UNIT – II
JAVASCRIPT

INTRODUCTION TO JAVASCRIPT:

Static web pages are useful and can be informative. A number of technologies have been developed that enable the creation of web applications rather than static web pages. The java programming language is probably the best known such technology. Few programming languages other than java have been adapted for use in client-side web applications. One such language that is used in programming client-side web applications is javascript.

Javascript originates from a language called Live Script and was developed by Sun microsystems and Netscape navigator. Scripts are small pieces of code which accomplish a single relatively simple task. Javascript is a scripting language that is used for the development of Client-side-in-browser applications. Javascript is a simple script based language which is only suitable for fairly simple tasks. Javascript is a language that is best suited to tasks that run for a short time. Most of the developers experience problems when they try to build web pages which have embedded javascript.

Important things about Javascript:

1. **Javascript is embedded into HTML:**

   Javascript code is usually embedded into HTML code and is executed within the HTML document. Javascript has no user interface and it relies on HTML to provide a means of interaction with the users. Most of the javascript objects have HTML tags and provide event-driven code to execute it.

2. **Javascript is browser dependent:**

   Javascript depends on the web browser to support it. If the browser does not support it javascript will be ignored. Javascript was given support from I.E 3.0 & N.N 2.0 onwards.

3. **Javascript is an interpreted language:**

   Javascript is interpreted at runtime by the browser before it is executed. It is not compiled into a separate program like .exe but remains part of HTML file.

4. **Javascript is a loosely typed language:**

   Javascript is very flexible when compared to java. There is no need to specify the data type of a variable while declaring it. We can declare variables whenever it is necessary i.e., variables can be declared explicitly.

5. **Javascript is an object-based language:**

   Javascript is an object-based language that means that we can work with objects that encapsulate data and behaviour. However, javascript object model is instance based and there is no inheritance.

6. **Javascript is Event-Driven:**

   HTML objects such as buttons are enhanced to support event handlers. We can specify functionality.

7. **Javascript is not Java:**

   Java is object oriented language, whereas javascript is object-based, interpreted, loosely-typed language meant for creating scripts.

8. **Javascript is multi functional:**

   Javascript can be used to:

   - Enhance HTML pages.
   - Develop client-side applications.
   - Build to a certain extent client/server web applications.
   - Create extensions to a web server.
   - Provide database connectivity without using CGI.

9. **Javascript language spans context:**

   Javascript can be used in server side Netscape live wire pro environment and microsoft's Active X server framework. It is not just a client side scripting tool.

Benefits of Javascript:

- It is widely supported in web browsers.
- It gives easy access to the document objects and can manipulate most of them.
- Javascript can give interesting animations without long download times associated with any multimedia data types.
- Web surfers do not need any special plug-in to use your scripts.
- Javascript is relatively secure i.e., javascript can neither read from a local hard drive nor write to it and we cannot get a virus directly from javascript.

Problems with Javascript:

- If our script does not work then our page is useless.
- Most of the web surfers disable javascript support in their browsers because of the problems of broken scripts.
- Scripts can run slowly and complex scripts can take a long time to start up.

Javascript Basics:

Javascript programs contain variables, objects and functions. The key points that we need to apply in all scripts are:

- Each line of code is terminated by semicolon.
- Blocks of code must be surrounded by a pair of curly braces. A block of code is a set of instructions that are to be executed together as a unit.
- Functions have parameters which are passed inside parentheses.
- Variables are declared using keyword **"var"**.
- Scripts neither require a main function nor an exit condition. Execution of scripts starts with the first line of code and runs until there is no more code.

Javascript and HTML page:

Javascript need to be included in a HTML page. We cannot execute these scripts from a command line as the interpreter is part of the browser. The script is included in the web page and run by the browser, usually as soon as the page has been loaded. The browser is able to debug the script and can display errors.

Javascript is embedded in HTML using <SCRIPT> element. Usually <SCRIPT> is included in <HEAD> element. A simple javascript code:

```
<html>
        <head>
                <title>Javascript</title>
                <script language="javascript">
                        document.write("Welcome to Javascript");
                </script>
        </head>
        <body>
```

```
                <h1>Javascript First Page</h1>
        </body>
</html>
```

If javascript is written as a separate file, then it must be saved with .js extension. External javascript file is included into HTML by using src attribute of <SCRIPT> element. Then also the script is evaluated when page loads and before any script action takes place.

In case of functions, all javascript statements within the function block are interpreted but executed only when the function is called from a javascript event. Javascript statements which are not in a function block are executed after document loads into browser.

JavaScript alert()

- The alert() method in JavaScript is used to display a virtual alert box. It is mostly used to give a warning message to the users. It displays an alert dialog box that consists of some specified message (which is optional) and an OK button. The syntax is:

alert("message");

JavaScript prompt()

The prompt() method in JavaScript is used to display a prompt box that prompts the user for the input. It is generally used to take the input from the user before entering the page. When the prompt box pops up, we have to click "OK" or "Cancel" to proceed. Syntax is:

prompt("message", "default")

Javascript Statements:

Programs are composed of data and code which manipulates that data. Program instructions are grouped into units called statements. We create programs from lot of statements.

If-else:

The JavaScript if-else statement is used to execute the code whether condition is true or false. There are three forms of if statement in JavaScript.

