Computer Engineering

Web Programming

3160713

IMP Question With Solution

APY MATERIAL

GUJARAT TECHNOLOGICAL UNIVERSITY
BE SEMESTER–VI • EXAMINATION

Subject Name: Web Technology and Programming

Q-1 (a) What is Cascading Style Sheet (CSS)? Explain different types of CSS with example.

What Is CSS?

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.Most web pages are made from **HTML**, or hypertext markup language. This is the standard way to decorate plain web text with fonts, colors, graphic doodads, and hyperlinks (clickable text that magically transports the user somewhere else). But websites can get really big. When that happens, HTML is a very hard way to do a very easy thing. **CSS** (cascading style sheets) can make decorating web sites easy again!

Three Types of CSS

CSS comes in three types:

In a separate file (**external**)
At the top of a web page document (**internal**)
Right next to the text it decorates (**inline**)

External style sheets are separate files full of CSS instructions (with the file extension .css). When any web page includes an external stylesheet, its look and feel will be controlled by this CSS file (unless you decide to override a style using one of these next two types). This is how you change a whole website at once. And that's perfect if you want to keep up with the latest fashion in web pages without rewriting every page!

Internal styles are placed at the top of each web page document, before any of the content is listed. This is the next best thing to external, because they're easy to find, yet allow you to 'override' an external style sheet -- for that special page that wants to be a nonconformist!

Inline styles are placed right where you need them, next to the text or graphic you wish to decorate. You can insert inline styles anywhere in the middle of your HTML code, giving you real freedom to specify each web page element. On the other hand, this can make maintaining web pages a real chore!

CSS Instructions

Before we introduce CSS, let's briefly review HTML. A simple web page is made of **tags**. Everything must go between the opening and closing <html> tags. The <head> section contains

invisible directions called **meta information**. The <body> section is where we put all the visible stuff. Here's a super simple HTML example:

```
<html>  <!-- Opening tag -->

    <head>
        <!-- The head section contains hidden meta information -->
    </head>

    <body>
        <!-- The body section contains our web page content -->

        <!-- Large headings are labeled h1 -->
        <h1>John Adams</h1>
    </body>

</html> <!-- Closing tag -->
```

HTML code for a typical HTML page

What the page would look like

Simple

External CSS

Now here is a short, simple example of CSS using an external file (we'll call it 'stylish.css'). We're going to tell our web page to be white and to make our h1 heading, noted in our simple HTML example as 'John Adams', appear in red at 24 units high:

```
/* CSS Document */

body {
    background-color: white;
}

h1 {
    color: red;
    font-size: 24px;
}
```

CSS Code as a Separate, External File

In the sample file, the top line is a comment and doesn't do anything. The next part (called body) tells the web page what background color to use for the**body** section. Right after that, the h1 part says we want our largest heading (h1) to be the color red and its font size to be 24 units high.

Now, to include this external CSS file ('stylish.css'), we have to include a link for it within the <head> section of our blank web page, as shown on screen:

```
<html>

    <head>
        <!-- An example of including an external CSS stylesheet -->
        <link rel="stylesheet" type="text/css" href="stylish.css">
    </head>

    <body>
        <!-- Large headings are labeled h1 -->
        <h1>John Adams</h1>
    </body>

</html>
```

How to include an external CSS file

What the page would look like

How to Include External CSS

Internal CSS

We don't need to include an external CSS file just to decorate one web page. We can just define our styles at the top of the page, in the <head> section. Here's a quick example in which we're making our heading (h1) blue at a font size of 28 px:

```
<html>

    <head>
        <!-- An example of an internal CSS style -->
        <style>
            h1 {
                color:blue;
                font-size: 28px;
            }
        </style>
    </head>

    <body>
        <!-- Large headings are labeled h1 -->
        <h1>John Adams</h1>
    </body>

</html>
```

How to write internal CSS

John Adams

What the page would look like

(b) Explain following HTML tags with example and important attributes.

1. <meta> OR What do you mean by meta tag? Explain different meta tags.

Definition and Usage

Metadata is data (information) about data.

The <meta> tag provides metadata about the HTML document. Metadata will not be displayed on the page, but will be machine parsable.

Meta elements are typically used to specify page description, keywords, author of the document, last modified, and other metadata.

The metadata can be used by browsers (how to display content or reload page), search engines (keywords), or other web services.

