WSA | Forward looking IT finishing school

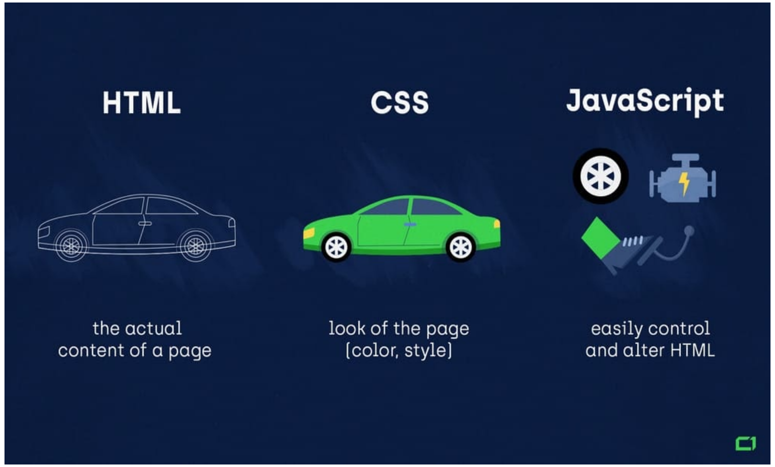
Introduction

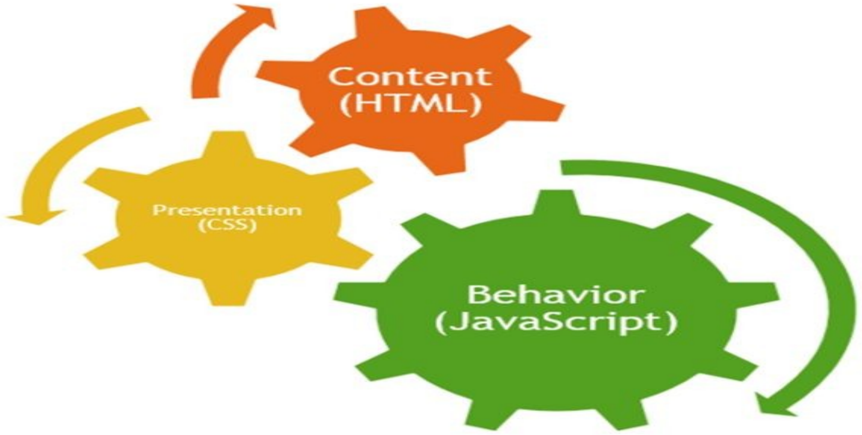
JavaScript

Table of Content

- Introduction to JavaScript
- Data Types
- Arrays
- Objects
- Statements
- Function basics

HTML, CSS & JavaScript


HTML
CSS
JavaScript
the actual content of a page
look of the page (color, style)
easily control and alter HTML



JS – What?

- A programming language designed for web pages
- It is embedded directly into HTML Pages
- An interpreted, client-side, event based language
- It is dynamic, lightweight and case-sensitive

JS - Internal

```
<!DOCTYPE html>
<html>
     <head>
            <script> . . . JS code . . . </script>
      </head>
      <body>
            <script> . . . JS code . . . </script>
      </body>
</html>
```

JavaScript - Output

Method	Description
window.alert()	Display data in alert dialog box
document.write()	Display data in browser display area
innerHTML()	Display data in HTML element
console.log()	Display data in browser console

