

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Internal Assessment Test 1 – June. 2021

Sub:	Web Technologies							Sub Code:	20MCA23
Date:	23/6/2021	Duration:	90 min's	Max Marks:	50	Sem & Sec:	II A & B	Branch:	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

	MAR KS	OBE	
		CO	RB
PART I			
1. Discuss the purpose of the following javascript functions with examples: a)Split() b)Match() c)Replace() d)Search()	[10]	CO2	L2
OR			
2. Write the java script code to find median of a set of numbers stored in an array.	[10]	CO4	L4
PART II			
3. Create a student registration form to accept name, gender, date of birth qualification, address and pincode. Provide reset and submit buttons.	[10]	CO2	L4
OR			
4. What is an event and event handler? Explain any two event handler with examples.	[10]	CO2	L2
PART III			
5. Write an XHTML (with CSS) program to display the following output: AIRCRAFT TYPES I) General aviation A. Single – Engine i) Tail wheel ii) Tricycle. B. Dual – Engine i) Wing mounted engines ii) Push-pull mounted engines. II) Commercial aviation A. Dual – Engine i) Wing mounted engines ii) Fuselage – Mounted engines B. Third engine i) Vertical stabilizer ii) Fuselage.	[10]	CO4	L4
OR			
6. Explain basic text formatting in XHTML with an example.	[10]	CO2	L2
PART IV			
7. Explain pattern matching using regular expressions in java script with example.	[10]	CO2	L2
OR			
8. Write XHTML and javascript code to compute and print binomial coefficient of given n and r values.	[10]	CO4	L4

PART V

9. Briefly explain the following
i) Web browsers ii) web servers iii) URL iv) MIME
- OR**
10. Write a program in java script to accept three numbers using prompt and find the largest of three numbers.

[10]	CO2	L2
[10]	CO4	L4

1. **Discuss the purpose of the following javascript functions with examples:**
a) Split() b) Match() c) Replace() d) Search()

- The split method of String splits its object string into substrings on the basis of a given string or pattern. The substrings are returned in an array. For example, consider the following code:

```
var str = "grapes:apples:oranges";var fruit =  
str.split(":");
```

In this example, fruit is set to [grapes, apples, oranges].

- The match method is the most general of the String pattern-matching methods.
- The match method takes a single parameter: a pattern. It returns an array of the results of the pattern-matching operation.
- If the pattern has the g modifier, the returned array has all of the substrings of the string that matched.

- The replace method is used to replace substrings of the String object that match the given pattern.
- The replace method takes two parameters: the pattern and the replacement string.
- The g modifier can be attached to the pattern if the replacement is to be global in the string, in which case the replacement is done for every match in the string.
- The matched substrings of the string are made available through the predefined variables \$1, \$2, and so on. For example, consider the following statements:

```
var str = "Fred, Freddie, and Frederica were siblings";
str.replace(/Fre/g, "Boy");
```

- The simplest pattern-matching method is search, which takes a pattern as a parameter.
- The search method returns the position in the String object (through which it is called) at which the pattern matched.
- If there is no match, search returns -1.
- Most characters are normal, which means that, in a pattern, they match themselves.
- The position of the first character in the string is 0.
- As an example, the following statements

```
var str = "Rabbits are furry";
var position = str.search(/bits/);
if (position >= 0)
    document.write("'bits' appears in position", position,
        "<br />");
else
    document.write("'bits' does not appear in str <br />");
```

produce the following output:

```
'bits' appears in position 3
```

- If the pattern does not include the g modifier, the returned array has the match as its first element, and the remainder of the array has the matches of parenthesized parts of the pattern if there are any:

```
var str =
    "Having 4 apples is better than having 3 oranges";
var matches = str.match(/\d/g);
```

In this example, matches is set to [4, 3].

2. Write the java script code to find median of a set of numbers stored in an array.

```
<html>
```

```
<body>
```

```
<script>
```

```
// Function for
```

```
// calculating median
```

```
function findMedian(a,n)
```

```
{
```

```

// First we sort
// the array
a.sort();
// check for
// even case
if (n % 2 != 0)
    return a[n / 2];
return (a[Math.floor((n-1)/2)] +
        a[n / 2]) / 2;
}

// Driver Code
let a = [1, 3, 4, 2, 7, 5, 8, 6]
let n = a.length;
// Function call
document.write("Median = " + findMedian(a, n));
</script>
</body>
</html>

```

3. Create a student registration form to accept name, gender, date of birth, qualification, address and pincode. Provide reset and submit buttons.

```

<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>My Details</title>
</head>
<body>
<form action=""><label>Name:</label>
<input type="text" name="text1" size="20" maxlength="15"/><br/>

