JECRC™
UNIVERSITY
BUILD YOUR WORLD

Bachelor Of Computer Application

Web Tech Assignment

Submitted To –

Mrs Kavita Maam

(Asst. Professor)

(Dept Of CA & IT)

Submitted By –

Parth Agarwal

18BCAN094

4th Semester

30/03/2020

RCM PAGE NO.: 1. DATE: / /

Assignment - III

Q1 How can you integrate css on a web page in three different ways? Describe the advantage and disadvantage of each method.

Internal css :

Internal or embedded css requires to add <style> tag in the <head> section of HTML document.

1. Open HTML page and locate <head> opening tag.
2. write <style type="text/css"> right after the <head> tag.
3. Add css rules on a new line. Ex:-

```
body {
    background-color: blue;
}
h1 {
    color: red;
    padding: 60px;
}
```

4. Type the closing tag </style>

```
<!Doctype html>
<html>
<head>
```

Teacher's Signature .....

```
<style>
body {
    background-color: blue;
}
h1 {
    color: red;
    padding: 60px;
}
</style>
</head>
<body>

<h1> Internal CSS </h1>
<p> This is a paragraph. </p>

</body>
</html>
```

Advantages of Internal CSS:

- You can use class and ID selectors in this style sheet.
- Since you'll only add the code within the same HTML file, you don't need to upload multiple files.

Disadvantages of Internal CSS:

- Adding the code to the HTML document can increase the page's size & loading time

Teacher's Signature.....

External Css:

With external Css, you'll link your web pages to an external .css file, which can be created by any text editor in your device (e.g. Notepad ++).

1. Create a new .css file with the text editor and add the style rules. Ex:

```
.xleftcol {
    float: left;
    width: 33%;
    background: #809900;
}
.xmiddlecol {
    float: left;
    width: 34%;
    background: #eff2df;
}
```

2. In the <head> section of your HTML sheet, add a reference to your external .css file right after <title> tag:

```
<link rel="stylesheet" type="text/css"
href="style.css" />
```

Advantages of External Css:

- Since the Css code is in a seperate document, your HTML files will have

Teacher's Signature.....

a cleaner structure and are smaller in size.
- You can use the same .css file for multiple pages.

Disadvantages of External CSS:

- Your pages may not be rendered correctly until the external css is loaded.
- Uploading or linking to multiple CSS files can increase your site's download time.

Inline CSS:

Inline css is used to style a specific HTML element. For this CSS style, you'll only need to add the style attribute to each HTML tag, without using selectors.

```
<!DOCTYPE html>
<html>
<body style="background-color:black;">
<h1 style="color:white;padding:30px;">
  Inline CSS </h1>
<p style="color:white;"> Something usefull
      here. </p>
</body> </html>
```

Teacher's Signature.....

Advantages of Inline CSS:

- You can easily and quickly insert CSS rules to an HTML page.
- You don't need to create and upload a separate doc as in external style.

Disadvantages of Inline CSS:

- Adding CSS rules to every HTML element is time-consuming and makes your HTML structure messy.
- Styling multiple elements can effect your page's size and download time.

Q2. What is the difference between padding & margin?

The difference between margin and padding is that while padding deals with the inner space (the space between an element content and the border), margin deals with the outer space (the space outside of the element) to the next outer element.
Thus padding and margin have crucial differences that allows them to play key roles in styling a web page.

Q3. How is the concept of inheritance applied in CSS?

Inheritance is a process of receiving values of properties by a child element from its parent element.

Ex:

```
<!DOCTYPE html>
<html>
<head>
<style>
 h1 {
      color: maroon;
 }
</style>
</head>
<body>
<h1> This is to test <i> inheritance </i>
 in Css </h1>
</body>
</html>
```

In the Internal Css, we have set color for H1 element as maroon. Now look at the html source code, we have an i element located within H1 element, to make the word inheritance italic. Because of inheritance, word inheritance

Teacher's Signature .....

has also become maroon since it is a child element of H1.

Q4. Explain All the selectors in CSS.

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship b/w them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

→ Simple selectors:

CSS element selector – selects HTML elements based on element name.

Ex:-
```
p {
    text-align: center;
    color: red;
}
```

Teacher's Signature.....

CSS id selector - uses id attribute of an HTML element to select a specific element.
character is written, followed by the id of the element.

Ex: #para1 {
text-align : center;
color: red;
}

 Something is written

CSS class selector - selects HTML elements with a specific class attribute.
To select elements with a specific class, write a period (.) character, followed by the class name.

Ex: .center {
text-align : center;
color: red;
}

 Something is written

Teacher's Signature.....