JS – First Program

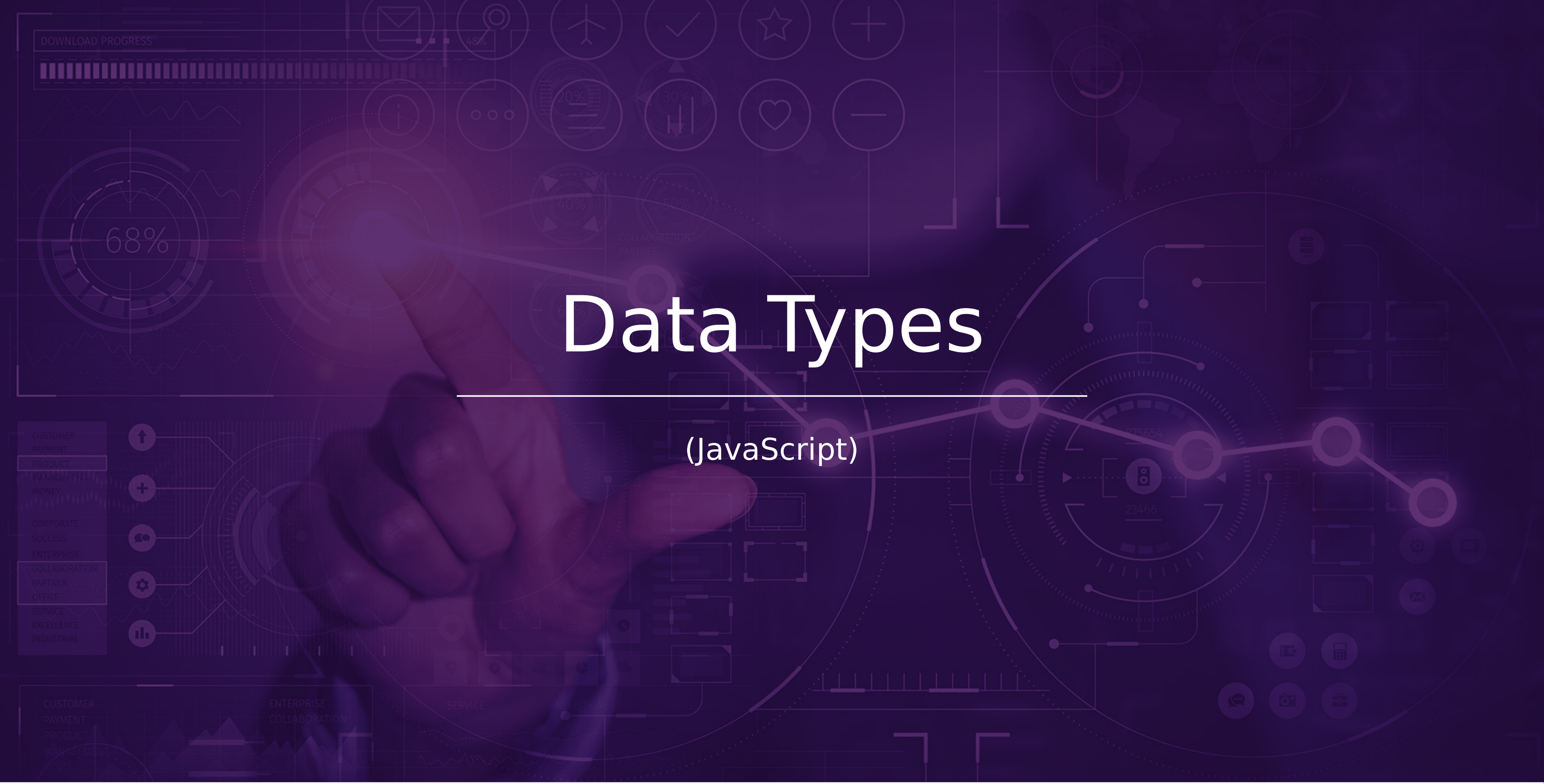
```
<body>
<script>
        document.write("Hello JavaScript!!");
</script>
</body>
```

JS - External

- External (or linked script) JavaScript can be inserted using src attribute

Syntax :

```
<script src="URL"> </script>

Absolute URL : http://www.example.com/example.js
Relative URL : /script/example.js
```



Data Types

(JavaScript)

JS - Variables

- Variables are container to store values
- Name must start with
  - A letter (a to z or A to Z)
  - Underscore(_)
  - Or dollar($) sign
- After first letter we can use digits (0 to 9)
  - Example: x1, y2, ball25, a2b

JS - Variables

- JavaScript variables are case sensitive, for example ‘sum’ and ‘Sum’ and ‘SUM’ are different variables

Example :

```
var x = 6;
var y = 7;
var z = x+y;
```

JS - Data Types

- There are two types of Data Types in JavaScript
  - Primitive data type
  - Non-primitive (reference) data type

Note : JavaScript is weakly typed. Every JavaScript variable has a data type , that type can change dynamically

Primitive data type

- String
- Number
- Boolean
- Null
- Undefined

Primitive data type (String)

- String - A series of characters enclosed in quotation marks either single quotation marks (') or double quotation marks (")

```
Example :
var name = “Webstack Academy”;
var name = 'Webstack Academy';
```

Primitive data type (Number)

- All numbers are represented in IEEE 754-1985 double precision floating point format (64 bit)
- All integers can be represented in -2^{53} to $+2^{53}$ range
- Largest floating point magnitude can be ± 1.7976×10^{308}
- Smallest floating point magnitude can be ± 2.2250×10^{-308}
- If number exceeds the floating point range, its value will be infinite

Primitive data type
(Number conversion)

- Converting from string

```
Example :
 var num1 = 0, num2 = 0;

 // converting string to number
 num1 = Number("35");
 num2 = Number.parseInt(“237”);
```

Primitive data type
(Number conversion)