```

```
<p>Enter your gender</p>
<label><input type="radio" name="gender" value="male"/>Male</label>
<label><input type="radio" name="gender" value="female"/>Female</label><br/>
<label>DOB:<input type="date" name="dob"/></label><br/>
```

```
<label>Qualification</label>
<select name="Qualify" size="2">
<option>Phd</option>
<option>PG</option>
<option>UG</option>
<option>XII</option>
<option>X</option>
</select><br/>
```

```
<label>Address</label>
<textarea rows = "5" cols = "50" name = "description">
</textarea><br/>
<label>Pincode:<input type="number" name="pin1"/></label><br/>
```

```
<input type="submit" value="go"/>
<input type="reset" value="clear"/>
</form>
</body>
</html>
```

4. What is an event and event handler? Explain any two event handler with examples.

- One important use of JavaScript for Web programming is to detect certain activities of the browser and the browser user and provide computation when those activities occur. These computations are specified with a special form of programming called event- driven programming.
- In conventional (non-event-driven) programming, the code itself specifies the order in which it is executed, although the order is usually affected by the program's input

data.

- In event-driven programming, parts of the program are executed at completely unpredictable times, often triggered by user interactions with the program that is executing.
- An event is a notification that something specific has occurred, either with the browser, such as the completion of the loading of a document, or because of a browser user action, such as a mouse click on a form button.
- An event handler is a script that is implicitly executed in response to the appearance of an event. Event handlers enable a Web document to be responsive to browser and user activities.
- One of the most common uses of event handlers is to check for simple errors and omissions in user input to the elements of a form, either when they are changed or when the form is submitted.
- This kind of checking saves the time of sending incorrect form data to the server.
- Because events are JavaScript objects, their names are case sensitive. The names of all event objects have only lowercase letters.
- Events are created by activities associated with specific XHTML elements.
- The process of connecting an event handler to an event is called registration.
- There are two distinct approaches to event handler registration, one that assigns tag attributes and one that assigns handler addresses to object properties.

EVENTS, ATTRIBUTES, AND TAGS

Event	Tag Attribute
blur	onblur
change	onchange
click	onclick
dblclick	ondblclick
focus	onfocus
keydown	onkeydown
keypress	onkeypress
keyup	onkeyup
load	onload
mousedown	onmousedown
mousemove	onmousemove
mouseout	onmouseout
mouseover	onmouseover
mouseup	onmouseup
reset	onreset
select	onselect
submit	onsubmit
unload	onunload

In many cases, the same attribute can appear in several different tags. The circumstances under which an event is created are related to a tag and an attribute, and they can be different for the same attribute when it appears in different tags.

Attribute	Tag	Description
onblur	<a>	The link loses the input focus
	<button>	The button loses the input focus
	<input>	The input element loses the input focus
	<textarea>	The text area loses the input focus
	<select>	The selection element loses the input focus
onchange	<input>	The input element is changed and loses the input focus
	<textarea>	The text area is changed and loses the input focus
	<select>	The selection element is changed and loses the input focus
onclick	<a>	The user clicks on the link
	<input>	The input element is clicked
ondblclick	Most elements	The user double-clicks the left mouse button
onfocus	<a>	The link acquires the input focus
	<input>	The input element receives the input focus
	<textarea>	A text area receives the input focus
	<select>	A selection element receives the input focus
onkeydown	<body>, form elements	A key is pressed down
onkeypress	<body>, form elements	A key is pressed down and released
onkeyup	<body>, form elements	A key is released
onload	<body>	The document is finished loading
onmousedown	Most elements	The user clicks the left mouse button
onmousemove	Most elements	The user moves the mouse cursor within the element
onmouseout	Most elements	The mouse cursor is moved away from being over the element
onmouseover	Most elements	The mouse cursor is moved over the element
onmouseup	Most elements	The left mouse button is unclicked
onreset	<form>	The reset button is clicked
onselect	<input>	Any text in the content of the element is selected
	<textarea>	Any text in the content of the element is selected
onsubmit	<form>	The Submit button is pressed
onunload	<body>	The user exits the document

As mentioned previously, there are two ways to register an event handler in the DOM 0 event model. One of these is by assigning the event handler script to an event tag attribute, as in the following example:

```
<input type = "button" id = "myButton"
      onclick = "alert('You clicked my button!');" />
```

In many cases, the handler consists of more than a single statement. In these cases, often a function is used and the literal string value of the attribute is the call to the function. Consider

the example of a button element:

```
<input type = "button" id = "myButton"
        onclick = "myButtonHandler();" />
```

An event handler function could also be registered by assigning its name to the associated event property on the button object, as in the following example:

```
document.getElementById("myButton").onclick =
        myButtonHandler;
```