→ Combinator selectors:

There are four different combinators in css:

- descendant selector (space) - matches all elements that are descendants of a specified element.

Ex: div p {
background-color: yellow;
}

- child selector (>) - selects all elements that are children of a specified element.

Ex: div > p {
background-color: yellow;
}

- Adjacent sibling selector (+) - selects all elements that are the adjacent siblings of a specified element.

Ex: div + p {
background-color: yellow;
}

- General Sibling selector (~) – selects all elements that are siblings of a specified element.

Ex:
```
div ~ p {
    background-color : yellow;
}
```

→ Pseudo-class selectors:

Syntax:

```
selector : pseudo-class {
    property : value ;
}
```

- Anchor Pseudo-classes –

Ex:

```
/* unvisited link */
a:link {
    color: #FF0000;
}
/* visited link */
a:visited {
    color: #00FF00;
}
/* mouse over link */
```

Teacher's Signature.....

```
a:hover {
    color: #FF00FF;
}

/* selected link */
a:active {
    color: #0000FF;
}
```

- Pseudo-classes & css classes — can be combined.

Ex:
```
a.highlight:hover {
    color: #FF0000;
}
```

- Hover on —

Ex:
```
div:hover {
    background-color: blue;
```

→ Pseudo-elements selector: is used to style specified parts of an element.

It can be used to:

- Style the first letter, or line, of an element.
- Insert content before, or after, the content of an element.

Teacher's Signature.....

Syntax :

```
selector::pseudo-element {
        property: value;
    }
```

- The ::first-line Pseudo-element - is used to add a special style to the first line of a text.

```
Ex: p::first-line {
        color: #ff0000;
        font-variant: small-caps;
    }
```

- The ::first-letter Pseudo-element - is used to add a special style to the first letter of a text.

```
Ex: p::first-letter {
        color: #ff0000;
        font-size: xx-large;
    }
```

→ Attribute Selectors :

- Css [attribute] selector - is used to select elements with a specified attribute.

Ex: a [target] {
background-color: yellow;
}

- Css [attribute = "value"] selector - is used to select elements with a specified attribute & value.

Ex: a [target = "_blank"] {
background-color: yellow;
}

- Css [attribute ~= "value"] selector - is used to select elements with an attribute value containing a specified word.

Ex: [title ~= "flower"] {
border: 5px solid yellow;
}

- Css [attribute |= "value"] selector - is used to select elements with the specified attribute starting with specified value.

Teacher's Signature.....

Ex: [class|="top"] {
background: yellow;
}

- Css [attribute^="value"] selector - is used to select elements whose attribute value begins with a specified value.

Ex: [class^="top"] {
background: yellow;
}

- Css [attribute$="Value"] selector - is used to select elements whose attribute value ends with a specified value.

Ex: [class$="test"] {
background: yellow;
}

- Styling Forms - The attribute selectors can be useful for styling forms without class or ID:

Ex: input[type="text"] {
width: 150px;
display: block;
margin-bottom: 10px; }

Q5. What is the difference between CDATA and PCDATA in XML?

CDATA → Unparsed Character Data: Contains the text which is not parsed further in an XML document. Tags inside the CDATA text are not treated as markup and entities will not be expanded.

Ex:-

```
<? xml version = "1.0" ?>
<! DOCTYPE employee SYSTEM " employee.
 dtd " >
<employee>
<! [ CDATA [
  <firstname> vimal </ firstname>
  <lastname> jaiswal </ lastname>
  <email> vimal @ gmail. com </ email>
]] >
</ employee>
```

In the above CDATA example, CDATA is used just after the element employee to make the data / text unparsed, so it will give the value of employee:

```
<firstname> vimal </ firstname> <lastname>
jaiswal </ lastname> <email> vimal @ gmail. com
```

Teacher's Signature.....

</email>

PCDATA → Parsed Character Data :

XML parsers are used to parse all the text in an XML document. PCDATA stands for Parsed Character data. PCDATA is the text that will be parsed by a parser. Tags inside the PCDATA will be treated as markup and entities will be expanded.

In other words you can say that a parsed character data means the XML parser examine the data and ensure that it doesn't content entity if it contains that will be replaced.

Ex:-

```
<? xml version="1.0" ?>
<! DOCTYPE employee SYSTEM "employee.dtd">
<employee>
   <firstname> vimal </firstname>
   <lastname> jaiswal </lastname>
   <email> vimal @ gmail.com </email>
</employee>
```

In the above example, the employee element contains 3 more elements 'first-name', 'lastname', and 'email', so it parses further to get the data / text

Teacher's Signature.....

& firstname, lastname and email to give the value of employee as: vimal jaiswal vimal @ gmail.com

Q6. What are the benefits of XML?