- Converting to string

```
Example :
 var str = “”; // Empty string
 var num1 = 125;

 // converting number to string
 str = num1.toString();
```

Primitive data type
(Number – special values)

Special Value	Cause	Comparison
Infinity, -Infinity	Number too large or too small to represent	All infinity values compare equal to each other
NaN (not-a-number)	Undefined operation	NaN never compare equal to anything (even itself)

Primitive data type
(Boolean)

- Boolean data type is a logical true or false

```
Example :
var ans = true;
```

Primitive data type
(Null)

- In JavaScript the data type of null is an object
- The null means empty value or nothing

```
Example :
var num = null; // value is null but still type is an object.
```

Primitive data type
(Undefined)

- A variable without a value is undefined
- The type is object

Example :

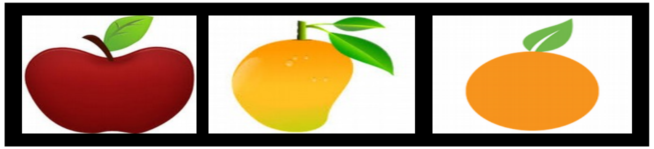
```
var num;    // undefined
```

Non-primitive data types

- Array
- Object

Arrays

- Array represents group of similar values
- Array items are separated by commas
- Array can be declared as :
  - var fruits = [“Apple” , “Mango”, “Orange”];



Objects

- An Object is logically a collection of properties
- Objects represents instance through which we can access members
- Object properties are written as name:value pairs separated by comma

Example :

```
var student={ Name:“Mac”, City:“Banglore”, State:“Karnataka”};
```

Statements

(JavaScript)

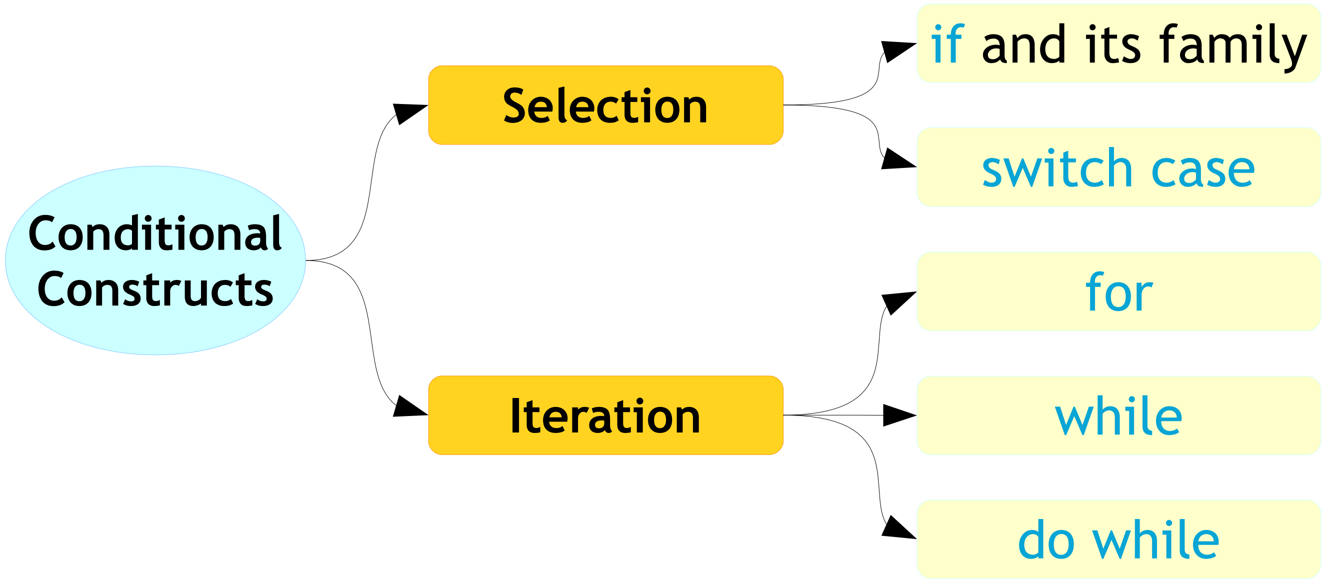
JS – Simple Statements

- In JavaScript statements are instructions to be executed by web browser (JavaScript core)

Example :

```
<script>
    var y = 4, z = 7; // statement
    var x = y + z;    // statement

    document.write("x = " + x);
</script>
```

JS – Conditional Construct



JS – Statements
(conditional - if)

```
Syntax :

if (condition) {

    statement(s);

}
```

```
Example :

<script>

var num = 2;
if (num < 5) {
    document.write("num < 5");
}

</script>
```

