Parul® University

Web Programming (203105353)

Prof. Mosam Patel, Assistant Professor,
Department of Computer Science and Engineering

Parul® University

DIGITAL LEARNING CONTENT

CHAPTER-3

JAVA SCRIPT

Overview of JavaScript

Hypertext Markup Language is used to create Web pages.

Static Web pages are combination of text and images and they are fixed and don't change.

Web designing in this internet era uses more dynamic objects and content.

This can be as simple as changing the size of a image when pointer is over it to complex interactive online registration pages.

A script consists of a set of instructions that are executed under certain circumstances.
JavaScript is a dynamic computer programming language

Dynamic web pages are interactive content and they are not fixed and change it.

Overview of JavaScript (Contd..)

JavaScript was first known as **LiveScript,** but Netscape changed its name to JavaScript.

It is client side scripting language developed by Netscape to provide us with dynamic Pages.

JavaScript is a lightweight, interpreted programming language with object-oriented capabilities.

The general-purpose core of the language is to make webpages interactive (e.g., having complex animations, clickable buttons, popup menus, etc.).
Web scripts can run in the two places:

1. The client side/ front-end. -To the web browser used to view a web page.

2. The server side/back-end. - To the server that hosts the website

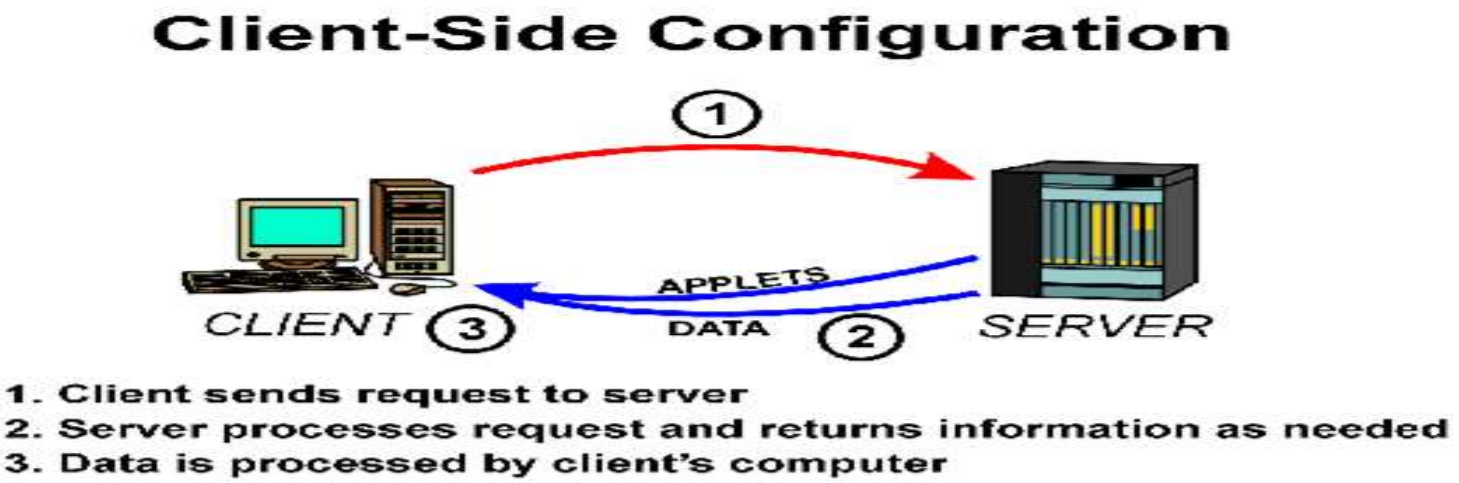
Overview of JavaScript (Contd..)

1)Client Side Scripting:

In this, scripts are executed on the local machine after you have loaded the web page.

When you interact with the web page in some way - for example, by clicking on a button - the script executes. The script is run on your local machine.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.



Overview of JavaScript (Conti..)

The script responds immediately to your click, checks your input and provides a response before you actually submit the form.

It is support many type of languages such as JavaScript is the most popular, but ActionScript, DART and VBScript are also used. Because it is run on the local computer.

Example you go to a shop and as a customer(client) you handover the list of items(request) you want to purchase to the shopkeeper(server) who then processes your order and returns you with all the items(response) and In case item is not available notifies you(error).similar is the client-server architecture.

Overview of JavaScript (Contd..)

2)Server Side Scripting:

Server side Scripting is code running over a server local resources.
Server side languages helps in making web page dynamic as well as interactive to the user.

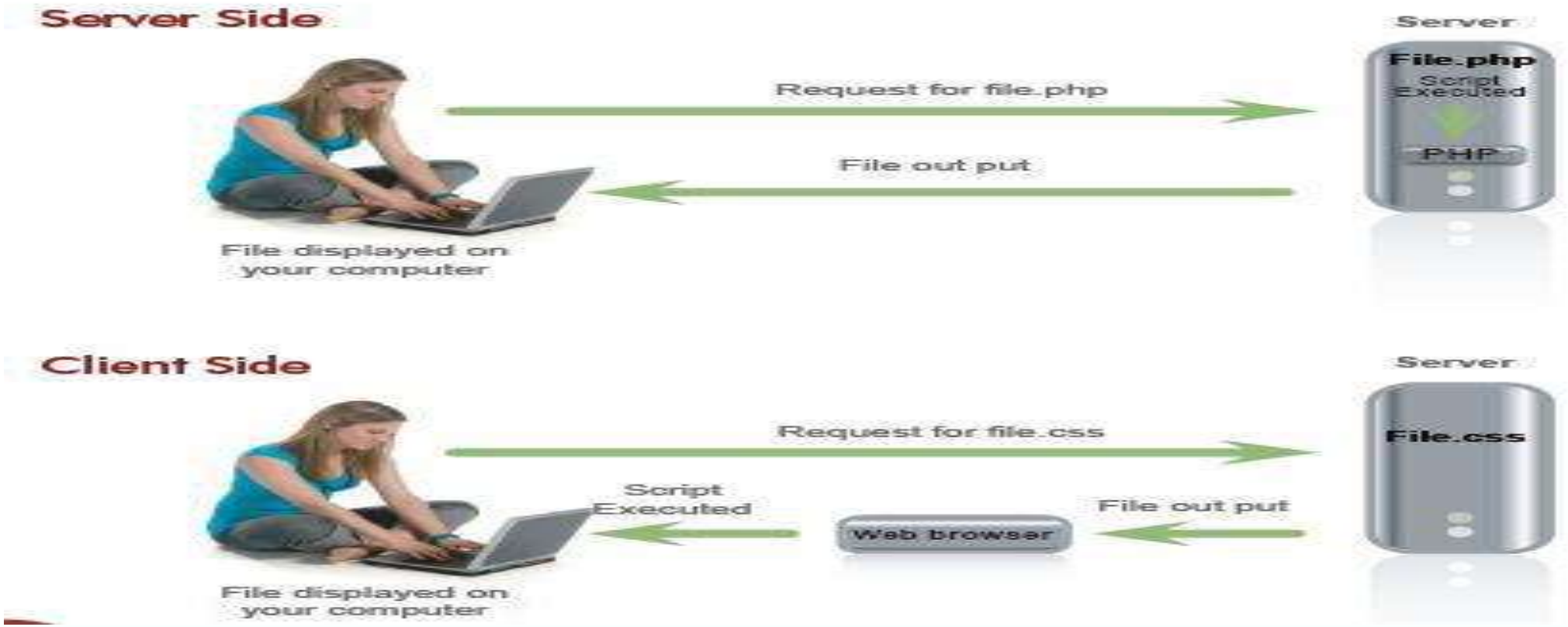
Server side scripts are only produced by the server side and it does not produce client side scripts.

The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.

Databases which are there on the web server are majorly connected by these server side languages only such as PHP, Python, Nodejs, Ruby etc. With this it is very easy to manage the file system which is there at the web server.

Overview of JavaScript (Contd..)

Client/Server Side Scripting – working



Overview of JavaScript (Contd..)

Programming language of the Website.

It is Simple, flexible and narrow learning curve.

Powerful manipulation of the DOM

JavaScript Can make Changes to HTML Styles (CSS)

It has capability to perform Validation

JavaScript Can make Changes to HTML Content

JavaScript Can make Changes to Attributes

Overview of JavaScript (Contd..)

Why Should you learn Java script?

JavaScript is a languages that all web developers must learn:

1. **HTML:** Defines Web sites content through semantic tags (headings, paragraphs, lists, …).

2. **CSS:**Defines 'rules' or 'styles' for presenting every aspect of an HTML document or to specify the layout of web page, Font (family, size, color, weight, etc.),Background (color, image, position, repeat)

3. **JavaScript:** defines dynamic behavior of web pages, Programming logic for interaction with the user, to handle events, etc.

Overview of JavaScript (Contd..)

Use of JavaScript:

Immediate feedback : No need of reload to see if they have forgotten to enter something.

Less server interaction: Allows one to perform client side validation due to which server is used less and thus traffic is reduced so server performance increases

Highly interactivity: As it allow us to create dynamic pages one can create objects that react when the mouse hovers over it or activates it thru the keyboard.

Richer interfaces: Allow usage of drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Extended functionality to web pages: If you use a website and require a certain feature to be included, you can write it yourself and implement it on the web page.

Overview of JavaScript (Contd..)

Response to user actions, Ex. mouse click
It has ability of events management
It Can read and write HTML elements
It Can modify the DOM tree
It Can validate form data(Performing form validation)
It Can access / modify browser cookies
It Can identify the user’s browser and OS
It Can be used as object-oriented language
It Can perform exception handling
Content loading and changing dynamically
Use AJAX functionality

Object orientation and JavaScript

Object-oriented (OO) languages usually are recognized through their use of classes for creating various objects which have similar properties and methods.

It's so deeply rooted in JavaScript that many of JavaScript's native functions and methods are written in the Object Oriented style.

Since JavaScript is an object-oriented programming language and so a programming language can be called object-oriented when it provides programmers with at least four basic capabilities to develop: Object, property, and method, Class, Encapsulation, Abstraction, inheritance, Polymorphism, Association, Aggregation, Composition

Implementing Java script!!!

JavaScript is placed either in the <body> and the <head> sections of an HTML page.

<script> Tag:

In HTML, JavaScript code must be inserted between <script> and </script> tags.

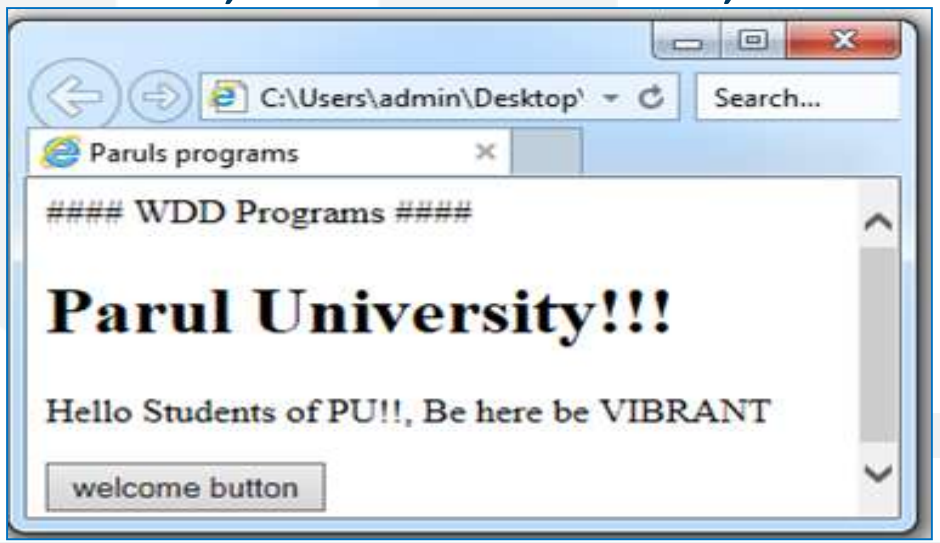
Example:

```
<script>
        document.getElementById("demo").innerHTML = “Welcome To Parul University";
</script>
```

Implementing Java script!!!

1) JavaScript in <head> : A JavaScript function is placed in the <head> section of an HTML page.

