Department of Computer Engineering

LAB MANUAL

Semester-V

Web Technologies Laboratory

Institute Vision, Mission & Quality Policy

Vision

To foster and permeate higher and quality education with value added engineering, technology programs, providing all facilities in terms of technology and platforms for all round development with societal awareness and nurture the youth with international competencies and exemplary level of employability even under highly competitive environment so that they are innovative adaptable and capable of handling problems faced by our country and world at large.

Mission

The Institution is committed to mobilize the resources and equip itself with men and materials of excellence thereby ensuring that the Institution becomes pivotal center of service to Industry, academia, and society with the latest technology. RAIT engages different platforms such as technology enhancing Student Technical Societies, Cultural platforms, Sports excellence centers, Entrepreneurial Development Center and Societal Interaction Cell. To develop the college to become an autonomous Institution & deemed university at the earliest with facilities for advanced research and development programs on par with international standards. To invite international and reputed national Institutions and Universities to collaborate with our institution on the issues of common interest of teaching and learning sophistication.

Quality Policy

ज्ञानधीनं जगत् सर्वम।

Knowledge is supreme.

Our Quality Policy

It is our earnest endeavour to produce high quality engineering professionals who are innovative and inspiring, thought and action leaders, competent to solve problems faced by society, nation and world at large by striving towards very high standards in learning, teaching and training methodologies.

Our Motto: If it is not of quality, it is NOT RAIT!

Dr. Vijay D. Patil
President, RAES

Department Vision & Mission

Vision

To impart higher and quality education in computer science with value added engineering and technology programs to prepare technically sound, ethically strong engineers with social awareness. To extend the facilities, to meet the fast changing requirements and nuture the youths with international competencies and exemplary level of employability and research under highly competitive environments.

Mission

- To mobilize the resources and equip the institution with men and materials of excellence to provide knowledge and develop technologies in the thrust areas of computer science and Engineering.
- To provide the diverse platforms of sports, technical, cocurricular and extracurricular activities for the overall development of student with ethical attitude.
- To prepare the students to sustain the impact of computer education for social needs encompassing industry, educational institutions and public service.
- To collaborate with IITs, reputed universities and industries for the technical and overall uplifment of students for continuing learning and entrepreneurship.

Departmental Program Educational Objectives (PEOs)

1. **Learn and Integrate**

   To provide Computer Engineering students with a strong foundation in the mathematical, scientific and engineering fundamentals necessary to formulate, solve and analyze engineering problems and to prepare them for graduate studies.

2. **Think and Create**

   To develop an ability to analyze the requirements of the software and hardware, understand the technical specifications, create a model, design, implement and verify a computing system to meet specified requirements while considering real-world constraints to solve real world problems.

3. **Broad Base**

   To provide broad education necessary to understand the science of computer engineering and the impact of it in a global and social context.

4. **Techno-leader**

   To provide exposure to emerging cutting edge technologies, adequate training & opportunities to work as teams on multidisciplinary projects with effective communication skills and leadership qualities.

5. **Practice citizenship**

   To provide knowledge of professional and ethical responsibility and to contribute to society through active engagement with professional societies, schools, civic organizations or other community activities.

6. **Clarify Purpose and Perspective**

   To provide strong in-depth education through electives and to promote student awareness on the life-long learning to adapt to innovation and change, and to be successful in their professional work or graduate studies.

Program Outcomes

PO1: **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Index

List of Experiments

Sr. No.	Experiments Name
1	To study and implement all basic HTML tags.
2	To implement Cascading Style Sheet
3	To design email registration form and validate it using Javascript
4	To implement animation using Javascript
5	To implement animation using jQuery.
6	To design home page for TOI using Quanta Plus/ Aptana/ Kompozer
7	To design online examination form using Quanta Plus/ Aptana/ Kompozer
8	To design home page for online mobile shopping using Quanta Plus/Aptana/ Kompozer
9	Design HTML form to accept the two numbers N1 and N2. Display prime numbers between N1 and N2 using PHP.
10	Design a login form to add username, id, password into database & validate it (use php).
11	Design course registration form and perform various database operations using PHP and MySQL database connectivity
12	To design XML document using XML schema for representing your semester marksheet using PHP.
13	To design DTD for representing your semester marksheet.
14	To design XML schema and DTD for railway reservation system.
15	Mini Project

Course Objectives, Course Outcome & Experiment Plan

Course Objectives:

1.	To provide students an overview of the concepts Web Technologies, and HTML.
2.	Learn, defining a CSS and unstaring its purpose different syntax and types of CSS.
3.	Learn how to define client side scripting and understand its advantages and disadvantages. Embedding JavaScript code into HTML document using script tag, and will understand dynamic HTML.
4.	Learn how will introduce editors for development of web pages.
5.	Learn about basics of XML and how it can be used to store information away from the mechanism of processing or formatting of such data. Will also learn how to build simple XML files and be able to manipulate and refer to them.
6.	Give a basic introduction of to PHP and dynamic programming on the server side.

Course Outcomes:

CO1	Students will be able to understand concepts of Web Technologies and will be able to design static client side web page using various HTML tags and different designing tools e.g. Quanta plus/Aptana/Kompozer
CO2	Student can create web pages using CSS to provide different styling layout
CO3	Students can create animation using client side scripting-JavaScript.
CO4	Students will be able to create XML file to store information.
CO5	Students will be able to create a server side script using PHP and they will be able to create dynamic web pages using database connectivity
CO6	Students will be able to develop different web applications

Experiment Plan

Module No.	Week No.	Experiments Name	Course Outcome	Weightage
1	W1	To study and implement all basic HTML tags.	CO1	3
2	W1	To implement Cascading Style Sheet	CO2	10
3	W2	To design email registration form and validate it using Javascript	CO3	2
4	W3	To implement animation using Javascript	CO3	6
5	W4	To implement animation using jQuery.	CO3	2
6	W5	To design home page for TOI using Quanta Plus/ Aptana/ Kompozer	CO1	3
7	W5	To design online examination form using Quanta Plus/ Aptana/ Kompozer	CO1	2
8	W6	To design home page for online mobile shopping using Quanta Plus/Aptana/ Kompozer	CO1	2
9	W7	Design HTML form to accept the two numbers N1 and N2. Display prime numbers between N1 and N2 using PHP.	CO5	2
10	W8	Design a login form to add username, id, password into database & validate it (use php).	CO5	2
11	W9	Design course registration form and perform various database operations using PHP and MySQL database connectivity	CO5	6
12	W10	To design XML document using XML schema for representing your semester marksheet using PHP.	CO4	3
13	W11	To design DTD for representing your semester marksheet.	CO4	3
14	W12	To design XML schema and DTD for railway reservation system.	CO4	4
15	W12	Mini Project	CO6	10

Mapping Course Outcomes (CO) - Program Outcomes (PO)

Subject Weight	**Course Outcomes**		**Contribution to Program outcomes (POs)**											
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
PRACTICAL/ ORAL + TERM WORK **100%**	CO1	Students will be able to understand concepts of Web Technologies and will be able to design static client side web page using various HTML tags and different designing tools e.g. Quanta plus/Aptana/Kompozer.	2	2	3		3							
	CO2	Student can create web pages using CSS to provide different styling layout.		1	3		3	1			2			
	CO3	Students can create animation using client side scripting-JavaScript.			2	2	1		2		2			1
	CO4	Students will be able to create XML file to store information.			1	2	2			1	2			2
	CO5	Students will be able to create a server side script using PHP and they will be able to create dynamic web pages using database connectivity.		2		2	2			1	1		1	1
	CO6	Students will be able to develop different web applications		1		3					1	2	2	1

Study and Evaluation Scheme

Course Code	Course Name	Teaching Scheme			Credits Assigned			
CPL501	Web Technologies Lab	Theory	Practical	Tutorial	Theory	Practical	Tutorial	Total
		--	04	--	--	02	--	02

Course Code	Course Name	Examination Scheme		
CPL501	Web Technologies Lab	Term Work	Practical	Total
		25	50	75

Term Work:

1. Term work will consist of small assignments testing all the technologies included in syllabus and a Mini project solving an appropriate problem using the above technology The distribution of marks for term work shall be as follows:

· Assignments: ……………………………………………… (20) Marks.
· Project Report Presentation…………………………….. (15) Marks.
· Group Discussion.………………………………………… (10) Marks.
· Attendance ………………………………………………… (05) Marks
TOTAL: ……………………………………………………. (50) Marks

Practical & Oral:

A. Oral examination is to be conducted by pair of internal and external examiners based on the mini projects undertaken by student groups.

Web Technologies

Experiment No. : 1

To study and implement all basic HTML tags.

Experiment No. 1

1. **Aim:** To study and implement all basic HTML tags.

2. **Objectives:** From this experiment, the student will be able to
   - Learn the overview of Web Technologies
   - Learn basic HTML tags
   - Learn to design effective website using different HTML tags

3. **Outcomes:** The learner will be able to

   - Apply knowledge of HTML tags and create Web Forms
   - Understand, identify, analyse and design the web page
   - Recognize the need for HTML and be able to engage continuing professional development

4. **Software Required :** Internet Explorer

5. **Theory:**

***HTML** is a language for describing web pages.*

- HTML stands for Hyper Text Markup Language
- HTML is not a programming language, it is a markup language
- A markup language is a set of markup tags
- HTML uses markup tags to describe web pages

***HTML Elements:** HTML documents are text files made up of HTML elements.HTML elements are defined using HTML tags.*

HTML Tags:

- HTML tags are used to mark-up HTML elements.
- HTML tags normally comes in pairs like <b>and </b>.
- The first tag in pair is the start tag; the second tag is the end tag.
- The text between the start and end tags is the element content.
- Some HTML elements have a missing end tag.
- HTML tags are not case sensitive; <b> means the same as <B>.

Tag Attributes: *Attributes can provide additional information about the HTML elements. Consider <body> tag with an added bgcolor attribute, we can set the background color of web page. For example, <body bgcolor=”red”> this will set the background color to red.*

- Attributes always come in name/value pairs like name=”value”.
- Attributes are always specified in the start tag.

HTML Headings:

- Headings are defined with the <h1> to <h6> tags.
- <h1> defines the largest heading. <h6> defines the smallest heading.

HTML Paragraphs:

- Paragraphs are defined with the <p> tag.
- HTML automatically adds an extra blank line before and after a paragraph.

HTML Formatting Tags:

- HTML uses tags like <b> and <i> for formatting text in bold and italic form.
- The <br> tag is used when we want to end a line, but don’t want to start a new paragraph.
- The <br> tag is an empty tag. It has no closing tag.