JS – Statements
(conditional : if-else)

```
Syntax :
if (condition) {
    statement(s);
}
else {
    statement(s);
}
```

JS – Statements
(conditional : if-else)

```
Example :
<script>
var num = 2;
if (num < 5) {
    document.write("num is smaller than 5");
}
else {
    document.write("num is greater than 5");
}
</script>
```

JS – Statements
(conditional : if-else if)

```
Syntax :
if (condition1) {
    statement(s);
}
else if (condition2) {
    statement(s);
}
else {
    statement(s);
}
```

JS – Statements
(conditional : if-else if)

```
Example :
<script>
var num = 2;
if (num < 5) {
    document.write("num is smaller than 5");
}
else if (num > 5) {
    document.write("num is greater than 5");
}
else {
    document.write("num is equal to 5");
}
</script>
```

JavaScript - Input

Method	Description
prompt()	It will asks the visitor to input Some information and stores the information in a variable
confirm()	Displays dialog box with two buttons ok and cancel

Example - prompt()

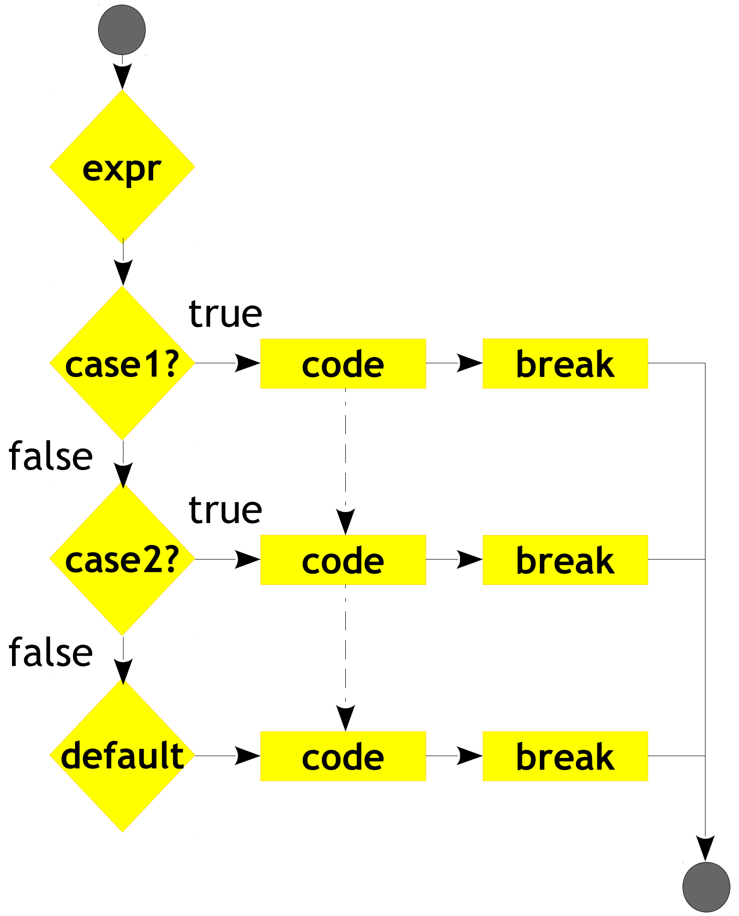
```
Example :

<script>

var person = prompt("Please enter your name", "");
if (person != null) {
    document.write("Hello " + person +
                   "! How are you today?");
}

</script>
```

Class Work

- WAP to find the max of two numbers
- WAP to print the grade for a given percentage
- WAP to find the greatest of given 3 numbers
- WAP to find the middle number (by value) of given 3 numbers

JS – Statements
(switch)

```
Syntax :

switch (expression) {
    case exp1:
        statement(s);
        break;
    case exp2:
        statement(s);
        break;
    default:
        statement(s);
}
```



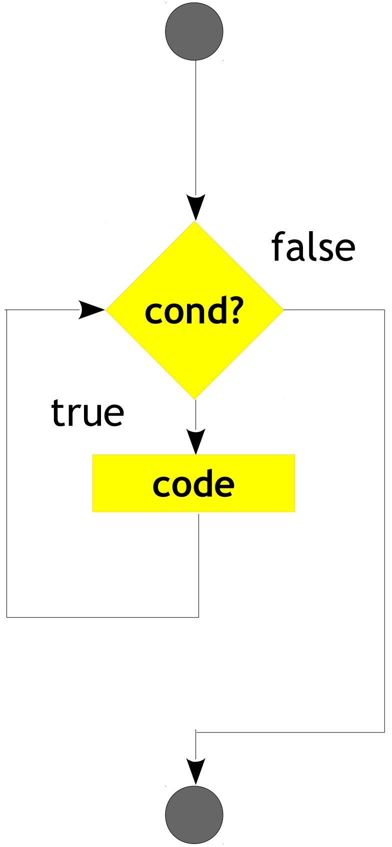
JS – Statements
(switch)

```
<script>
    var num = Number(prompt("Enter the number!", ""));

    switch(num) {
        case 10 : document.write("You have entered 10");
            break;
        case 20 : document.write("You have entered 20");
            break;
        default : document.write("Try again");
    }

</script>
```