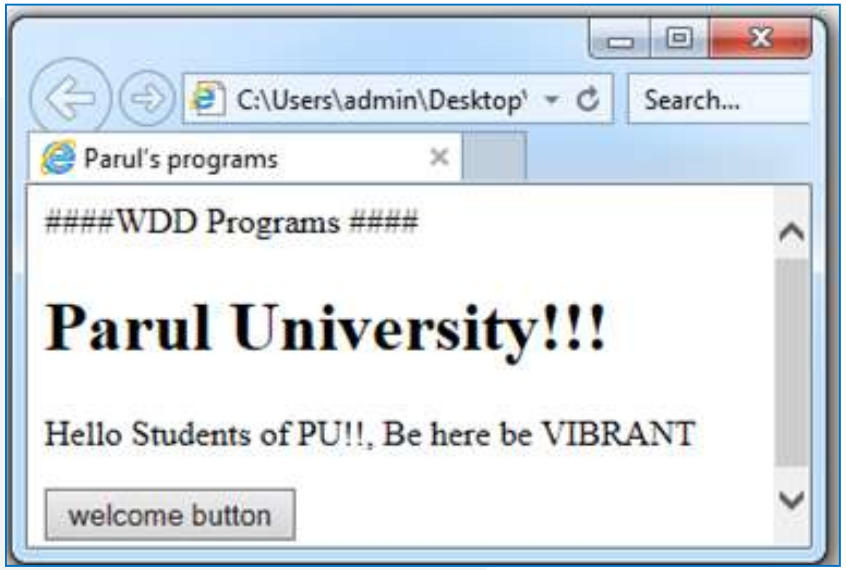
```
<html><head>
<title>Paruls programs</title>
<script>
function parul() {
  document.getElementById("PIT").innerHTML = "Hello Students of PU!!, Be here be VIBRANT";
}
</script>
</head>
<body>
<p>#### WDD Programs ####</p>
<h1>Parul University!!!</h1>
<p id="PIT"></p>
<button onclick="parul()">welcome button</button>
</body><html>
```



Implementing Java script!!!

2) JavaScript in <body>: A JavaScript function is placed in the <body> section of an HTML page.

```
 <html><head>
<title>Parul's programs</title>
</head><body>
<p>####WDD Programs ####</p>
<h1>Parul University!!!</h1>
<p id="PIT"></p>
<button onclick="parul()">welcome button</button>
<script>
    function parul() {
    document.getElementById("PIT").innerHTML = "Hello Students of PU!!, Be here be VIBRANT";
    }
</script>
</body><html>
```



Implementing Java script!!!

Scripts can also be placed in external files. file extension of JavaScript files is **.js**.

It is use when the same code is used in many different web pages.

External scripts cannot contain <script> tags.

To use an external script, put the name of the script file in the src (source) attribute of the <script> tag:

Example:

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

The use of JavaScripts external files are:

Cached JavaScript files can load page faster.

HTML and code are separated

It makes JavaScript and HTML easier to read and maintain

Syntactic characteristics

1. JavaScript Syntax:

JavaScript syntax is the set of rules, how JavaScript programs are constructed.

JavaScript Programs:

A computer program is a list of "instructions" to be "executed" by the computer.

In a programming language, these program instructions are called statements.

JavaScript is a programming language & statements are separated by semicolon.

Example:

```
var parul = 5;
var university= 6;
var piet= w + q;
```

Syntactic characteristics (Contd..)

2. JavaScript Comments: JavaScript comments can be used to explain JavaScript code, and to make it more readable.

Comments are ignored, and will not be executed:
Code after double slashes **//** or between **/*** and ***/** is treated as a comment.

A. Single Line Comments

Single line comments start with //.
Any text between // and the end of a line, will be ignored by JavaScript

Example:

```
// Change heading
document.getElementById("PIT").innerHTML = "I Love PU";
```

Syntactic characteristics (Contd..)

B. Multi-line Comments

Multi-line comments start with /* and end with */.

Any text between /* and */ will be ignored by JavaScript.

This example uses a multi-line comment (a comment block) to explain the code:

Example

```
/*
The code below will change
the heading with id = "Hp"
and the paragraph with id = "myP"
in my web page:
*/
document.getElementById("Hp").innerHTML = “I Love PU";
```

Syntactic characteristics (Contd..)

3. JavaScript keywords: JavaScript keywords are used to identify actions to be performed. Keyword have a fixed meaning and that meaning cannot change.
Below table describe the list of keywords:

Keyword	Description
break	Terminates a switch or a loop
continue	Jumps out of a loop and starts at the top
debugger	Stops the execution of JavaScript, and calls (if available) the debugging function

Syntactic characteristics (Contd..)

Keyword	Description
do ... while	Executes a block of statements, and repeats the block, while a condition is true
for	Marks a block of statements to be executed, as long as a condition is true
function	Declares a function
if ... else	Marks a block of statements to be executed, depending on a condition
return	Exits a function
switch	Marks a block of statements to be executed, depending on different cases
try ... catch	Implements error handling to a block of statements

Syntactic characteristics (Contd..)

4.JavaScript Variables: Variables are used to store data values. All JavaScript variables must be identified with unique names.

var keyword is use to define variables.

The var keyword tells the browser to create a new variable ,In blow example x and y are defined as a variable :

Example : **var x = 5 + 6;**

var y = x * 10;

The basic rules for declaring variables are:

1.Variables Names can contain letters, digits, underscores, and dollar signs.

2. Variables Names must begin with a letter

3. Variables Names can also begin with $ and _

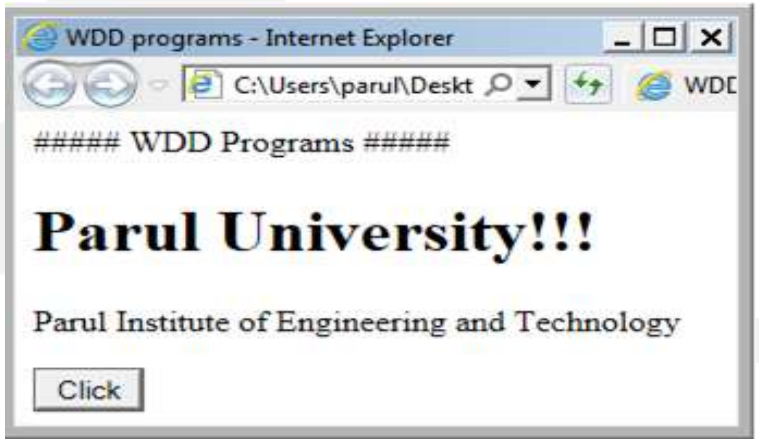
4. Variables Names are case sensitive

5.Reserved words (like JavaScript keywords) cannot be used as names

Syntactic characteristics (Contd..)

A. Local JavaScript Variables: Variables declared inside a JavaScript function, is called LOCAL variable.
Local variables have local scope: They can only be accessed inside the function.

```
<html><head><title>WDD programs</title>
<script>
function parul()
 {
   var Name = "Technology";
   document.getElementById("PIT").innerHTML = "Parul Institute of Engineering and " + Name;
  }
</script></head><body>
<p>##### WDD Programs #####</p>
<h1>Parul University!!!</h1>
<p id="PIT"></p>
<button onclick="parul()">Click</button>
</body><html>
```

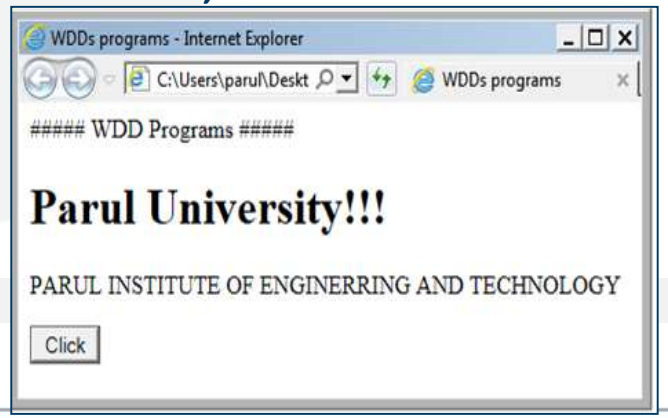


Syntactic characteristics (Contd..)

B. Global JavaScript Variables: variable declared outside a function, is called GLOBAL.

A global variable has global scope: It is accessible anywhere.

```
<html><head>
<title>WDDs programs</title>
<script>
var Name = "TECHNOLOGY";
function parul()
{
document.getElementById("PIET").innerHTML = "PARUL INSTITUTE OF ENGINEERING AND" + Name;
}
</script> </head><body>
<p>##### WDD Programs #####</p>
<h1>Parul University!!!</h1>
<p id="PIET"></p>
<button onclick="parul()“>Click</button>
</body><html>
```



Syntactic characteristics (Contd..)

5. JavaScript Operator: JavaScript Operator is a symbol to perform a specific Task. JavaScript operators are used to assign values, compare values, perform arithmetic operations, and more.

1) Arithmetic Operators (+, - ,*, / ,%)
2) Assignment Operators (=, +=.-+,/=,*=)
3) String Operators
4) Comparison(Relational) Operators (==, <=,>=,<,===,!=)
5) Logical Operators (&&,||,!)
6) Conditional Operator
7) Bitwise Operators (&,|,^,<<,>>,~)

Syntactic characteristics (Contd..)

6. JavaScript statements: JavaScript statements are "instructions" to be "executed" by the web browser. JavaScript statements are combination : Values, Operators, Expressions, Keywords, and Comments.

Example:
```
<html><head>
<title>WDD programs</title>
</head><body>
<p>##### WDD Programs ######</p>
<p># Author:- Tejal K Patel</p>
<h1>Parul University!!!</h1>
<p id="PIET"></p>
<script>
document.getElementById("PIET").innerHTML = "Welcome to Tejals classroom!";
</script>
</body>
<html>
```

Syntactic characteristics (Contd..)

7. JavaScript Data Type: JavaScript variables can hold many data types: numbers, strings, arrays, objects and more:

The type of a variable can be determined at any instant of time by using typeof operator. The typeof operator returns a string describing the type of value.

Example:

```
var Numeric = 1690;                                    // Number
var check= 2<31                                        //Boolean(return true or false)
var fname = “parul";                                   // String
var faculty = [“TKP", “DKS", “RM"];                    // Array
var details = {firstName:“Hiren", lastName:“Patel"};   // Object
```

Syntactic characteristics (Contd..)

If the second operand is a string, JavaScript will also treat the first operand as a string.

```
var x = 20 + “Parul";
```

O/P: Parul

JavaScript evaluates expressions from left to right. Different sequences can produce different results:

```
var x = 20 + 4 + " Parul ";
```

O/P: 24 Parul

The first operand is a string, all operands are considered as strings.

```
var x = " Parul " + 20 + 4;
```

O/P: Parul 204

Screen output

JavaScript does not support for built-in display functions.

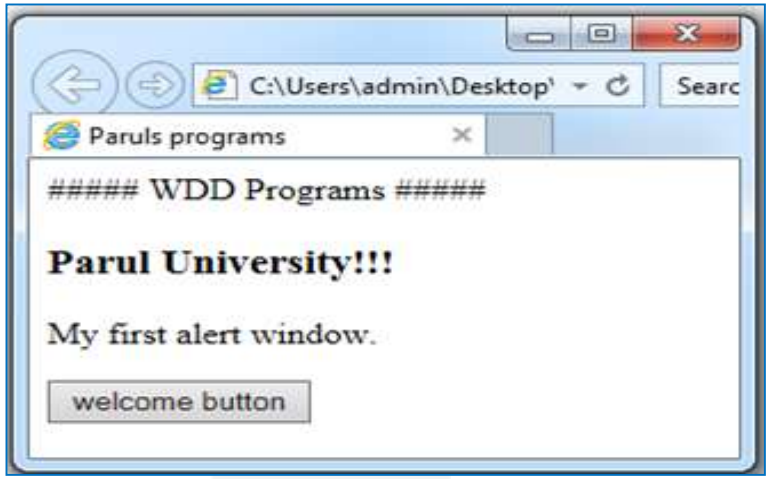
There are some ways to display/print output:

1. Writing into an alert box, using **window.alert()**.
2. Writing into the HTML output using **document.write()**.
3. Writing into an HTML element, using **innerHTML**.
4. Writing into the browser console, using **console.log()**.

Screen output (Contd..)

1) Using window.alert():can be use to display message and data.

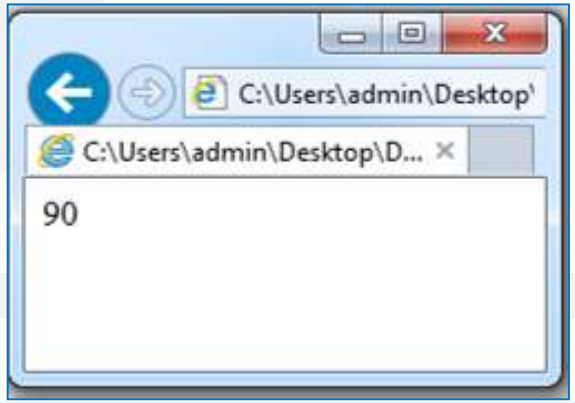
```
<html><head>
<title>Paruls programs</title>
  <script>
     function parul() {
     window.alert(2020 + 2021 );
   }
</script></head>
<body>
<p>##### WDD Programs #####</p>
<h3>Parul University!!!</h3>
<p>My first alert window.</p>
<button onclick="parul()">welcome button</button>
</body><html>
```



Screen output (Contd..)