HANDLING EVENTS FROM BODY ELEMENTS

The events most often created by body elements are load and unload. As our first example of event handling, we consider the simple case of producing an alert message when the body of the document has been loaded. In this case, we use the onload attribute of <body> to specify the event handler:

```
<?xml version = "1.0" encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- load.html
    A document for load.js
-->
<html xmlns = "http://www.w3.org/1999/xhtml">
    <head>
        <title> load.html </title>
        <script type = "text/javascript" src = "load.js" >
        </script>
    </head>
    <body onload="load_greeting();">
        <p />
    </body>
</html>

// load.js
// An example to illustrate the load event

// The onload event handler
function load_greeting () {
    alert("You are visiting the home page of \n" +
        "Pete's Pickled Peppers \n" + "WELCOME!!!");
}
```

OUTPUT



The unload event is probably more useful than the load event. It is used to do some cleanup before a document is unloaded, as when the browser user goes on to some new document. For example, if the document opened a second browser window, that window could be closed by an unload event handler.

HANDLING EVENTS FROM BUTTON ELEMENTS

Buttons in a Web document provide an effective way to collect simple input from the browser user. Example:

```
<?xml version = "1.0" encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- radio_click.html
  A document for radio_click.js
  Creates four radio buttons that call the planeChoice
  event handler to display descriptions
-->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> radio_click.html </title>
    <script type = "text/javascript" src = "radio_click.js" >
    </script>
  </head>
  <body>
    <h4> Cessna single-engine airplane descriptions </h4>
    <form id = "myForm" action = "">
      <p>
        <label> <input type = "radio" name = "planeButton"
          value = "152"
          onclick = "planeChoice(152)" />
        Model 152 </label>
        <br />
        <label> <input type = "radio" name = "planeButton"
          value = "172"
          onclick = "planeChoice(172)" />
```

```

    Model 172 (Skyhawk) </label>
    <br />
    <label> <input type = "radio" name = "planeButton"
              value = "182"
              onclick = "planeChoice(182)" />
    Model 182 (Skylane) </label>
    <br />
    <label> <input type = "radio" name = "planeButton"
              value = "210"
              onclick = "planeChoice(210)" />
    Model 210 (Centurian) </label>
  </p>
</form>
</body>
</html>

// radio_click.js
// An example of the use of the click event with radio buttons,
// registering the event handler by assignment to the button
// attributes

// The event handler for a radio button collection
function planeChoice (plane) {

// Produce an alert message about the chosen airplane
switch (plane) {
  case 152:
    alert("A small two-place airplane for flight training");
    break;
  case 172:
    alert("The smaller of two four-place airplanes");
    break;
  case 182:
    alert("The larger of two four-place airplanes");
    break;
  case 210:
    alert("A six-place high-performance airplane");
    break;
  default:
    alert("Error in JavaScript function planeChoice");
    break;
}
}