- if Statement
- if else statement
- if else if statement

If statement:

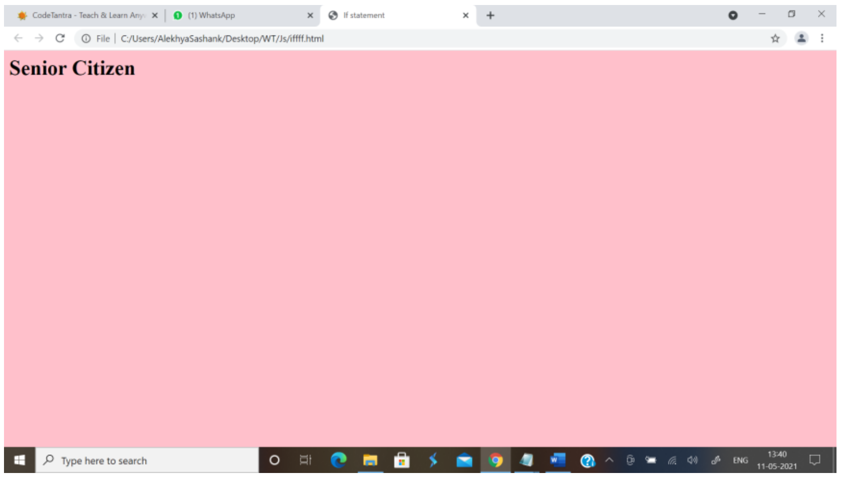
It evaluates the content only if expression is true. The signature of JavaScript if statement is:

```
if(expression){
    //content to be evaluated
}
```

Example:

```
<html>
<head>
<title>If statement</title>
</head>
<body bgcolor="pink">
<script type="text/javascript">
var age=parseInt(prompt("Enter age of the person",""));
if(age>60){
document.write("<h1>Senior Citizen</h1>");
}
</script>
</body>
</html>
```

Output:


CodeTantra - Teach & Learn Any
(1) WhatsApp
If statement
File | C:/Users/AlekhyaSashank/Desktop/WT/Js/iffff.html
Type here to search
ENG
13:40
11-05-2021

```
Senior Citizen
```

JavaScript If...else Statement:

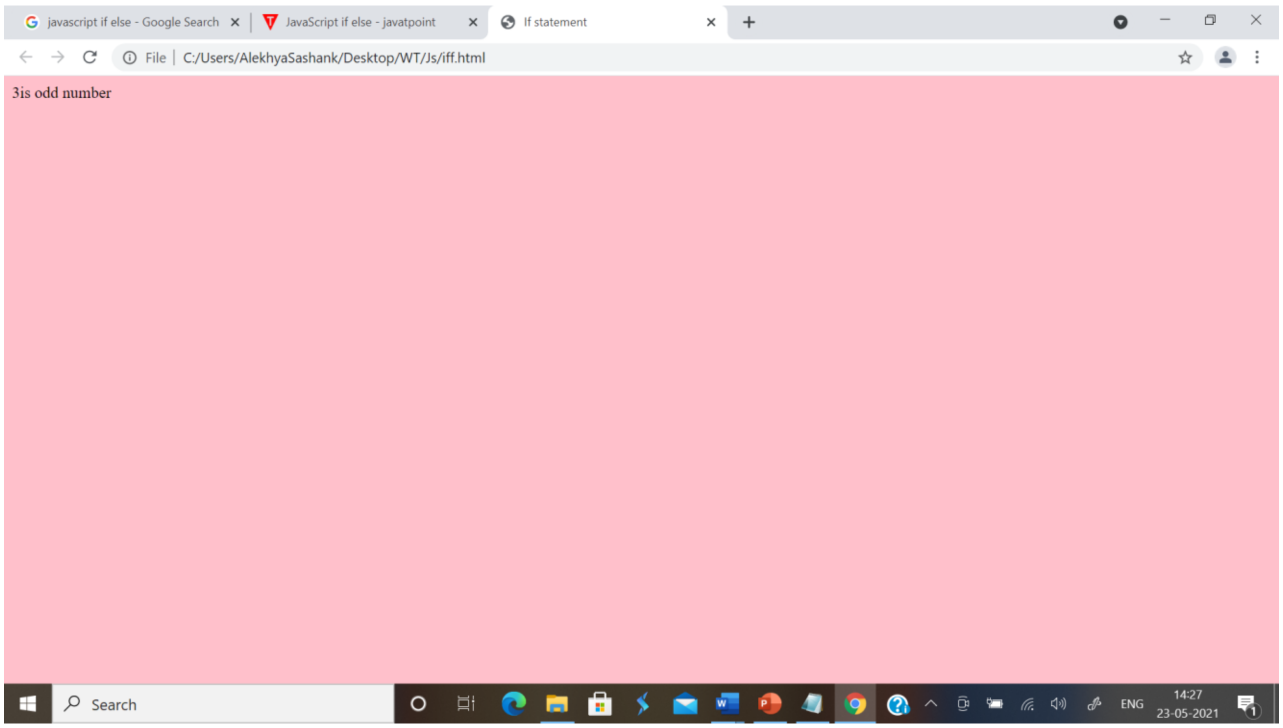
It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is:

```
if(expression){
//content to be evaluated if condition is true
}
else{
//content to be evaluated if condition is false
}
```

Example:

```
<html>
<head>
<title>If statement</title>
</head>
<body bgcolor="pink">
<script language="javascript">
var n=parseInt(prompt("Enter a value"," "));
if(n%2==0){
document.write(n+"is even number");
}
else{
document.write(n+"is odd number");
}
</script>
</body>
</html>
```