JS – Statements
(while)

```
Syntax:

while (condition)
{
    statement(s);
}
```

- **Controls** the loop.
- Evaluated **before** each execution of loop body



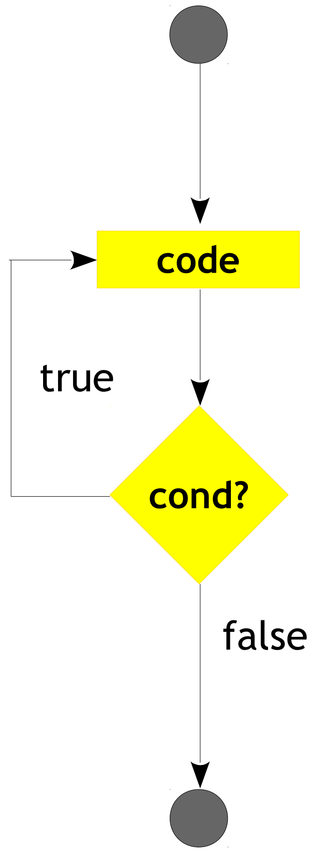
JS – Statements
(while)

```
Example:

<script>
    var iter = 0;

    while(iter < 5)
    {
        document.write("Looped " + iter + " times <br>");
        iter = iter + 1;
    }
</script>
```

JS – Statements
(do - while)

```
Syntax:

do {
    statement(s);
} while (condition);
```

- **Controls** the loop.
- Evaluated **after** each execution of loop body



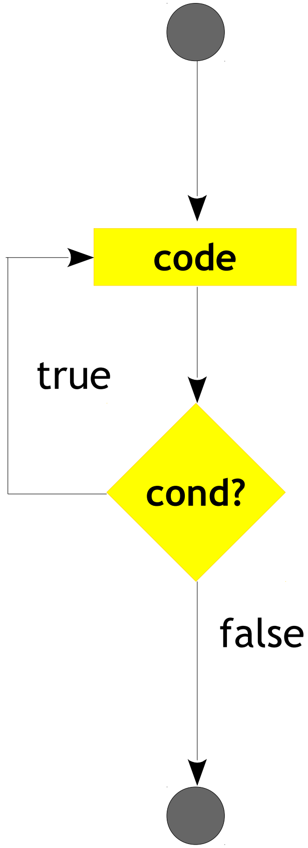
JS – Statements
(do-while)

```
Example:

<script>
    var iter = 0;

    do {
        document.write("Looped " + iter + " times <br>");
        iter = iter + 1;
    } while ( iter < 5 );

</script>
```

JS – Statements
(for loop)

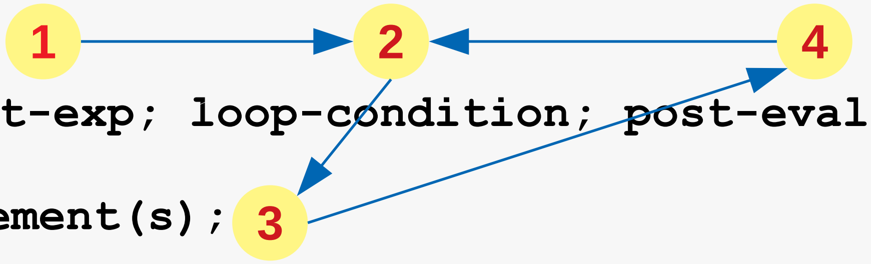
```
Syntax:

for (init-exp; loop-condition; post-eval-exp) {

    statement(s);

};
```

- **Controls** the loop.
- Evaluated **before** each execution of loop body



JS – Statements
(for loop)

```
Execution path:

for (init-exp; loop-condition; post-eval-exp) {

    statement(s);

};
```



JS – Statements
(for loop)