2) Using document.write(): will delete all existing HTML.

```
<html>
<head>
<title>Paruls programs</title>
</head>
<body>
<p>#### WDD Programs ####</p>
<h3>Parul University!!!</h3>
<button onclick="document.write(50+40)">welcome button</button>
</body>
<html>
```

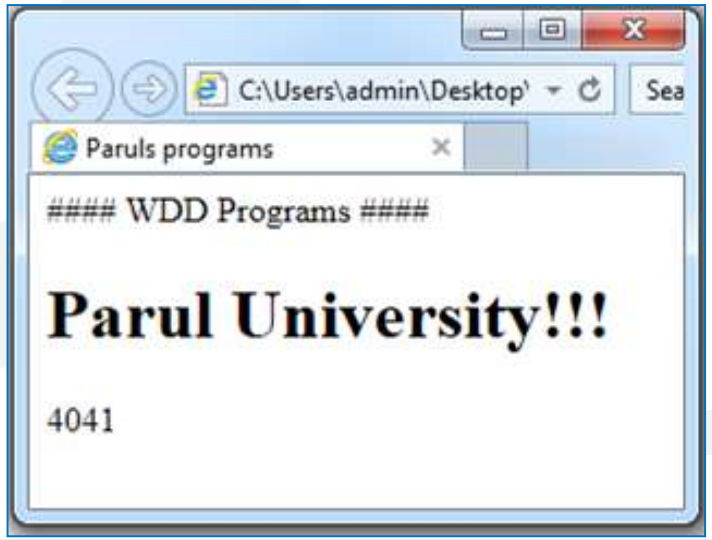


Screen output (Contd..)

3) Using innerHTML: To access an HTML element, JavaScript can use the **document.getElementById(id)**method.

The **id** attribute defines the HTML element. The **innerHTML** property defines the HTML content:

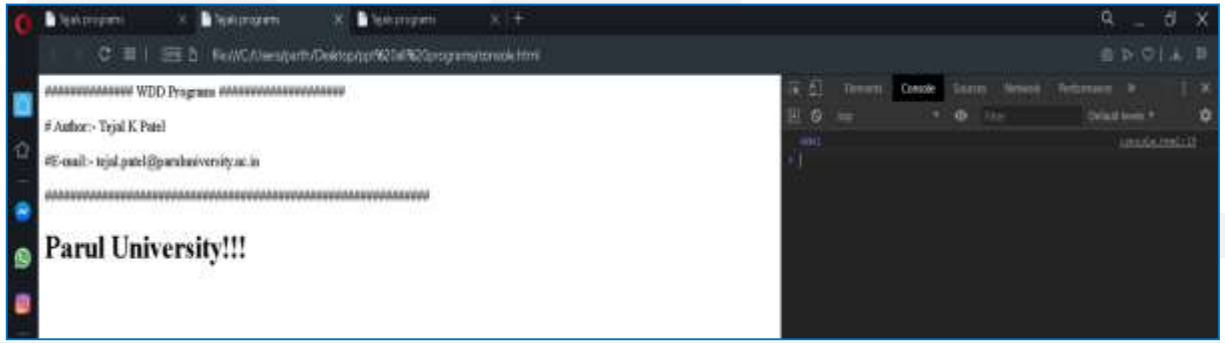
```
<html><head>
<title>Paruls programs</title>
</head>
<body>
<p>#### WDD Programs ####</p>
<h1>Parul University!!!</h1>
<p id="PIET"></p>
<script>
 document.getElementById("PIET").innerHTML = 2020 + 2021;
</script>
</body><html>
```



Screen output (Contd..)

4) Using console.log(): Activate the browser console with F12, and select "Console" in the menu.

```
<html><head>
<title>Tejals programs</title>
</head>
<body>
<p>############## WDD Programs ####################</p>
<p># Author:- Tejal K Patel</p>
<p>#E-mail:- tejal.patel@paruluniversity.ac.in</p>
<p>##################################################################</p>
<h1>Parul University!!!</h1>
<p id="PIET"></p>
<script>
 console.log("Parul University!!!");
</script>
</body><html>
```



JavaScript Program

The statements are executed in order as they are written.
In this example, pu, piet, and pit is given values, and finally pu is displayed:

Example:
```
var piet = 5;
var pit = 6;
var pu = piet + pit;
document.getElementById("demo").innerHTML = pu;
```

If a statement does not fit on one line, the best place to break it, is after an operator:

Example:
```
document.getElementById("demo").innerHTML =
"Parul University.";
```

Control Statements

The control structures within JavaScript allow the program flow to change within a unit of code or function.

These statements can determine whether or not given statements are executed - and provide the basis for the repeated execution of a block of code.

A. JavaScript Condition: Conditional statements are used to perform different actions based on different conditions.

B. JavaScript Loop: Loops are control structures that execute other statements repetitively until some conditions are satisfied.

C. JavaScript jumps: Jumps are control structures that cause a jump to another part of the program.

Control Statements(Contd..)

A. JavaScript Condition: Conditional statements are used to perform different actions based on different conditions.

In JavaScript there are following conditional statements:

1. if : to executed a code block, if a specified condition is true
2. else: to executed a code block, if the same condition is false
3. else if :to specify a new condition to test, if the first condition is false
4. switch: to check many alternative condition and execute a block of code when condition is satisfied.

Control Statements (Contd..)

1. The if Statement

Use the if statement to execute a block of JavaScript code if a condition is true.

```
if (condition)
{
    block of code to be executed if the condition is true
}
```

Example:

Make a "Good day" greeting if the hour is less than 17:00:

```
if (h< 17)
    {
 message= “Parul university is open";
    }
```

Control Statements (Contd..)

Example:

```
<html><head>
<title>WDD programs</title>
<script>
function parul()
{
    if (new Date().getHours() < 17)
     {   document.getElementById("PIET").innerHTML = " Parul university is open!";  }
 }
</script></head><body>
<p>##### WDD Programs #####</p>
<h1>Parul University!!!</h1>
<p id="PIET"></p>
<button onclick="parul()">welcome button</button>
</body><html>
```

Control Statements (Contd..)

2. The else Statement

Use the else statement to specify a block of code to be executed if the condition is false.

Syntax:

```
if (condition)
    {
        block of code (True)
    }
    else
    {
        block of code (False)
    }
```

Example:

```
if (h< 17)
 {
 message= " Parul university is open ";
 } else {
 message= " Parul university is closed ";
}
```

Control Statements (Contd..)

Example :
```
<html><head>
<title>WDDs programs</title>
<script>
function parul()
{
   if (new Date().getHours() < 9)
     {
      document.getElementById("PIET").innerHTML = " Parul university is Not open!";  }
   else
    {  document.getElementById("PIET").innerHTML = "Parul university is open!"; }
 }</script>
</head><body>
<p>##### WDD Programs #####</p>
<h1>Parul University!!!</h1>
<p id="PIET"></p>
<button onclick="parul()">welcome button</button>
</body><html>
```

Control Statements (Contd..)

3. The else if Statement :Use the **else if** statement to create a new condition if the first condition is false.

Syntax: **if (*condition1*)**

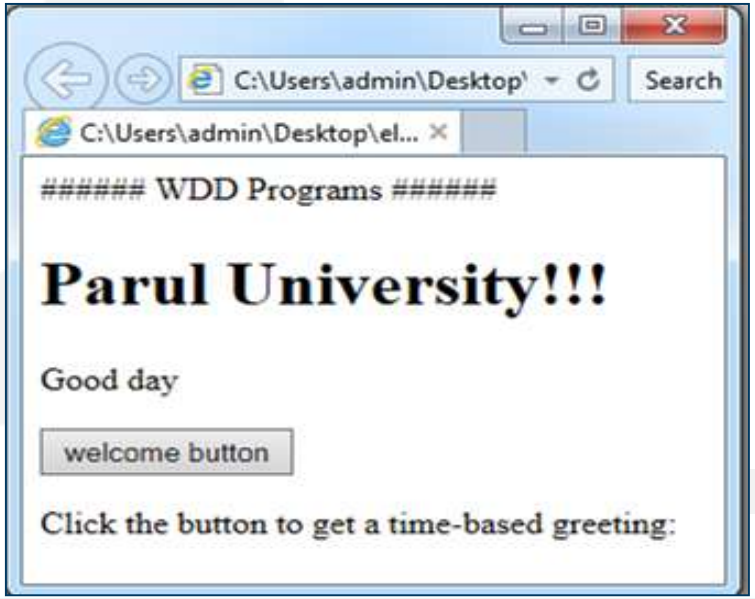
```
{
    block of code [condition1 is true]
} else if (condition2) {
    block of code [if the condition1 is false and condition2 is true]
} else {
    block of code [if the condition1 is false and condition2 is false]
}
```

Example:

```
if (x < 10)
  {
    message= “I have up to 9 Pen";
  } else if (x< 20) {
    message= “I have up to 19 Pen";
  } else {
    message= “I have more than 20 Pen”;   }
```

Control Statements (Contd..)

Example :
```
<html><body>
<p>###### WDD Programs ######</p>
<h1>Parul University!!!</h1><p id="PIET"></p>
<button onclick="parul()">welcome button</button>
<p>Click the button to get a time-based greeting:</p>
<script>
function parul() {
   var wishes;
   var time = new Date().getHours();
   if (time < 10) {
      wishes = "Good morning";
   } else if (time < 20) {
      wishes = "Good day";
   } else {
      wishes = "Good evening";}
document.getElementById("PIET").innerHTML = wishes;}</script></body></html>
```



Control Statements (Contd..)

4. The Switch Statement: It is used to perform different action based on different conditions.
To select one of many blocks of code to be executed using the switch statement .

Syntax:

```
switch(expression)
{
   case n:
      code block
      break;
   case n:
      code block
      break;
   default:
      default code block
}
```

Control Statements (Contd..)

Example: switch (new Date().getDay()) {

```
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday"; break;}
```

Control Statements (Contd..)

Example

```
<html><body>
<p>###### WDD Programs ######</p>
<h1>Parul University!!!</h1><p id="PIET"></p>
<button onclick="parul()">welcome button</button>
<p>Click the button to get a which day it is:</p>
<script>
function parul() {
   var wd;
switch (new Date().getDay()) {
   case 0:
      wd = "Sunday";
      break;
case 1:
      wd = "Monday";
      break;
case 2:
      wd = "Tuesday";    break;
```

Control Statements (Contd..)

```
case 3:
      wd = "Wednesday";
      break;
   case 4:
      wd = "Thursday";
      break;
   case 5:
      wd = "Friday";
      break;
   case  6:
      wd = "Saturday";
      break;
 default:text = "Looking forward to the Weekend";
}
document.getElementById("PIET").innerHTML = wd;
}</script></body></html>
```

Control Statements (Contd..)

B. JavaScript Loop: Loops are control structures that execute other statements repetitively until some conditions are satisfied.

If you want to run the same code over and over again, each time with a different value.

Instead of writing:

```
text += fruit[0] + "<br>";
text += fruit [1] + "<br>";
text += fruit [2] + "<br>";
text += fruit [3] + "<br>";
```

You can write:

```
for (i = 0; i < fruit.length; i++)
{
    text += fruit [i] + "<br>";
}
```

There are four type of Loops:

1. for - loops through a block of code a number of times
2. for/in - loops through the properties of an object
3. while - loops through a block of code while a specified condition is true
4. do/while - also loops through a block of code while a specified condition is true

Control Statements (Contd..)

1. The For Loop

The for loop repeats as long as the condition is met.

If the condition is true then the body of the loop is executed.

Syntax: **for (*statement 1*; *statement 2*; *statement 3*)**

```
{
   block of  code
}
```

Example:

```
for (t = 5; t < 0; i--)
{
text += "The number is " + i + "<br>";
}
```

Control Statements (Contd..)

Example:

```
<html><body>
<p>###### WDD Programs ######</p>
<h1>Parul University!!!</h1>
<p id="PIET"></p>
<button onclick="parul()">welcome button</button>
<script>
function parul() {
var looper = "";
var t;
for (t = 6; t > 0; t--)
{
looper += "Parul University Count Down " + t + "<br>";
document.getElementById("PIET").innerHTML = looper;
}}
</script></body></html>
```

Control Statements (Contd..)

2. The For/In Loop

The JavaScript for/in statement loops through the properties of an object.

In each iteration the name of the property is stored in the variable.