Output:


javascript if else - Google Search
JavaScript if else - javatpoint
If statement
File | C:/Users/AlekhyaSashank/Desktop/WT/Js/iff.html
Search
ENG
14:27
23-05-2021

```
3is odd number
```

JavaScript If...else if statement:

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is:

```
if(expression1){
//content to be evaluated if expression1 is true
}
else if(expression2){
//content to be evaluated if expression2 is true
}
else if(expression3){
//content to be evaluated if expression3 is true
}
else{
//content to be evaluated if no expression is true
}
```

Example:

```
<html>
<head>
<title>if statement</title>
</head>
<body bgcolor="pink">
<script language="javascript">
var m1=parseInt(prompt("Enter Marks1","0"));
var m2=parseInt(prompt("Enter Marks2","0"));
var m3=parseInt(prompt("Enter Marks3","0"));
var avg=parseInt((m1+m2+m3)/3);
if(avg>70){
document.write("<h1>Distinction</h1>");
}
else if(avg<70&&avg>60){
document.write("<h1>First Class</h1>");
}
else if(avg<60&&avg>50){
document.write("<h1>Second class</h1>");
}
else{
document.write("<h1>Fail</h1>");
}
</script>
</body>
</html>
```

JavaScript Switch:

The JavaScript switch statement is used to execute one code from multiple expressions. The signature of JavaScript switch statement is :

```
switch(expression){
```

```
case value1:
 code to be executed;
 break;
case value2:
 code to be executed;
 break;
......

default:
 code to be executed if above values are not matched;
}
```

Example:

```
<html>
<head>
<title>Switch</title>
</head>
<body bgcolor="pink">
<script language="javascript">
var a=parseInt(prompt("Enter a value"," "));
var b=parseInt(prompt("Enter b value"," "));
var ch=parseInt(prompt("Enter your choice"," "));
switch(ch){
case 1: document.write("<h1>Addition is:"+(a+b)+"<h1>");
            break;
case 2: document.write("<h1>Subtraction is:"+(a-b)+"<h1>");
            break;
case 3: document.write("<h1>Multiplication is:"+(a*b)+"<h1>");
            break;
case 4: document.write("<h1>Division is:"+(a/b)+"<h1>");
            break;
default: document.write("<h1>Invalid Choice</h1>");
}
</script>
</body>
</html>
```

Javascript Loops:

The JavaScript loops are used to iterate the piece of code. Various types of loops are:

1. for loop
2. while loop
3. do-while loop

JavaScript For loop:

The JavaScript for loop iterates the elements for the fixed number of times. Syntax is:

```
 for (initialization; condition; increment)
{
   code to be executed
}
```

JavaScript while loop:

The JavaScript while loop iterates the elements for the infinite number of times. Syntax is:

```
while (condition)
{
   code to be executed
}
```

JavaScript do while loop:

The JavaScript do while loop iterates the elements for the infinite number of times like while loop. But, code is executed at least once whether condition is true or false. Syntax is:

```
do{
   code to be executed
}while (condition);
```

Example:

```
<html>
<head>
<title>Multiplication</title>
</head>
<body bgcolor="pink">
<script language="javascript">
var n=parseInt(prompt("Enter a number",""));
var i;
for(i=1;i<=10;i++){
document.write("<b>"+n+"*"+i+"="+(n*i)+"<br>");
}
</script>
</body>
</html>
```

Javascript Events:

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page. When the page loads, it is called an event. When the user clicks a button, that click too is an event. Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

Form Events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Window/Document events:

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

Javascript Functions:

A javascript function contains code that will be executed by an event or by a call to function. We may call a function from anywhere within a page. Functions can be defined both in <head> and <body> section of a document. Syntax to define a function is:

```
function function_name(var1, var2,…..,varn)
{
//Block of code
}
```

- A function with no parameters must include the parentheses () after function name.
- The keyword function must be written in lower case, otherwise javascript error occurs.

Example:

```
<html>
<head>
<title>functions</title>
<script language="javascript">
function displayMessage(){
alert("This is a function");
}
</script>
</head>
```

```
<body bgcolor="pink">
<form>
<input type="button" value="Click Me" onClick="displayMessage()">
</form>
</body>
</html>
```

Function with parameters:

```
<html>
<head>
<title>Function</title>
</head>
<body onLoad="pinfo('abc',30)">
<script language="javascript">
function pinfo(name,age){
document.write("<center><h1>User Information</h1><br>");
document.write("<h3>Name is:"+name+"</h3>");
document.write("<br><h3>Age is:"+age+"</h3>");
document.write("</center>");
document.close();
}
</script>
</body>
</html>
```

Objects in Javascript:

Window object:

Various properties & methods supported by window object are:

- **open():** It is used to open a new window. Two arguments are provided. **URL** that specifies the path of documents which should be loaded in the window and **Name of window.**
- **close():** This method is used to close the current window.
- **scroll():** By using this method contents of a given window can be easily scrolled.