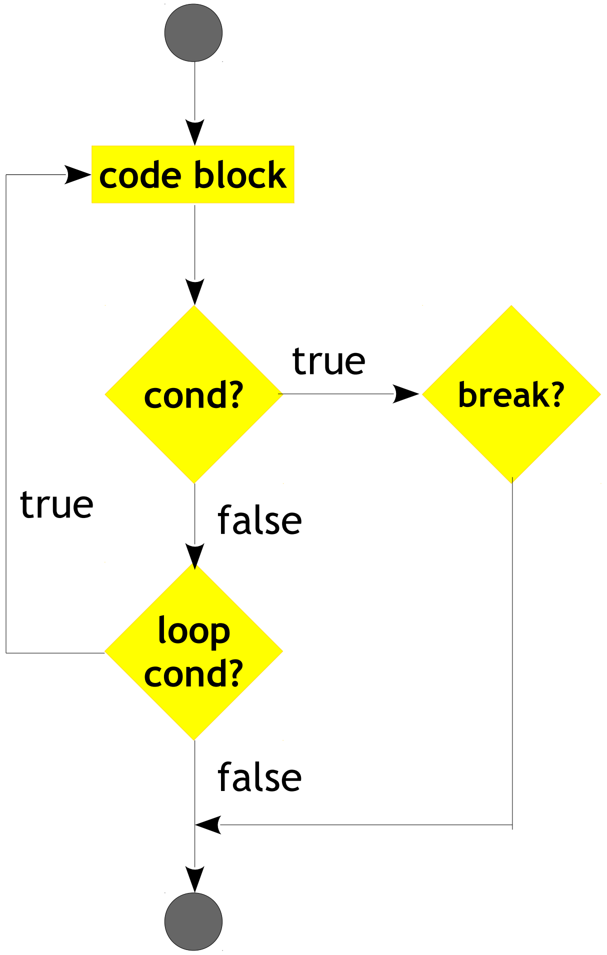
```
Example:


<script>
    for (var iter = 0; iter < 5; iter = iter + 1) {
        document.write("Looped " + iter + " times <br>");
    }
</script>
```

JS – Statements
(break)

- A break statement shall appear only in “switch body” or “loop body”
- “break” is used to exit the loop, the statements appearing after break in the loop will be skipped
- “break” without label exits/‘jumps out of’ containing loop
- “break” with label reference jumps out of any block

JS – Statements
(break)

```
Syntax:
while (condition) {
     conditional statement
       break;
}
```



JS – Statements
(break)

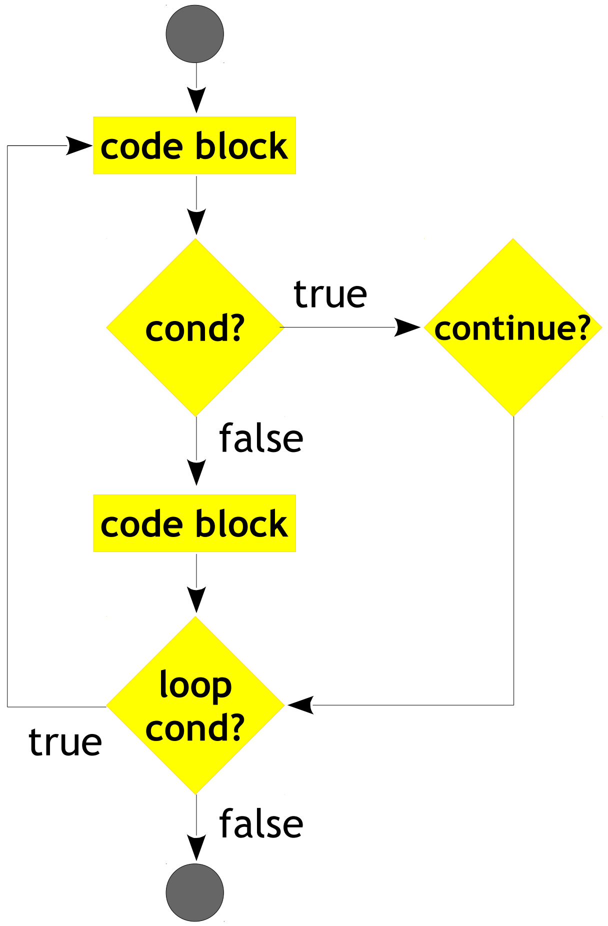
```
<script>
for (var iter = 0; iter < 10; iter = iter + 1) {
    if (iter == 5) {
        break;
    }
    document.write("<br>iter = " + iter);
}
</script>
```

JS – Statements
(continue)

- A continue statement causes a jump to the loop-continuation portion, that is, to the end of the loop body
- The execution of code appearing after the continue will be skipped
- Can be used in any type of multi iteration loop