It iterates over all the properties of the object and writes properties names and their values.

Syntax:

```
for (variable in object)
 statement;
```

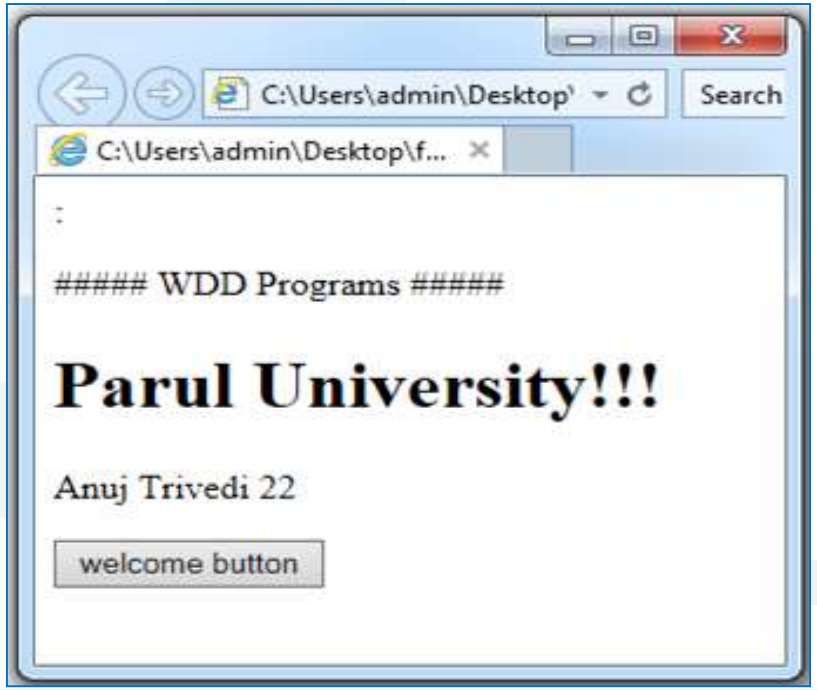
Example:

```
var person = {fname:“Anuj", lname:“Trivedi", age:22};
      var looper = "";
      var t;
      for (t in person)
  {
            looper += person[x];
   }
```

Control Statements (Contd..)

Example:

```
<html><body>
<p>##### WDD Programs #####</p>
<h1>Parul University!!!</h1><p id="PIET"></p>
<button onclick="parul()">welcome button</button>
<script>
function parul() {
var looper = "";
var t= {fname:"Anuj", lname:"Trivedi", age:22};
var pu;
for (pu in t) {
looper += t[pu] + " ";
document.getElementById("PIET").innerHTML = looper;
} }
</script></body></html>
```



Control Statements (Contd..)

3. The While Loop

The while loop loops through a block of code as long as a specified condition is true.

Syntax:

```
while (condition)
  {
      block of code;
  }
```

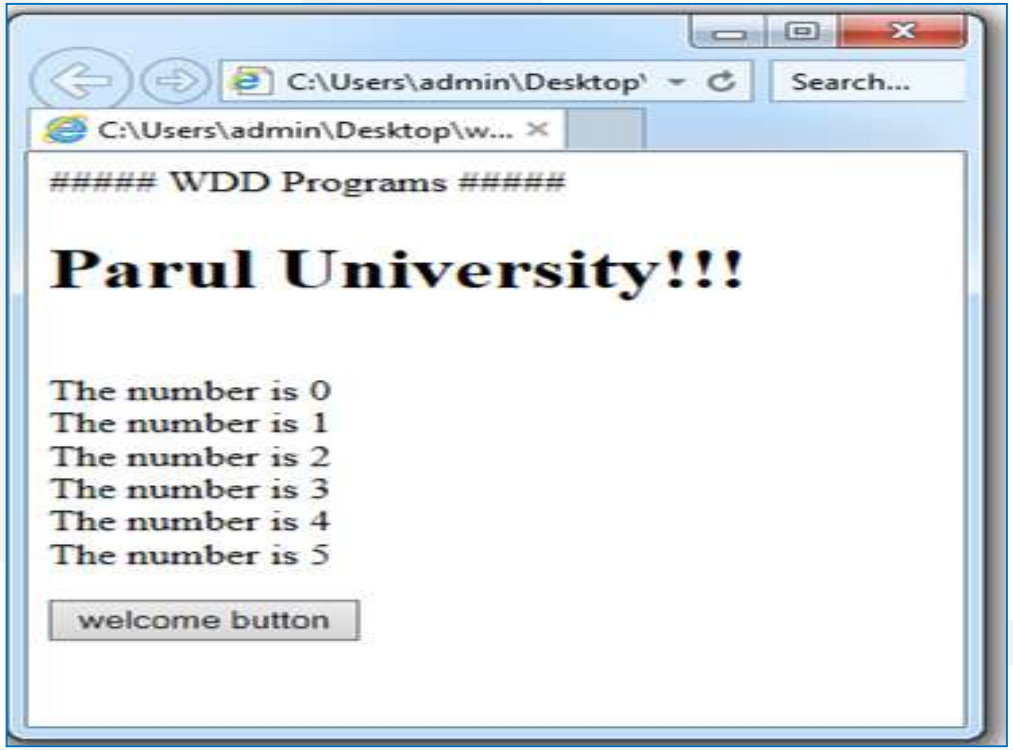
Example:

```
while (j < 10)
  {
      val += "The value is " + j;
      j++;
  }
```

Control Statements (Contd..)

Example: <html><body>

```
<p>##### WDD Programs #####</p>
<h1>Parul University!!!</h1>
<p id="PIET"></p>
<button onclick="parul()">welcome button</button>
<script>
function parul() {
 var val = "";
   var i = 0;
     while (i < 6) {
     val += "<br>The number is " + i;
      i++;
     document.getElementById("PIET").innerHTML = val; }
  }
</script> </body> </html>
```



Control Statements (Contd..)

4. The Do/While Loop

It will execute the block of code once, before checking if the condition is true, then it will repeat the loop until the condition is true.

The loop must be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

Syntax:

```
do {
    block of code
}
while (condition);
```

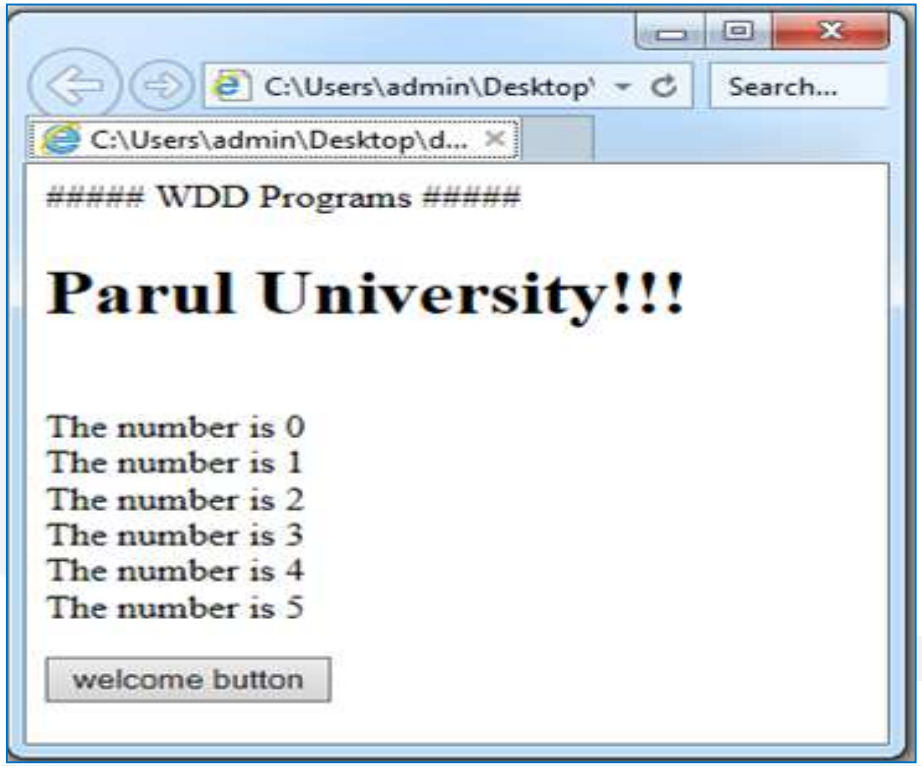
Example:

```
do {
    text += "The number is " + i;
    i++;
}
while (i < 10);
```

Control Statements (Contd..)

Example:

```
<html><body>
<p>##### WDD Programs #####</p>
<h1>Parul University!!!</h1><p id="PIET"></p>
<button onclick="parul()">welcome button</button>
<script>
function parul() {
   var val = "";
   var i = 0;
      do {
      val += "<br>The number is " + i;
      i++;
document.getElementById("PIET").innerHTML = val;
}while (i < 10);
}
</script></body></html>
```



Control Statements (Contd..)

C. JavaScript jumps:

Jumps are control structures that cause a jump to another part of the program.

There are two types of jump statement.

1. The break Statement

2. The continue statement

Control Statements (Contd..)

1. The break Statement:

The break statement causes an immediate exit from a loop (while, do-while, for, for-in, for-of) or switch.

The break statement breaks the loop and continues executing the code after the loop.

syntax:

```
break [label];
```

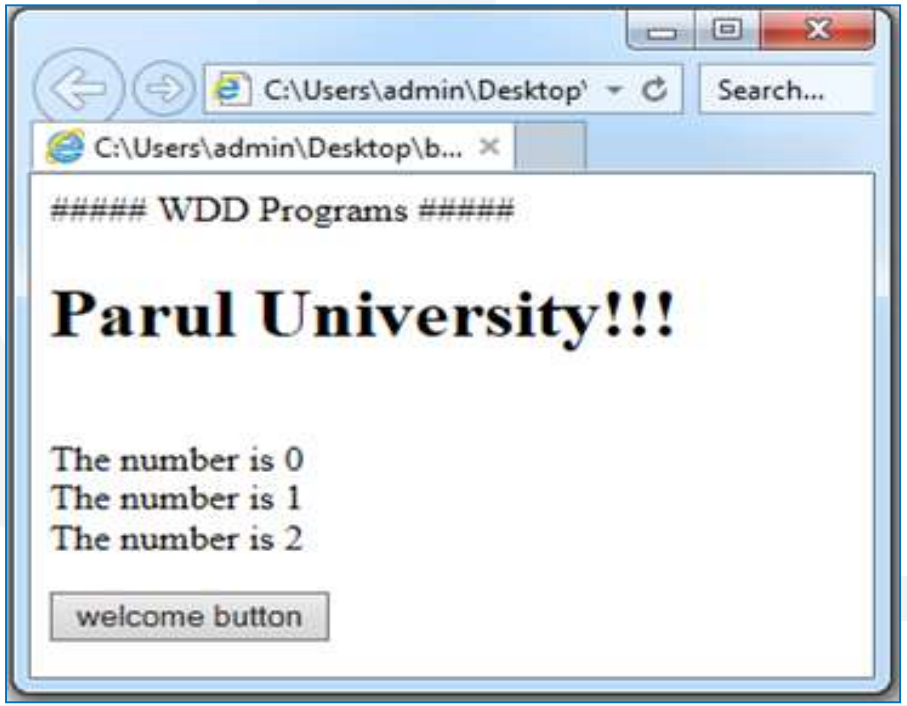
If the break statement is used without a label, it exits from the current loop or switch.

If the break is used with a label it terminates the specified labeled statement.

Control Statements (Contd..)

Example:

```
<html><body>
<p>##### WDD Programs #####</p>
<h1>Parul University!!!</h1><p id="PIET"></p>
<button onclick="parul()">welcome button</button>
<script>
function parul() {
   var val = "";
   var i = 0;
      do {
      val += "<br>The number is " + i;
      i++;
        document.getElementById("PIET").innerHTML = val;
         if (i === 3) {
               break;  }
}while (i < 10);
}
</script></body></html>
```



Control Statements (Contd..)