Anchor Tag and Href attribute:

- HTML uses the <a> (anchor) tag to create a link to another document.
- An anchor can point to any resource on web for example an HTML page, an image, a sound file etc.
- The syntax of creating an anchor:

  <a href=”url”> Text to be displayed </a>
- **HREF** stands for **H**ypertext **REF**erence. The href attribute is used to create a hypertext link.

The target Attribute:

- The **target attribute** defines **where** the linked document will be opened.

The name Attribute:

- The **name attribute** is used to create a named anchor.

The Image Tag and the Src Attribute :

- In HTML, images are defined with the <img> tag.
- The <img> tag is empty, which means that it contains attributes only and it has no closing tag.
- To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display on your page.URL points to the location where the image is stored.
- The syntax of defining an image:

  <img src="url">

The Alt Attribute :

- The alt attribute is used to define an "alternate text" for an image. The value of the alt attribute is an author-defined text:

  `<img src="boat.gif" alt="Big Boat">`

- The "alt" attribute tells the reader what he or she is missing on a page if the browser can't load images. The browser will then display the alternate text instead of the image. It is a good practice to include the "alt" attribute for each image on a page, to improve the display and usefulness of your document for people who have text-only browsers.

List Tags:

Tag	*Description*
<ol>	*Defines an ordered list*
<ul>	*Defines an unordered list*
<li>	*Defines a list item*
<dl>	*Defines a definition list*
<dt>	*Defines a definition term*
<dd>	*Defines a definition description*

Table Tags:

Tag	Description
<table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell
<caption>	Defines a table caption
<colgroup>	Defines groups of table columns
<col>	Defines the attribute values for one or more columns in a table
<thead>	Defines a table head
<tbody>	Defines a table body
<tfoot>	Defines a table footer

With frames, you can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others. Frames allow content and navigation to be separated from each other. It is used typically to present a left hand navigation menu and a top banner giving access to the main parts of the site.

A frame can contain interactive tables of the contents with links that when clicked display results in adjoining frame. Frames designed side by side permit queries to be passed and answered on the same page, with one frame holding the query frame and the other presenting the results.

The Frameset Tag:

- The <frameset> tag defines how to divide the window into frames
- Each frameset defines a set of rows or columns

- The values of the rows/columns indicate the amount of screen area each row/column will occupy

The Frame Tag:

- The <frame> tag defines what HTML document to put into each frame In the example below we have a frameset with two columns. The first column is set to 25% of the width of the browser window. The second column is set to 75% of the width of the browser window. The HTML document "frame_a.htm" is put into the first column, and the HTML document "frame_b.htm" is put into the second column:

```
<frameset cols="25%,75%">
  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
</frameset>
```

The frameset column size value can also be set in pixels (cols="200,500"), and one of the columns can be set to use the remaining space (cols="25 %,*").

Attributes of Frame Tag:

1. src = ”url” :- Specifies the address of the document to be displayed in the frame.
2. name = ”window name” :- This is used to assign the name to frame so that it can be targeted by links in other documents.
3. frameborder = ”1/0” :- When set to 1 a separator is drawn on every side next to another frame. Default value is 1. When set to 0 there is no separator.
4. marginwidth=”value in pixel”:- This is used when some margin for the frame is required horizontally.
5. marginheight=”value in pixel”:- This is used when some margin for the frame is required vertically.
6. noresize :- It means frame is not resizable by the user. It does not have any value.
7. scrolling = ”yes/no” :-This indicates whether the frame should have scrollbars or not. Default value is yes.

Frame Tags :

Tag	Description
<frameset>	Defines a set of frames
<frame>	Defines a sub window (a frame)
<noframes>	Defines a noframe section for browsers that do not handle frames
<iframe>	Defines an inline sub window (frame)

The Form Tag:

Forms are used to collect information from user and transmit that information to server for processing. It is an area that can contain form elements. Form elements are elements that allow the user to enter information like text fields, text area fields, drop-down menus, radio buttons, checkboxes, etc. in a form.

A form is defined with the <form> tag. For example:

```
<form>
    <input>
    </input>
</form>
```

Input: The most used form tag is the <input> tag. The type of input is specified with the type attribute. The most commonly used input types are explained below.

Text Fields: Text fields are used when you want the user to type letters, numbers, etc. in a form. The width of the text field is 20 characters by default.

Consider the following example:

```
<form>
First name:
<input type="text" name="first name">
<br>
Last name:
<input type="text" name="last name">
</form>
```

Radio Buttons: Radio Buttons are used when user want to select one of a limited number of choices. Only one option can be chosen at a time.

Consider the following example:

```
<form>
<input type="radio" name="sex" value="male"> Male
<br>
<input type="radio" name="sex" value="female"> Female
</form>
```

Checkboxes: Checkboxes are used when you want the user to select one or more options of a limited number of choices.

Consider the following example:

```
<form>
I have a bike:
<input type="checkbox" name="vehicle" value="Bike">
<br>
I have a car:
```

```
<input type="checkbox" name="vehicle" value="Car">
<br>
I have an airplane:
<input type="checkbox" name="vehicle" value="Airplane">
</form>
```

The Form's Action Attribute and the Submit Button:

When the user clicks on the "Submit" button, the content of the form is sent to the server. The form's action attribute defines the name of the file where the user wants to send the contents of form. The file defined in the action attribute usually does something with the received input.

Consider the following example:

```
<form name="input" action="html_form_submit.asp" method="get">
 Username:    <input type="text" name="user">
<input type="submit" value="Submit">
</form>
```

From Event Handlers :

i. onSubmit Event Handler :

It is executed when form is submitted using a submt <input> tag. The onSubmit event is used to validate ALL form fields before it is send to a server.

Below is an example of how to use the onSubmit event. The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be cancelled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<form method=”post” action=”xxx.htm”
onsubmit=”return checkForm()”>
```

ii. onReset Event Handler :

It is executed when a reset button is selected on a form. It is used to confirm that the user really wants to reset the contents of a form.

iii. onLoad and onUnload Event Handler:

The onLoad and onUnload events are triggered when the user enters or leaves the page.The onLoad event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

iv. onFocus, onBlur and onChange Event Handler :

The onFocus, onBlur and onChange events are often used in combination with validation of form fields. Below is an example of how to use the onChange event. The checkEmail() function will be called whenever the user changes the content of the field:

```
<input type="text" size="30" id="email" onchange="checkEmail()">
```

v. onMouseOver and onMouseOut Event Handler :

onMouseOver and onMouseOut are often used to create "animated" buttons.

Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected.

6. Conclusion:

We have studied different HTML tags, HTML Elements, HTML paragraphs, formatting tags. We also analysed tag attribute, anchor tag, frame tag, table tag, list tags for creating web pages. Thus able to understand how to design web page for given problem by analysing it.

7. Viva Questions:

- What are web pages?
- Which language is interpreted by browser?
- Other than HTTP, which protocol is supported by a web server?

8. References:

- Ralph Moseley, M.T. Savaliya, “Developing Web Applications”, Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- “Web Technology Black Book”, Dreamtech Press, First Edition, 978-7722-997
- www.tutorialspoint.com/html/html_tutorial.pdf
- codesparta.com/2015/11/01/complete-html-tags-list-with-examples-free-pdf/
- https://www.projecta.com/Files/BasicHtmlTags.pdf

Web Technologies

Experiment No. : 2

To implement cascaded style sheet.

Experiment No. 2

1. **Aim:** To implement cascaded style sheet.

2. **Objectives:** From this experiment, the student will be able to
   - Learn cascaded style sheet
   - Analyse its purpose, different syntax and types of CSS
   - Create web page using CSS to provide different styling layout

3. **Outcomes:** The learner will be able to
   - Use current styles in creating web pages
   - Identify, formulate and solve engineering problems
   - Recognize the need for CSS and be able to engage continuing professional development

4. **Software Required :** Internet Explorer

5. **Theory:**

CSS (CASCADED STYLE SHEET) :

- **CSS** is a simple mechanism for adding style to web documents.
- Style sheet enables you to change the appearance and layout of the web page.
- Style sheet define how to display HTML elements.
- With plain Html we define the colors and sizes of the text and tables throughout our pages. If we want to change a certain element we will therefore have to work through the document and change it. Whereas with css we define the colors and sizes in styles. If we change a certain style it will change the look of entire site.
- Even multiple external style sheets can be referenced inside a single HTML document.

Syntax:

The CSS syntax is made up of three parts : a selector, a property and a value
selector {property: value}
The selector is normally the HTML element/tag you wish to define, the property is the attribute you wish to change, and each property can take a value. The property and value are separated by a colon, and surrounded by curly braces:
body {color: black}

If the value is multiple words, put quotes around the value:
p {font-family: "sans serif"}

If we wish to specify more than one property, we must separate each property with a semicolon. The example below shows how to define a center aligned paragraph, with a red text color:
p {text-align:center;color:red}

To make the style definitions more readable, you can describe one property on each line, like this:

```
p{
text-align: center;
color: black;
font-family: arial
}
```

Types Of CSS :

A. External Style Sheet :

- External CSS is a file that contains only CSS code and is saved with a .css file extension.
- With an external style sheet we can change the look of an entire web site by changing one file.
- Each page must link to the style sheet using the <link> tag.
- The <link> tag goes inside the head section.
- We can use one external style sheet with multiple HTML files.

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

B. Internal Style Sheet :

- Internal style sheet is defined inside the head section by using the <style> tag.
- An internal style sheet should be used when a single document has a unique style.

```
<style type=”text/css”>
hr{color:sienna}
p{margin-left:20px}
body{background-image:url(“images/back40.gif”)}
</style>
```

C. Inline Style Sheet :

- Inline style sheet is also called as embedded style sheet.
- This is defined along with the HTML tags.
- Inline Style Sheet has more priority than Internal and External Style Sheet.