```

OUTPUT

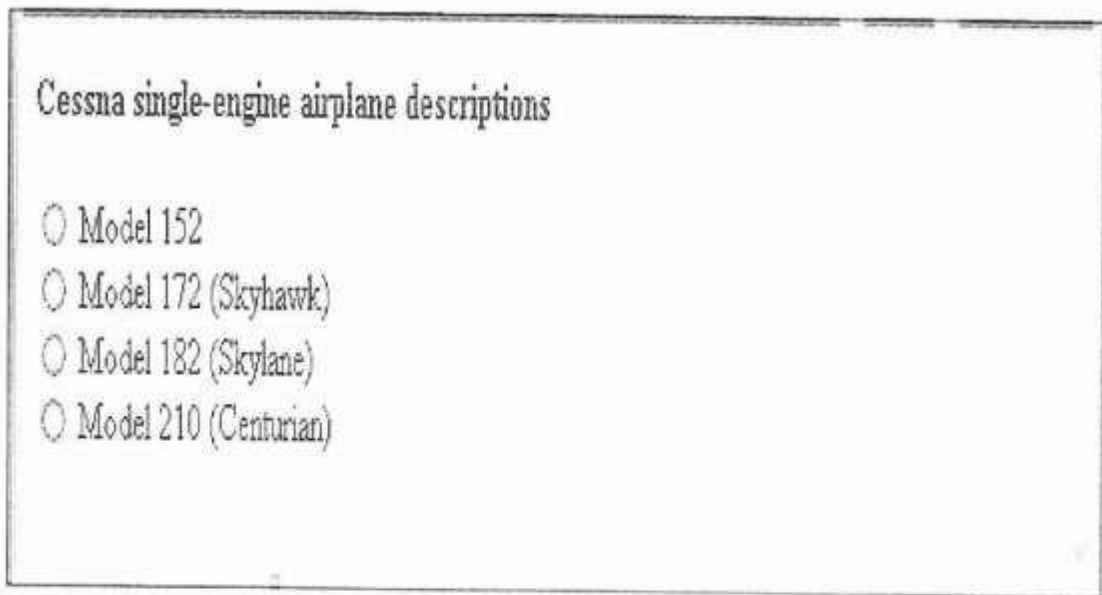


Figure 5.3 Display of `radio_click.html`

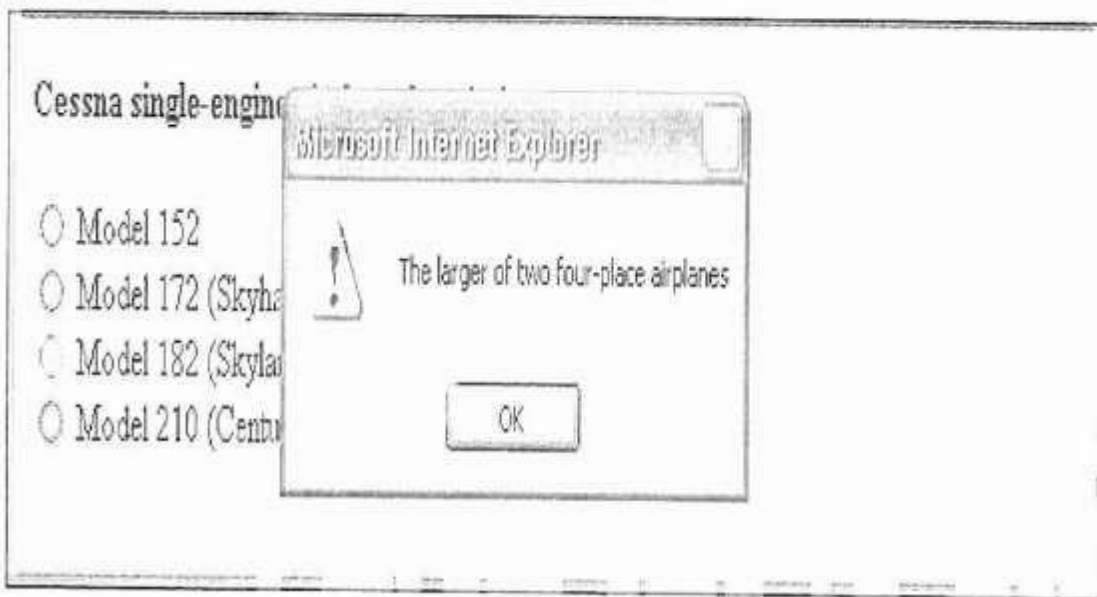


Figure 5.4 The result of pressing the Model 182 button in `radio_click`

The next example, `radio_click2.html`, whose purpose is the same as that of `radio_click.html`, registers the event handler by assigning the name of the handler to the event properties of the radio button objects.

```

<!-- radio_click2.html
  A document for radio_click2.js
-->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> radio_click2.html </title>

<!-- Script for the event handler -->
  <script type = "text/javascript" src = "radio_click2.js" >
  </script>

</head>
<body>
  <h4> Cessna single-engine airplane descriptions </h4>
  <form id = "myForm" action = "">
    <p>
      <label> <input type = "radio" name = "planeButton"
        value = "152" />
      Model 152 </label>
      <br />
      <label> <input type = "radio" name = "planeButton"
        value = "172" />
      Model 172 (Skyhawk) </label>
      <br />
      <label> <input type = "radio" name = "planeButton"
        value = "182" />
      Model 182 (Skylane) </label>
      <br />
      <label> <input type = "radio" name = "planeButton"
        value = "210" />
      Model 210 (Centurian) </label>
    </p>
  </form>

<!-- Script for registering the event handlers -->
  <script type = "text/javascript" src = "radio_click2r.js" >
  </script>

  </body>
</html>

```

```

// radio_click2.js
// An example of the use of the click event with radio buttons,
// registering the event handler by assigning an event property

// The event handler for a radio button collection
function planeChoice (plane) {

// Put the DOM address of the elements array in a local variable
var dom = document.getElementById("myForm");

// Determine which button was pressed
for (var index = 0; index < dom.planeButton.length;
index++) {
if (dom.planeButton[index].checked) {
plane = dom.planeButton[index].value;
break;
}
}

// Produce an alert message about the chosen airplane
switch (plane) {
case "152":
alert("A small two-place airplane for flight training");
break;
case "172":
alert("The smaller of two four-place airplanes");
break;
case "182":
alert("The larger of two four-place airplanes");
break;
case "210":
alert("A six-place high-performance airplane");
break;
default:
alert("Error in JavaScript function planeChoice");
break;
}
}
}

```

```

// radio_click2r.js
// The event registering code for radio_click2
var dom = document.getElementById("myForm");
dom.elements[0].onclick = planeChoice;
dom.elements[1].onclick = planeChoice;
dom.elements[2].onclick = planeChoice;
dom.elements[3].onclick = planeChoice;

```


There are two advantages to registering handlers as properties over registering them in XHTML attributes.