2. The Continue Statement:

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Syntax:

```
continue [label];
```

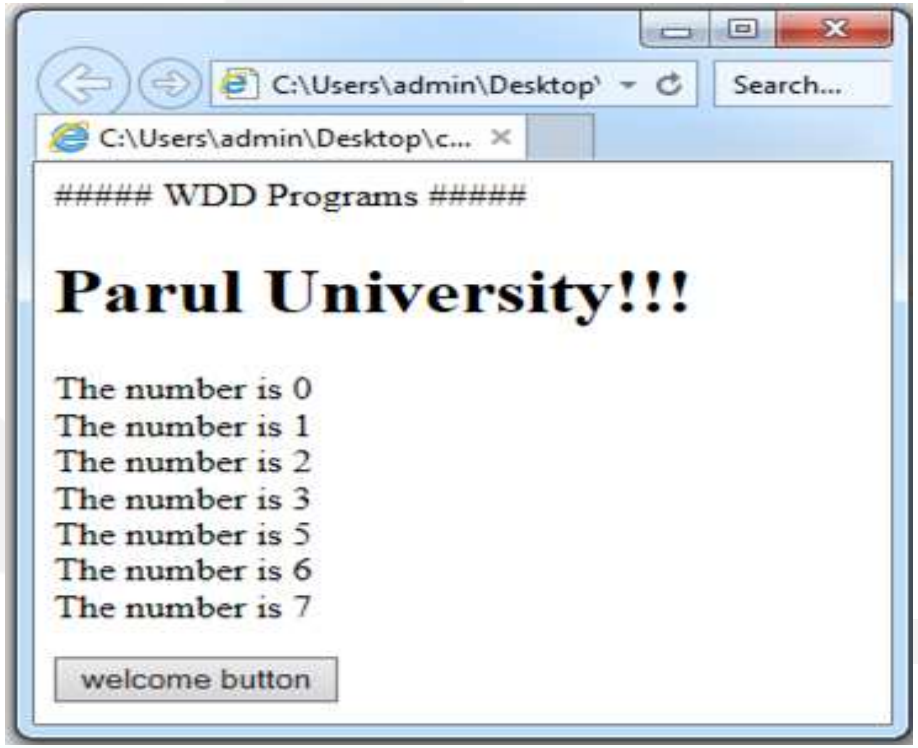
The continue statement used without label terminates the current iteration of immediately enclosing loop. Whereas using continue with a label inside nested loops it terminates the current iteration of loop identified with that label.

Example:

```
for (i = 0; i < 10; i++)
 {
 if (i === 5)                    //skips the value of 5
      continue;
 text += "The number is " + i + "<br>";
}
```

Control Statements (Contd..)

Example:
```
<html><body>
<p>##### WDD Programs #####</p>
<h1>Parul University!!!</h1><p id="PIET"></p>
<button onclick="parul()">welcome button</button>
<script>
function parul() {
        var msg = "";
        var i;
        for (i = 0; i < 8; i++) {
        if (i === 4) {
          continue;
        }
  msg += "The number is " + i + "<br>";
 }
 document.getElementById("PIET").innerHTML = msg;
}</script></body></html>
```



Object creation and modification

JavaScript variables are containers that store data values.

Below code assigns a simple value (Apple) to a variable named mobile:

```
var mobile= " Apple";
```

Objects are considered variables too. But objects can contain multiple values.

Below code assigns multi values (Apple,black) to a variable named car:

```
var mobile = {name:“Apple", color:“black“};
```

The values are written as name:value pairs separated by a colon.

In real life, mobile is an object.

A mobile has properties like name, color, size and methods like lock() & unlock() .

All mobiles have the same properties, but the property values differ from mobile to mobiles

All mobiles have the same methods, but the methods are executed at different times.

Object creation and modification(Contd..)

1.Object Properties :

The name:values pairs are called properties.

var person = {firstName:“Akash", lastName:“Shah", age:24, eyeColor:"blue"};

2.Object Methods:

Methods are procedure that can be performed on objects.

Methods are stored in properties as function definitions.

3.Object Definition:

You define and create a JavaScript object with an object literal:

Example:

```
var person = {firstName:“Akash", lastName:“Shah", age:24, eyeColor:"blue"};
```

Object creation and modification(Contd..)

4.Accessing Object Properties:

```
var person = {
        firstName:“Akash",
        lastName:“Shah",
         age:24,
         eyeColor:"blue“    };
```

We can access object properties in two ways:

objectName.propertyName

OR

objectName[propertyName]

Example:

```
person.lastName;
        OR
person["lastName"];
```

Object creation and modification(Contd..)

5.Accessing Object Methods:

Syntax for accessing object Method is:

objectName.methodName()

Example:

```
name = person.fullName();
```

The fullName **property**, without (), it will return the **function definition**:

Example:

```
name = person.fullName;
```

JavaScript Array

It is use to store multiple values under a single variable.

An array is a special variable, which can hold more than one value at a time.

If you have a list of faculty names, then storing the faculty in single variables such as:

```
var f1 = “parul";
var f2 = “priyanka";
var f3 = “kuldeep";
```

And what if you had not 3 faculty, but 40000?

The solution is an array!

```
var teachers = [" parul", “priyanka", “kuldeep"];
```

JavaScript Array (Contd..)

1.How to Create an Array?

Syntax:

var *array-name* = [*item1*, *item2*, ...];

Example:

```
var teachers = [" parul", “priyanka", “kuldeep"];
```

We can create using the JavaScript Keyword new.

Example:

```
var teachers = new Array(" parul ", “priyanka", “kuldeep");
```

JavaScript Array (Contd..)

2. How to Access the Elements of an Array?

Using the index number you refer to an array element .

This below statement accesses the value of the first element in teachers:

```
var name = teachers[0];
```

This statement modifies the first element in teachers:

```
teachers[0] = “Hiren";
```

JavaScript Array (Contd..)

3. How to Add Array Elements?

We can use length property to add array elements:

Example:

```
var subject = [“WDD", “JAVA", “DAA", “COA"];
subject[subject.length] = “ASP";
```

Adding elements with high indexes can create undefined "holes" in an array:

Example:

```
var subject = [“WDD", “JAVA", “DAA", “COA"];
fruits[10] = “ASP";
```

JavaScript Array (Contd..)

4.Looping Array Elements :

The best way to loop through an array, is using a "for" loop:

Example:

```
var index;
var subject = [“WDD", “JAVA", “DAA", “COA"];
for (index = 0; index <subject.length; index++)
 {
      text += subject[index];
 }
```

JavaScript Array (Contd..)

5. Arrays are Objects

Arrays are a special type of objects. The **typeof** operator in JavaScript returns "object" for arrays.

Arrays use numbers to access its "elements". In this example, person[0] returns Hiren:

Array:

```
var person = [“Hiren", “Patel", 23];
```

Objects use names to access its "members". In this example, person.firstName returns Hiren:

Object:

```
var person = {firstName:“Hiren", lastName:“Patel", age:23};
```

You should use objects when *objects use named indexes* [strings].

You should use arrays when *arrays use numbered indexes.*[numbers.] .

JavaScript Array (Contd..)

6.Array Properties and Methods

The real strength of JavaScript arrays are the built-in array properties and methods:

Examples:

```
var pu = subject.length;
var ds = subject.sort();
```

The length Property

The **length** property of an array returns the length of an array (the number of array elements).

Example:

```
var subject = [“WDD", “JAVA", “DAA", “COA"];
subject.length;  // the length of subject is 4
```

JavaScript Array (Contd..)

Array Methods are:

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

JavaScript Array (Contd..)

Method	Description
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 Array (Contd..)

Array push() Method: To Add a new item in array.

Example:

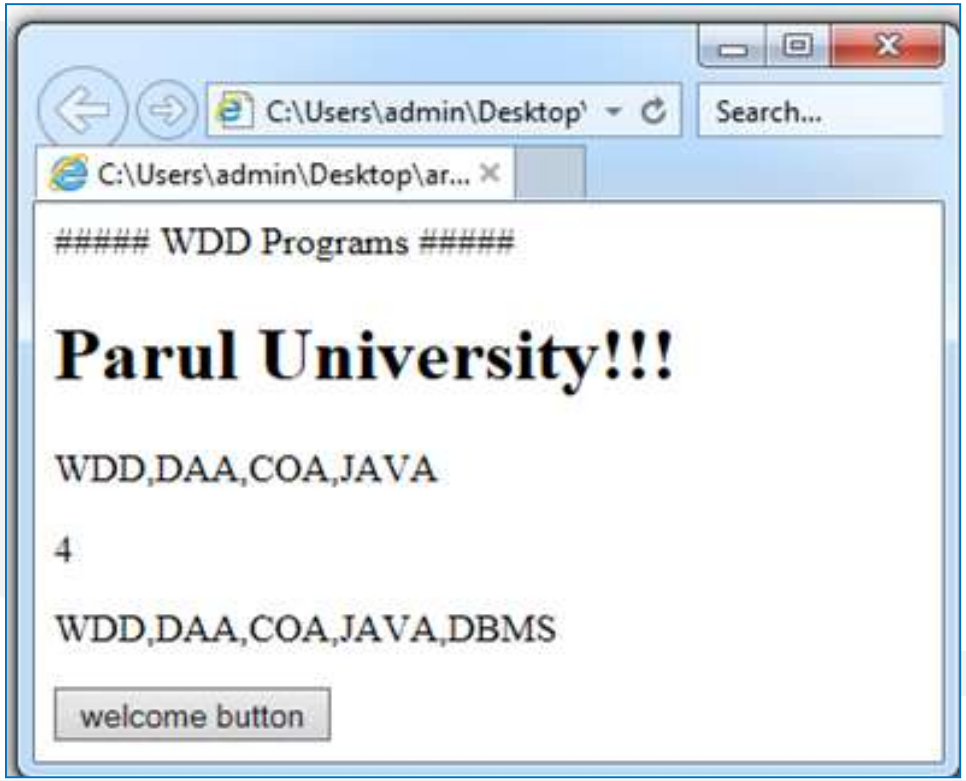
```
var items= [“Pen", “Pencil", “Eraser"];
items.push(“Sharpener");
```

The result of items will be:

Pen, Pencil, Eraser, Sharpener

JavaScript Array (Contd..)

Example:
```
<html><body>
<p>##### WDD Programs #####</p>
<h1>Parul University!!!</h1>
<p id="PIET1"></p>
<p id="PIET2"></p>
<p id="PIET3"></p>
<button onclick="parul()">welcome button</button>
<script>
function parul() {
var subject= ["WDD", "DAA" , "COA", "JAVA"];
document.getElementById("PIET1").innerHTML =subject;
document.getElementById("PIET2").innerHTML =subject.length;
subject.push("DBMS");
document.getElementById("PIET3").innerHTML =subject;
}</script>
</body></html>
```



JavaScript Array (Contd..)

Array Properties are:

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

JavaScript Functions

A JavaScript function is a block of code which perform a specific task.
To execute a JavaScript function "something" must invokes it (calls it).
function keyword is used to defined the JavaScript function , followed by a name, followed by parentheses ().
Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

Syntax:

```
function name(parameter1, parameter2, parameter3)
  {
    code to be executed
  }
```

Example:

```
function parul(t1, t2)
 {
  return t1 * t2;          // The function returns the product of t1 & t2
 }
```

JavaScript Functions(Contd..)

Use of Functions:

Allows reusability of code segment : write the code once, and use it multiple times.

To produce different results using different arguments (can be passed for the same code multiple times)

Example: Convert Fahrenheit to Celsius:

```
function ConCelsius(fahrenheit)
  {
     return (5/9) * (fahrenheit-32);
  }
document.getElementById("demo").innerHTML = ConCelsius(32);
```

JavaScript Functions(Contd..)

Function Return

When JavaScript executes a return statement, the function will stop executing.

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

Functions often have a return value statement that perform computation. The return value is "returned" back to the "caller":

Example: Calculate the product of two numbers, and return the result:

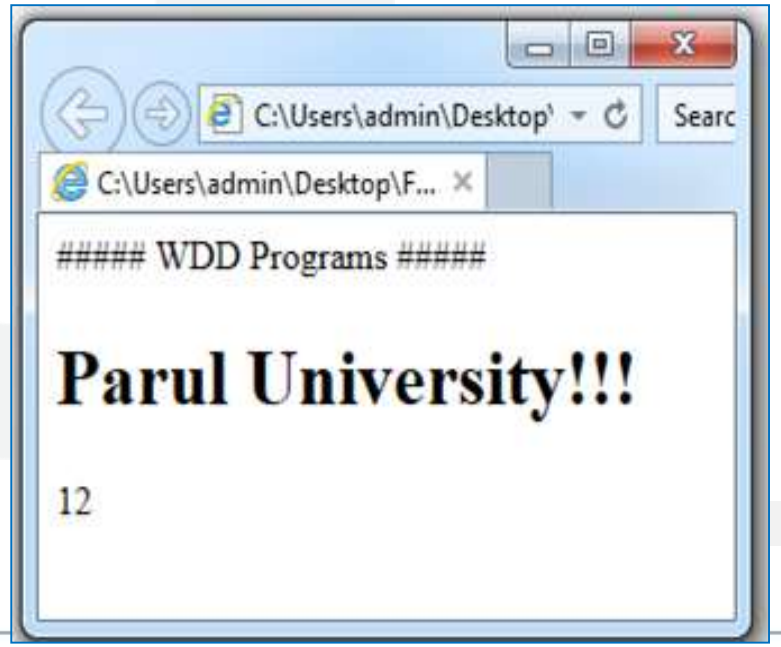
```
var x = parul(4, 3); // Function is called, return value will end up in x

function  parul(a, b)
  {
      return a * b;     // It returns the product of a and b
  }
```

Output: 12

JavaScript Functions(Contd..)