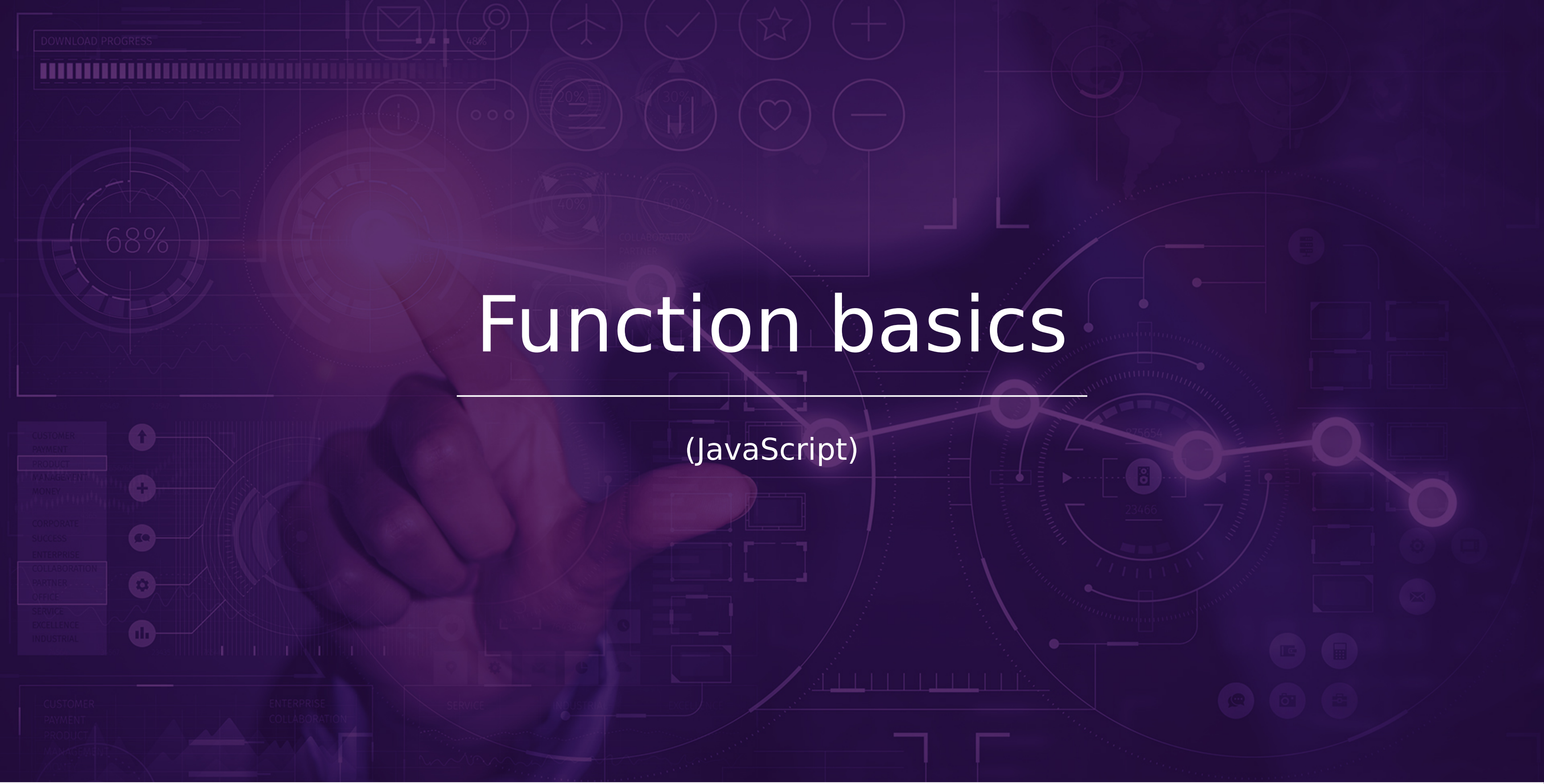
JS – Statements
(continue)

```
Syntax:
while (condition) {
    conditional statement
        continue;
}
```



JS – Statements
(continue)

```
<script>
for (var iter = 0; iter < 10; iter = iter + 1) {
    if (iter == 5) {
        continue;
    }
    document.write("<br>iter = " + iter);
}
</script>
```



Function basics

(JavaScript)

What is function?