- First, it is good to keep XHTML and Java-Script separated in the document. This allows a kind of modularization of XHTML documents, resulting in a cleaner design that will be easier to maintain.
- Second, having the handler function registered as the value of a property allows for the possibility of changing the function during use.

Write an XHTML (with CSS) program to display the following output:

AIRCRAFT TYPES

I) General aviation

A. Single – Engine

i) Tail wheel

ii) Tricycle.

B. Dual – Engine

i) Wing mounted engines

ii) Push-pull mounted engines.

II) Commercial aviation

A. Dual – Engine

i) Wing mounted engines

ii) Fuselage – Mounted engines

B. Third engine

i) Vertical stabilizer

5. ii) Fuselage.

```
<html><head>
```

```
<style type = "text/css">
```

```
ol {list-style-type: upper-roman;}
```

```
ol ol {list-style-type: upper-alpha;}
```

```
ol ol ol {list-style-type: lower-roman;}
```

```
</style>
```

```
</head>
```

```
<body>
```

<h3> Aircraft Types </h3>

 General Aviation (piston-driven engines)

 Single-Engine Aircraft

 Tail wheel

 Tricycle

 Dual-Engine Aircraft

 Wing-mounted engines

 Push-pull fuselage-mounted engines

 Commercial Aviation (piston-driven engines)


```
<li> Dual-Engine Aircraft</li>
```

```
<ol>
```

```
<li> Wing Mounted </li>
```

```
<li> Fusilage </li>
```

```
</ol>
```

```
<li> Third Engine</li>
```

```
<ol>
```

```
<li> Vertical </li>
```

```
<li> Fusilage </li>
```

```
</ol>
```

```
</ol>
```

```
</ol>
```

```
</body>
```

```
</html>
```

6. Explain basic text formatting in XHTML with an example.

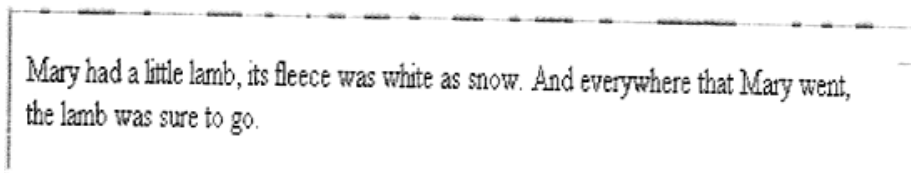
- 1) Paragraphs**
- 2) Line Breaks**
- 3) Preserving whitespace**
- 4) Headings**

- 5) **Block Quotations**
- 6) **Font styles and Sizes**
- 7) **Character Entities**
- 8) **Horizontal rules**
- 9) **The meta Element**

1) **Paragraphs**

- XHTML does not allow text to be directly placed into the document. Text is normally organised into paragraphs.
- It begins with `<p>` and ends with `</p>`. Multiple paragraphs may appear in a single document

```
<p>
  Mary had
a
  little lamb, its fleece was white as snow. And
  everywhere that
  Mary went, the lamb
  was sure to go.
</p>
```

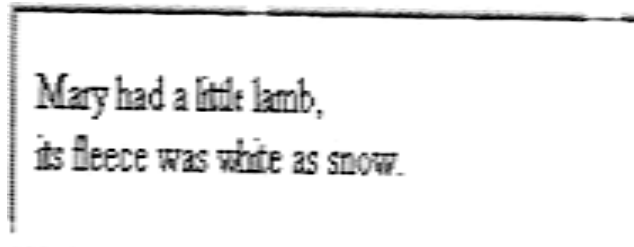


Mary had a little lamb, its fleece was white as snow. And everywhere that Mary went, the lamb was sure to go.

2) **Line Breaks**

- Sometime text requires a line break without the preceding blank line. This exactly what the break tag does.
- Break tag does not have any content therefore it is self-closing tag
- The break tag is specified as `
`. The slash indicates that the tag is both an opening and closing tag.

