Hall Ticket Number:

IV/IV B.Tech (Regular) DEGREE EXAMINATION

December, 2024 — **Common to CE,EC,EE, EIE& ME**

Seventh Semester — **Web Technologies**

Time: Three Hours — **Maximum:**70 Marks

Answer question 1 compulsorily. **(14X1 = 14Marks)**

Answer one question from each unit. **(4X14=56Marks)**

			CO	BL	M
1	a)	Define HTML.	CO1	L1	1M
	b)	Write the syntax for creating a hyper link in HTML.	CO1	L1	1M
	c)	Differentiate between ***cell padding*** and ***cell spacing.***	CO1	L1	1M
	d)	List any four character entities.	CO1	L1	1M
	e)	Give the purpose of CSS.	CO2	L1	1M
	f)	Mention any four CSS Selectors.	CO2	L1	1M
	g)	Name any three pre-defined functions in Java Script.	CO2	L1	1M
	h)	Outline the syntax for creating an event in Java Script.	CO2	L1	1M
	i)	Define Document Object Model.	CO3	L1	1M
	j)	List any three document object methods.	CO3	L1	1M
	k)	What are DOM Levels?	CO3	L1	1M
	l)	Distinguish between ***Well-formed*** and ***Valid XML***.	CO4	L2	1M
	m)	Write the significance of **CDATA** and **PCDATA**.	CO4	L1	1M
	n)	List any three XSLT elements.	CO4	L1	1M
		Unit-I			
2	a)	How do block-level and inline elements differ in HTML? Provide examples of their usage in webpage layout.	CO1	L2	7M
	b)	Describe the key elements and attributes used in creating image maps and explain with an example.	CO1	L2	7M
		(OR)			
3	a)	Explain the role of hyperlinks in HTML5. Describe the <a> tag and how to create both internal and external links with suitable examples.	CO1	L2	7M
	b)	Create a Student Registration form which contains the following fields: Name, Date of birth, Gender, Languages Known, Phone number, Address, Educational Qualifications ,Contact Number and Submit button.	CO1	L3	7M
		Unit-II			
4	a)	Discuss the three different ways of applying CSS to a webpage (inline, internal, and external). Provide examples of each method and explain the advantages and disadvantages.	CO2	L2	7M
	b)	Discuss how to apply backgrounds and borders in CSS. Explain the different propertieshelps to set a background and border to a web page.	CO2	L2	7M
		(OR)			
5	a)	Discus in detail about various Dialog boxes used in Java Script with an example.	CO2	L2	7M
	b)	Write a Java Script program to demonstrate simple Animation.	CO2	L3	7M
		Unit-III			
6		Explain in detail about the following Java Script Objects with suitable examples.	CO3	L2	14M
		i) Array Object ii) String Object iii) Date Object			
		(OR)			
7	a)	How does JavaScript use the document object to access, modify, and delete HTML elements on a webpage? Illustrate with examples of creating and removing elements dynamically.	CO3	L3	7M
	b)	Discuss the hierarchical structure of the DOM, which represents the document as a tree of nodesand alsoexplain the types of nodes (Element, Text, Attribute, etc.).	CO3	L2	7M

		Unit-IV			
8	a)	Explain in detail about Internal DTD and External DTD with suitable examples.	CO4	L2	7M
	b)	Write an XML document to represent a book catalog containing at least five books with attributes like title, author, publisher and price.	CO4	L3	7M
		(OR)			
9	a)	Differentiate between DTD and XSD with an example.	CO4	L2	7M
	b)	Create an XML schema (XSD) to validate an XML file containing employee details.	CO4	L3	7M

Scheme & Solutions

1. a) Define HTML

Ans: HTML stands for Hypertext Markup Language. It's a text-based language that defines the structure and meaning of web content. HTML is used to tell a web browser how to display text, images, and other content on a web page. -→ 1M

b) Write the syntax for creating a hyper link in HTML.

Ans: The <a> tag defines a hyperlink, which is used to link from one page to another. The most important attribute of the <a> element is the href attribute, which indicates the link's destination.

Syntax: <a href="http://Internet URL goes here."> -→ 1M

c) Differentiate between *cell padding* and *cell spacing.*

Ans: Cell padding is the space between the content and border of a cell, while cell spacing is the space between adjacent cells. Both are used to format tables and improve their layout and aesthetics.

-→ 1M

d) List any four-character entities.

Ans: Here are some examples of character entities:

- < Less than sign
- > Greater than sign
- & Ampersand
- " Double quotation mark -→ 1M

e) Give the purpose of CSS.

Ans: Cascading Style Sheets (CSS) is a coding language that controls the appearance of a website. Its purpose is to separate the content of a website from its presentation. This makes it easier to design and maintain a website. -→ 1M

f) Mention any four CSS Selectors.