```
<div style=”color:red;”><p style=”font-size:12px”>
```

6. Conclusion:

CSS is used for specific web layout which is not possible with normal HTML tags. For example, h1-h6 tag with colour, placing an image and paragraph at one line. We can use different style sheet to design web page as per the requirement.

7. Viva Questions:

- What is inline style sheet?
- What is purpose of CSS in HTML?
- What are different types of CSS?

8. References:

- Ralph Moseley, M.T. Savaliya, “Developing Web Applications”, Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- “Web Technology Black Book”, Dreamtech Press, First Edition, 978-7722-997
- www.w3schools.com/css/css_reference.asp (list of all CSS properties)
- www.glish.com/css/
- www.html.net/tutorials/css/
- http://blog.html.it/layoutgala/

Web Technologies

Experiment No. : 3

To design email registration form and validate it using Javascript.

Experiment No. 3

1. **Aim:** Design bank transaction form using JavaScript.

2. **Objectives:** From this experiment, the student will be able to

   - Learn how to define document and window object
   - Learn how to define client side scripting
   - Understand its advantages and disadvantages
   - Embed javascript code into HTML document using script tag

3. **Outcomes:** The learner will be able to
   - Communicate effectively with range of audiences in form of dialog box
   - Use current skills to embed javascript into HTML document
   - Identify, formulate and solve the engineering problem

4. **Software Required :** Web Browser

5. **Theory:**

JavaScript is the world's most popular programming language. It is the language for HTML, for the web, for servers, PCs, laptops, tablets, phones, and more.

- JavaScripts in HTML must be inserted between <script> and </script> tags.
- JavaScripts can be put in the <body> and in the <head> section of an HTML page.

The <script> Tag

To insert a JavaScript into an HTML page, use the <script> tag. The <script> and </script> tells where the JavaScript starts and ends. The lines between the <script> and </script> contain the JavaScript:

```
<script>
alert("My First JavaScript");
</script>
```

JavaScript in <body>

Example

```
<!DOCTYPE html>
<html>
<body>..........
<script>
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph</p>");
</script>
..........</body>
</html>
```

JavaScript Functions and Events

- The JavaScript statements, in the example above, are executed while the page loads.

- More often, we want to execute code when an **event** occurs, like when the user clicks a button.
- If we put JavaScript code inside a **function**, we can call that function when an event occurs.

JavaScript in <head> or <body>

- You can place an unlimited number of scripts in an HTML document.
- Scripts can be in the <body> or in the <head> section of HTML, and/or in both.
- It is a common practice to put functions in the <head> section, or at the bottom of the page. This way they are all in one place and do not interfere with page content.

External JavaScripts

Scripts can also be placed in external files. External files often contain code to be used by several different web pages. External JavaScript files have the file extension .js. To use an external script, point to the .js file in the "src" attribute of the <script> tag:

Example

```
<!DOCTYPE html>
<html>
<body><script src="myScript.js"></script></body>
</html>
```

<u>JavaScript Objects</u>

"Everything" in JavaScript is an Object. Even primitive data types (except null and undefined) can be treated as objects.

- Booleans can be objects or primitive data treated as objects
- Numbers can be objects or primitive data treated as objects
- Strings are also objects or primitive data treated as objects
- Dates are always objects
- Math and Regular Expressions are always objects
- Arrays are always objects
- Even functions are always objects

An object is just a special kind of data, with **properties** and **methods**.

<u>Accessing Object Properties</u>

Properties are the values associated with an object.
The syntax for accessing the property of an object is:

```
objectName.propertyName
```

This example uses the length property of the String object to find the length of a string:

```
var message="Hello World!";
var x=message.length;
```

The value of x, after execution of the code above will be:

```
12
```

<u>Accessing Objects Methods</u>

Methods are the actions that can be performed on objects.
You can call a method with the following syntax:

```
objectName.methodName()
```

This example uses the toUpperCase() method of the String object, to convert a text to uppercase:

```
var message="Hello world!";
var x=message.toUpperCase();
```

The value of x, after execution of the code above will be:

```
HELLO WORLD!
```

JavaScript Math Object

Math Object Properties

Property	Description
E	Returns Euler's number (approx. 2.718)
LN2	Returns the natural logarithm of 2 (approx. 0.693)
LN10	Returns the natural logarithm of 10 (approx. 2.302)
LOG2E	Returns the base-2 logarithm of E (approx. 1.442)
LOG10E	Returns the base-10 logarithm of E (approx. 0.434)
PI	Returns PI (approx. 3.14)
SQRT1_2	Returns the square root of 1/2 (approx. 0.707)
SQRT2	Returns the square root of 2 (approx. 1.414)

Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of E^x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x

JavaScript String Object

String Object Properties

Property	Description

constructor	Returns the function that created the String object's prototype
length	Returns the length of a string
prototype	Allows you to add properties and methods to an object

String Object Methods

Method	Description
charAt()	Returns the character at the specified index
charCodeAt()	Returns the Unicode of the character at the specified index
concat()	Joins two or more strings, and returns a copy of the joined strings
fromCharCode()	Converts Unicode values to characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
localeCompare()	Compares two strings in the current locale
match()	Searches for a match between a regular expression and a string, and returns the matches
replace()	Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring
search()	Searches for a match between a regular expression and a string, and returns the position of the match
slice()	Extracts a part of a string and returns a new string
split()	Splits a string into an array of substrings
substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices
toLocaleLowerCase()	Converts a string to lowercase letters, according to the host's locale
toLocaleUpperCase()	Converts a string to uppercase letters, according to the host's locale
toLowerCase()	Converts a string to lowercase letters
toString()	Returns the value of a String object
toUpperCase()	Converts a string to uppercase letters
trim()	Removes whitespace from both ends of a string
valueOf()	Returns the primitive value of a String object

JavaScript Boolean Object

Boolean Object Properties

Property	Description
constructor	Returns the function that created the Boolean object's prototype
prototype	Allows you to add properties and methods to a Boolean object

Boolean Object Methods

Method	Description
toString()	Converts a Boolean value to a string, and returns the result
valueOf()	Returns the primitive value of a Boolean object

JavaScript RegExp Object

RegExp Object Properties

Property	Description
constructor	Returns the function that created the RegExp object's prototype
global	Specifies if the "g" modifier is set
ignoreCase	Specifies if the "i" modifier is set
lastIndex	Specifies the index at which to start the next match
multiline	Specifies if the "m" modifier is set
source	Returns the text of the RegExp pattern

RegExp Object Methods

Method	Description
compile()	Deprecated in version 1.5. Compiles a regular expression
exec()	Tests for a match in a string. Returns the first match
test()	Tests for a match in a string. Returns true or false
toString()	Returns the string value of the regular expression

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Brackets are used to find a range of characters:

Expression	**Description**
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any digit between the brackets
[^0-9]	Find any digit NOT between the brackets
(x|y)	Find any of the alternatives specified
Meta character	**Description**
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word
\B	Find a match not at the beginning/end of a word
\0	Find a NUL character
\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character
\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx

\xdd	Find the character specified by a hexadecimal number dd
\uxxxx	Find the Unicode character specified by a hexadecimal number xxxx

Quantifiers

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{X}	Matches any string that contains a sequence of *X* *n*'s
n{X,Y}	Matches any string that contains a sequence of X to Y *n*'s
n{X,}	Matches any string that contains a sequence of at least X *n*'s
n$	Matches any string with *n* at the end of it
^n	Matches any string with *n* at the beginning of it
?=n	Matches any string that is followed by a specific string *n*
?!n	Matches any string that is not followed by a specific string *n*

JavaScript Number Object

Number Object Properties

Property	Description
constructor	Returns the function that created the Number object's prototype
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
NEGATIVE_INFINITY	Represents negative infinity (returned on overflow)
NaN	Represents a "Not-a-Number" value
POSITIVE_INFINITY	Represents infinity (returned on overflow)
prototype	Allows you to add properties and methods to an object

Number Object Methods

Method	Description
toExponential(x)	Converts a number into an exponential notation
toFixed(x)	Formats a number with x numbers of digits after the decimal point
toPrecision(x)	Formats a number to x length
toString()	Converts a Number object to a string
valueOf()	Returns the primitive value of a Number object

JavaScript Array Object

Array Object Properties

Property	Description
constructor	Returns the function that created the Array object's prototype
length	Sets or returns the number of elements in an array
prototype	Allows you to add properties and methods to an Array object

Array Object Methods

Method	Description

concat()	Joins two or more arrays, and returns a copy of the joined arrays
indexOf()	Search the array for an element and returns its position
join()	Joins all elements of an array into a string
lastIndexOf()	Search the array for an element, starting at the end, and returns its position
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array

JavaScript Date Object

Date Object Properties

Property	Description
constructor	Returns the function that created the Date object's prototype
prototype	Allows you to add properties and methods to an object

Date Object Methods

Method	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 (four digits)
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
getTimezoneOffset()	Returns the time difference between UTC time and local time, in minutes
getUTCDate()	Returns the day of the month, according to universal time (from 1-31)
getUTCDay()	Returns the day of the week, according to universal time (from 0-6)
getUTCFullYear()	Returns the year, according to universal time (four digits)
getUTCHours()	Returns the hour, according to universal time (from 0-23)
getUTCMilliseconds()	Returns the milliseconds, according to universal time (from 0-999)
getUTCMinutes()	Returns the minutes, according to universal time (from 0-59)
getUTCMonth()	Returns the month, according to universal time (from 0-11)
getUTCSeconds()	Returns the seconds, according to universal time (from 0-59)
getYear()	Deprecated. Use the getFullYear() method instead
parse()	Parses a date string and returns the number of milliseconds since midnight of Jan 1, 1970
setDate()	Sets the day of the month of a date object
setFullYear()	Sets the year (four digits) of a date object
setHours()	Sets the hour of a date object

setMilliseconds()	Sets the milliseconds of a date object
setMinutes()	Set the minutes of a date object
setMonth()	Sets the month of a date object
setSeconds()	Sets the seconds of a date object
setTime()	Sets a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970
setUTCDate()	Sets the day of the month of a date object, according to universal time
setUTCFullYear()	Sets the year of a date object, according to universal time (four digits)
setUTCHours()	Sets the hour of a date object, according to universal time
setUTCMilliseconds()	Sets the milliseconds of a date object, according to universal time
setUTCMinutes()	Set the minutes of a date object, according to universal time
setUTCMonth()	Sets the month of a date object, according to universal time
setUTCSeconds()	Set the seconds of a date object, according to universal time
setYear()	Deprecated. Use the setFullYear() method instead
toDateString()	Converts the date portion of a Date object into a readable string
toGMTString()	Deprecated. Use the toUTCString() method instead
toISOString()	Returns the date as a string, using the ISO standard
toJSON()	Returns the date as a string, formatted as a JSON date
toLocaleDateString()	Returns the date portion of a Date object as a string, using locale conventions
toLocaleTimeString()	Returns the time portion of a Date object as a string, using locale conventions
toLocaleString()	Converts a Date object to a string, using locale conventions
toString()	Converts a Date object to a string
toTimeString()	Converts the time portion of a Date object to a string
toUTCString()	Converts a Date object to a string, according to universal time
UTC()	Returns the number of milliseconds in a date string since midnight of Jan 1, 1970,
valueOf()	Returns the primitive value of a Date object

Window Object

The window object represents an open window in a browser. If a document contain frames (<frame> or <iframe>), the browser creates one window object for the HTML document, and one additional window object for each frame.

Note: There is no public standard that applies to the Window object, but all major browsers support it.

Property	Description
closed	Returns a Boolean value indicating whether a window has been closed or not
defaultStatus	Sets or returns the default text in the statusbar of a window
document	Returns the Document object for the window (See Document object)
frameElement	Returns the <iframe> element in which the current window is inserted
frames	Returns all <iframe> elements in the current window
history	Returns the History object for the window (See History object)
innerHeight	Returns the inner height of a window's content area
innerWidth	Returns the inner width of a window's content area
length	Returns the number of <iframe> elements in the current window
localStorage	Returns a reference to the local storage object used to store data. Stores data with no expiration date
location	Returns the Location object for the window (See Location object)

name	Sets or returns the name of a window
navigator	Returns the Navigator object for the window (See Navigator object)
opener	Returns a reference to the window that created the window
outerHeight	Returns the outer height of a window, including toolbars/scrollbars
outerWidth	Returns the outer width of a window, including toolbars/scrollbars
pageXOffset	Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window
pageYOffset	Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window
parent	Returns the parent window of the current window
screen	Returns the Screen object for the window (See Screen object)
screenLeft	Returns the horizontal coordinate of the window relative to the screen
screenTop	Returns the vertical coordinate of the window relative to the screen
screenX	Returns the horizontal coordinate of the window relative to the screen
screenY	Returns the vertical coordinate of the window relative to the screen
sessionStorage	Returns a reference to the local storage object used to store data. Stores data for one session (lost when the browser tab is closed)
scrollX	An alias of pageXOffset
scrollY	An alias of pageYOffset
self	Returns the current window
status	Sets or returns the text in the statusbar of a window
top	Returns the topmost browser window

Method	**Description**
alert()	Displays an alert box with a message and an OK button
atob()	Decodes a base-64 encoded string
blur()	Removes focus from the current window
btoa()	Encodes a string in base-64
clearInterval()	Clears a timer set with setInterval()
clearTimeout()	Clears a timer set with setTimeout()
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
focus()	Sets focus to the current window
getComputedStyle()	Gets the current computed CSS styles applied to an element
getSelection()	Returns a Selection object representing the range of text selected by the user
matchMedia()	Returns a MediaQueryList object representing the specified CSS media query string
moveBy()	Moves a window relative to its current position
moveTo()	Moves a window to the specified position
open()	Opens a new browser window
print()	Prints the content of the current window
prompt()	Displays a dialog box that prompts the visitor for input
resizeBy()	Resizes the window by the specified pixels
resizeTo()	Resizes the window to the specified width and height
scroll()	Deprecated. This method has been replaced by the scrollTo() method.
scrollBy()	Scrolls the document by the specified number of pixels
scrollTo()	Scrolls the document to the specified coordinates

setInterval()	Calls a function or evaluates an expression at specified intervals (in milliseconds)
setTimeout()	Calls a function or evaluates an expression after a specified number of milliseconds
stop()	Stops the window from loading

HTML DOM Nodes

In the HTML DOM (Document Object Model), everything is a node:

- The document itself is a document node
- All HTML elements are element nodes
- All HTML attributes are attribute nodes
- Text inside HTML elements are text nodes
- Comments are comment nodes

The Document Object

When an HTML document is loaded into a web browser, it becomes a document object. The document object is the root node of the HTML document and the "owner" of all other nodes:
(element nodes, text nodes, attribute nodes, and comment nodes). The document object provides properties and methods to access all node objects, from within JavaScript
The Document Object is supported in all major browsers.

Document Object Properties and Methods

Property / Method	**Description**
document.activeElement	Returns the currently focused element in the document
document.addEventListener()	Attaches an event handler to the document
document.adoptNode()	Adopts a node from another document
document.anchors	Returns a collection of all <a> elements in the document that have a name attribute
document.applets	Returns a collection of all <applet> elements in the document
document.baseURI	Returns the absolute base URI of a document
document.body	Sets or returns the document's body (the <body> element)
document.close()	Closes the output stream previously opened with document.open()
document.cookie	Returns all name/value pairs of cookies in the document
document.charset	Deprecated. Use document.characterSet instead. Returns the character encoding for the document
document.characterSet	Returns the character encoding for the document
document.createAttribute()	Creates an attribute node
document.createComment()	Creates a Comment node with the specified text
document.createDocumentFragment()	Creates an empty DocumentFragment node
document.createElement()	Creates an Element node
document.createTextNode()	Creates a Text node
document.doctype	Returns the Document Type Declaration associated with the document
document.documentElement	Returns the Document Element of the document (the <html> element)
document.documentMode	Returns the mode used by the browser to render the document
document.documentURI	Sets or returns the location of the document
document.domain	Returns the domain name of the server that loaded the document
document.domConfig	Obsolete. Returns the DOM configuration of the document
document.embeds	Returns a collection of all <embed> elements the document
document.forms	Returns a collection of all <form> elements in the document
document.getElementById()	Returns the element that has the ID attribute with the specified value

document.getElementsByClassName()	Returns a NodeList containing all elements with the specified class name
document.getElementsByName()	Returns a NodeList containing all elements with a specified name
document.getElementsByTagName()	Returns a NodeList containing all elements with the specified tag name
document.hasFocus()	Returns a Boolean value indicating whether the document has focus
document.head	Returns the <head> element of the document
document.images	Returns a collection of all <img> elements in the document
document.implementation	Returns the DOMImplementation object that handles this document
document.importNode()	Imports a node from another document
document.inputEncoding	Returns the encoding, character set, used for the document
document.lastModified	Returns the date and time the document was last modified
document.links	Returns a collection of all <a> and <area> elements in the document that have a href attribute
document.normalize()	Removes empty Text nodes, and joins adjacent nodes
document.normalizeDocument()	Removes empty Text nodes, and joins adjacent nodes
document.open()	Opens an HTML output stream to collect output from document.write()
document.querySelector()	Returns the first element that matches a specified CSS selector(s) in the document
document.querySelectorAll()	Returns a static NodeList containing all elements that matches a specified CSS selector(s) in the document
document.readyState	Returns the (loading) status of the document
document.referrer	Returns the URL of the document that loaded the current document
document.removeEventListener()	Removes an event handler from the document (that has been attached with the addEventListener() method)
document.renameNode()	Renames the specified node
document.scripts	Returns a collection of <script> elements in the document
document.strictErrorChecking	Sets or returns whether error-checking is enforced or not
document.title	Sets or returns the title of the document
document.URL	Returns the full URL of the HTML document
document.write()	Writes HTML expressions or JavaScript code to a document
document.writeln()	Same as write(), but adds a newline character after each statement

JavaScript Form Validation

JavaScript can be used to validate data in HTML forms before sending off the content to a server. Form data that typically are checked by a JavaScript could be:

- has the user left required fields empty?
- has the user entered a valid e-mail address?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Required Fields
The function below checks if a field has been left empty. If the field is blank, an alert box alerts a message, the function returns false, and the form will not be submitted:

```
function validateForm() {
   var x = document.forms["myForm"]["fname"].value;
   if (x == null || x == "") {
        alert("First name must be filled out");
        return false;
    }
}
```

The function above could be called when a form is submitted:

Example

```
<form name="myForm" action="demo_form.asp" onsubmit="return validateForm()"
method="post">
First name:
<input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

E-mail Validation

The function below checks if the content has the general syntax of an email.

This means that the input data must contain an @ sign and at least one dot (.). Also, the @ must not be the first character of the email address, and the last dot must be present after the @ sign, and minimum 2 characters before the end:

```
function validateForm() {
    var x = document.forms["myForm"]["email"].value;
    var atpos = x.indexOf("@");
    var dotpos = x.lastIndexOf(".");
    if (atpos< 1 || dotpos<atpos+2 || dotpos+2>=x.length) {
        alert("Not a valid e-mail address");
        return false;
    }
}
```

The function above could be called when a form is submitted:

Example

```
<form name="myForm" action="demo_form.asp" onsubmit="return validateForm();"
method="post">
Email:
 <input type="text" name="email">
<input type="submit" value="Submit">
</form>
```

6. Conclusion:

We have studied JavaScript scripting language. We used different window object, window object properties, window object methods and also different document object properties and document object methods to create Javascript window and document object. Validation form and JavaScript code was embedded into HTML document. The different fields of form can be validated using javascript.

7. Viva Questions:

- How do you validate registration form using Javascript?
- What is the need for validation?
- Where do you embed java script code in HTML documents?
- What are different attributes of java script?

8. References:

- Ralph Moseley, M.T. Savaliya, "Developing Web Applications", Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- "Web Technology Black Book", Dreamtech Press, First Edition, 978-7722-997
- www.larryullman.com › JavaScript
- https://docs.juspay.in/javascript/
- https://www.simplify.com/commerce/docs/tutorial/index
- www.cs.iusb.edu/thesis/SKhodali_thesis.pdf
- www.javatpoint.com/javascript-form-validation
- www.webcodegeeks.com › JavaScript
- www.forgetcode.com › JavaScript
- www.scriptiny.com/2012/12/register-or-signup-form-validation-using-javascript/

Web Technologies

Experiment No. : 4

To design animation using Javascript.

Experiment No. 4

1. **Aim:** To design animation using Javascript.

2. **Objectives:** From this experiment, the student will be able to

- Learn how to design animation in javascript.
- Understand its advantages and disadvantages
- Embed javascript code into dynamic HTML document using script tag

3. **Outcomes:** The learner will be able to

- Communicate effectively with range of audiences in form of animation.
- Use current skills to embed javascript into dynamic HTML document
- Formulate solution with the help of animation.

4. **Software Required :** Web Browser

5. **Theory:**

JavaScript can be used to move a number of DOM elements (<img />, <div> or any other HTML element) around the page according to some sort of pattern determined by a logical equation or function.
JavaScript animations can handle things that CSS can't.
JavaScript provides the following two functions to be frequently used in animation programs.

setTimeout(function, duration) − This function calls function after duration milliseconds from now.
setInterval(function, duration) − This function calls function after every duration milliseconds.
clearTimeout(setTimeout_variable) − This function calls clears any timer set by the setTimeout() functions.

JavaScript can also set a number of attributes of a DOM object including its position on the screen. You can set top and left attribute of an object to position it anywhere on the screen.

Styling the Elements
To make an animation possible, the animated element must be animated relative to a "parent container".
The container element should be created with style = "position: relative".
The animation element should be created with style = "position: absolute".