Apart from above methods, window object may also have properties such as **toolbar, location, menubar, scrollbar, resizeable etc..**

Examples:

Creating a new window and loading existing page:

```
<html>
<head>
<title>Window</title>
<script language="javascript">
function showNewPage(){
window.open("login.html");
}
</script>
</head>
<body bgcolor="pink">
<center>
<form>
<input type="button" value="SeeLogin" onClick="showNewPage()">
</form>
</center>
</body>
</html>
```

Creating a new window:

```
<html>
<head>
<title>Window</title>
<script language="javascript">
function showNewPage(){
var newWindow=window.open("window.html","height=200,width=450");
newWindow.document.write("<center><h1>This is a new Window</h1></center>");
}
```

```
</script>
</head>
<body bgcolor="pink">
<center>
<form>
<input type="button" value="Open page" onClick="showNewPage()">
</form>
</center>
</body>
</html>
```

Document object:

Document refers to page which will be displayed as soon as the browser window is opened. Various methods/properties supported by the document object are:

- **write()/writeln():** We can create HTML pages using javascript by using write()/writeln() methods. Data can also be inserted.
- **bgcolor/fgcolor:** These are the same properties that can be set in the <body> tag. The only difference here is that the values can be set from javascript.
- **anchors:** It is an array holding the names of anchor elements appearing on web page.
- **links:** It is an array holding all links appearing on web page.
- **forms:** It is an array that contains all of the HTML forms.
- **close():** The document is not completely written until the close() method is called. The browser keep waiting for more data if user do not call this method.

Form object:

Two aspects of form can be manipulated through javascript.

- Most commonly the data that is entered onto the form can be checked at submission.
- We can actually build forms through javascript.

Various form events are:

- **onClick:** This event is triggered when the user clicks on an element.
- **onSubmit:** This event can only be triggered when the form is submitted.
- **onReset:** This event is triggered when the form is reset by the user.

Simple form validation:

```
<html>
<head>
<title>Validation</title>
<script language="javascript">
function validate(){
var value=document.forms[0].elements[0].value;
if(value!="abc"){
document.forms[0].reset();
}
else{
alert("Hi abc");
}
}
</script>
</head>
<body bgcolor="pink">
<form>
Name:<input type="text" name="txt1" value=""><br>
<input type="submit" value="Submit" onClick="validate()">
</form>
</body>
</html>
```

Math object:

Methods supported by math object are:

- **min():** Displays minimum of two numeric values entered. (min(45,65)).
- **max():** Displays maximum of two numeric values entered. (min(45,65)).
- **abs():** Displays the absolute value of numeric entity entered into it.
- **ceil():** Displays the rounded value of the integer entered into it. (ceil(5.2)=6)
- **round():** It rounds the value entered to its nearest integer. (round(5.2)=5)

- **sqrt():** Displays the square root of the value entered into it.
- **sin():** Displays the trigonometric sine value.
- **cos():** Displays the trigonometric cosine value.
- **tan():** Displays the trigonometric tangent value.
- **log():** Displays the logarithmic equivalent value.

Browser object:

The browser object is also called as navigator object. Methods supported by browser object are:

- **navigator.appcodeName:** The internal name for the browser.
- **navigator.appName:** This is the public name of the browser.
- **navigator.appversion:** The version number, platform on which the browser is running and the version of navigator with which it is compatible.
- **navigator.userAgent:** The strings appcodeName and appVersion concatenated together.
- **navigator.plugins:** An array containing details of all installed plugins.
- **navigator.mimeTypes:** An array of all supported MIME types.

Date object:

Javascript Date provides functions to perform many different date manipulations. In javascript dates and times represent the number of milliseconds since 01/01/1970 UTC. Javascript like most programming systems has two separate notions of time:

- UTC is universal time, also known as Greenwich Mean Time, which is the standard time throughout the world.
- Local time is the time on the machine which is executing the script.

Date object Methods:

1. **Date():** Construct an empty date object.
2. **Date(milliseconds):** Construct a new date object based upon the number of milliseconds which have elapsed since 00:00:00 hours on 01/01/1970.
3. **Date(String):** Create a date object based upon the contents of a text string.
4. **Date(year,month,day[hour,minute,second]):** Create a new date object based upon numeric values for year, month and day. Optional time values may also be supplied.
5. **Parse(string):** Returns the number of milliseconds since midnight on 01/01/1970.

6. **getDate():** Return the day of the month.
7. **getDay():** Return the day of the week.
8. **getHours():** Return the hours.
9. **getMilliseconds():** Return the milliseconds.
10. **getMinutes():** Return the minutes.
11. **getSeconds():** Return the seconds.
12. **getMonth():** Return the month.
13. **getFullYear():** Return the year as a four digit number.
14. **getTime():** Return the number of milliseconds since midnight on 01/01/1970.
15. **setDate(day):** Set the day value of the object. Accept values in the range 1 to 31.
16. **setFullYear(year[,month,day]):** sets the year value of the object.
17. **setHours(hours[,mins,secs,ms]):** Set the hours value of the object to an integer in the range 0 – 23.
18. **setMilliseconds(ms):** Set the milliseconds value of the object in the range 0 - 999.
19. **setMinutes(min[,secs,ms]):** Set the minutes value of the object in the range 0 - 59.
20. **setSeconds(secs[,ms]):** Set the seconds value of the object to an integer in the range 0 - 59.
21. **setMonth(month[,day]):** Set the month value of the object to an integer in the range 0 - 11.

Example:

```
<html>
<head>
<title>Date</title>
</head>
<body onLoad="Dater()">
<script language="javascript">
function Dater(){
var today=new Date();
var yesterday=new Date();
var diff=today.getDate()-1;
Yesterday.setDate(diff);
document.write("<h3>The date is:"+today+"</h3>");
document.write("<h3>The date yesterday was:"+yesterday+"</h3>");
document.close();
}
</script>
</body>
</html>
```

Javascript Arrays:

The basic array operations are:

- Creation of Arrays.
- Adding elements.
- Accessing individual elements.
- Removing elements.