Ans: Basic selectors in CSS are simple tools used to target specific HTML elements for styling. These include selecting by element name (e.g., h1), class (. class Name), ID (#idName), or universally (* for all elements). Any four -→ 1M

g) Name any three pre-defined functions in Java Script.

Ans: Some of the pre-defined functions in Java Script are:alert(), confirm(), prompt() and parseInt().

Any four -→ 1M

h) Outline the syntax for creating an event in Java Script.

Ans: Syntax for defining event is **onEventname="eventdescription"** -→ 1M

i) Define Document Object Model.

Ans: Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document. -→1M

j) List any three document object methods.

Ans: Some of the methods of document object are:

- getElementById()
- getElementsByTagName()
- createElement()
- appendChild()
- write()

any valid three methods →1M

k) What are DOM Levels?

Ans: DOM Levels are the versions of the specification for defining how the Document Object Model should work. DOM Level 1 defines the core elements of the Document Object Model. DOM Level 2 extends those elements and adds events. DOM Level 3 extends DOM level 2 and adds more elements and events. --→ 1M

l) Distinguish between *Well-formed* and *Valid XML*.

Ans: Well-formed XML follows XML grammar rules and syntax, while valid XML is well-formed and also conforms to a document type definition (DTD). ---→ 1M

m) Write the significance of CDATA and PCDATA.

Ans: PCDATA is text that will be parsed by a parser. Tags inside the text will be treated as markup and entities will be expanded. CDATA is text that will not be parsed by a parser. Tags inside the text will not be treated as markup and entities will not be expanded. ---→ 1M

n) List any three XSLT elements.

Ans: Some of the XSLT elements are: xsl: for-each, xsl: apply-templates, xsl:choose, xsl:element,etc.

Any valid three elements→1M

Unit-I

1. a) How do block level and inline elements differ in html? Provide examples of their usage in web development. [7M]

Ans: In HTML, **block-level elements** and **inline elements** serve different purposes and have distinct characteristics, which affect how they behave in a web layout.

Block-level Elements

Characteristics:

1. **Occupy the full width** of their parent container, stacking one on top of the other (like a block).
2. **Start on a new line** in the document flow.
3. Can contain other block-level elements or inline elements.
4. Typically used for structural purposes.

Common Examples:

-
-
- <h1> to <h6>
- and <ol> (and their <li> items)
- <section>, <article>, <header>, <footer>

Example Usage:

<h1>This is a heading</h1>

This is a paragraph within a block-level element (div).

In this example, the groups a heading and a paragraph. Each block-level element (e.g., <h1> and) will take up its own line. --→ 3M

Inline Elements

Characteristics:

1. **Do not start on a new line** and only take up as much width as necessary.
2. **Stay inline** with the content surrounding them.
3. Cannot contain block-level elements but can contain other inline elements.
4. Typically used for styling or small content pieces.

Common Examples:

-
- <a>
- , , <b>, <i>
- <img>

Example Usage:

This is a bold word and an inline link in a sentence.

In this example, the and <a> elements are inline and do not disrupt the flow of the paragraph.

Key Differences

Aspect	Block-Level Elements	Inline Elements
Display Behavior	Start on a new line	Stay in the same line
Width	Occupy full width by default	Occupy only necessary width
Containment	Can contain block or inline	Can only contain inline

--→ 3M

Practical Usage in Web Development

- **Block-level elements** are ideal for defining sections or layouts:

<section>

<h2>Section Title</h2>

This is a section of the website.

</section>

- **Inline elements** are used for styling or linking parts of content:

Click here to visit the website. --→ 1M

2. b) Describe the key elements and attributes used in creating image maps and explain with an example. [7M]

An **image map** in HTML allows you to define clickable areas on an image, linking each area to a specific URL or triggering specific actions. This is achieved using the <map> and <area> elements.

Key Elements and Attributes for Image Maps

1. **<img> Element**:
    - Used to display the image.
    - Must include the usemap attribute to associate the image with a <map> element.

Attributes:

- src: Specifies the image file.
- alt: Provides alternative text for accessibility.
- usemap: Links the image to the <map> element (e.g., usemap="#mapName").

2. **<map> Element**:
    - Defines the image map and its clickable areas.
    - Requires a name attribute, which links it to the usemap attribute of the <img> element.

Attributes:

- name: Specifies the name of the map (e.g., name="mapName").

3. **<area> Element**:
    - Defines specific clickable areas within the image map.