Example :
```
<html>
<head>
<style>
#train {
```

```
position: relative;
cursor: pointer; }
</style>
</head>
<body>
<img id="train" src="https://js.cx/clipart/train.gif">
<script>
train.onclick = function() {
let start = Date.now();
let timer = setInterval(function() {
let timePassed = Date.now() - start;
train.style.left = timePassed / 5 + 'px';
if (timePassed > 2000) clearInterval(timer);
}, 20);
}
</script>
</body>
</html>
```

Request animation frame

Let’s imagine we have several animations running simultaneously. If we run them separately, each one with its own setInterval(..., 20), then the browser would have to repaint much more often than every 20ms. Each setInterval triggers once per 20ms, but they are independent, so we have several independent runs within 20ms. These several independent redraws should be grouped together, to make it easier for the browser.

In other words, this:

```
setInterval(function() {
animate1(); animate2(); animate3(); }, 20)
```

…Is lighter than this:

```
setInterval(animate1, 20);
setInterval(animate2, 20);
setInterval(animate3, 20);
```

There’s a standard Animation timing that provides the function requestAnimationFrame. The window.requestAnimationFrame() method tells the browser that you wish to perform an animation and requests that the browser call a specified function to update an animation before the next repaint. The method takes as an argument a callback to be invoked before the repaint.

The syntax:

```
window. requestAnimationFrame(callback);
```

Parameters

callback

A parameter specifying a function to call when it's time to update your animation for the next repaint. The callback has one single argument, a DOMHighResTimeStamp, which indicates the current time (the time returned from performance.now()) for when requestAnimationFrame starts to fire callbacks.

Return value
A long integer value, the request id, that uniquely identifies the entry in the callback list. This is a non-zero value, but you may not make any other assumptions about its value. You can pass this value to window.cancelAnimationFrame() to cancel the refresh callback request.

Example

```
var start = null;
var element = document.getElementById('SomeElementYouWantToAnimate');
element.style.position = 'absolute';
function step(timestamp) {
if (!start) start = timestamp;
var progress = timestamp - start;
element.style.left = Math.min(progress / 10, 200) + 'px';
if (progress < 2000) {
window.requestAnimationFrame(step);
}
}
window.requestAnimationFrame(step);
```

6. Conclusion:

We have studied animation using JavaScript scripting language. We used different functions to create animation.

7. Viva Questions:

- Explain the different function used in creation of animation using javascript?
- What is the need for animation?

8. References:

- Ralph Moseley, M.T. Savaliya, “Developing Web Applications”, Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- “Web Technology Black Book”, Dreamtech Press, First Edition, 978-7722-997
- http://ebook-dl.com/book/3634
- www.webcodegeeks.com › JavaScript
- www.forgetcode.com › JavaScript
- https://javascript.info/js-animation#requestanimationframe

Web Technologies

Experiment No. : 5

To design animation using JQuery

Experiment No. 5

1. **Aim:** To design animation using JQuery.
2. **Objectives:** From this experiment, the student will be able to
   - Learn how to design animation in JQuery.
   - Understand its advantages and disadvantages

3. **Outcomes:** The learner will be able to
   - Communicate effectively with range of audiences in form of animation.
   - Use current skills to embed jquery into dynamic HTML document
   - Formulate solution with the help of animation.

4. **Software Required :** Web Browser
5. **Theory:**

JQuery is a lightweight, "write less, do more" and JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website. JQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code. JQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.
The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

jQuery Syntax:
The jQuery syntax is tailor made for **selecting** HTML elements and performing some **action** on the element(s). With jQuery you select (query) HTML elements and perform "actions" on them.
Basic syntax is: **$(*selector*).*action*()**

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

The Document Ready Event:
$(document).ready (function(){ *// jQuery methods go here...* });
This is to prevent any jQuery code from running before the document is finished loading (is ready).
It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head jQuery selectors are one of the most important parts of the jQuery library. jQuery selectors allow you to select and manipulate HTML element(s). jQuery selectors are used to "find" (or select) HTML elements based on their id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.
All selectors in jQuery start with the dollar sign and parentheses: $().

The element Selector:
The jQuery element selector selects elements based on the element name.
You can select all <p> elements on a page like this:
$("p")

Example
When a user clicks on a button, all <p> elements will be hidden:
Example
$(document).ready(function(){ $("button").click(function(){ $("p").hide(); }); });

The #id Selector:
The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.
An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.
To find an element with a specific id, write a hash character, followed by the id of the element:
$("#test")

Example
When a user clicks on a button, the element with id="test" will be hidden:
Example
$(document).ready(function(){ $("button").click(function(){ $("#test").hide(); }); });

The .class Selector:
The jQuery class selector finds elements with a specific class.
To find elements with a specific class, write a period character, followed by the name of the class:
$(".test")

Example
When a user clicks on a button, the elements with class="test" will be hidden:
Example
$(document).ready(function(){ $("button").click(function(){ $(".test").hide(); }); });

Functions In a Separate File:

If your website contains a lot of pages, and you want your jQuery functions to be easy to maintain, you can put your jQuery functions in a separate .js file. However, sometimes it is preferable to place them in a separate file, like this (use the src attribute to refer to the .js file):<head><script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script><script src="my_jquery_functions.js"></script></head>

Examples of jQuery Selectors Syntax	**Description**
$("*")	Selects all elements
$(this)	Selects the current HTML element
$("p.intro")	Selects all <p> elements with class="intro"
$("p:first")	Selects the first <p> element
$("ulli:first")	Selects the first <li> element of the first <ul>
$("ulli:first-child")	Selects the first <li> element of every <ul>
$("[href]")	Selects all elements with an href attribute
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"
$(":button")	Selects all <button> elements and <input> elements of type="button"
$("tr:even")	Selects all even <tr> elements
$("tr:odd")	Selects all odd <tr> elements

6. Conclusion:

We have studied and implemented animation using JQuery. We studied and implemented syntax of JQuery, document ready event, syntax of element selector, syntax of id selector and syntax of class selector.

7. Viva Questions:

- What is jQuery and how it is different from JavaScript?
- Is jQuery for client scripting or server scripting?

- Should we have multiple document.ready() function on the same page?
- Why do we use jQuery instead JavaScript?

8. References:

- https://www.tutorialrepublic.com/jquery-tutorial/jquery-animation-effects.php
- https://jqueryui.com/animate/
- http://api.jquery.com/animate/
- https://www.w3schools.com/jquery/eff_animate.asp
- https://www.w3schools.com/jquery/jquery_animate.asp
- https://jqueryui.com/animate/

Web Technologies

Experiment No. : 6

To design home page for TOI using Kompozer

Experiment No. 6

1. **Aim**: To design home page for TOI using Kompozer.

2. **Objectives:** From this experiment, the student will be able to
   - Learn basics of editor
   - Create web page for mobile shopping using editor tool
   - Understand the development of web pages using any web tool.

3. **Outcomes:** The learner will be able to

   - Understand the usage of web tool
   - Identify, formulate and design web page using web tool.
   - Recognize the need for learning web tool and be able to engage continuing professional development

4. **Software Required :** Internet Explorer

5. **Theory:**

KompoZer is a complete Web Authoring System which integrates web page development and web file management. KompoZer is basically the same product as Nvu but has been developed further removing some of the faults found in Nvu and improving the user interface.

It provides a web page editor which has a simple graphical (wysiwyg – what you see is what you get) interface. With KompoZer, newcomers will quickly and easily be able to produce new web pages.The output code is compliant to a high extent with the latest issues of the appropriate web language specifications and pages may be checked for validity directly from KompoZer using the official W3C validator.

KompoZer incorporates a Site Manager; this gives rapid access to the files on both local machines and remote servers. It can cater for several sites and switch rapidly between them. From within KompoZer pages and associated files may be uploaded to a remote server.

KompoZer supports the use of “Styles” through Cascading Style sheets (CSS) both embedded and external. It has an editor which generates CSS code conforming with CSS 2.1 specifications.

KompoZer is suitable for anyone wishing to have a modern, free of charge, program for developing small web sites and who would like to learn modern web design techniques such as the use of CSS.

Methodology

The approach taken by the guide attempts to satisfy the needs both of the comparative novice who has little knowledge of web design tools, and those making the transition from other tools and who may have considerable knowledge of the field.

Although the Guide is not intended for the complete beginner, concepts are introduced progressively and gradually added to. To avoid being too repetitive I have, in a number of places, referred to subjects yet to come. This applies, in particular to the use of styles in which KompoZer is strong. I suggest that these references are disregarded at first reading and returned to later. This will not prevent understanding later parts.

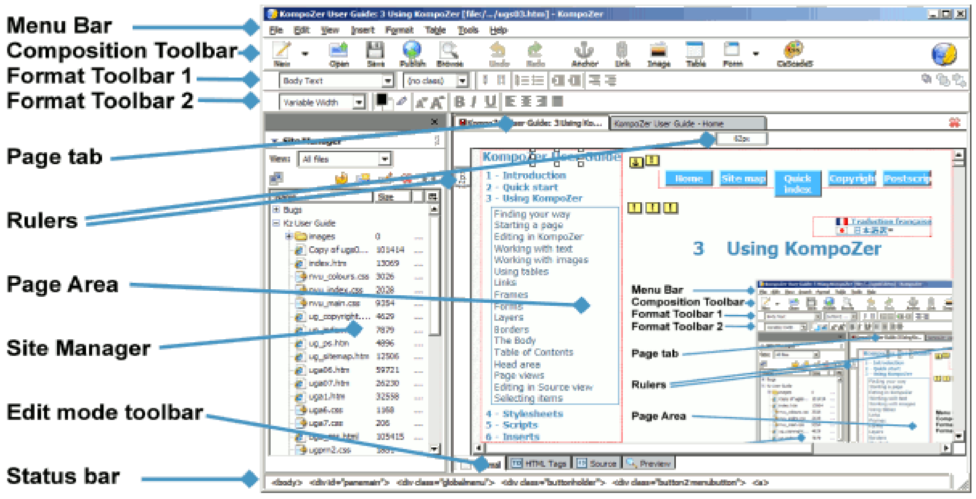
Navigation

Navigate around the ten sections of the Guide and the appendices using the menu pane on the left hand side. In each section the menu expands to show more detail within that section. Cross references throughout are hyperlinked.

The buttons at the top of each page give access to some supplementary aids and information. The 'Site map' contains a detailed table of contents of the whole guide while the 'Quick index' gives access by subject.

Open KompoZer

The main window opens. At the top are a number of toolbars. The topmost is the Menu Bar. This carries a number of items (File, Edit etc) used to make selections. The next is the ‘Composition Toolbar’ which carries a number of ‘Buttons’ labelled ‘New’, ‘Open’ etc.