Example:

```
<html>
<body>
<p>##### WDD Programs #####</p>
<h1>Parul University!!!</h1>
<p id="PIET1"></p>
<script>
   function parul(a,b)
     {
       return a * b
     }
document.getElementById("PIET1").innerHTML =parul(3,4);
</script>
</body></html>
```



Pattern matching using regular expressions

Regular expressions are patterns used to match character combinations in strings.
In JavaScript, regular expressions are also objects.
Regular expression patterns include the use of letters, digits, punctuation marks, etc., plus a set of special regular expression characters (do not confuse with the HTML special characters).
The characters that are given special meaning within a regular expression, are:
. * ? + [ ] ( ) { } ^ $ | \.
You will need to backslash these characters whenever you want to use them literally.
For example, if you want to match ".", you'd have to write \.. All other characters automatically assume their literal meanings.
The following sections describe the various options available for formulating patterns:

Pattern matching using regular expressions (Contd..)

The following sections describe the various options available for formulating patterns:

1.Pattern Modifiers: **Modifiers** can be used to perform case-insensitive more global searches. A pattern modifier allows you to control the way a pattern match is handled.

Example: /pattern/i.

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

Pattern matching using regular expressions (Contd..)

2. Character Classes: Brackets are used to find a range of characters.
Square brackets surrounding a pattern of characters are called a character class e.g. [abc].
A character class always matches a single character out of a list of specified characters that means the expression [abc] matches only a, b or c character.

Expression	Description
[abc]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x|y)	Find any of the alternatives separated with |
[^abc]	Matches any one character other than a, b, or c.

Pattern matching using regular expressions (Contd..)

3. Predefined Character Classes: Metacharacters are characters with a special meaning. Some character classes such as digits, letters, and whitespaces are used so frequently that there are shortcut names for them.

Metacharacter	Description
.	Matches any single character except newline \n.
\d	Find a digit Same as [0-9]
\w	Matches any word character (defined as a to z, A to Z,0 to 9, and the underscore) like [a-zA-Z_0-9]
\s	Find a whitespace character. (space, tab, newline or carriage return character) [\t\n\r]
\b	Find a match at the beginning of a word like this: \b WORD, or at the end of a word like this: WORD\b
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Pattern matching using regular expressions (Contd..)

4.Repetition Quantifiers: With quantifiers you can specify how many times a character in a regular expression should match.

Quantifiers can be applied to the individual characters, as well as classes of characters, and groups of characters contained by the parentheses.

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{2}	Matches exactly two occurrences of the letter n.
n{2,3}	Matches at least two occurrences of the letter n, but not more than three occurrences.
n{2,}	Matches two or more occurrences of the letter n.

Pattern matching using regular expressions (Contd..)

5.Position Anchors: There are certain situations where you want to match at the beginning or end of a line, word, or string. To do this you can use anchors.

Two common anchors are caret (^) which represent the start of the string, and the dollar ($) sign which represent the end of the string.

Anchors	Description
^p	Matches the letter p at the beginning of a line.
p$	Matches the letter p at the end of a line.

Pattern matching using regular expressions (Contd..)

Example: The Regular expression for email id is

/^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/

To understand the regular expression we will divide it into smaller components:

/^[a-zA-Z0-9._-]+: Means that the email address must begin with alpha-numeric characters (both lowercase and uppercase characters are allowed). It may have periods, underscores and hyphens.

@: There must be a '@' symbol after initial characters.

[a-zA-Z0-9.-]+: After the '@' sign there must be some alpha-numeric characters. It can also contain period ('.') and hyphens('-').

\.: After the second group of characters there must be a period ('.'). This is to separate domain and sub domain names.

[a-zA-Z]{2,4}$/: Finally, the email address must end with two to four alphabets. Having a-z and A-Z means that both lowercase and uppercase letters are allowed.

{2,4} indicates the minimum and maximum number of characters. This will allow domain names with 2, 3 and 4 characters e.g.; us, tx, org, com, net, wxyz).

Error in JavaScript

Errors are statements which don't let the program run properly.

There are three main types of errors that can occur while compiling a JavaScript program.

1. syntax errors
2. runtime errors
3. logical errors

Error in JavaScript(Contd..)

1.Syntax Errors

Syntax errors are the most common type of error that occurs in any programming language.

As the name suggests, something incorrect in the syntax of the program body raises this error.

Syntax errors are also known as **parsing errors**. In JavaScript, they occur at the interpretation time.

Example:

```
<script type=''text/javascript''>
        window.show(;
</script>
```

Error in JavaScript(Contd..)

2. Runtime Errors

This type of error occurs during the runtime of the program, after it is interpreted by the compiler.

Example:

```
<script type=''text/javascript''>
 window.show();
</script>
```

Notice that there is no show function defined.

This program will raise an error at runtime as the function which is not present is called, although the syntax is correct.

Error in JavaScript(Contd..)

3. Logical Errors

These type of errors are the most difficult to find.

Example: ''player is playing cricket.''
This statement is logically correct and its syntax is also correct.

But : ''cricket is playing player.''
This statement is correct with respect to its syntax but is logically incorrect.

These types of errors cause a serious problem as they change the whole path of how your program will work.

Popup Boxes

JavaScript has three types of popup boxes: Alert box, Confirm box, and Prompt box.

1. Alert Box:

If you want to make sure information is displayed to the user then an alert box is used. When an alert box pops up, the user will have to click "OK" to proceed.

Syntax:

```
window.alert("message");
```

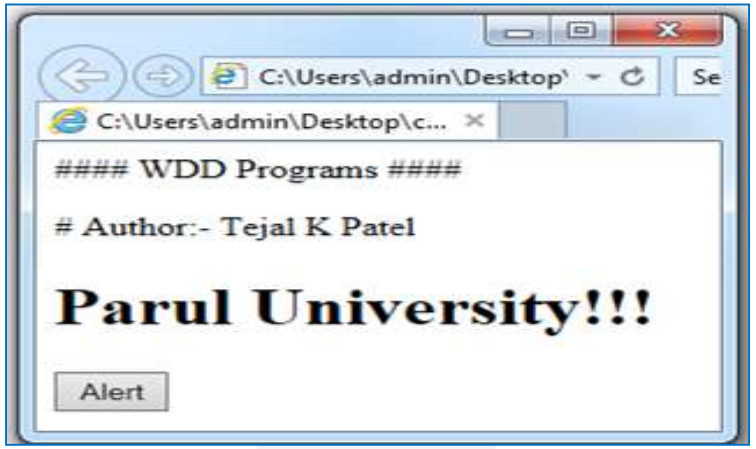
The window.alert method can be written without the window prefix.

Example:

```
alert("Parul University!");
```

Popup Boxes (Contd..)

Example:
```
<html><body>
<p>#### WDD Programs ####</p>
<p># Author:- Tejal K Patel</p>
<h1>Parul University!!!</h1>
<button onclick="parul()">Alert</button>
<script>
function parul()
{
   alert("Hello! I am an Tejal Patel your instructor!");
}
</script>
</body></html>
```



Popup Boxes (Contd..)

2. Confirm Box: A confirm box is used if you want the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax:

```
window.confirm("message");
```

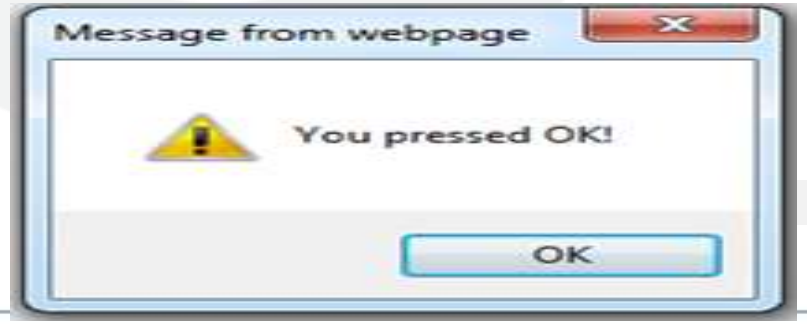
Example:

```
var r = confirm("Press a button");
if (r == true)
    {
  x = "You pressed OK!";
    }
else {
  x = "You pressed Cancel!";
  }
```

Popup Boxes (Contd..)

Example: <html><body>

```
<p>#### WDD Programs ####</p>
<p># Author:- Tejal K Patel</p>
<h1>Parul University!!!</h1>
<button onclick="parul()">Confirm Button</button>
<script>
function parul() {
var r=confirm("Are you from PU??!");
        if (r==true)
        {
            alert("You pressed OK!");
        }
    else
      {
            alert("You pressed Cancel!");
        }   }
</script></body></html>
```



Popup Boxes (Contd..)

3.Prompt Box: A prompt box is used to get input from user before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax:

```
window.prompt("sometext","defaultText");
```

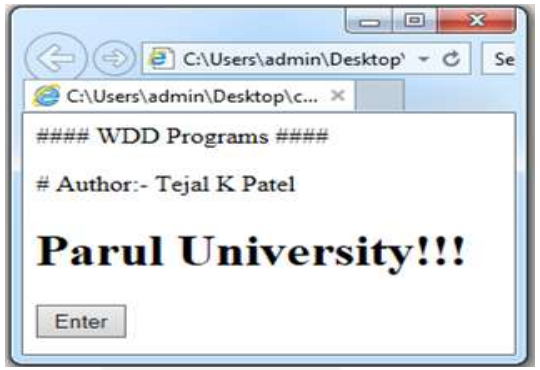
Example:

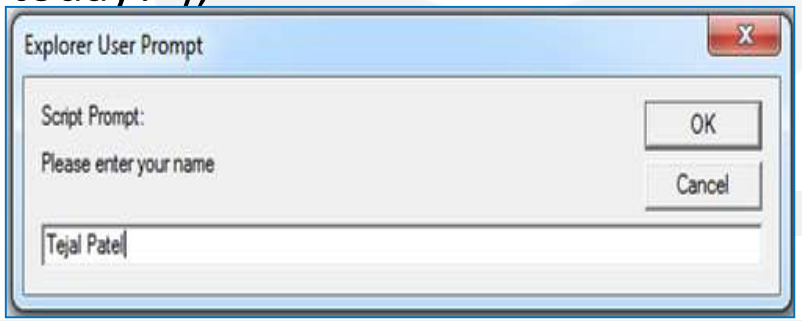
```
var person = prompt("Please enter your name", “Aksah Shah");
if (person != null)
  {
   document.getElementById("demo").innerHTML =  "Hello " + person + "! How are you today?";
  }
```

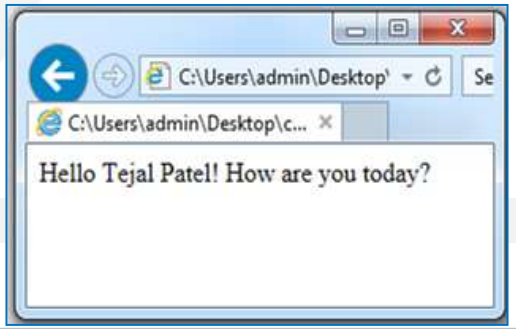
Popup Boxes (Contd..)