Attributes:

- shape: Defines the shape of the clickable area. Possible values:
    - rect: Rectangle.
    - circle: Circle.
    - poly: Polygon (multiple points).
- coords: Specifies the coordinates for the clickable area. Format depends on the shape:
    - For rect: x1,y1,x2,y2 (top-left and bottom-right corners).
    - For circle: x,y,r (center coordinates and radius).
    - For poly: x1,y1,x2,y2,... (list of points).
- href: Specifies the URL to link to when the area is clicked.
- alt: Provides alternative text for the area. --→ 4M

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Image Map Example</title>
</head>
<body>
<!-- Image with a usemap attribute -->
<imgsrc="world-map.jpg" alt="World Map" usemap="#worldMap" width="600">

<!-- Defining the map -->
<map name="worldMap">
<!-- Rectangular area linking to North America -->
<area shape="rect" coords="50,50,150,150" href="https://example.com/north-america" alt="North
America">
<!-- Circular area linking to Europe -->
<area shape="circle" coords="300,120,50" href="https://example.com/europe" alt="Europe">
<!-- Polygonal area linking to Australia -->
<area shape="poly" coords="450,300,480,320,470,350,440,340" href="https://example.com/australia"
alt="Australia">
</map>
</body>
</html>
```

----→ 3M

(OR)

3. a) Explain the role of hyperlinks in HTML5. Describe the <a> tag and how to create both internal and external links with suitable examples. [7M]

Ans: **Role of Hyperlinks in HTML5**

Hyperlinks are a fundamental feature of HTML that allows users to navigate between different resources. These resources can be:

- **Internal links**: Navigate to a different section or page within the same website.
- **External links**: Navigate to resources on other websites.
- **Other resources**: Link to email addresses, phone numbers, downloadable files, or scripts.

The <a> (anchor) tag is used to create hyperlinks in HTML. **--→ 1M**

The <a> Tag

The <a> tag defines a hyperlink and is one of the most commonly used elements in HTML. It has several attributes to define its behaviour and appearance.

Attributes of <a>

1. **href**: Specifies the URL or path of the linked resource.
    - Example: href="https://example.com"
2. **target**: Defines where to open the linked document.
    - _self (default): Opens in the same tab/window.
    - _blank: Opens in a new tab/window.
    - _parent: Opens in the parent frame.
    - _top: Opens in the full body of the window.
3. **rel**: Specifies the relationship between the current document and the linked document.
    - Example: rel="nofollow" (used for SEO purposes).
4. **title**: Provides additional information about the link(appears as a tooltip on hover)

-----→ 2M

Creating Links

1. Internal Links

- Internal links navigate within the same website.
- **Example**:

```
<!DOCTYPE html>
<html>
<head>
<title>Internal Links Example</title>
</head>
<body>
<h1>Welcome to My Website</h1>
<p>Go to the Contact Section of the Page</p>
<p>Jump to a section on the same page: <a href="#contact">Contact Section</a>.</p>
<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
<h2 id="contact">Contact Us</h2>
<p>You can reach us at email@example.com.</p>
</body>
</html>
```

Explanation:

- The link Contact Section navigates to a section with the id="contact".

2. External Links

- External links navigate to other websites.
- **Example**:

```
<!DOCTYPE html>
```

```
<html >

<head>

<title>External Links Example</title>

</head>

<body>

<h1>Learn More</h1>

<p>Visit

<a href="https://www.google.com" target="_blank">Google</a> for more information.</p>

</body>

</html>
```

Explanation:

- The link Google navigates to an external website.
- target="_blank" ensures the link opens in a new tab. ----→ 4M

3. b)Create a Student Registration form which contains the following fields: Name, Date of birth, Gender, Languages Known, Phone number, Address, Educational Qualifications, Contact Number and Submit button. [7M]

Ans:

```
<!DOCTYPE html>

<html>

<head>

<title>Student Registration Form</title>

</head>

<body>

<h2>Student Registration Form</h2>

<form>                                                                    ----→ 1M

<!-- Name -->

  Name: <input type="text" name="name" placeholder="Enter your name" required><br><br>

<!-- Date of Birth -->

  Date of Birth: <input type="date" name="dob" required><br><br>

<!-- Gender -->

  Gender:

<input type="radio" name="gender" value="Male" required> Male

<input type="radio" name="gender" value="Female" required> Female

<input type="radio" name="gender" value="Other" required> Other<br><br>
```

```
<!-- Languages Known -->
  Languages Known:
<input type="checkbox" name="languages[]" value="English"> English
<input type="checkbox" name="languages[]" value="Hindi"> Hindi
<input type="checkbox" name="languages[]" value="Other"> Other<br><br>
<!-- Phone Number -->
  Phone Number: <input type="tel" name="phone" placeholder="Enter your phone number" required><br><br>
<!-- Address -->
  Address: <textarea name="address" rows="4" cols="30" placeholder="Enter your address"></textarea><br><br>
<!-- Educational Qualifications -->
  Educational Qualifications:
<select name="education" required>
<option value="">Select your qualification</option>
<option value="High School">High School</option>
<option value="Undergraduate">Undergraduate</option>
<option value="Postgraduate">Postgraduate</option>
<option value="Doctorate">Doctorate</option>
</select><br><br>
<!-- Emergency Contact -->
  Emergency Contact Number: <input type="tel" name="contact" placeholder="Enter emergency contact number" required><br><br>
<!-- Submit Button -->
<input type="submit" value="Submit">
</form>
</body>
</html>
```

----→ 3M

-----→ 3M

4. a) Discuss the three different ways of applying CSS to a webpage (inline, internal, and external). Provide examples of each method and explain the advantages and disadvantages. [7M]

Ans: CSS (Cascading Style Sheets) can be applied to a webpage in three main ways: **inline**, **internal**, and **external**. Each method has its specific use cases, advantages, and disadvantages. Below is a detailed discussion of each:

1. Inline CSS

Inline CSS applies styles directly to an HTML element using the style attribute.

Example:

```
<p style="color: blue; font-size: 18px;">This is a styled paragraph.</p>
```

Advantages:

- **Specificity**: The style is applied directly to the element, making it easy to target specific elements.
- **Quick adjustments**: Useful for testing or overriding styles temporarily.

Disadvantages:

- **Difficult to maintain**: Changes require modifying each element individually.
- **Not reusable**: Styles cannot be reused across multiple elements or pages.
- **Code clutter**: Inline CSS mixes presentation with content, violating the principle of separation of concerns. **--→ 3M**

2. Internal CSS

Internal CSS involves placing styles within a `<style>` tag in the `<head>` section of the HTML document.

Example**:**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Internal CSS Example</title>
<style>
    p {
color: green;
        font-size: 20px;
    }
</style>
</head>
<body>
<p>This paragraph is styled with internal CSS.</p>
</body>
</html>
```

Advantages:

- **Centralized control**: Styles for the entire document are in one place.
- **Specific to the page**: Useful for single-page styling where styles won't be reused.

Disadvantages:

- **Limited reusability**: Styles cannot be shared across multiple pages.
- **Larger file size**: Combining HTML and CSS in the same file can make the file larger and harder to manage. **----→ 2M**

3. External CSS

External CSS involves linking to a separate CSS file using the `<link>` tag in the `<head>` section of the HTML document.

Example: **HTML File**

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>External CSS Example</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<p>This paragraph is styled with external CSS.</p>

</body>

</html>
```

CSS File:

```
p {
color: red;
   font-size: 22px;
}
```

Advantages:

- **Reusability**: The same CSS file can be used for multiple pages, promoting consistency.
- **Separation of concerns**: Keeps HTML and CSS separate, improving code organization and readability.
- **Scalability**: Easier to maintain and update styles for large projects.

Disadvantages:

- **Dependency on external files**: If the CSS file fails to load (e.g., due to a broken link or network issue), the page may lose its styling.
- **Initial load time**: The browser must load the external CSS file, which can slightly delay the initial rendering. **---→ 2M**

Comparison Table**:**

Method	Best for	Advantages	Disadvantages
Inline	Quick, small changes or overrides	Easy to apply and test	Hard to maintain, not reusable, clutters code
Internal	Single-page projects	Centralized, specific to the page	Not reusable across pages, larger file size
External	Multi-page projects, consistent styling	Reusable, scalable, organized	Dependent on external file, slower initial load

4. b) Discuss how to apply backgrounds and borders in CSS. Explain the different properties helps to set a background and border to a web page. [7M]

Ans:**CSS Background Properties**

CSS background property is used to define the background effects on element. Some of the background properties are:

1. background-color
2. background-image
3. background-repeat
4. background-position
5. background-size
6. background-attachment
7. background-clip
8. background-origin
9. background-blend-mode
10. background (short hand notation)

Here is the list of **CSS background properties** and their corresponding values:

1. background-color

- **Values**:
    - Color names (e.g., red, blue, green).
    - Hex codes (e.g., #FF5733).
    - RGB values (e.g., rgb(255, 87, 51)).
    - HSL values (e.g., hsl(0, 100%, 50%)).
    - Transparent (transparent).

2. background-image

- **Values**:
  - url('path-to-image'): Sets an image as the background.
  - none: No background image.

3. background-repeat

- **Values**:
  - repeat: Repeats the background image both horizontally and vertically.
  - repeat-x: Repeats the background image horizontally.
  - repeat-y: Repeats the background image vertically.
  - no-repeat: Prevents the background image from repeating.
  - space: Ensures even spacing between repeated images.
  - round: Scales the background image to ensure it fits perfectly without gaps.

4. background-position

- **Values**:
  - Keywords: left, right, top, bottom, center.
  - Length units: px, %, em, etc. (e.g., 50px, 10%).
  - Combination: Horizontal and vertical values (e.g., center bottom, 10px 20px).

5. background-size

- **Values**:
  - auto: Uses the original size of the background image.
  - cover: Scales the image to cover the element, maintaining aspect ratio.
  - contain: Scales the image to fit inside the element, maintaining aspect ratio.
  - Specific dimensions: width height (e.g., 100px 50px).
  - Keywords: initial, inherit.

6. background-attachment

- **Values**:
  - scroll: Background scrolls with the element (default).
  - fixed: Background stays fixed relative to the viewport.
  - local: Background scrolls within the element’s content.

7. background-clip

- **Values**:
  - border-box: Background extends to the border edge.
  - padding-box: Background extends to the padding edge.
  - content-box: Background extends to the content edge.
  - text: Clips the background to the text's content.

8. background-origin

- **Values**:
  - border-box: Background starts at the border edge.
  - padding-box: Background starts at the padding edge.
  - content-box: Background starts at the content edge.

9. background-blend-mode

- **Values**:
  - normal: Default blending.
  - multiply, screen, overlay, darken, lighten, etc., to blend background layers.

10. background (Shorthand Property)

- **Values** (combined from individual properties):
  - background-color, background-image, background-repeat, background-attachment, background-position, background-size.
  - Example order: color image position/size repeat attachment. **Any three -→3M**

CSS Border Properties:

Here is the list of **CSS border properties** and their corresponding values:

1. border-width

- **Values**:
  - Keywords: thin, medium, thick (default values).
  - Specific lengths: Any unit like px, em, %, rem, etc. (e.g., 1px, 2em).

2. border-style

- **Values**:
  - none: No border (default).
  - solid: Single solid line.
  - dotted: Dotted line.
  - dashed: Dashed line.
  - double: Two solid lines.
  - groove: Carved effect based on the element's color.
  - ridge: Raised effect based on the element's color.
  - inset: Appears embedded into the page.
  - outset: Appears raised from the page.

3. border-color

- **Values**:
  - Color names (e.g., red, blue).
  - Hexadecimal colors (e.g., #FF5733).
  - RGB values (e.g., rgb(255, 87, 51)).
  - HSL values (e.g., hsl(0, 100%, 50%)).

- Transparent (transparent).

4. border-radius

- **Values**:
  - Length units: px, em, %, etc. (e.g., 5px, 50% for circular shapes).
  - Combination of values for individual corners:
    - border-top-left-radius, border-top-right-radius, border-bottom-left-radius, border-bottom-right-radius.

5. border (Shorthand Property)

- **Values** (combination of properties):
  - border-width, border-style, border-color.

6. border-top, border-right, border-bottom, border-left

- **Values**:
  - Same as border but applied to specific sides.

7. border-collapse (Used with tables)

- **Values**:
  - collapse: Merges adjacent borders.
  - separate: Keeps borders separate (default).

8. border-spacing (Used with tables)

- **Values**:
  - Length: Any unit like px, em, etc. (e.g., 5px**).** **-→Any Three properties-- 4M**

(OR)

5. a) Discuss in detail about various Dialog boxes used in Java Script wit an example. [7M]

Ans: In JavaScript, dialog boxes are used to interact with users by displaying messages or receiving inputs. These are built-in methods of the window object, which is implicitly available in browser environments. The three main types of dialog boxes in JavaScript are:

1. alert() dialog box
2. confirm() dialog box
3. prompt() dialog box **----→ 1M**

1. **alert()** Dialog Box

The `alert()` method is used to display a simple message to the user. This dialog box only contains an **OK** button and is typically used to show information or warnings.

Syntax: alert(message);

Example: alert("This is an alert dialog box."); E**xplanation with any relevant example→ 2M**

2. confirm() Dialog Box

The confirm() method is used to display a dialog box with a message and OK and Cancel buttons. It allows the user to confirm or cancel an action.

Syntax: var result = confirm(message);

Example:

```
var userChoice = confirm("Do you want to proceed?");
if (userChoice) {
console.log("User clicked OK.");} else {
console.log("User clicked Cancel."); }
```

Explanation with any relevant example--→**2M**

3. prompt() Dialog Box

The prompt() method is used to display a dialog box that asks the user for input. It contains a text field for input, along with **OK** and **Cancel** buttons.

Syntax: var userInput = prompt(message, defaultValue);

Parameters:

- message: The text to display to the user.
- defaultValue (optional): The default text displayed in the input field.

Example:

```
var name = prompt("What is your name?", "John Doe");
    if (name) {
    console.log("Hello, " + name + "!");
    } else {
    console.log("User canceled the prompt.");
    }
```

--→Explanation with any relevant example **2M**

5. b) Write a Java Script program to demonstrate simple Animation. [7M]

Ans:

```
<!DOCTYPE html>
<html>
<head>
<title>Car Animation</title>
<style>
    #road {
      position: relative;
      width: 100%;
      height: 200px;
      bottom: 0;
      background: gray;
      margin-top: 300px;
    }
    #car {
      position: absolute;
      width: 150px;
      height: auto;
```

```
        left: 0;
        top: 50px;
    }
</style>
</head>
<body>
<div id="road">
<img id="car" src="car.jpg" alt="Car">
</div>
<script>
const car = document.getElementById("car");
    let position = 0; // Starting position of the car
constscreenWidth = window.innerWidth;
    // Function to move the car
    function moveCar() {
        if (position >screenWidth) {
            position = -150; // Reset the car position when it goes off-screen
        }
        position += 5; // Increment the position
car.style.left = position + "px"; // Update the car's position
    }
    // Start the animation
setInterval(moveCar, 20); // Move the car every 20 milliseconds
</script>
</body>
    </html>
```

-----→ 2M

--→ 5M

Unit-III

6.Explain in detail about the following Java Script Objects with suitable examples.

i) Array Object ii) String Object iii) Date Object [14M]

Ans:

i) <u>Array object:</u>

- An **Array** is used to store a number of values (called as elements) in order with a single variable.
- An array object can created in two ways

Using array constructor:

- Creates an empty array: var ar = new Array()
- Creates an array with given size: var ar = new Array(size)
- Creates an array based on number of elements:

  var ar = new Array(element0, element1, ..., elementN)

Using array literal notation:

- Array literal notation are coma separated list of items enclosed by square brackets.
- creates an empty array: var ar = [ ]
- creates an empty array with elements: var ar = [ele1,ele2,...,elen]
- example: var ar = [5,"hello", true]

JavaScript Array Objects Property

Name	Description
Length	Returns the number of elements in an array
prototype	Use to add new properties to all objects.
constructor	Specifies the function which creates an object's prototype.

JavaScript object methods:

Name	Description
concat()	Use to join two or more arrays and returns a new array.
join()	Use to join all elements of an array into a string.
pop()	Use to remove the last element from an array.
push()	Use to add one or more elements at the end of an array.
reverse()	Use to reverse the order of the elements in an array.
shift()	Use to remove first element from an array.
slice()	Use to extract a section of an array.
sort()	Use to sort the elements of an array.
toString()	Returns a string represent the array and its elements.
unshift()	Use to add one or more elements to the beginning of an array and returns the new length of the array.
valueOf()	Returns the primitive value of an array.

Explanation with any example -→ 5M

ii) String Objects:

- String is a series of characters.
- String objects are used to perform operations on text.

  Syntax:

  var variable_name = new String(string);

  Or

  var variable_name="string";

- **Example:**

  var s = new String("Hello");

  Or

var s=”Hello”;

String object properties

Properties	Description
Length	It returns the length of the string.
Prototype	It allows you to add properties and methods to an object.
Constructor	It returns the reference to the String function that created the object.

String object methods

Methods	Description
charAt ()	It returns the character at the specified index.
charCodeAt()	It returns the ASCII code of the character at the specified position.
concat ()	It combines the text of two strings and returns a new string.
indexOf()	It returns the index within the calling String object.
lastIndexOf()	It returns the position of last occurrence of specified value in string.
match()	It is used to match a regular expression against a string.
quote()	It writes the quotes in JavaScript
replace()	It is used to replace the matched substring with a new substring.
search()	It executes the search for a match between regular expressions.
slice()	It extracts a session of a string and returns a new string.
split()	It splits a string into substrings.
toLowerCase()	It returns the calling string value converted lower case.
toUpperCase()	Returns the calling string value converted to uppercase.

Explanation with any example -→ 5M

iii) Date Object:

- The Date object used to display the date on web page or time stamp in numerical or mathematical calculations.
- Date objects are created with **new Date ().**
- There are five ways of instantiating (creating) a date:

var date1 = new Date();

or

var date1 = new Date(*milliseconds*);

or

var date1 = new Date(*“mm dd, yyyy”*);

or

var date1 = new Date(*“mm dd, yyyyhr:min:sec”*);

or

var date1 = new Date(*yyyy*, *mm*, *dd*, [, *hr*, *min*, *sec*, *millisec]*);

Properties:

Name	Description
Constructor	Returns the function that created the Date object's prototype
Prototype	Allows you to add properties and methods to an object

Methods

Name	Description
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year
getHours()	Returns the hour (from 0-23)