To create a new page:

On the Composition toolbar Click the ‘New’ button.

To open an existing page

Assuming that the page is stored on your local disk in HTML format:

On the menu Bar click ‘File’ then ‘Open File’. Browse to the file and click ‘Open’.

Editing a web page

Your web page – blank or otherwise – is in the large pane in the centre right of the KompoZer application window. Many editing functions are very similar to those in a word processor. The top four toolbars on the KompoZer application window provide a number of editing functions – to see what any do hover the cursor over an item and a hint will appear.

Saving a Page - ***To save a page:***

On the Composition toolbar click ‘Save’.

If it was a new document a dialog window will ask you to enter a title for the page. This will appear in the tab at the top of the page display area. NB this is NOT the file name. Click ‘OK’; you will then be offered a normal save window which allows you to browse to a suitable location and name the file. The file extension offered will be HTML.

Browsing a page

To see how your page will look in your default browser on the Composition toolbar click 'Browse'.

Starting a Page:

A new page can be created either from a blank page or by opening any pre-existing page.

Working with text:

> *Text typed directly onto the KompoZer page defaults to appearing in the format for the 'body' element. HTML defines a small number of elements specifically for text and it is usually preferable to use these.*

Formatting text:

Text can be formatted in a number of ways using a format toolbar. The changes listed in the table can be applied.

Choose a font*
Choose text colour*   Choose background colour*
Choose highlight colour*
Make text smaller*   Make text larger*
Embolden*   Italicise*   Underline*
Format as a numbered list   Format as bulleted list
Align left   Align Centre   Align right   Justify
Indent text   Outdent text
Emphasise*   Strongly emphasise*

Numbered and Bulleted lists:

Kompozer can format a list of items giving each item a sequential number in any of several formats (HTML calls these ordered lists <ol>) or presenting them bulleted (unordered lists <ul>). By clicking on icon below.

Special Characacters:

To insert a Special characters: Insert > Characters and Symbols.

Link:

To insert a link a in webpage select a text and click on   and specify the address of the page to be linked.

Table: Similarly table can be added by clicking on following icon.

6. Conclusion:

Kompozer provides a GUI icons for probably all the basic HTML elements such as table tag, anchor tag, form tag etc. Thus we have understood the tools necessary for creating web pages. We were able to understand, identify, analyze and design the web page using software tool.

7. References:

- Ralph Moseley, M.T. Savaliya, "Developing Web Applications", Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- "Web Technology Black Book", Dreamtech Press, First Edition, 978-7722-997
- https://sourceforge.net/projects/quanta/

Web Technologies

Experiment No. : 7

To design online examination form using Kompozer

Experiment No. 7

1. **Aim**: To design online-examination form using Kompozer.

2. **Objectives:** From this experiment, the student will be able to
   - Learn basics of editor
   - Create web page for mobile shopping using editor tool
   - Understand the development of web pages using any web tool.

3. **Outcomes:** The learner will be able to
   - Understand the usage of web tool
   - Identify, formulate and design web page using web tool.
   - Recognize the need for learning web tool and be able to engage continuing professional development

4. **Software Required :** Internet Explorer

5. **Theory:**

Forms

Forms provide a mechanism by which a visitor to a site may send data to as sever for processing. This may be as simple as a box to write a message for e-mailing or as complex as looking up items from a catalogue and sending an order to a supplier. Forms collect data typed in or collected from check boxes or lists and pass them to the server. It follows that forms can be used only in association with compatible software running on the server.

Forms may be placed on standard web pages and act as block level elements. In normal view KompoZer shows forms surrounded by a dotted cyan box. Forms may contain other standard block level elements (paragraphs, headings …) as well as several specific elements known as 'form controls' which are designed for data collection. Since these form controls are essentially inline elements they must be laid out inside block level elements (typically paragraphs though divs would be equally suitable).

Each item of data sent to the server is tagged with information about which control has sent it. This is done by naming each control. Thus designers must give a unique name to each control to be used.

The data collected by a form will be sent to a URL specified in an element ‘action’ which KompoZer will add to the form. This URL is often on the server that hosts the web page but does not have to be. The data will be processed using one of two methods known as ‘GET’ or ‘POST’ which also need to be specified. These details will be available from the provider of the associated software.

Steps to set up a form

1. Click the form button.
2. In the Form properties window give the form a name of your choosing
3. Complete the Action box with the correct URL and select the appropriate method
4. ‘Encoding’ and ‘Target Frame’ will frequently not be required but, if they are, select ‘More Properties’ and complete the boxes
5. Click OK
6. On the form place any headings, paragraphs and images ensuring that there is a placeholder for any controls needed. (If blank placeholders are needed it is probably sensible to put some dummy text in now and delete it later.)
7. Where controls are needed click the corresponding placeholder and using the drop down box beside the Form button select the required control
8. Give each control a unique name
9. Each control has specific information which needs to be entered. Enter it into the box in the window which appears

6. Conclusion:

Kompozer provides easy way to design a form. Thus we have understood the tools necessary for creating web pages. We were able to understand, identify, analyze and design the web page using software tool.

7. References:

- Ralph Moseley, M.T. Savaliya, “Developing Web Applications”, Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- “Web Technology Black Book”, Dreamtech Press, First Edition, 978-7722-997
- https://sourceforge.net/directory/?q=kompozer+templates

Web Technologies

Experiment No. : 8

To design home page for online mobile shopping using Kompozer

Experiment No. 8

1. **Aim**: To design home page for online mobile shopping using Kompozer.

2. **Objectives:** From this experiment, the student will be able to
    - Learn basics of editor
    - Create web page for mobile shopping using editor tool
    - Understand the development of web pages using any web tool.

3. **Outcomes:** The learner will be able to
    - Understand the usage of web tool
    - Identify, formulate and design web page using web tool.
    - Recognize the need for learning web tool and be able to engage continuing professional development

4. **Software Required :** Internet Explorer

5. **Theory:**

Introduction to style

The use of styles is already well established in word processing and desk top publishing. In web site design it assumes an even more important role and can control almost every aspect of presentation.

Power of styles

Styles specify how particular **elements** on a page appear on the screen, in print or whatever. This guide limits itself to on-screen considerations. By ‘elements’ we mean parts of the page structure, typically headings and paragraphs, but also stretching to many others including tables, bulleted and numbered lists etc. In fact most HTML ‘Tags’ may be specified though the same style may be applied to several.

Style may typically define such aspects of presentation as the font face, size and variant, the font colour, the background colour, whether an element is to be aligned right, centre or left, whether spaced away from other elements, surrounded by a border applied.

Using style

Inline style

Within a page styles may be used in three ways. These ways can be mixed and matched as you wish. The first, easiest and crudest is to define a style for the nonce, there and then. Such a style is listed in the code (in Source view) with the tag to which it applies (using the structure "`style = ….`" if you look at the code).

Don't worry KompoZer hides all this from you.

If you have another item with the same style this code must be repeated. This bloats the page. This is known as an 'inline style'. For the most part KompoZer users do not have to be concerned with this method although KompoZer will sometimes use it without you knowing.

Internal style

The second way to use a style is to embed a list of style definitions within the HEAD section of a page. (These definitions are referred to as 'style rules' or just 'rules')

These rules may be of one of two sorts. The first sort applies to all elements of a particular type (e.g. p, h1, table) the second is the class as discussed above.

External style – Linked stylesheets

While the first two methods are valid and have their uses, the third is the preferred method specifically because it is economical, reuses the same styles for many pages and helps in achieving consistency of appearance right across a complete web site. The method uses an external style sheet which is 'Linked' to a page, or to several pages (though each page must include the linking information for itself).

An external style sheet contains the same list of rules which would otherwise have been included in the internal list referred to above. (It is actually a simple text document such as you could construct using a text editor like Windows® Notepad.) The file is usually located in the same folder (directory) as the page to which it is linked (though it can be elsewhere) and has the extension 'css'.

How does a page know to use this style sheet? A line of code which, KompoZer will insert for you in the head section of your page, will see to this.

The code looks like< `link rel="stylesheet" type="text/css" href="mystylesheet.css>`

CREATING INTERNAL STYLES USING NVU

Styles are created and edited using the built in Style sheet editor called Cascades. Cascades has two modes of operation 'Beginner Mode' and 'Expert Mode'. When it opens it opens in expert mode, should you wish to switch to Beginner Mode clear the check box. The difference is that in Beginner mode External style sheets cannot be created.

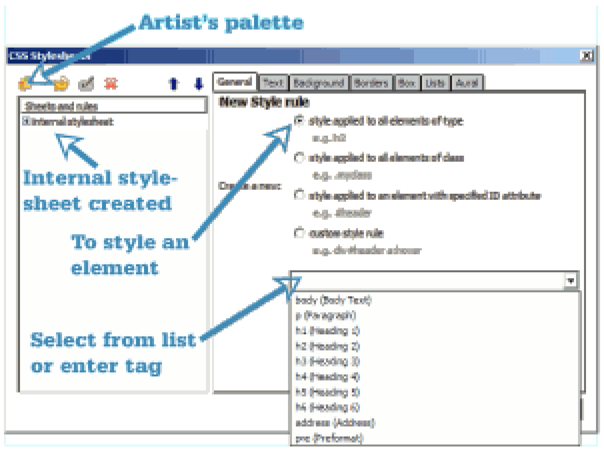
When internal styles are created Nvu will include the the rules for these in the 'head' area of the document.

To start creating styles:

1. Open Tools> CSS Editor.
2. Click 'Style elt', 'Create Stylesheet'.
3. Click RULE.
4. The General tab opens and offers the options 'Named Style' or 'Style applied to all elements of type' or 'Style applied to all elements matching the following selector'.

To define how you want elements to look:

1. Click the tag for the element in the left hand pane. The General tab should show, listing the 'Selector' (at present the element tag) and any style 'Declarations' (initially blank).
2. Now select in turn as required the tabs for 'Text', 'Background', 'Border' etc and specify exactly how you wish that element to appear. The next section amplifies some details of how to do this.
3. Return to the general tab to see the full declarations that you have set for the Selector.



To create a style rule for an element

1. Click the Cascades button on the Composition toolbar. The CSS Stylesheets window opens.
2. Click on the artist's palette button . In the 'Sheets and rules' pane you will see an internal stylesheet has been created for you.
3. To create a rule click 'Style applied to all elements of type'

4. Beside the blank box click the drop down arrow. You will see listed a number of common elements. To create a style for one of these click it alternatively enter the tag for any other element.
5. Click 'Create Style rule'
6. You are now presented with a window headed 'Selector' followed by the tag for the element. The window actually lists the style declarations for that element, but of course that is now blank.