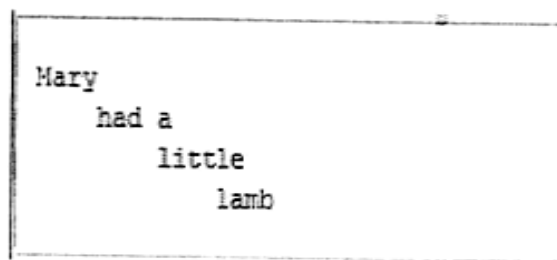
```
<p>
Mary had a little lamb, <br />
  its fleece was white as snow.
</p>
```



3) Preserving Whitespace

Sometimes it is desirable to preserve the white space in text—that is, to prevent the browser from eliminating multiple spaces and ignoring embedded line breaks. This can be specified with the `<pre>` tag.

```
<p><pre>
Mary
  had a
    little
      lamb
</pre>
```



4) Headings

In XHTML, there are six levels of headings, specified by the tags `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, and `<h6>`, where `<h1>` specifies the highest-level heading. Headings are usually displayed in a boldface font whose default size depends on the number in the heading tag. On most browsers, `<h1>`, `<h2>`, and `<h3>` use font sizes that are larger than that of the default size of text, `<h4>` uses the default size, and `<h5>` and `<h6>` use smaller sizes. The heading tags always break the current line, so their content always appears on a new line. Browsers usually insert some vertical space before and after all headings

`<html>`

```
<head><title> Headings </title></head>
```

```
<body>
```

```
<h1> Heading 1 </h1>
```

```
<h2> Heading 2 </h2>
```

```
<h3> Heading 3 </h3>
```

```
<h4> Heading 4 </h4>
```

```
<h5> Heading 5 </h5>
```

```
<h6> Heading 6 </h6>
```

```
</body>
```

```
</html>
```



Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

5) Block Quotations

The `<blockquote>` tag is used to make the contents look different from the surrounding text.

```
<p>Here is a quote from WWF's website:</p>
```

```
<blockquote cite="http://www.worldwildlife.org/who/index.html">
```

For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by 1.2 million members in the United States and close to 5 million globally.

```
</blockquote>
```

About WWF

Here is a quote from WWF's website:

For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by 1.2 million members in the United States and close to 5 million globally.

6) Font Styles and sizes

- ``, `<i>` and `<u>` specifies bold, italics and underline respectively.
- The emphasis tag, ``, specifies that its textual content is special and should be displayed in some way that indicates this distinctiveness. Most browsers use italics for such content.
- The strong tag, `` is like the emphasis tag, but more so. Browsers often set the content of strong elements in bold.
- The code tag, `<code>`, is used to specify a monospace font, usually for program code.
- Subscript and Superscript characters can be specified by the `<sub>` and `<sup>` tags, respectively. These are not content-based tags. Eg

`x₂³ + y₁²`

Illustration of Font Sizes (subscripts and superscripts)

$$x_2^3 + y_1^2$$

- XHTML tags can be categorised as block or inline.
- The content of inline tag appears on the current line.
- A block tag breaks the current line so that its content appears on a new line
- ```` are inline tags whereas heading and `<blockquote>` are block tags

7) Character Entities

XHTML provides a collection of special characters that are sometimes needed in a document but cannot be typed as themselves. In some cases, these characters are used in XHTML in some special way—for example, `>`, `<`, and `&`. In other cases, the characters do not appear on keyboards, such as the small raised circle that represents “degrees” in a reference to temperature. These special characters are defined as entities, which are codes for the characters. An entity in a document is replaced by its associated character by the browser.

Table 2.1 Some commonly used entities

Character	Entity	Meaning
&	<code>&amp;</code>	Ampersand
<	<code>&lt;</code>	Less than
>	<code>&gt;</code>	Greater than
"	<code>&quot;</code>	Double quote
'	<code>&apos;</code>	Single quote (apostrophe)
$\frac{1}{4}$	<code>&frac14;</code>	One quarter
$\frac{1}{2}$	<code>&frac12;</code>	One half
$\frac{3}{4}$	<code>&frac34;</code>	Three quarters
°	<code>&deg;</code>	Degree
(space)	<code>&nbsp;</code>	Nonbreaking space

8) Horizontal Rules

The parts of a document can be separated from each other, making the document easier to read, by placing horizontal lines between them. Such lines are called horizontal rules. The block tag that creates them is `<hr />`. The `<hr />` tag causes a line break (ending the current line) and places a line across the screen.

Note again the slash in the `<hr />` tag, indicating that this tag has no content and no closing tag.

9) The meta Element

The meta element is used to provide additional information about a document. The meta tag has no content; rather, all of the information provided is specified with attributes. The two attributes that are used to provide information are name and content. The user makes up a name as the value of the name attribute and specifies information through the content attribute. One commonly chosen name is keywords; the values of the content attribute associated with the keywords are those which the author of a document believes characterizes his or her document. An example is

```
<meta name="Keywords" content="binary trees, linked lists, stacks"/>
```

Web search engines use the information provided with the meta element to categorize Web documents in their indices.