Example: <html><body>

```
<p>#### WDD Programs ####</p>
<p># Author:- Tejal K Patel</p>
<h1>Parul University!!!</h1>
<button onclick="parul()">Enter</button>
<script>
function parul() {
var name=prompt("Please enter your name","Akash Shah");
if (name!=null && name!=""){
  document.write("Hello " + name + "! How are you today?");
  }
}
</script>
</body></html>
```





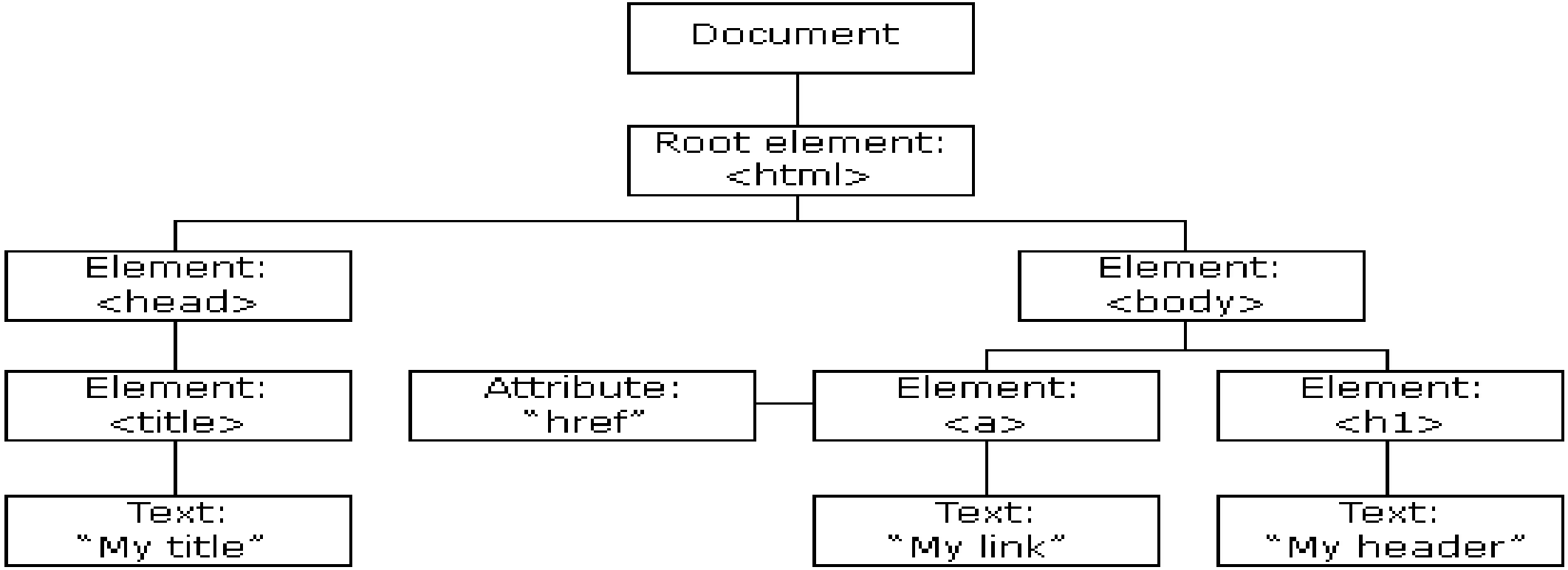


HTML DOM

The Document Object Model (DOM) is a programming API for HTML and XML documents.
It is a standard object model and programming interface for HTML.
It defines the logical structure of documents and the way a document is accessed and manipulated.
The browser creates a Document Object Model of the page when a web page is loaded.
The DOM model is constructed and represented as a tree of Objects.
The root node of the HTML document and the "owner" of all other nodes is the document object .
It provides properties and methods to access all node objects, from within JavaScript.
It is a standard for how to get, change, add, or delete HTML elements.

HTML DOM (Contd..)

The HTML DOM Tree of Objects:



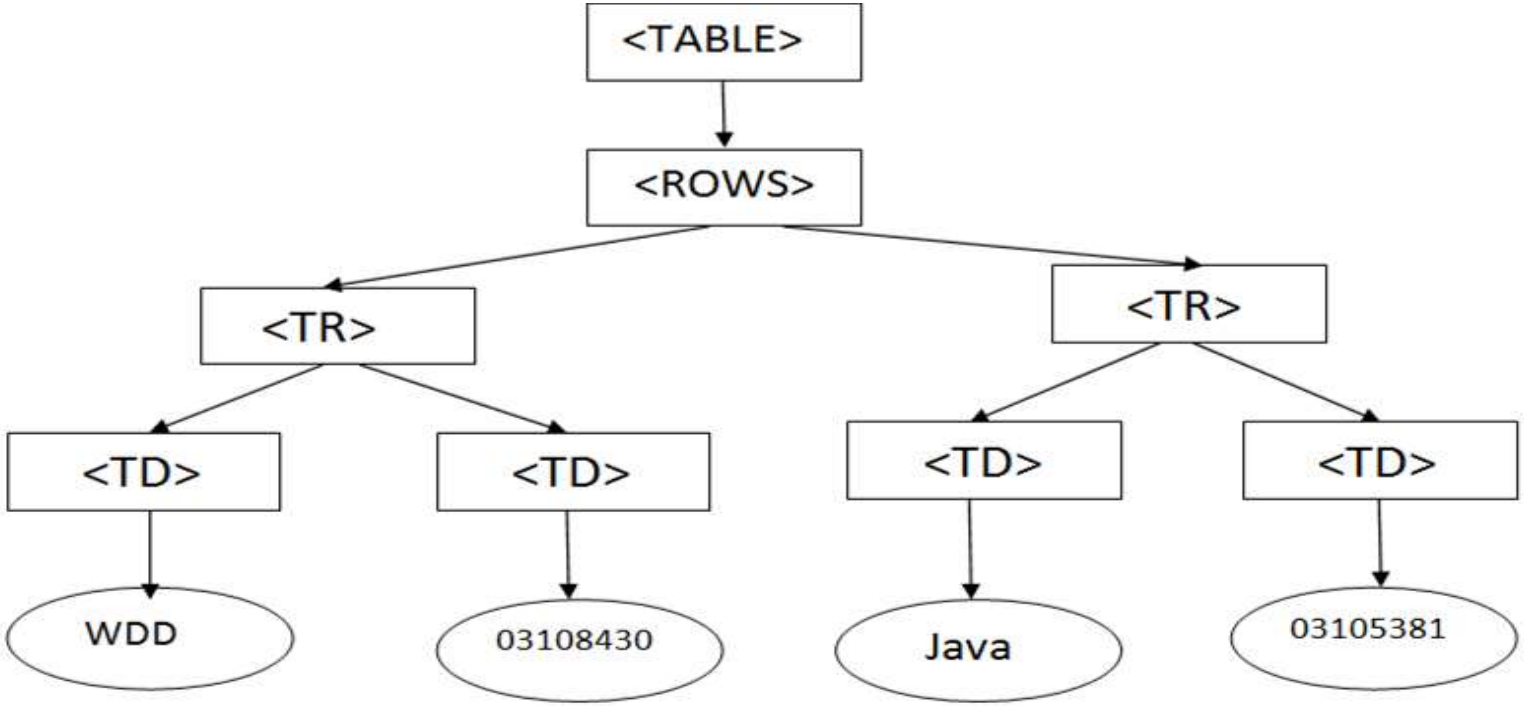
HTML DOM (Contd..)

Example:

```
<TABLE>
    <ROWS>
      <TR>
        <TD>WDD</TD>
        <TD>03108430</TD>
      </TR>
      <TR>
        <TD>Java</TD>
        <TD>03105381</TD>
      </TR>
    </ROWS>
</TABLE>
```

HTML DOM (Contd..)

The Document Object Model represents previous table like this:



DOM representation of the example table

HTML DOM (Contd..)

JavaScript gets extra powers due to the object model and then it has the following ability to do:

JS can change all the HTML attributes in the page
JS can change all the CSS styles in the page
JS can remove existing HTML elements and attributes
JS can add new HTML elements and attributes
JS can react to all existing HTML events in the page
JS can create new HTML events in the page
JS can change all the HTML elements in the page

HTML DOM (Contd..)

DOM Methods & properties:

Method:

A method is an task you can do –Ex. add or deleting an HTML element.

HTML DOM methods are actions you can perform on HTML Elements.

Property:

A property is a value that you can get or set-Ex. changing the content of an HTML element).

HTML DOM properties are values of HTML Elements that you can set or change

Example:

```
<body>
<p id="PIET"></p>
<script>
document.getElementById("PIET").innerHTML = "Hello students of PU!!";
</script>
</body>
```

HTML DOM (Contd..)

The following properties and methods can be used on HTML documents:

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()

HTML DOM (Contd..)

The following properties and methods can be used on HTML documents:

Property / Method	Description
document.cookie	Returns all name/value pairs of cookies in 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)

HTML DOM (Contd..)

The following properties and methods can be used on HTML documents:

Property / Method	Description
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

HTML DOM (Contd..)

The following properties and methods can be used on HTML documents:

Property / Method	Description
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

HTML DOM (Contd..)

The following properties and methods can be used on HTML documents:

Property / Method	Description
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

HTML DOM (Contd..)

The following properties and methods can be used on HTML documents:

Property / Method	Description
document.referrer	Returns the URL of the document that loaded the current document
document.removeEventListener()	Removes an event handler from the document.
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

The Browser Object Model

The Browser Object Model (BOM) gives JavaScript the capability to “interact" with the browser.

All modern browsers have implemented almost the same methods and properties for JavaScript thus it is often refers to, as methods and properties of the BOM.

1) The Window Object:

The window object represent the browser's window it is supported by all browsers

All global JavaScript objects, functions, and variables automatically comes under the window object.

Even the document object of the HTML DOM is a property of the window object:

```
window.document.getElementById(“parul");
```

is the same as:

```
document.getElementById(“parul");
```

The Browser Object Model (Contd..)

2. Window Size: To find the size of the browser window.

For Internet Explorer, Chrome, Firefox, Opera, and Safari:

window.innerHeight - the inner height of the browser window

window.innerWidth - the inner width of the browser window

For Internet Explorer 8, 7, 6, 5:

document.documentElement.clientHeight

document.documentElement.clientWidth

or

document.body.clientHeight

document.body.clientWidth

The Browser Object Model (Conti..)

Example: To displays the browser window's height and width

```
<html><body>
<p>#### WDD Programs ####</p>
<p># Author:- Tejal K Patel</p>
<h1>Parul University!!!</h1>
<button onclick="parul()">Size</button>
<p id="PIET"></p>
<script>
var x = window.innerWidth
|| document.documentElement.clientWidth
|| document.body.clientWidth;
var y = window.innerHeight
|| document.documentElement.clientHeight
|| document.body.clientHeight;
var size = document.getElementById("PIET");
size.innerHTML = "Browser inner window width: " + x + ", height: " + y + ".";
</script>
</body><html>
```

The Browser Object Model (Contd..)

3.Window Methods

window.open() - open a new window

window.close() - close the current window

window.moveTo() -move the current window

window.resizeTo() -resize the current window

The Browser Object Model (Contd..)

4.Window Screen:

The window.screen object contains information about the user's screen.

The window.screen object can be written without the window prefix.

Properties:

screen.colorDepth
screen.pixelDepth
screen.width
screen.height
screen.availWidth
screen.availHeight

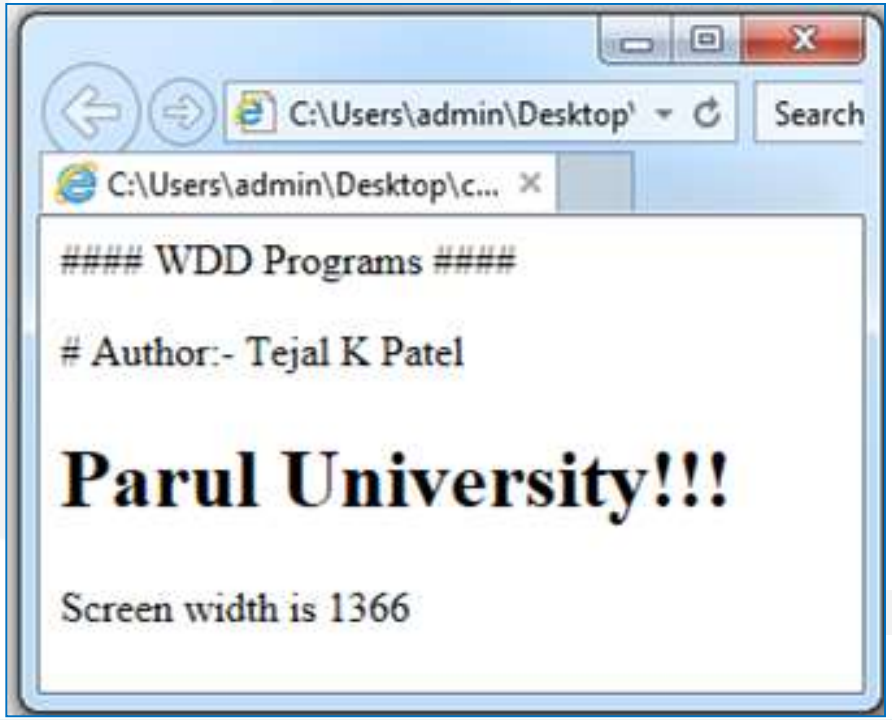
Example:

The screen.width property returns the width of the visitor's screen in pixels.

The Browser Object Model (Contd..)