To define how you want elements to look:

I. Select in turn as required the tabs for 'Text', 'Background', 'Border' etc and specify exactly how you wish that element to appear. The next section amplifies some details of how to do this.
II. Return to the general tab to see the full declarations that you have set for the Selector
III. If you click the 'General' tab you will see all the declarations for the rule. You can edit these here but it is better to leave the job to CaScadeS because if you make any errors the declaration will be deleted
IV. When you are satisfied click 'OK'

Once created, rules may be edited by reopening Cascades, Click and expand the stylesheet and click the rule involved.

In the 'Sheets and rules' pane on the left you should see a structure showing each tag as it is defined.

First, do note you are not obliged to specify any particular properties. You specify as much or as little as you wish. It's just that the browser will revert to defaults if you don't. There are fairly complicated rules for what it uses, and browsers don't always follow the rules, however chances are that you'll be happy with whatever it does – if you're not – change it!

You may like to try setting declarations for the 'body' and apply it to all elements. In most cases declarations specified here with trickle down to everything else (they are 'inherited') unless you specify it explicitly elsewhere. Generally this works but can generate surprises.

6. Conclusion:

Using Kompozer we can design attractive web pages by applying styling and animation. Thus we have understood the tools necessary for creating web pages. We were able to understand, identify, analyze and design the web page using software tool.

7. References:

- Ralph Moseley, M.T. Savaliya, "Developing Web Applications", Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- "Web Technology Black Book", Dreamtech Press, First Edition, 978-7722-997
- www.aptana.com/

Web Technologies

Experiment No. : 9

Design HTML form to accept the two numbers N1 and N2. Display prime numbers between N1 and N2 using PHP

Experiment No. 9

1. **Aim:** Design HTML form to accept the two numbers N1 and N2. Display prime numbers between N1 and N2 using PHP.

2. **Objectives:** From this experiment, the student will be able to
   - Learn basics of PHP
   - Understand the need of server side scripting
   - Embed PHP code into HTML document.

3. **Outcomes:** The learner will be able to
   - Communicate effectively with range of audiences in form of dialog box
   - Use current skills to embed PHP into dynamic HTML document
   - Recognize the need for PHP and be able to engage continuing professional development

4. **Software Required :** Internet Explorer, MYSQL database, WAMP/XAMPP server

5. **Theory:**

PHP is a scripting language originally designed for producing dynamic web pages. It has evolved to include a command line interface capability and can be used in standalone graphical applications. PHP is a widely-used general-purpose scripting language that is especially suited for web development and can be embedded into HTML. It generally runs on a web server, which is configured to take PHP code as input and create web page content as output. It can be deployed on most web servers and on almost every operating system and platform free of charge.

- PHP stands for PHP: Hypertext Preprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use
- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML

- PHP files have a file extension of ".php", ".php3", or ".phtml"
- PHP code is executed on the server, and the plain HTML result is sent to the browser.

Basic PHP Syntax

- A PHP scripting block always starts with <?php and ends with ?>. A PHP scripting block can be placed anywhere in the document.
- On servers with shorthand support enabled you can start a scripting block with <? and end with ?>.
- For maximum compatibility, we recommend that you use the standard form (<?php) rather than the shorthand form.

A PHP script starts with <?php and ends with ?>. This script can be placed anywhere in the document. PHP usually has HTML tags, along with some PHP coding.

PHP Syntax:

```
<?php
PHP Code Here
?>
```

Note: The PHP file should be saved using the *.php extension, not *.html, if saved in *.html extension then the PHP code will not be executed.

For Example:
To display an message "Good Morning" following code is executed.

```
<html>
<body>
   <?php
echo "Good Morning";
//This is a comment
# This is a comment
/*    This is
a comment
block    */
?>
</body>
</html>
```

Data Types in PHP include Boolean, Integer, Double, String, Array, Object, Resource and NULL

6. Conclusion:

PHP is a scripting language originally designed for producing dynamic web pages. In the program two numbers were read from user and the prime numbers between them is displayed.

7. Viva Questions:

- What is PHP?
- What is need of scripting language?
- What is dynamic web page?

8. References:

- Ralph Moseley, M.T. Savaliya, “Developing Web Applications”, Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- “Web Technology Black Book”, Dreamtech Press, First Edition, 978-7722-997
- www.w3schools.com/php/php_form_complete.asp
- 3.www.html-form-guide.com/php-form/php-registration-form.html

Web Technologies

Experiment No. : 10

Design a login form to add username, id, password into database & validate it (use PHP)

Experiment No. 10

1. **Aim:** Design a login form to add username, id, password into database & validate it (use PHP)
2. **Objectives:** From this experiment, the student will be able to
   - Learn basics of dynamic programming
   - Understand the need of server side scripting
   - Embed PHP code into HTML document.

3. **Outcomes:** The learner will be able to

   - Communicate effectively with range of audiences in form of dialog box
   - Use current skills to embed PHP into dynamic HTML document
   - Recognize the need for PHP and be able to engage continuing professional development

4. **Software Required :** Internet Explorer, MYSQL database, WAMP/XAMPP server

5. **Theory:**

Design a form with Name, E-mail, Website, Comment, Gender fields.

The validation rules for the form above are as follows:

Field	**Validation Rules**
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one.

First we will look at the plain HTML code for the form:

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male

The Form Element

The HTML code of the form looks like this:

<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">

When the form is submitted, the form data is sent with method="post".

The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script. So, the $_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as he form.
The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with &lt; and &gt;. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

Validate Form Data With PHP
The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.
When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:
<script>location.href('http://www.hacked.com')</script>
- this would not be executed, because it would be saved as HTML escaped code, like this:
&lt;script&gt;location.href('http://www.hacked.com')&lt;/script&gt;
The code is now safe to be displayed on a page or inside an e-mail.
We will also do two more things when the user submits the form:
Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
Remove backslashes (\) from the user input data (with the PHP stripslashes() function)
The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).
We will name the function test_input().
Now, we can check each $_POST variable with the test_input() function, and the script look like this:

Example

```
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}
function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

Notice that at the start of the script, we check whether the form has been submitted using $_SERVER["REQUEST_METHOD"]. If the REQUEST_METHOD is POST, then the form has been submitted - and it should be validated. If it has not been submitted, skip the validation and display a blank form.

However, in the example above, all input fields are optional. The script works fine even if the user do not enter any data. The next step is to make input fields required and create error messages if needed.

6. Conclusion:

We analyzed different text fields, radio buttons, form element for creating login page using PHP, and the validation was applied on its username, id and password. These credentials were validated with the database contents.

7. Viva Questions:

- What is the $_SERVER["PHP_SELF"] variable?
- What is the lifetime of data stored through localstorage?
- Which is the function used to retrieve a value?

8. References:

- Ralph Moseley, M.T. Savaliya, “Developing Web Applications”, Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- Web Technology Black Book”, Dreamtech Press, First Edition, 978-7722-997
- www.tutorialspoint.com/php/php_login_example.htm
- www.sourcecodester.com/tutorials/php/4341/how-create-login-page-phpmysql.html

Web Technologies

Experiment No. : 11

Database operations using PHP and MySQL database connectivity

Experiment No. 11

1. **Aim:** Design course registration form and perform various database operations using PHP and MySQL database connectivity

2. **Objectives:** From this experiment, the student will be able to
   - Learn basics of dynamic programming
   - Understand the need of server side scripting
   - Understand the commands to connect to database.

3. **Outcomes:** The learner will be able to

   - Communicate effectively with range of audiences in form of dialog box
   - Use current skills to embed PHP into dynamic HTML document
   - Recognize the need for PHP and be able to engage continuing professional development

4. **Software Required :** Internet Explorer, MYSQL database, WAMP/XAMPP server

5. **Theory:**

PHP MySQL Databse connectivity

For any database related programming you just keep in mind the following steps in order.

1. Create a connection.
2. Select a database.
3. Execute Queries.
4. Get The Results.
5. Close the connection. Not necessary, because the connection get closed, when the web page is sent from the server.
   1. First, we will create a database according to the following details in MySQL.
      Database name: AddressBook
      Table name: Addresses
      Table fields: ID, Name, DOB, HouseNumber, Street, City, Country, Telephone, Fax, Email, Remarks.

   2. Well, you can just use the following SQL script for creating the database and the table.

```
CREATE DATABASE AddressBook;
USE AddressBook;
CREATE TABLE Addresses (
   ID SMALLINT NOT NULL AUTO_INCREMENT,
   Name VARCHAR(60) NOT NULL,
   DOB DATE NOT NULL,
   HouseNumber VARCHAR(5) NOT NULL,
   Street VARCHAR(30) NOT NULL,
   City VARCHAR(15) NOT NULL,
   Country VARCHAR(30) NOT NULL,
   Telephone VARCHAR(15) NOT NULL,
   Fax VARCHAR(15) NOT NULL,
   Email VARCHAR(30) NOT NULL,
   Remarks TEXT,
   PRIMARY KEY(ID)
);
INSERT INTO Addresses VALUES ('3','Tom','1966-05-03','No 31','Paradise Street,','Paradise
City,','Haven.','666-666','999-999','tom@paradise.haven','Good Old Tommy');
```

In php we have to write the following code for database connectivity
First, Lets make a connection

```
$db=mysql_connect("localhost","root");

/*Parameters are: mysql_connect([server],[username],[password]) and will return a link identifier on
success otherwise a FALSE. i.e. you can use $db as the connection when you do the subsequence steps. */

/*Now, Lets Select the AddressBook database. */
mysql_select_db("AddressBook",$db);

/* Parameters are: mysql_select_db(database name,[link identifier]) and will return TRUE on success and
FALSE otherwise. */

/* Lets execute a query. */
$results=mysql_query("SELECT * FROM Addresses",$db);
```

6. Program:

```
   <?php
$serverName = "SQL Server";
$uid = "sqlusername";
$pwd = "sqlpassword";
$databaseName = "DBName";
$connectionInfo = array( "UID"=>$uid,
                         "PWD"=>$pwd,
                         "Database"=>$databaseName);
/* Connect using SQL Server Authentication. */
$conn = sqlsrv_connect( $serverName, $connectionInfo);
$tsql = "SELECT id, FirstName, LastName, Email FROM tblContact";
```