```
<meta name = "Title" content = "Don Quixote" />
<meta name = "Author" content = "Miguel Cervantes" />
<meta name = "keywords" content = "novel,
  Spanish literature, groundbreaking work" />
```

7. Explain pattern matching using regular expressions in java script with example.

PATTERN MATCHING BY USING REGULAR EXPRESSIONS

- JavaScript has powerful pattern-matching capabilities based on regular expressions.
- There are two approaches to pattern matching in JavaScript: one that is based on the RegExp object and one that is based on methods of the String object.
- The simplest pattern-matching method is search, which takes a pattern as a parameter.
- The search method returns the position in the String object (through which it is called) at which the pattern matched.
- If there is no match, search returns -1.
- Most characters are normal, which means that, in a pattern, they match themselves.

- The position of the first character in the string is 0.
- As an example, the following statement

```
var str = "Rabbits are furry";
var position = str.search(/bits/);
if (position >= 0)
    document.write("'bits' appears in position", position,
                    "<br />");
else
    document.write("'bits' does not appear in str <br />");
```

produce the following output:

```
'bits' appears in position 3
```

CHARACTER AND CHARACTER-CLASS PATTERNS

- Metacharacters are characters that have special meanings in some contexts in patterns.
- The following are the pattern metacharacters:

`\ | () [] { } ^ $ * + ? .`

- Metacharacters can themselves be matched by being immediately preceded by a backslash.
- A period matches any character except newline.
- Example: `/snow./` matches “snowy”, “snowe”, and “snowd”
- Example: `/3\.4/` matches 3.4. *but* `/3.4/` would match 3.4 and 374, among others.
- Example: `[abc]` matches ‘a’, ‘b’ & ‘c’
- Example: `[a-h]` matches any lowercase letter from ‘a’ to ‘h’
- Example: `[^aeiou]` matches any lowercase letter except ‘a’, ‘e’, ‘i’, ‘o’ & ‘u’

Name	Equivalent Pattern	Matches
<code>\d</code>	<code>[0-9]</code>	A digit
<code>\D</code>	<code>[^0-9]</code>	Not a digit
<code>\w</code>	<code>[A-Za-z_0-9]</code>	A word character (alphanumeric)
<code>\W</code>	<code>[^A-Za-z_0-9]</code>	Not a word character
<code>\s</code>	<code>[\r\t\n\f]</code>	A whitespace character
<code>\S</code>	<code>[^ \r\t\n\f]</code>	Not a whitespace character

ANCHORS

- A pattern is tied to a string position with an anchor. A pattern can be specified to match only at the beginning of the string by preceding it with a circumflex (^) anchor.
- For example, the following pattern matches “pearls are pretty” but does not match “My pearls are pretty”:

`/^pearl/`

- A pattern can be specified to match at the end of a string only by following the pattern with a dollar sign anchor. For example, the following pattern matches “I like gold” but does not match “golden”:

`/gold$/`

- Anchor characters are like boundary-named patterns: They do not match specific characters in the string; rather, they match positions before, between, or after characters.

PATTERN MODIFIERS

- The modifiers are specified as letters just after the right delimiter of the pattern.
- The `i` modifier makes the letters in the pattern match either uppercase or lowercase letters in the string.
- For example, the pattern `/Apple/i` matches ‘APPLE’, ‘apple’, ‘APple’, and any other combination of uppercase and lowercase spellings of the word “apple.”
- The `x` modifier allows white space to appear in the pattern.

```

/\d+           # The street number
\s           # The space before the street name
[A-Z][a-z]+   # The street name
/x

```

is equivalent to

```

/\d+\s[A-Z][a-z]+/

```

OTHER PATTERN-MATCHING METHODS OF String

- The `replace` method is used to replace substrings of the `String` object that match the given pattern.
- The `replace` method takes two parameters: the pattern and the replacement string.
- The `g` modifier can be attached to the pattern if the replacement is to be global in the string, in which case the replacement is done for every match in the string.
- The matched substrings of the string are made available through the predefined variables `$1`, `$2`, and so on. For example, consider the following statements:

```

var str = "Fred, Freddie, and Frederica were siblings";str.replace(/Fre/g,
"Boy");

```

- In this example, `str` is set to “Boy, Boydie, and Boyderica were siblings”, and `$1`, `$2`, and `$3` are all set to “Fre”.
- The `match` method is the most general of the `String` pattern-matching methods.
- The `match` method takes a single parameter: a pattern. It returns an array of the results of the pattern-matching operation.
- If the pattern has the `g` modifier, the returned array has all of the substrings of the string that matched.

- If the pattern does not include the g modifier, the returned array has the match as its first element, and the remainder of the array has the matches of parenthesized parts of the pattern if there are any:

```
var str =  
    "Having 4 apples is better than having 3 oranges";  
var matches = str.match(/\d/g);
```

In this example, matches is set to [4, 3].

- The split method of String splits its object string into substrings on the basis of a given string or pattern. The substrings are returned in an array. For example, consider the following code:

```
var str = "grapes:apples:oranges"; var fruit =  
str.split(":");
```

In this example, fruit is set to [grapes, apples, oranges].

8. Write XHTML and javascript code to compute and print binomial coefficient of given n and r values.

