Registration Form</h2>

  <!-- Text Input -->
  Name: <input type="text" name="username" required><br><br>

  <!-- Email Input -->
  Email: <input type="email" name="email" required><br><br>

  <!-- Password Input -->
  Password: <input type="password" name="password" required><br><br>

  <!-- Radio Buttons -->
  Gender:
  <input type="radio" name="gender" value="male"> Male
  <input type="radio" name="gender" value="female"> Female<br><br>

  <!-- Checkbox -->
  Hobbies:
  <input type="checkbox" name="hobby" value="sports"> Sports
  <input type="checkbox" name="hobby" value="music"> Music
  <input type="checkbox" name="hobby" value="reading"> Reading<br><br>

  <!-- Dropdown -->
  Country:
  <select name="country">
    <option value="india">India</option>
    <option value="usa">USA</option>
    <option value="uk">UK</option>
  </select><br><br>

  <!-- Buttons -->
  <input type="submit" value="Register">
  <input type="reset" value="Clear">
</form>
```

Q8. Explain the importance of Progressive Web Apps (PWA) in modern development. How do they improve user experience

compared to traditional websites? Create a basic layout of a PWA homepage using HTML and internal CSS.

☑ **Definition:**

A **Progressive Web App (PWA)** is a type of web application that uses modern web technologies to deliver an app-like experience. PWAs work offline, load quickly, and can be installed on devices like native apps.

☑ **Importance of PWAs (Key Points):**

1. **Offline Support:** PWAs can work without internet using service workers.
2. **Installable:** Users can add PWAs to their home screen like mobile apps.
3. **Faster Performance:** PWAs load quickly and feel smoother than traditional websites.
4. **Cross-platform:** They work on all devices (desktop, mobile, tablets) with the same code.
5. **Better Engagement:** PWAs support push notifications and background sync.

☑ **Example – Basic PWA Homepage Layout:**

```
<!DOCTYPE html>
<html>
<head>
 <title>PWA Homepage</title>
 <style>
  body { font-family: Arial; margin:0; padding:0; text-align:center; }
  header { background:#0073e6; color:white; padding:20px; }
  nav a { margin:10px; text-decoration:none; color:white; }
  section { padding:20px; }
  footer { background:#333; color:white; padding:10px; }
 </style>
</head>
<body>
 <header>
  <h1>My PWA</h1>
  <nav>
   <a href="#">Home</a>
   <a href="#">About</a>
   <a href="#">Services</a>
  </nav>
 </header>
 <section>
  <h2>Welcome to My Progressive Web App</h2>
  <p>Fast, reliable, and installable on your device.</p>
 </section>
 <footer>
  <p>&copy; 2025 My PWA</p>
 </footer>
</body>
</html>
```

Q9. How does CSS positioning work (static, relative, absolute, fixed, and sticky)? Explain each type and provide examples showing how they influence element placement on a webpage.

☑ **Main Definition:**

The **CSS position property** determines how an element is placed on a webpage. It decides whether an element stays in the normal document flow, moves relative to its parent, or sticks to the screen

during scrolling.

☑ **1. static (default)**
Definition: Static positioning is the default behavior where elements appear in the normal document flow without any special positioning.

```
p { position: static; }
```

☑ **2. relative**
Definition: Relative positioning places an element relative to its original position, allowing small adjustments with top, left, right, or bottom.

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

☑ **3. absolute**
Definition: Absolute positioning removes the element from the normal flow and positions it relative to the nearest ancestor with a position other than static.

```
.box { position: absolute; top: 50px; right: 30px; }
```

☑ **4. fixed**
Definition: Fixed positioning attaches an element to the browser window, so it stays in the same place even when the user scrolls.

```
nav { position: fixed; top: 0; width: 100%; }
```

☑ **5. sticky**
Definition: Sticky positioning is a hybrid of relative and fixed; the element behaves like relative until a certain scroll point, after which it sticks like fixed.

```
header { position: sticky; top: 0; background: yellow; }
```

Q10. Create a CSS stylesheet that demonstrates the use of RGB, RGBA, HEX, HSL, and HSLA to style different HTML elements.

☑ **Main Definition:**
In CSS, colors can be written in different formats depending on how precise or flexible you want to be. The most common formats are **RGB, RGBA, HEX, HSL, and HSLA**. These notations allow developers to control colors, brightness, and transparency when styling web elements.

☑ **1. RGB (Red, Green, Blue)**
Definition: Defines colors using red, green, and blue values, each ranging from 0 to 255. By mixing these three, millions of colors can be created.
Example:

```
h1 { color: rgb(255, 0, 0); }  /* Pure red */
```

☑ **2. RGBA (RGB + Alpha)**
Definition: Similar to RGB but with an extra **alpha channel** for transparency, where 0 = fully transparent and 1 = fully opaque.
Example:

```
p { background-color: rgba(0, 0, 255, 0.5); }  /* Semi-transparent blue */
```

☑ **3. HEX (Hexadecimal)**

Definition: Uses six hexadecimal digits (#RRGGBB) to represent red, green, and blue values. It's compact and widely used in web design.
Example:

```
div { border: 2px solid #00ff00; }  /* Green */
```

☑ **4. HSL (Hue, Saturation, Lightness)**
Definition: Defines colors using **hue (0–360° on the color wheel)**, **saturation (%)**, and **lightness (%)**. It is easier for designers to adjust shades.
Example:

```
h2 { color: hsl(200, 100%, 50%); }  /* Bright blue */
```

☑ **5. HSLA (HSL + Alpha)**
Definition: Adds transparency to HSL by including an **alpha channel** (0–1). This gives more control over opacity.
Example:

```
section { background-color: hsla(120, 100%, 50%, 0.3); }  /* Transparent green */
```

☑ **Full Example CSS:**

```
h1 { color: rgb(255, 0, 0); }              /* RGB - Red */
p { background-color: rgba(0, 0, 255, 0.5); } /* RGBA - Semi-transparent Blue */
div { border: 2px solid #00ff00; }         /* HEX - Green */
h2 { color: hsl(200, 100%, 50%); }         /* HSL - Bright Blue */
section { background-color: hsla(120, 100%, 50%, 0.3); } /* HSLA - Transparent Green */
```

☑ **Example HTML:**

```
<h1>RGB Example</h1>
<p>RGBA Example</p>
<div>HEX Example</div>
<h2>HSL Example</h2>
<section>HSLA Example</section>
```