Creating Arrays:

Javascript arrays can be created in three different ways:

1. The easiest way is simply to declare a variable and pass it some elements in array format.

   var days=[“Monday”,”Tuesday”,”Wednesday”];

2. The second approach is to create an array object using the keyword new and a set of elements to store:

   var days=new Array(“Monday”,”Tuesday”,”Wednesday”);

3. Finally, an empty array object which has a space for a number of elements can be created:

   var days=new Array(3);

Adding elements to Arrays:

Adding an element to an array can be done as shown below:

var days[4]=“Thursday”;

In javascript we have a benefit. If the array is full, then also we can add elements to array. The interpreter simply extends the array and inserts new item. For example:

Accessing Array elements:

The elements in the array are accessed through their index. The same access method is used to find elements and to change their values.

Length:

When accessing array elements we need to know how many elements have been stored into the array. This is done through the length attribute. The index number runs from 0 – length-1.

Example:

```
<html>
<head>
<title>Arrays</title>
</head>
<body>
<script language="javascript">
document.write("<h1> Looping through the Array</h1>");
var data =[10,20,30,40];
var len=data.length;
for(var count=0;count<len;count++){
		document.write(data[count]+",");
}
document.close();
</script>
</body>
</html>
```

Removing Array elements:

Javascript does not provide a built-in function to remove array members. To remove array elements we use the following procedure:

1. Read each element in the array.
2. If the element is not the one which we want to delete, copy it to temporary array.
3. If the element is the one we want to delete, do nothing.
4. Increment loop counter.
5. Repeat the process and copy the array elements again into original array.

Example:

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

```
<title>Arrays</title>
</head>
<body>
<script language="javascript">
document.write("<h1> Removing Array elements</h1>");
var data =[10,20,30,40];
var len=data.length;
for(var count=0;count<len;count++){
        document.write(data[count]+",");
}
var rem=prompt("Enter item to remove","");
var tmp=new Array(data.length-1);
var count2=0;
for(count=0;count<len;count++){
if(data[count]==rem){
}
else{
tmp[count2]=data[count];
count2++;
}
}
data=tmp;
var len=data.length;
for(var count=0;count<len;count++){
        document.write(data[count]+",");
}
document.close();
</script>
</body>
</html>
```

Dynamic HTML using Javascript:

In case of HTML dynamic means subject to change at any time. The DHTML is simply HTML that has the ability to change after the browser has loaded the page. DHTML is also called as **"Animated HTML"** i.e., anything that is written in HTML can be redone after the page loads.

This dynamic capability is achieved through a combination of several key DHTML components that work in conjunction with HTML. DHTML can also be described as the combination of HTML, stylesheets and scripts that allow documents to be animated. The features of dynamic HTML are:

1. **Changing tags and contents:**

   In this we deal with technology of changing the contents and tags of text using the concept of DOM.
2. **Live positioning of elements:**

   In this DHTML has introduced a concept of dynamically positioning tags in the document.
3. **Dynamic fonts:**

   We can increase the font of text dynamically.
4. **Data Binding:**

   This enables page elements such as table cells to attach themselves to data records. Changes to a record are updated on page, and user modification updated on data record.

Example:

```
<html>
<head>
<title>DHTML</title>
</head>
<body bgcolor="pink">
<center>
<h1 onMouseOver="this.style.color='red';" onMouseOut="this.style.color='blue';">
Text changes to red with the mouse
</h1>
</center>
```

```
</body>
</html>
```

Dynamic content – Changing web pages:

There are several methods that insert HTML in pages:

1. insertAdjacentHTML.
2. insertAdjacentText.
3. innerText.
4. outerText.
5. innerHTML.
6. outerHTML.

insertAdjacentHTML & insertAdjacentText:

The insertAdjacentHTML method lets us to insert HTML next to an element that already exists and insertAdjacentText method lets us insert text in the same way.

We can determine where the new text or HTML will go with respect to existing element by passing the constraints **“BeforeBegin, AfterBegin, BeforeEnd or AfterEnd”**.

innerText: Let us change the text between the start and end tags.

outerText: Let us change all the text including start and end tags.

innerHTML: Changes the contents of elements between start and end tags.

outerHTML: Changes contents of an element including the start and end tags.

createElement: It is used to create new web page elements and use methods like insertBefore() and insertAfter() to insert those elements into web page.

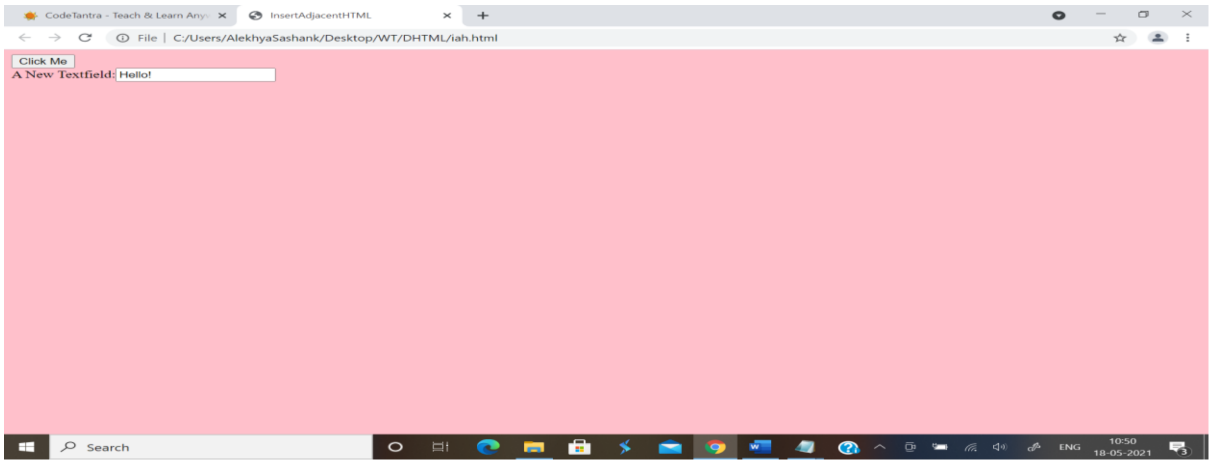
Examples:

insertAdjacentHTML():

```
<html>
<head>
<title>InsertAdjacentHTML</title>
<script language="javascript">
function showMore(){
div1.insertAdjacentHTML('afterend','A New Textfield:<input type="text" value="Hello!">');
}
```

```
</script>
</head>
<body bgcolor="pink">
<div id="div1">
<input type="button" value="Click Me" onClick="showMore()">
</div>
</body>
</html>
```

Output:



innerText:

```
<html>
<head>
<title>InnerText</title>
<script language="javascript">
function changeHeader(){
header.innerText="This is new Header";
}
</script>
</head>
```

```
<body bgcolor="pink">
<center>
<h1 id="header" onClick="changeHeader()">Dynamic HTML</h1>
</center>
</body>
</html>
```

innerHTML:

```
<html>
<head>
<title>InnerText</title>
<script language="javascript">
function changeHeader(){
header.innerHTML="<marquee>This is new Header</marquee>";
}
</script>
</head>
<body bgcolor="pink">
<center>
<h1 id="header" onClick="changeHeader()">Dynamic HTML</h1>
</center>
</body>
</html>
```

outerHTML:

```
<html>
<head>
<title>InnerText</title>
<script language="javascript">
function changeHeader(){
header.outerHTML="<marquee style='font-size:50'>This is new Header</marquee>";
}
```

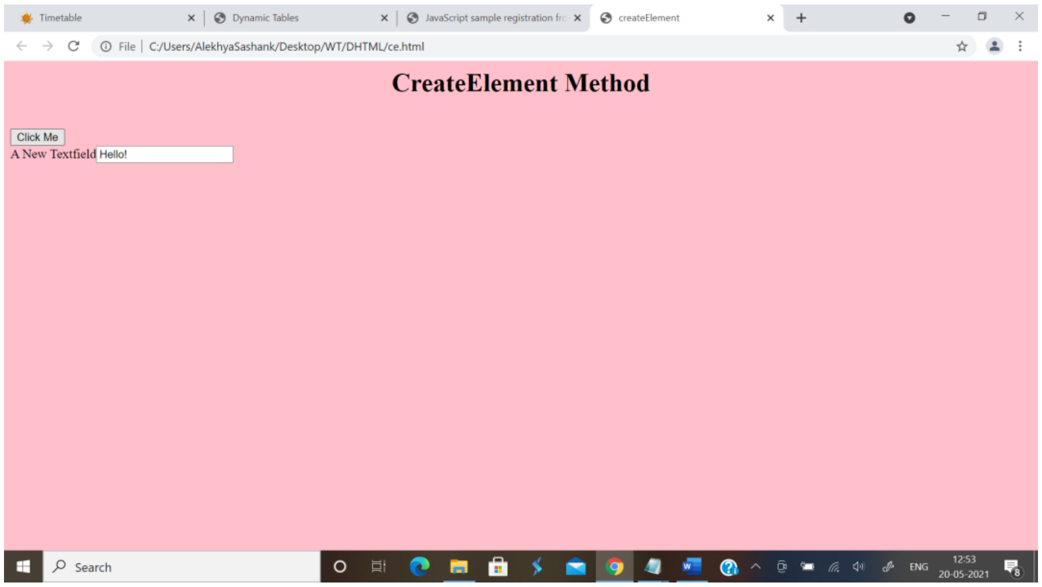
```
</script>
</head>
<body bgcolor="pink">
<center>
<h1 id="header" onClick="changeHeader()">Dynamic HTML</h1>
</center>
</body>
</html>
```

createElement() & createTextNode():

```
<html>
<head>
<title>createElement</title>
<script language="javascript">
function showMore(){
var newDiv,newTextfield,newText;
newDiv=document.createElement("DIV");
newDiv.id="div1";
newTextfield=document.createElement("INPUT");
newTextfield.type="text";
newTextfield.value="Hello!";
newText=document.createTextNode("A New Textfield");
newDiv.insertBefore(newText,null);
newDiv.insertBefore(newTextfield,null);
document.body.insertBefore(newDiv,null);
}
</script>
</head>
<body bgcolor="pink">
<h1 align="center">CreateElement Method</h1><br>
<input type="button" value="Click Me" onClick="showMore()">
```

```
</body>
</html>
```

Output:



Timetable
Dynamic Tables
JavaScript sample registration fr
createElement
File | C:/Users/AlekhyaSashank/Desktop/WT/DHTML/ce.html
Search
ENG
12:53
20-05-2021