getMilliseconds()	Returns the milliseconds (from 0-999)
getMinutes()	Returns the minutes (from 0-59)
getMonth()	Returns the month (from 0-11)
getSeconds()	Returns the seconds (from 0-59)
getTime()	Returns the number of milliseconds since midnight Jan 1 1970, and a specified date

Explanation with any example -→ 4M

(OR)

7. a) How does JavaScript use the document object to access, modify, and delete HTML elements on a webpage? Illustrate with examples of creating and removing elements dynamically. [7M]

Ans: JavaScript uses the document object, which represents the entire HTML document, to access, modify, and delete elements on a webpage. The document object provides methods and properties that allow interaction with the DOM (Document Object Model).

Accessing HTML Elements

You can access HTML elements using various methods:

1. getElementById()
2. getElementsByClassName()
3. getElementsByTagName()
4. querySelector() and querySelectorAll()

Modifying HTML Elements

Once an element is accessed, you can modify its content, attributes, and styles using properties such as innerHTML, style, or methods like setAttribute().

Creating and Removing Elements Dynamically

The document object also allows creating new elements and removing existing ones.

Key Methods Used

1. **Creating Elements**:
   - document.createElement('tagName') creates a new element.
   - element.appendChild(newElement) appends the created element to a parent.
2. **Removing Elements**:
   - parent.removeChild(child) removes a child element from its parent.
3. **Querying Elements**:
   - document.querySelector() selects the first element matching a CSS selector.
   - document.querySelectorAll() selects all elements matching a CSS selector.

Explanation+ Program = 4+3=7M

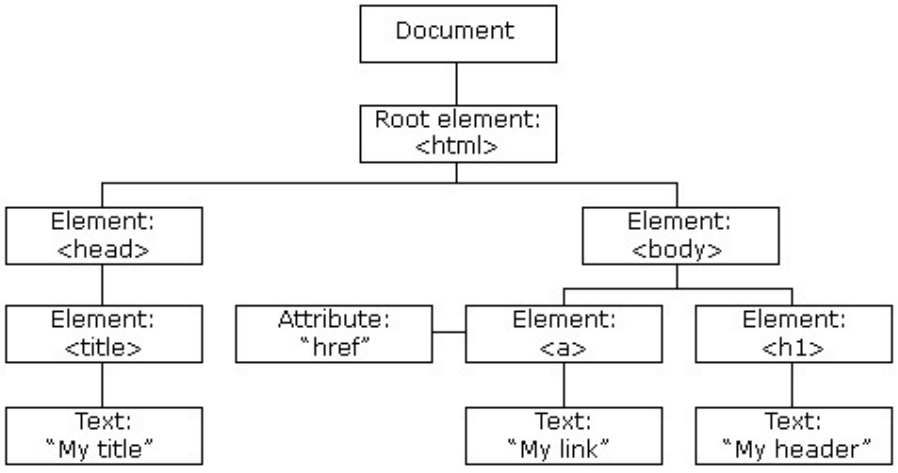
7. b) Discuss the hierarchical structure of the DOM, which represents the document as a tree of nodesand alsoexplain the types of nodes (Element, Text, Attribute, etc.). [7M]

Ans: **Hierarchical Structure of the DOM**

The DOM organizes an HTML document as a tree-like structure:

- The root of the tree is the **document** object.
- Each HTML element is represented as a **node**.
- Each node can have child nodes (e.g., text, attributes, or other elements).

HTML DOM Tree of Objects



---→ 3M

Types of Nodes in the DOM

The DOM defines 12 node types, but the most commonly used types are:

1. Document Node (nodeType = 9):
    - Represents the root of the document tree (the document object).
    - Provides entry points to the DOM, such as document.body or document.getElementById().
2. Element Nodes (nodeType = 1):
    - Represents HTML or XML elements (e.g., <html>, <body>,).
    - Can have attributes and child nodes.
    - Example: This is a paragraph. represents an element node.
3. Text Nodes (nodeType = 3):
    - Represents the actual text content within an element.
    - Text nodes are always children of element nodes.
    - Example: "This is a paragraph." in This is a paragraph. .
4. Attribute Nodes (nodeType = 2):
    - Represent attributes of an element (e.g., id, class, href).
    - These are not direct children of element nodes but can be accessed via methods like getAttribute().
5. Comment Nodes (nodeType = 8):
    - Represent comments in the document**.**
    - **Example: <!-- This is a comment -->.**

Key Node Properties

Each node has properties that reflect its place in the hierarchy:

1. Parent Node: Accessed using node.parentNode.
2. Child Nodes: Accessed using node.childNodes or node.children (excludes text and comment nodes).
3. Sibling Nodes:
    - node.previousSibling (for the previous node).
    - node.nextSibling (for the next node). Program+Theory ---→4M

8. a) Explain in detail about Internal DTD and External DTD with suitable examples. [7M]

Ans:DocumentType Definition (DTD) defines the structure and rules of an XML document, ensuring that the document follows a specific format. It specifies the elements, attributes, and their relationships. There are two types of DTDs based on how they are defined: **Internal DTD** and **External DTD**.

1. Internal DTD

An **Internal DTD** is defined within the XML document itself using a <!DOCTYPE> declaration. It is embedded inside the XML file and is typically used for small or simple XML documents where portability is important.

Syntax

The <!DOCTYPE> declaration is used at the top of the XML document, followed by the structure of the DTD enclosed in square brackets [ ].

Example: Internal DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE employees [
<!ELEMENT employees (employee+)>
<!ELEMENT employee (name, designation, salary)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT designation (#PCDATA)>
<!ELEMENT salary (#PCDATA)>
<!ATTLIST employee id ID #REQUIRED>
]>
<employees>
<employee id="101">
<name>John Doe</name>
<designation>Software Engineer</designation>
<salary>75000</salary>
</employee>
<employee id="102">
<name>Jane Smith</name>
<designation>Project Manager</designation>
<salary>95000</salary>
</employee>
</employees>
```