Examples

Example 1 - Define keywords for search engines:

<meta name="keywords" content="HTML, CSS, XML, XHTML, JavaScript">

Example 2 - Define a description of your web page:

<meta name="description" content="Free Web tutorials on HTML and CSS">

Example 3 - Define the author of a page:

<meta name="author" content="Hege Refsnes">

Example 4 - Refresh document every 30 seconds:

<meta http-equiv="refresh" content="30">

Attributes

Attribute	Value	Description
charset	*character_set*	Specifies the character encoding for the HTML document
content	*text*	Gives the value associated with the http-equiv or name attribute
http-equiv	content-type default-style refresh	Provides an HTTP header for the information/value of the content attribute
name	application-name author description generator keywords	Specifies a name for the metadata
scheme	*format/URI*	Not supported in HTML5. Specifies a scheme to be used to interpret the value of the content attribute

2. <a>

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.

By default, links will appear as follows in all browsers:

An unvisited link is underlined and blue
A visited link is underlined and purple
An active link is underlined and red

Attributes

Attribute	Value	Description
charset	*char_encoding*	Not supported in HTML5. Specifies the character-set of a linked document
coords	*coordinates*	Not supported in HTML5. Specifies the coordinates of a link
download	*filename*	Specifies that the target will be downloaded when a user clicks on the hyperlink
href	*URL*	Specifies the URL of the page the link goes to
hreflang	*language_code*	Specifies the language of the linked document
media	*media_query*	Specifies what media/device the linked document is optimized for
name	*section_name*	Not supported in HTML5. Use the global id attribute instead. Specifies the name of an anchor
rel	alternate author bookmark	Specifies the relationship between the current document and the linked document

	help license next nofollow noreferrer prefetch prev search tag	
rev	*text*	Not supported in HTML5. Specifies the relationship between the linked document and the current document
shape	default rect circle poly	Not supported in HTML5. Specifies the shape of a link
target	_blank _parent _self _top *framename*	Specifies where to open the linked document
type	*media_type*	Specifies the media type of the linked document

3. <form>

Description

The HTML <form> tag is used for creating a form for user input. A form can contain textfields, checkboxes, radio-buttons and more. Forms are used to pass user-data to a specified URL.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML form Tag</title>
</head>
<body>
<form action="/cgi-bin/hello_get.cgi" method="get">
First name:
<input type="text" name="first_name" value="" maxlength="100" />
<br />
Last name:
<input type="text" name="last_name" value="" maxlength="100" />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

Attribute	**Value**	**Description**
Accept	MIME_type	Specifies a comma-separated list of content types that the

		server accepts.
accept-charset	charset list	Specifies a list of character encodings that the server accepts. The default value is "unknown".
Action	URL	Specifies a URI/URL of the back-end script that will process the form
autocomplete	on off	Specifies whether form should have autocomplete on or off
Enctype	mimetypes	The mime type used to encode the content of the form.
Method	get post	Specifies the HTTP method to use when the form is submitted. Possible values: get (the form data is appended to the URL when submitted) post (the form data is not appended to the URL)
Name	form name	Defines a unique name for the form.
novalidate	novalidate	Specifies that the form should not be validated when submitted.
target	_blank _self _parent _top	Target to open the given URL. _blank - the target URL will open in a new window _self - the target URL will open in the same frame as it was clicked _parent - the target URL will open in the parent frameset _top - the target URL will open in the full body of the window

4. <img>

Description

The HTML <img> tag is used to put an image in an HTML document.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Tag</title>
</head>
<body>
<img src="http://www.tutorialspoint.com/images/html.gif" alt="HTML Tutorial" height="150" width="140" />
</body>
</html>
```

Specific Attributes

The HTML <img> tag also supports following additional attributes:

Attribute	**Value**	**Description**
Align	top bottom middle left right	*Deprecated*-Specifies the alignment for the image.
Alt	text	Specifies alternate text
Border	pixels	*Deprecated* - Specifies the width of the image border.

crossorigin	anonymous use-credentials	It allows images from third-party sites that allow cross-origin access to be reused with canvas.
Height	pixels or %	Specifies the height of the image.
Hspace	pixels	*Deprecated* - Amount of white space to be inserted to the left and right of the object.
Ismap	URL	Defines the image as a server-side image map.
longdesc	text	*Deprecated*-Specifies a URI/URL of a long description - this can elaborate on a shorter description specified with the alt attribute.
Src	URL	the url of an image
Usemap	#mapname	Defines the image as a client-side image map and used alongwith <map> and <area> tags.
Vspace	pixels	*Deprecated* - Amount of white space to be inserted to the top and bottom of the object.
Width	pixels or %	Sets the width of an image in pixels or in %.

Q-2 (a) Write a javascript code to find whether given number is prime or not.