CreateElement Method

Click Me

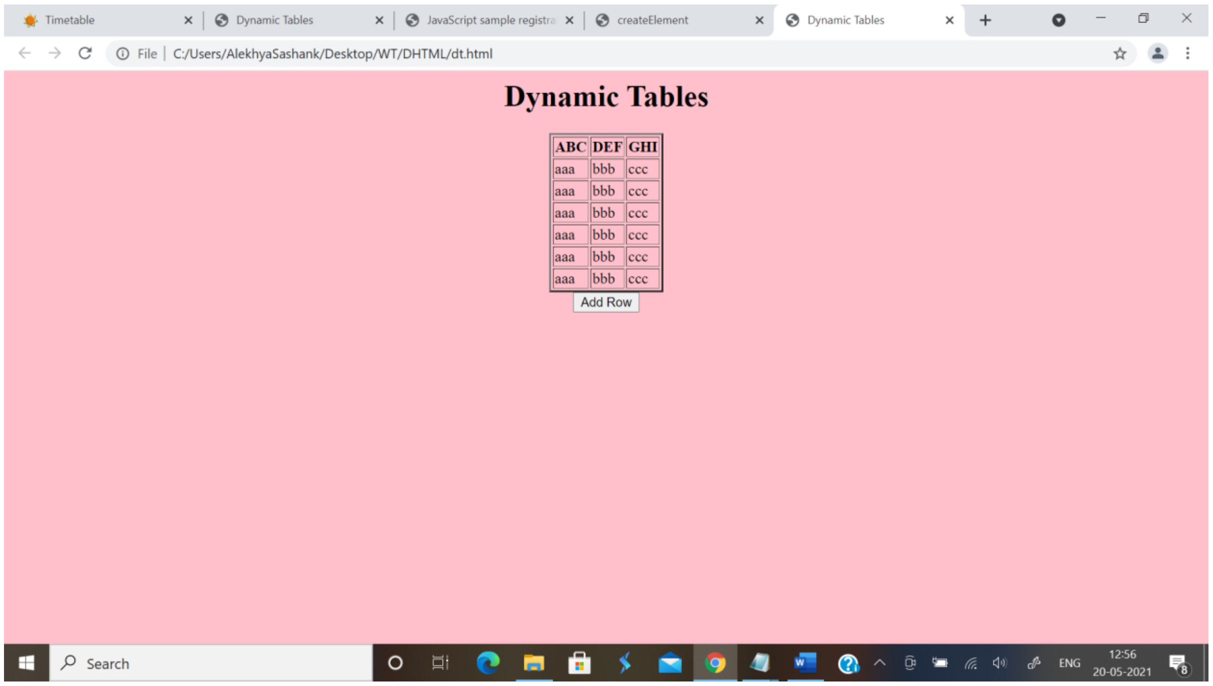
A New Textfield Hello!

Creating Dynamic Tables:

```
<html>
<head>
<title>Dynamic Tables</title>
<script language="javascript">
function addRow(){
var newRow=table1.insertRow(3);
var newCell=newRow.insertCell(0);
newCell.innerText="aaa";
var newCell=newRow.insertCell(1);
newCell.innerText="bbb";
var newCell=newRow.insertCell(2);
newCell.innerText="ccc";
}
</script>
</head>
```

```
<body bgcolor="pink">
<center>
<h1>Dynamic Tables</h1>
<table id="table1" border="2">
<tr>
<th>ABC</th>
<th>DEF</th>
<th>GHI</th>
</tr>
<tr>
<td>aaa</td>
<td>bbb</td>
<td>ccc</td>
</tr>
<tr>
<td>aaa</td>
<td>bbb</td>
<td>ccc</td>
</tr>
</table>
<input type="button" value="Add Row" onClick="addRow()">
</center>
</body>
</html>
```

Output:


Timetable
Dynamic Tables
JavaScript sample registra
createElement
Dynamic Tables
File | C:/Users/AlekhyaSashank/Desktop/WT/DHTML/dt.html
Add Row
Search
ENG
12:56
20-05-2021

Dynamic Tables

ABC	DEF	GHI
aaa	bbb	ccc
aaa	bbb	ccc
aaa	bbb	ccc
aaa	bbb	ccc
aaa	bbb	ccc
aaa	bbb	ccc

Data Validation using Javascript:

Data validation is a process of verifying/validating the data entered by the user. Data validation can be done in two ways:

1. Client-side data validation.
2. Server-side data validation.

Client-side data validation is a process of verifying the data entered by the user before it is submitted to the server.

Server-side data validation is a process of verifying the data at server side with the database data.

Example:

Registration Form Validation:

```
<html>
<head>
<title>JavaScript sample registration from validation</title>
<script type='text/javascript'>
function formValidation()
```

