

**Web Programming (203105353)**

**Prof. Mosam Patel,** Assistant Professor,

Department of Computer Science and Engineering



**CHAPTER-3JAVA 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.

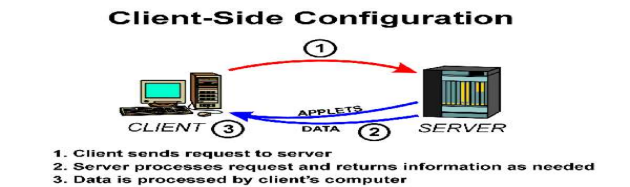
2. The server side/back-end. - To the server that hosts the website

**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.

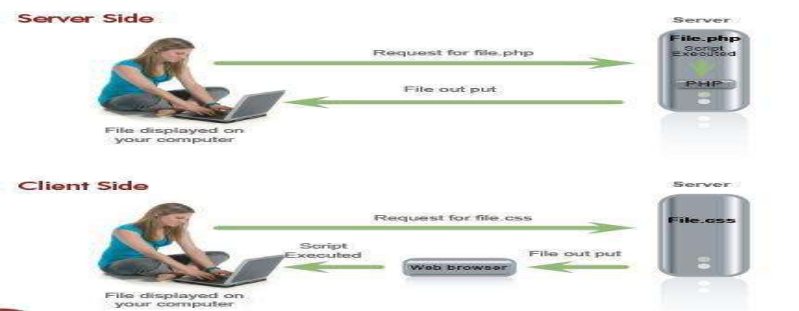
Server side languages helps in making web page dynamic as well as interactive to the user. Server side scripts are only produced by the server side and it does not produce client side scripts. 

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 t~~here 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>

<bod><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"; }**

**</s~~cript>~~**

</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.

HTM~~L 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;

v~~ar 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

do~~cument.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**  break | **Description**  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**  do ... while | **Description**  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. V~~ariables 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>

<bu~~tton 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>

<h~~tml>~~

**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;**

**</s~~cript>~~**

</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>**

**co~~nsole.log(“Parul University!!!”);~~**

**</script>**

<bod><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)

{

m~~essage= “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"; }

doc~~ument.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>";

}

do~~cument.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};

Yo~~u 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**  concat() | **Description**  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**  reverse() | **Description**  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**  constructor | **Description**  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);

}

do~~cument.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**