```
<html>
<head><TITLE></TITLE>
<script language="JavaScript">
var a=prompt("enter a value");
alert(a);
var c=0;

 for(var i=2;i<a;i++)
  {
   if(a%i==0)
   {
    c=i;
    //alert("Now C IS:"+c);
   }
  }
  if(c==0)
  {
   alert("given no is prime");
  }
  else
  {
   alert("given no is not a prime");
  }
</script>
</head>
<body></body>
```

</html>
}

Q.2 (b) What is DHTML? Write a code to display animation using DHTML.

DHTML was essentially a combination of HTML for markup, CSS for style, and JavaScript for manipulating both of those (mostly in the forms of mouseovers and form validation).

DHTML was good in theory, but the two companies pushing the technology each developed distinct ways of coding the same behavior and accessing the same parts of a document. This led many developers down the dark path to "code forking" --that is, writing code to do the same thing in at least two different ways, in order to supply each browser with code only it understands through an intricate (and often fragile) system of browser-detection scripts.

For instance, Netscape enabled you to interact with its proprietary label elements when you gave them a unique ID:

document.layers['myLayer'];

Microsoft enabled similar access, but through a slightly different method:

document.all['myLayer'];

And the differences didn't end there.

Programming in **DHTML** was time-consuming both in initial development as well as maintenance. Scripts, which should have been short and sweet, sprawled out over hundreds of lines of code. It was incredibly inefficient and frustrating for many. Eventually, **DHTML** became something of a black art, and the topic wasn't often discussed in polite web design circles. You will sometimes hear the collective access methods of this period in the history of the DOM referred to as "DOM Level 0."

Q-3 (a) What is Schema and DTD? Explain it with example.

A Document Type Definition (DTD) is a file associated with SGML and XML documents that defines how markup tags should be interpreted by the application reading the document. The DTD uses SGML syntax to explain precisely which elements and attributes may appear in a document and the context in which they may be used. DTDs were briefly introduced earlier in this chapter. In this section, we'll take a closer look.

A DTD is a text document that contains a set of rules, formally known as element declarations , attlist (attribute) declarations, and entity declarations. DTDs are most often stored in a separate

file (with the .dtd suffix) and shared by multiple documents; however, DTD information can be included inside the XML document as well. Both methods are demonstrated later in this section.

The following example is made up of lines taken from the XHTML Strict DTD (the full DTD is over 1,500 lines long). It contains samples of element , attlist (attribute), and entity declarations.

```
<!ELEMENT title (#PCDATA)>
<!ELEMENT meta EMPTY>
<!ELEMENT ul (li)+>

<!ENTITY % i18n
 "lang       %LanguageCode; #IMPLIED
  xml:lang   %LanguageCode; #IMPLIED
  dir        (ltr|rtl)      #IMPLIED"
  >

<!ATTLIST title
  %i18n;
  id         ID             #IMPLIED
  >

<!ATTLIST meta
  %i18n;
  id         ID             #IMPLIED
  http-equiv  CDATA          #IMPLIED
  name       CDATA          #IMPLIED
  content    CDATA          #REQUIRED
  scheme     CDATA          #IMPLIED
  >
```


Q 3 (b) What are cookies in PHP? Explain it with example.

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Only the *name* parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
   echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
   echo "Cookie '" . $cookie_name . "' is set!<br>";
   echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Q.4 (a) Write PHP and HTML code to upload file from client to server.

Form content:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title></title>
</head>
<body>
   <form action="sendEmail.php" method="post" name="mainform" enctype="multipart/form-data">
```

```
  <table width="500" border="0" cellpadding="5" cellspacing="5">
  <tr>
     <th>To Email</th>
     <td><input name="toEmail" type="text"></td>
  </tr>

  <tr>
     <th>Subject</th>
     <td><input name="fieldSubject" type="text" id="fieldSubject"></td>
  </tr>
  <tr>
    <th>Attach Your File</th>
    <td><input name="attachment" type="file"></td>
  </tr>
  <tr>
     <td colspan="2" style="text-align:center;"><input type="submit" name="Submit" value="Send File"></td>
  </tr>
  </table>
  </form>
</body>
</html>
```