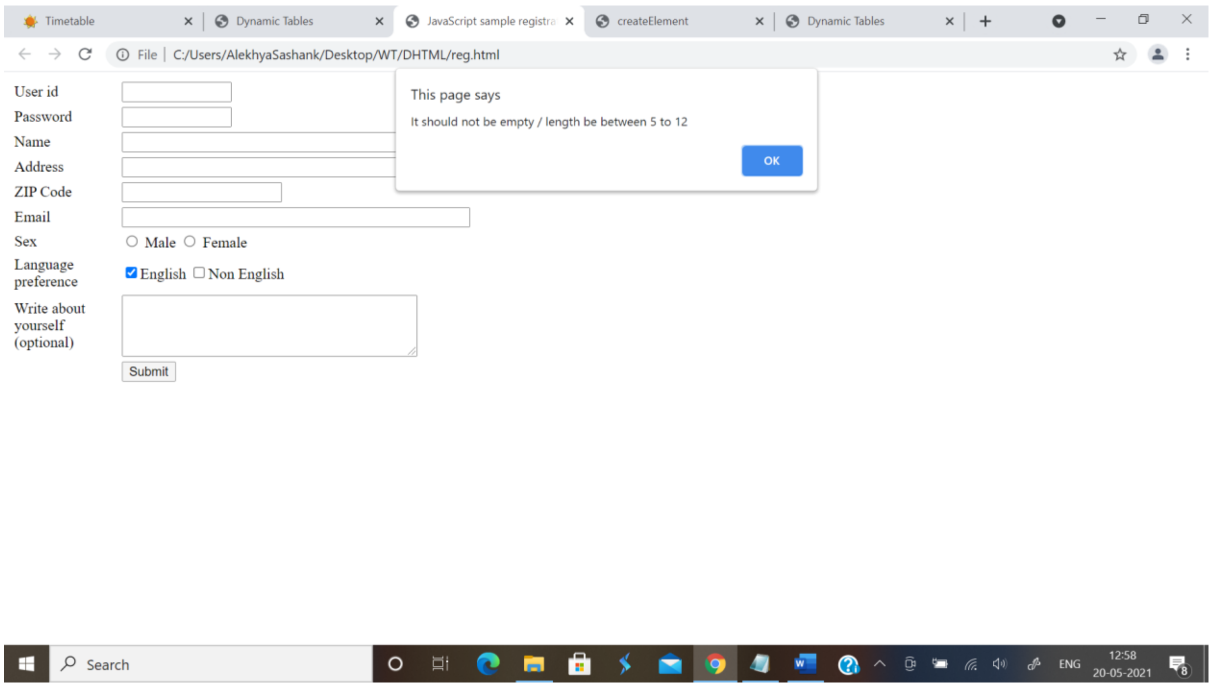
```
{
var uid = document.form1.userid;
var passid = document.form1.passid;
var uname = document.form1.username;
var uadd = document.form1.address;
var uzip = document.form1.zip;
var uemail = document.form1.email;
if(userid_validation(uid,5,12))
{
if(userid_validation(passid,7,12))
{
if(allLetter(uname))
{
if(alphanumeric(uadd))
{
if(allnumeric(uzip))
{
if(ValidateEmail(uemail))
{
}
}
}
}
}
}
return false;
}
function userid_validation(uid,mx,my)
{
var uid_len = uid.value.length;
```

```
if (uid_len == 0 || uid_len >= my || uid_len < mx)
{
alert("It should not be empty / length be between "+mx+" to "+my);
uid.focus();
return false;
}
return true;
}
function allLetter(uname)
{
var letters = /^[A-Za-z]+$/;
if(uname.value.match(letters))
{
return true;
}
else
{
alert('Please input alphabet characters only');
uname.focus();
return false;
}
}
function alphanumeric(uadd)
{
var letters = /^[0-9a-zA-Z]+$/;
if(uadd.value.match(letters))
{
return true;
}
else
```

```
{
alert('Please input alphanumeric characters only');
uadd.focus();
return false;
}
}
function allnumeric(uzip)
{
var numbers = /^[0-9]+$/;
if(uzip.value.match(numbers))
{
return true;
}
else
{
alert('Please input numeric characters only');
uzip.focus();
return false;
}
}
function ValidateEmail(uemail)
{
var mailformat = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
if(uemail.value.match(mailformat))
{
return true;
}
else
{
alert("You have entered an invalid email address!");
uemail.focus();
return false;
}
}
</script>
</head>
<body>
<form name='form1' onsubmit='return formValidation()' >
<table width="500" cellpadding="3" style="border-collapse: collapse;">
<tr>
<td>User id </td>
<td><input type="text" name="userid" size="12" /></td>
</tr>
<tr>
<td>Password</td>
<td><input type="password" name="passid" size="12" /></td>
</tr>
<tr>
<td>Name</td>
<td><input type="text" name="username" size="50" /></td>
</tr>
<tr>
<td>Address</td>
<td><input type="text" name="address" size="50" /></td>
</tr>
<tr>
<td>ZIP Code </td>
<td><input type="text" name="zip" /></td>
</tr>
<tr>
```

```
<td>Email</td>
<td><input type="text" name="email" size="50" /></td>
</tr>
<tr>
<td>Sex</td>
<td><input type="radio" name="msex" value="Male" /> Male
<input type="radio" name="fsex" value="Female" /> Female</td>
</tr>
<tr>
<td>Language preference</td>
<td><input type="checkbox" name="en" value="en" checked />English
<input type="checkbox" name="nonen" value="noen" />Non English</td>
</tr>
<tr>
<td>Write about yourself<br>
(optional)</td>
<td><textarea name="desc" rows="4" cols="40"></textarea></td>
</tr>
<tr>
<td>&nbsp;</td>
<td><input type="submit" name="submit" value="Submit" /></td>
<td>&nbsp;</td>
</tr>
</table>
</form>
</body>
</html>
```

Output:



C:/Users/AlekhyaSashank/Desktop/WT/DHTML/reg.html

This page says

It should not be empty / length be between 5 to 12

OK

User id	
Password	
Name	
Address	
ZIP Code	
Email	
Sex	○ Male ○ Female
Language preference	☑ English ☐ Non English
Write about yourself (optional)	

Submit