```
<html>  
<head>  
<title>Binomial Coefficient</title>  
</head>  
<body>  
<script>  
function binomialCoefficient (n, k){  
  
    // Checkuing if n and k are integer  
    if(Number.isNaN(n) || Number.isNaN(k)){  
        return NaN;  
    }  
  
    if(k < 0 || k > n){  
        return 0  
    }  
  
    if(k === 0 || k === n){  
        return 1  
    }  
  
    if(k === 1 || k === n - 1){  
        return n  
    }  
  
    let res = n;  
    for(let i = 2; i <= k; i++){  
        res *= (n - i + 1) / i;  
    }  
  
    return Math.round(res);  
}
```

```

}

var n = parseFloat(prompt("Enter n value: "));

var r= parseFloat(prompt("Enter r value: "));

document.write(binomialCoefficient(n, r));
</script>
</body>
</html>

```

9. Briefly explain the following

i) Web browsers ii) web servers iii) URL iv) MIME

i) Web browsers:

- Documents provided by servers on the Web are requested by browsers, which are programs running on client machines.
- They are called browsers because they allow the user to browse the resources available on servers.
- Mosaic was the first browser with a graphical user interface.
- A browser is a client on the Web because it initiates the communication with a server, which waits for a request from the client before doing anything.
- In the simplest case, a browser requests a static document from a server.
- The server locates the document among its servable documents and sends it to the browser, which displays it for the user.
- Sometimes a browser directly requests the execution of a program stored on the server.
- The output of the program is then returned to the browser.
- Examples: Internet Explorer, Mozilla Firefox, Netscape Navigator, Google Chrome, Opera etc.,

ii) Web servers :

Web servers are programs that provide documents to requesting browsers. Example: Apache

All the communications between a web client and a web server use the HTTP

- When a web server begins execution, it informs the OS under which it is running & it runs as a background process
- A web client or browser, opens a network connection to a web server, sends information requests and possibly data to the server, receives information from the server and closes the connection.
- The primary task of web server is to monitor a communication port on host machine, accept HTTP commands through that port and perform the operations specified by the commands.
- When the URL is received, it is translated into either a filename or a program name

iii) URL:

- Uniform Resource Locators (URLs) are used to identify different kinds of resources on Internet.
- If the web browser wants some document from web server, just giving domain name is not sufficient because domain name can only be used for locating the server.
- It does not have information about which document client needs. Therefore, URL should be provided.
- The general format of URL is: **scheme: object-address**

- Example: **http: www.vtu.ac.in/results.php**
- The scheme indicates protocols being used. (http, ftp, telnet...)
- In case of http, the full form of the object address of a URL is as follows:
//fully-qualified-domain-name/path-to-document
- URLs can never have embedded spaces
- It cannot use special characters like semicolons, ampersands and colons
- The path to the document for http protocol is a sequence of directory names and a filename, all separated by whatever special character the OS uses. (Forward or backward slashes)
- The path in a URL can differ from a path to a file because a URL need not include all directories on the path
- A path that includes all directories along the way is called a **complete path**.

iv)MIME:

- MIME stands for Multipurpose Internet Mail Extension.
- The server system apart from sending the requested document, it will also send MIME information.
- The MIME information is used by web browser for rendering the document properly.
- The format of MIME is: **type/subtype**
- Example: text/html , text/doc , image/jpeg , video/mpeg
- When the type is either text or image, the browser renders the document without anyproblem
- However, if the type is video or audio, it cannot render the document
- It has to take the help of other software like media player, win amp etc.,
- These softwares are called as **helper applications or plugins**
- These non-textual information are known as **HYPER MEDIA**
- Experimental document types are used when user wants to create a customized information & make it available in the internet
- The format of experimental document type is: **type/x-subtype**
- Example: database/x-xbase , video/x-msvideo
- Along with creating customized information, the user should also create helper applications.
- This helper application will be used for rendering the document by browser.
- The list of MIME specifications is stored in configuration file of web server.

10. Write a program in java script to accept three numbers using prompt and find the largest of three numbers.

```
// program to find the largest among three numbers

<html>

<body>

<script>

// take input from the user

const num1 = parseFloat(prompt("Enter first number: "));

const num2 = parseFloat(prompt("Enter second number: "));
```

```
const num3 = parseFloat(prompt("Enter third number: "));
```

```
const largest = Math.max(num1, num2, num3);
```

```
// display the result
```

```
document.write("The largest number is " + largest);
```

```
</script>
```

```
</body>
```

```
</html>
```