Example:

```
<html><body>
<p>#### WDD Programs ####</p>
<p># Author:- Tejal K Patel</p>
<h1>Parul University!!!</h1>
<p id="PIET"></p>
<script>
document.getElementById("PIET").innerHTML =
"Screen width is " + screen.width;

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



The Browser Object Model (Contd..)

5. Window Location

The window.location object can be used to get the current page address (URL) and to redirect the browser to a new page.
It can be written without the window prefix.

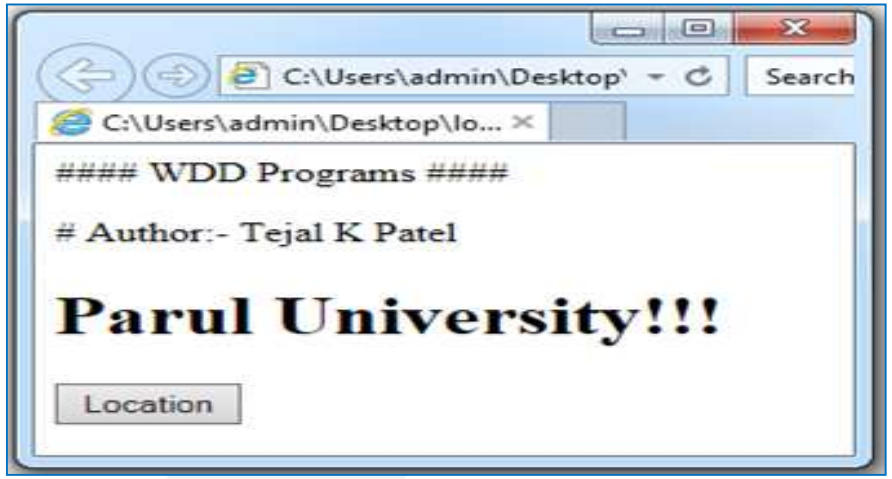
Some examples:
window.location.href returns the href (URL) of the current page
window.location.hostname returns the domain name of the web host
window.location.pathname returns the path and filename of the current page
window.location.protocol returns the web protocol used (http:// or https://)
window.location.assign loads a new document

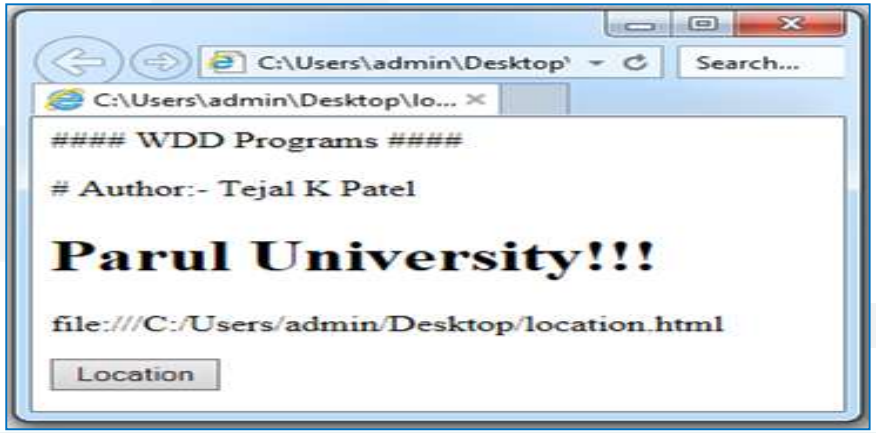
Example:
The window.location.href property returns the URL of the current page.

The Browser Object Model (Contd..)

Example:

```
<html><body>
<p>#### WDD Programs ####</p>
<p># Author:- Tejal K Patel</p>
<h1>Parul University!!!</h1><p id="PIET"></p>
<button onclick="parul()">Location</button>
<script>
function parul(a,b) {
document.getElementById("PIET").innerHTML = window.location.href;
}
</script>
</body></html>
```





The Browser Object Model (Contd..)

6.Window History:

The window.history object contains the browsers history.
It can be written without the window prefix.
To protect the privacy of the users, there are limitations to how JavaScript can access this object.

Methods:

history.back() - same as clicking back in the browser
history.forward() - same as clicking forward in the browser

Example:

The history.back() method loads the previous URL in the history list.
This is the same as clicking the Back button in the browser.
We can Create a back button on a page.

The Browser Object Model (Contd..)

7.Window Navigator

The window.navigator object contains information about the visitor's browser.
It can be written without the window prefix.

Examples:

navigator.appName :return the name of browser
navigator.appCodeName :return the name of browser
navigator.product : returns the engine name of the browser
navigator.appVersion: returns version information about the browser
navigator.platform : returns the browser platform (operating system)
navigator.language : returns the browser's language

The Browser Object Model (Contd..)

Event Handling :

Functions that manage events are called event handlers.

Event handlers can be used to manage, and verify, user input, user actions, and browser actions.

HTML DOM events allow JavaScript to work with different event handlers on elements in an HTML document.

An HTML event can be initiated by the browser operation, or user action.

Events are normally used in combination with functions, and the function will not be executed before the event occurs.

A JavaScript may get executed when an event occurs, like when a user clicks on an HTML element.

To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

Example:

onclick=*JavaScript*

The Browser Object Model (Contd..)

HTML events:

User clicks the mouse

Once a web page has loaded

Once an image has been loaded

Once the mouse moves over an element

When an input field is changed

When an HTML form is submitted

When a user strokes a key

Example: The content of the <h1> element is changed when a user clicks on it:

```
<html><body>
<h1 onclick="this.innerHTML='parul!'">Clickhere!</h1>
</body></html>
```

The Browser Object Model (Contd..)

Common HTML Events are:

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

The Browser Object Model (Contd..)

HTML events:

Media Events: Events triggered by medias like videos, images and audio applies to all HTML elements, but is most common in media elements, like <audio>, <embed>, <img>, <object>, and <video>.

Form Events: Events triggered by actions inside a HTML form applies to almost all HTML elements, but is most used in form elements like onsubmmit(), onchange()

Keyboard Events: onkeydown(), onkeypress()

Mouse Events: Events triggered by a mouse, or similar user actions like onmouseover(), onmousout()

Window Event Attributes: Events triggered for the window object (applies to the <body> tag) like onload(), onunload()

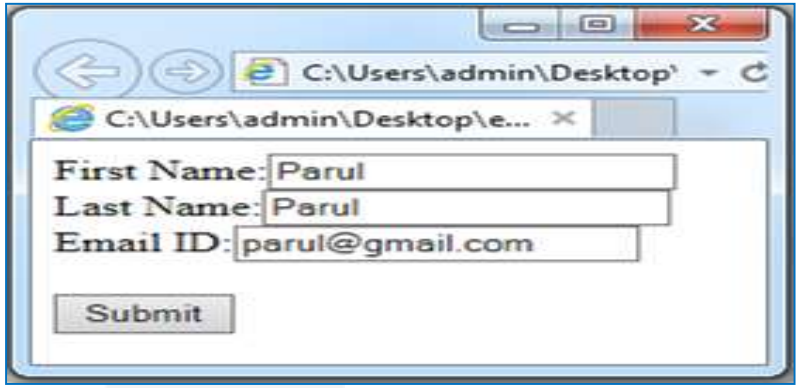
Clipboard Events: oncopy(), oncut()

Misc Events: onerror(), onshow()

The Browser Object Model (Contd..)

Example: <body>

```
<form onsubmit="alert('Are you autherized user of parul university only the submit data!')">
First Name:<input type="text" name="fname">
<br>
Last Name:<input type="text" name="lname">
<br>
Email ID:<input type="text" name="email">
<br>
<br>
<input type="submit" value="Submit">
</form>
</body>
```



Form Validation

JavaScript provide validation to validate form's data on the client's side.

HTML form validation can be done through a JavaScript.

If a form field is empty, this code will give a alert message, and returns false, to prevent the form from being submitted:

```
function validateForm()
 {
   var parul = document.forms["Formforpiet"]["fname"].value;
   if (parul == null || parul == "") {
     alert("Empty Feilds");
   return false;
    }
}
```

Form Validation (Contd..)

Form validation generally performs two functions.

1. Basic Validation - The form must be checked to make sure data was entered into each form field that required it.
This would need just loop through each field in the form and check for data.

2.Data Format Validation - The data that is entered must be checked for correct form and value.
This would need to put more logic to test correctness of data.

Form Validation (Contd..)

Example:<head>

```
<script >
    <!–   // Form validation code will come here.
 function validate()
 {
   if( document.myForm.Name.value == "" )
      {
          alert( "Please provide your name!" );
          document.myForm.Name.focus() ;
          return false;
      }
    if( document.myForm.EMail.value == "" )
      {
           alert( "Please provide your Email!" );
           document.myForm.EMail.focus() ;
          return false;
      }
```

Form Validation (Contd..)

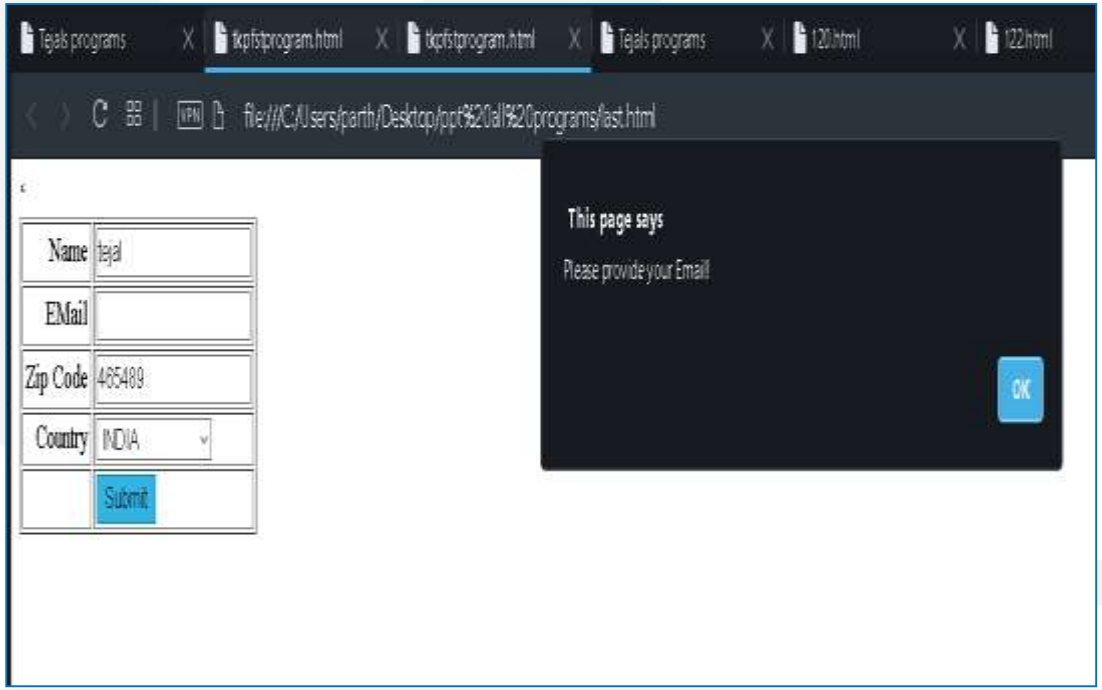
```
if( document.myForm.Country.value == "-1" )
    {
      alert( "Please provide your country!" );
      return false;
    }
   return( true );
  } //-->
 </script>
</head>
```

Form Validation (Contd..)

```
<body>
 <form action="/cgi-bin/test.cgi" name="myForm"
onsubmit="return(validate());">
 <table cellspacing="2" cellpadding="2" border="1">
 <tr>
   <td align="right">Name</td>
   <td><input type="text" name="Name" /></td>
 </tr>
 <tr>
   <td align="right">EMail</td>
   <td><input type="text" name="EMail" /></td>
 </tr>
```

Form Validation (Contd..)

```
<tr>
 <td align="right">Country</td>
 <td>
 <select name="Country">
  <option value="-1" selected>[choose yours]</option>
  <option value="1">USA</option>
  <option value="2">UK</option>
  <option value="3">INDIA</option>
 </select>
 </td>
 </tr>
 <tr>
 <td align="right"></td>
 <td><input type="submit" value="Submit" /></td>
 </tr>
 </table>
 </form></body></html>
```



Form Validation (Contd..)

2) Data Format Validation:

Now we will see how we can validate/check our entered form data before submitting it to the web server.

This example shows how to validate an entered email address which means email address must contain at least an @ sign and a dot (.). Also, the @ must not be the first character of the email address, and the last dot must at least be one character after the @ sign:

Form Validation (Contd..)

```
<script > <!—
 function validateEmail()
 {
 var emailID = document.myForm.EMail.value;
atpos = emailID.indexOf("@");
dotpos = emailID.lastIndexOf(".");
if (atpos < 1 || ( dotpos - atpos < 2 ))
{
 alert("Please enter correct email ID");
 document.myForm.EMail.focus() ;
 return false;
}
return( true );
} //-->  </script>
```

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in