```
/* Execute the query. */
$stmt = sqlsrv_query( $conn, $tsql);
if ( $stmt )
{
    echo "Statement executed.<br>\n";
}
else
{
    echo "Error in statement execution.\n";
    die( print_r( sqlsrv_errors(), true));
}
/* Iterate through the result set printing a row of data upon each iteration.*/
while( $row = sqlsrv_fetch_array( $stmt, SQLSRV_FETCH_NUMERIC))
{
    echo "Col1: ".$row[0]."\n";
    echo "Col2: ".$row[1]."\n";
    echo "Col3: ".$row[2]."<br>\n";
    echo "-----------------<br>\n";
}
/* Free statement and connection resources. */
sqlsrv_free_stmt( $stmt);
sqlsrv_close( $conn);
?>
```

7. Conclusion:

We have studied different steps for creating database connectivity using MySql and PHP for various applications. We have created database connectivity using PHP and performed various operations on database.

8. Viva Questions:

- What is the need for connecting to database?
- What is the command for connection?

9. References:

- Ralph Moseley, M.T. Savaliya, “Developing Web Applications”, Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- Web Technology Black Book”, Dreamtech Press, First Edition, 978-7722-997
- www.w3schools.com/php/php_mysql_insert.asp
- www.tutorialrepublic.com/php-tutorial/php-mysql-insert-query.php
- people.cis.ksu.edu/~hankley/d764/tut06/GopisettyPHP.html

Web Technologies

Experiment No. : 12

To design XML document using XML schema for representing your semester marksheet using PHP

Experiment No. 12

1. **Aim**: To design XML document using XML schema for representing your semester marksheet using PHP.

2. **Objectives:** From this experiment, the student will be able to
   - Learn basics of XML
   - Create XML file to store information
   - Understand the formatting of stored data.

3. **Outcomes:** The learner will be able to
   - Understand the fundamentals of XML
   - Analyse local and global impact of computing on organization.
   - Recognize the need for XML and be able to engage continuing professional development

4. **Software Required :** Internet Explorer

5. **Theory:**

XML:

- XML stands for EXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to carry data, not to display data
- XML tags are not predefined. You must define your own tags
- XML is designed to be self-descriptive
- XML is a W3C Recommendation
- All XML Elements Must Have a Closing Tag

In HTML, you will often see elements that don't have a closing tag:

```
<p>This is a paragraph
<p>This is another paragraph
```

In XML, it is illegal to omit the closing tag. All elements **must** have a closing tag:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

XML Tags are Case Sensitive:
XML elements are defined using XML tags.

XML tags are case sensitive. With XML, the tag <Letter> is different from the tag <letter>.

Opening and closing tags must be written with the same case:

```
<Message>This is incorrect</message>
<message>This is correct</message>
```

XML Elements must be properly nested:

In HTML, you will often see improperly nested elements:

```
<b><i>This text is bold and italic</b></i>
```

In XML, all elements **must** be properly nested within each other:

```
<b><i>This text is bold and italic</i></b>
```

In the example above, "Properly nested" simply means that since the <i> element is opened inside the <b> element, it must be closed inside the <b> element.

XML Documents Must Have a Root Element:

XML documents must contain one element that is the **parent** of all other elements. This element is called the **root** element.

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

XML Attribute Values must be quoted:

XML elements can have attributes in name/value pairs just like in HTML.

In XML the attribute value must always be quoted. Study the two XML documents below. The first one is incorrect, the second is correct:

```
<note date=12/11/2007>
<to>Tove</to>
<from>Jani</from>
</note>

<note date="12/11/2007">
<to>Tove</to>
<from>Jani</from>
</note>
```

The error in the first document is that the date attribute in the note element is not quoted.

Entity References:

Some characters have a special meaning in XML.

If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

This will generate an XML error:

```
<message>if salary < 1000 then</message>
```

To avoid this error, replace the "<" character with an **entity reference**:

```
<message>if salary &lt; 1000 then</message>
```

There are 5 predefined entity references in XML:

&lt;	<	less than

&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	Apostrophe
&quot;	"	quotation mark

Comments in XML:
The syntax for writing comments in XML is similar to that of HTML.
<!-- This is a comment -->

With XML, White Space is preserved:

HTML reduces multiple white space characters to a single white space:

HTML:	Hello my name is Tove
Output:	Hello my name is Tove

With XML, the white space in your document is not truncated.

Difference between XML & HTML

HTML	XML
HTML is acronym for HyperText Markup Language	XML is acronym for eXtensible Markup Language
HTML was designed to display data with focus on how data looks	XML was designed to be a software and hardware independent tool used to transport and store data, with focus on what data is.
HTML is a markup language	XML provides a framework for defining markup languages.
HTML is a presentation language	XML is neither a programming language nor a presentation language.
HTML is used for designing a web-page to be rendered on the client side	XML is used basically to transport data between the application and the database.
HTML has it own predefined tags	XML is flexible is that it has custom tags can be defined and the tags are invented by the author of the XML document.
HTML is static as it displays data	XML is dynamic as it is ud=sed for storage of data.

HTML is not strict if the user does not use the closing tags.	XML makes it mandatory for the user the close each tag that has been used.

6. Conclusion:

We have learnt XML language. In this we have analyzed XML elements, XML documents, XML tags, XML attributes, XML commands. Basic difference between of HTML and XML can be understood to write XML schema.

7. Viva Questions:

- What delimits the end of the portion of the document?
- Which component in XML is modelled as a tree?

8. References:

- Ralph Moseley, M.T. Savaliya, "Developing Web Applications", Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- "Web Technology Black Book", Dreamtech Press, First Edition, 978-7722-997
- www.w3schools.com/xml/schema_example.asp
- www.xmlmaster.org/en/article/d01/c04/
- examples.oreilly.com/9780596002527/

Web Technologies

Experiment No. : 13

To design DTD for representing your semester marks sheet

Experiment No. 13

1. **Aim:** To design DTD for representing your semester marksheet.

2. **Objectives:** From this experiment, the student will be able to
   - Learn basics of DTD
   - Create DTD to represent marksheet
   - Embed DTD in XML document

3. **Outcomes:** The learner will be able to
   - Understand the fundamentals of DTD
   - Analyse local and global impact of computing on organization.
   - Recognize the need for DTD and be able to engage continuing professional development

4. **Software Required :** Internet Explorer

5. **Theory:**

DTD: A document type definition (DTD) is a set of markup declarations that define a document type for a markup language. A Document Type Definition (DTD) defines the legal building blocks of an XML document. It defines the document structure with a list of legal elements and attributes. A DTD can be declared inline inside an XML document, or as an external reference.

All XML documents (and HTML documents) are made up by the following building blocks:

- Elements
- Attributes
- Entities
- PCDATA
- CDATA

E.g. :-
```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```
The DTD above is interpreted like this:

- !DOCTYPE note defines that the root element of the document is note
- !ELEMENT note defines that the note element contains four elements: "to, from, heading, body"
- !ELEMENT to defines the to element to be of type "#PCDATA"
- !ELEMENT from defines the from element to be of type "#PCDATA"
- !ELEMENT heading defines the heading element to be of type "#PCDATA"
- !ELEMENT body defines the body element to be of type "#PCDATA"

#PCDATA means parse-able text data.
#CDATA means character data. (CDATA is text that will NOT be parsed by a parser.)

6. Conclusion:

The XML of user application will be developed with the help of DTD as this only will help us to define different elements and attributes of the user applications

7. Viva Questions:

- What is DTD?
- What does CDATA stands for?

8. References:

- Ralph Moseley, M.T. Savaliya, “Developing Web Applications”, Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- “Web Technology Black Book”, Dreamtech Press, First Edition, 978-7722-997
- www.w3schools.com/xml/xml_dtd.asp
- www.w3schools.com/xml/xml_dtd_examples.asp
- www.xmlmaster.org/en/article/d01/c03/

Web Technologies

Experiment No. : 14

To design XML schema and DTD for railway reservation system

Experiment No. 14

1. **Aim:** To design XML schema and DTD for railway reservation system.

2. **Objectives:** From this experiment, the student will be able to
   - Learn basics of XSL
   - Build simple XML file to store information
   - Understand how to manipulate and refer to stored data

3. **Outcomes:** The learner will be able to
   - Understand the fundamentals of XML
   - Analyse local and global impact of computing on organization.
   - Recognize the need for XML and be able to engage continuing professional development

4. **Software Required :** Internet Explorer

5. **Theory:**

The user application developed with the help XML and DTD are not good at designing part. For the designing of the XML file we go ahead with the XSLT.

XSL: XSL stands for EXtensible Stylesheet Language, and is a style sheet language for XML documents.

XSLT stands for XSL Transformations. In this tutorial you will learn how to use XSLT to transform XML documents into other formats, like XHTML.

XPath : An expression language used by XSLT (and many other languages) to access or refer to parts of an XML document.

XML Schema is an XML-based alternative to DTD:

Example:

```
<xs:element name="reminder">
<xs:complexType>
  <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

The Schema above is interpreted like this:

- <xs:element name="note"> defines the element called "reminder"
- <xs:complexType> the "reminder" element is a complex type
- <xs:sequence> the complex type is a sequence of elements
- <xs:element name="to" type="xs:string"> the element "to" is of type string (text)
- <xs:element name="from" type="xs:string"> the element "from" is of type string
- <xs:element name="heading" type="xs:string"> the element "heading" is of type string
- <xs:element name="body" type="xs:string"> the element "body" is of type string

Everything is wrapped in "Well Formed" XML.

6. Conclusion:

The designing of the XML of user application will be developed with the help of DTD is done with XSL transformation. It helps to dynamically change the data of the static HTML page.

7. Viva Questions:

- What is XSL?
- What is need of XSL?
- What are built-in data types of XML schema?

8. References:

- Ralph Moseley, M.T. Savaliya, “Developing Web Applications”, Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- “Web Technology Black Book”, Dreamtech Press, First Edition, 978-7722-997
- www.nmt.edu/tcc/help/pubs/dtd/dtd.pdf

Web Technologies

Experiment No. : 15

Mini Project

Experiment No. 15

1. **Aim:** Design a website which includes all required validations, styles and database connectivity

2. **Objectives:** From this experiment, the student will be able to
   - Learn basics of web development
   - Understand concepts of HTML, CSS, PHP, XML, database connectivity

3. **Outcomes:** The learner will be able to
   - Communicate effectively with range of audiences
   - Use current skills to acquire knowledge of HTML, CSS, PHP, XML, database connectivity
4. **Software Required :** Internet Explorer, MYSQL database, WAMP/XAMPP server

5. **Conclusion:** Thus the complete package of website was developed, by applying current skill and technologies. The contemporary issues were considered.

6. **References:**

- Ralph Moseley, M.T. Savaliya, "Developing Web Applications", Wilely India, Second Edition, ISBN: 978-81-265-3867-6
- Web Technology Black Book", Dreamtech Press, First Edition, 978-7722-997