- Simplicity
- Openness
- Extensibility
- Self-Description
- Contains machine-readable context information
- Seperates content from presentation
- Supports multilingual documents and Unicode
- Facilitates the comparison and aggregation of data
- Can embed multiple data types.
- Can embed existing data.

30/03/2020

Assignment - IV

Q1. Differentiate between client side Java Script and server side Java script?

1. Server-side scripting is used at the backend, where the source code is not viewable or hidden at the client side (browser). On the other hand, client-side scripting is used at the front end which users can see from the browser.
2. When a server-side script is processed it communicates to the server. As against, client-side scripting does not need any server interaction.
3. The client-side scripting language involves languages such as HTML, CSS and JavaScript. In contrast, programming languages such as PHP, ASP.net, Ruby, ColdFusion, Python, C#, Java, C++, etc.
4. Server-side scripting is useful in customizing the web pages and implement the dynamic changes in the websites. Conversely, the client-side script can effectively minimize the load to the server.
5. Server-side scripting is more secure than client-side scripting as the server side scripts are usually hidden from the client end, while a client-side script is visible to the users.

Teacher's Signature.....

Q2. What are the pop up boxes available in JavaScript?

There are three types of pop up boxes in JavaScript namely Alert Box, Confirm Box and Prompt Box.

Alert Box: It is used when a warning message is needed to be produced. When the alert box is displayed to the user, the user needs to press OK and proceed

Syntax: alert("your Alert here")

Prompt Box: It is a type of pop up box which is used to get the user input for further use. After entering the required details user have to click OK to proceed next stage else by pressing the cancel button user returning the null value.

Syntax: prompt("your Prompt here")

Confirm Box: It is a type of pop up box which is used to get the authorization or permission from the user. The user has to press the OK or cancel button to proceed.

Syntax : confirm(" your query here ")

Q3. How can you validate a form in java script?

HTML form validation can be done by Javascript.
If a form field (fname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted:

```
<!DOCTYPE html>
<html>
<head>
<script>
function validateForm() {
    var x = document.forms["myform"]["fname"].value;
    if (x == "") {
        alert("Name must be filled out");
        return false;
    }
}
</script>
</head>
<body>
<form name="myform" action="/action_page.php" onsubmit="return validateForm()" method="post">
```

Teacher's Signature.....

Name: <input type = "text" name = "fname"> <input type = "Submit" value = "Submit">
</form>
</body>
</html>

Q4. How can you validate email in java script?

To get a valid email id we use a regular expression (RegEx)
/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/

Javascript code to validate an email id:

```
function ValidateEmail(mail)
{
if (/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/.test(myform.emailAddr.value))
    {
      return (true)
    }
    alert("You have entered an invalid email address!")
    return (false)
}
```

Qs. ~~Explain Java Script Objects.~~
What is innerHTML property in Java Script?

The innerHTML property sets or returns the HTML content (inner HTML) of an element.

Syntax:

Return the innerHTML property:

HTMLElementObject.innerHTML

Set the innerHTML property:

HTMLElementObject.innerHTML = text

Ex:

```
<!DOCTYPE html>
<html>
<body>
<p id="myp"> I am a paragraph.</p>
<p> Click the button get the HTML
 content of the p element.</p>

<button onclick="myFunction()">Try it
 </button>
<p id="demo"></p>
<script>
```

Teacher's Signature.....

```
function myFunction () {
    var x = document.getElementById
("myP").innerHTML;
document.getElementById("demo").
innerHTML = x;
}
</script>
</body>
</html>
```

Q6. Explain Java Script objects.

Objects, in Javascript, is it's most important data-type and forms the building blocks for modern Javascript. These objects are quite different from Javascript's primitive data-types in the sense that while these primitive data-types all store a single value each (depending on their types).

Ex:

```
let school = {
        name: "vivekananda School",
        location: "Delhi",
        established: "1971"
    }
```

Each of these keys is referred to as properties of the object.

Teacher's Signature.....

- Properties of javascript object : The property names are numbers, they must be accessed using the "bracket notation" like this :

```
let school = {
    name: 'vivekananda school',
    location: 'Delhi',
    established: '1971',
    20:100,
    displayInfo : function () {
        console.log(`The value of the key 20 is ${school['20']}`);
    }
}
school.displayInfo();
```

- Creating objects : Several ways to create objects :

1. Using the Object literal syntax
2. Object Constructor
3. Constructors
4. Prototypes

- Accessing object members :

1. Dot Notation - (objectName.memberName)
2. Bracket Notation - objectName["memberName"]

Delete operator to delete property of object.

Teacher's Signature.....