2. External DTD

An **External DTD** is defined in a separate file, making it reusable across multiple XML documents. It is linked to the XML file using the <!DOCTYPE> declaration with a SYSTEM or PUBLIC identifier.

Syntax: <!DOCTYPE root-element SYSTEM "file.dtd">

SYSTEM: Specifies the path to the external DTD file (either local or a URL).

PUBLIC: Specifies a public identifier along with the DTD file location.

Example: External DTD

Employees.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE employees SYSTEM "employees.dtd">
<employees>
<employee id="101">
<name>John Doe</name>
<designation>Software Engineer</designation>
<salary>75000</salary>
</employee>
<employee id="102">
<name>Jane Smith</name>
<designation>Project Manager</designation>
<salary>95000</salary>
</employee>
</employees>
```

employees.dtd (External DTD File):

```
<!ELEMENT employees (employee+)>
<!ELEMENT employee (name, designation, salary)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT designation (#PCDATA)>
<!ELEMENT salary (#PCDATA)>
<!ATTLIST employee id ID #REQUIRED>
```

Comparison of Internal and External DTD

Aspect	**Internal DTD**	**External DTD**
Definition Location	Inside the XML document itself.	In a separate external file.
Reusability	Not reusable for other XML documents.	Can be reused across multiple XML documents.
Portability	Self-contained; portable as a single file.	Requires an additional DTD file.
Complexity Handling	Suitable for simple or small XML documents.	Suitable for complex or large XML documents.

Differences + Example Program = 4+3 = 7M

8. b) Write an XML document to represent a book catalog containing at least five books with attributes like title, author, publisher and price. [7M]

Ans:

```
<?xml version="1.0" encoding="UTF-8"?>
<bookCatalog>
<book title="To Kill a Mockingbird" author="Harper Lee" publisher="J.B. Lippincott & Co." price="18.99" />
<book title="1984" author="George Orwell" publisher="Secker & Warburg" price="14.99" />
<book title="The Great Gatsby" author="F. Scott Fitzgerald" publisher="Charles Scribner's Sons" price="10.99" />
<book title="Pride and Prejudice" author="Jane Austen" publisher="T. Egerton" price="12.99" />
<book title="Moby Dick" author="Herman Melville" publisher="Harper & Brothers" price="16.50" />
</bookCatalog>
```

Any other valid solution = 7M

(OR)

9. a) Differentiate between DTD and XSD with an example. [7M]

Ans: The important differences between DTD and XSD are given below:

S.No	DTD	XSD
1)	DTD stands for **Document Type Definition**.	XSD stands for XML Schema Definition.
2)	DTDs are derived from **SGML** syntax.	XSDs are written in XML.
3)	DTD **doesn't support datatypes**.	XSD **supports datatypes** for elements and attributes.
4)	DTD **doesn't support namespace**.	XSD **supports namespace**.
5)	DTD **doesn't define order** for child elements.	XSD **defines order** for child elements.
6)	DTD is **not extensible**.	XSD is **extensible**.
7)	DTD is **not simple to learn**.	XSD is **simple to learn** because you don't need to learn new language.
8)	DTD provides **less control** on XML structure.	XSD provides **more control** on XML structure.

Differences + Example Program = 4+3 = 7M

9. b) Create an XML schema (XSD) to validate an XML file containing employee details. [7M]

Ans:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schemaxmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<!-- Root element -->
<xs:element name="employees">
<xs:complexType>
<xs:sequence>                                                    ------→ 2M
<!-- Employee element -->
<xs:element name="employee" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="name" type="xs:string" />
<xs:element name="designation" type="xs:string" />
<xs:element name="department" type="xs:string" />
<xs:element name="email" type="xs:string" />
<xs:element name="salary" type="xs:decimal" />
</xs:sequence>
<!-- Employee ID attribute -->
<xs:attribute name="id" type="xs:int" use="required" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>                                                      ------→ 5M
```

Scheme Prepared By: **Signature of the HOD**

K. Suresh Kumar,

Assistant Professor,

Department of IT