Script to handle:

```
<?php
$to = $_POST['toEmail'];
$subject = $_POST['fieldSubject'];

// Get file info
$tmpName = $_FILES['attachment']['tmp_name'];
$fileType = $_FILES['attachment']['type'];
$fileName = $_FILES['attachment']['name'];

if (file($tmpName)) {
 $file = fopen($tmpName,'rb');
 $data = fread($file,filesize($tmpName));
 fclose($file);

 $randomVal = md5(time());
 $mimeBoundary = "==Multipart_Boundary_x{$randomVal}x";

 $headers = "From: $fromName";

 $headers .= "\nMIME-Version: 1.0\n";
 $headers .= "Content-Type: multipart/mixed;\n" ;
 $headers .= " boundary=\"{$mimeBoundary}\"";

 $message = "This is test email\n\n";
```

```
 $data = base64_encode($data);
}

$result = mail ("$to", "$subject", "$message", "$headers");

if($result){
  echo "A email has been sent to: $to";
 }
else{
  echo "Error while sending email";
}
```

(b) List Five Tags in HTML. Write code to display following table.

A			
B	C		D
	E	F	G
H	I	J	

```
<html>
<body>

   <table align="center">
      <tr>
         <td colspan="4" rowspan="1" align="center"> A </td>
      </tr>
      <tr>
         <td rowspan="2" align="center"> B </td>
         <td colspan="2" align="center"> C </td>
          <td align="center"> D </td>
      </tr>
      <tr>
           <td align="center"> E </td>
           <td align="center"> F </td>
           <td align="center"> G </td>
      </tr>
      <tr>
           <td align="center"> H </td>
           <td align="center"> I </td>
          <td colspan="2" align="center"> J </td>
      </tr>
   </table>

</body>
</html>
```

Q-5(a) List seven mouse events in JavaScript. Explain mouse move and mouse click with proper example.

Seven mouse events:

click
dblclick
mousedown
mouseup
mousemove
mouseover
mouseout

Mousemove

The mousemove event works fine, but you should be aware that it may take quite some system time to process all mousemove events. If the user moves the mouse one pixel, the mousemove event fires. Even when nothing actually happens, long and complicated functions take time and this may affect the usability of the site: everything goes very slowly, especially on old computers.
Therefore it's best to register an onmousemove event handler only when you need it and to remove it as soon as it's not needed any more:

```
element.onmousemove = doSomething;
// later
element.onmousemove = null;
```

mouseclick

If the user clicks on an element no less than three mouse events fire, in this order:

1. mousedown, user depresses the mouse button on this element
2. mouseup, user releases the mouse button on this element
3. click, one mousedown and one mouseup detected on this element

In general mousedown and mouseup are more useful than click. Some browsers don't allow you to read out mouse button information onclick. Furthermore, sometimes the user does something with his mouse but noclick event follows.
Suppose the user depresses the mouse button on a link, then moves his mouse off the link and then releases the mouse button. Now the link only registers a mousedown event. Similarly, if the user depresses the mouse button, then moves the mouse over a link and then releases the mouse button, the link only registers a mouseup. In neither case does a clickevent fire.
Whether or not this is a problem depends on the user interaction you want. But you should generally register your script onmousedown/up, unless you're completely sure you want the click event and nothing else.
If you use alerts in handling these events, the browser may lose track of which event took place on which element and how many times it took place, messing up your scripts. So it's best not to use alerts.

Dblclick

The dblclick event is rarely used. Even when you use it, you should be sure *never* to register both an onclick and an ondblclick event handler on the same HTML element. Finding out what the user has actually done is nearly impossible if you register both.
After all, when the user double–clicks on an element one click event takes place before the dblclick. Besides, in Netscape the second clickevent is also separately handled before the dblclick. Finally, alerts are dangerous here, too.
So keep your clicks and dblclicks well separated to avoid complications.

```
<input type="button" value="click me" id="btn">
<input type="button" value="right-click me" id="btn2">
```

```
<script>
document.getElementById('btn').onclick = function()
{
  alert('click!')
}
document.getElementById('btn2').oncontextmenu = function()
{
  alert('right click!')
}
</script>
```