- A function is a block of JavaScript code that is defined once but may be executed, or invoked, any number of times
- A function can be used to return a value, construct an object, or as a mechanism to simply run code
- JavaScript functions are defined with the function keyword
- Either function declaration or a function expression can be used

Function Declaration

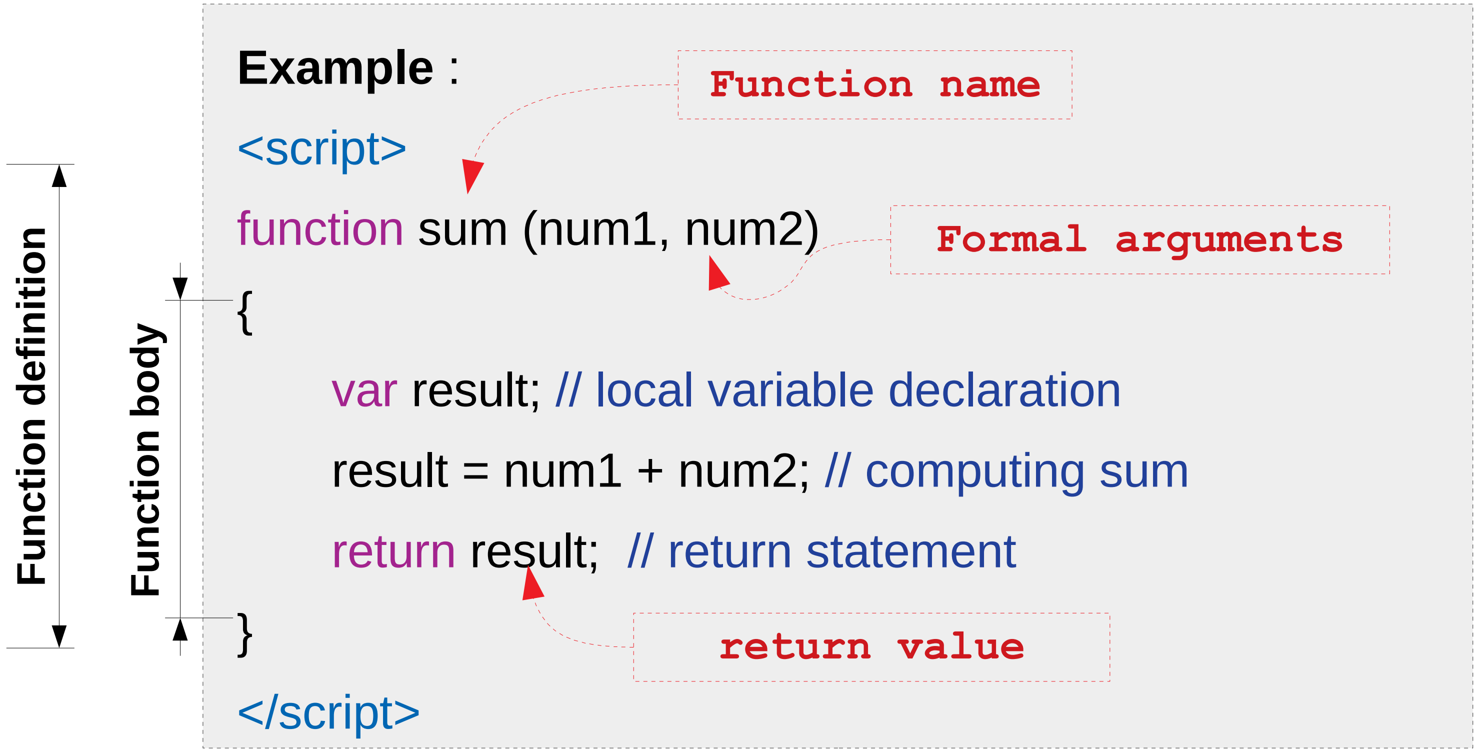
Syntax:

```
function functionName (param-1, param-2, . . . , param-n) {
  statement(s);
}
```

Parts of functions

- Name – A unique name given by developer
- Parameters / arguments – to pass on input values to function
- Body – A block of statement(s) to be executed
  - Local variable declaration
  - Flow of computing statement(s)
  - Return statement

Function Example



Example :

```
<script>
function sum (num1, num2)
{
        var result; // local variable declaration
        result = num1 + num2; // computing sum
        return result;  // return statement
}
</script>
```

Function Execution

- Merely defining a function does not result in execution of the function; it must be called for execution

```
<script>
      . . . function definition . . .
      var x = 3, y = 5, z;  // global variable declaration
                                   x and y are actual arguments
      z = sum (x, y);   // calling function for execution
      document.write("The sum of numbers is : " + z);
</script>
```

Actual Vs formal arguments

- Formal arguments are the names listed within parenthesis in function definition (also known as function parameters)
- Formal arguments are initialized through actual arguments at run time
- Actual arguments are variables or literals passed to the function at the time of invocation (call to execute)
- The formal arguments are visible to function only

Thank you

Web Stack Academy (P) Ltd

#83, Farah Towers,

1st floor,MG Road,

Bangalore – 560001

M: +91-80-4128 9576

T: +91-98862 69112

E: info@www.webstackacademy.com