INTRODUCTION TO SCRIPTING

Script

- What is Script?
- The term “scripting language” is also used loosely to refer to dynamic high-level general-purpose languages, such as JavaScript, VBScript, with the term “script”
- Scripting languages are becoming more popular due to the emergence of web based applications.

Advantages

- More Advantages of Scripting Languages:
- Easy to learn and use
- Minimum programming knowledge or experience required.
- Allow complex tasks to be performed in relatively few steps
- Allow simple creation and editing in a variety of text editors
- Allow the addition of dynamic and interactive activates to web pages.
- Editing and running code is fast.

Type of Scripting:

- Scripts are classified into the following two types:
- Client Side Scripts
- Server side Scripts

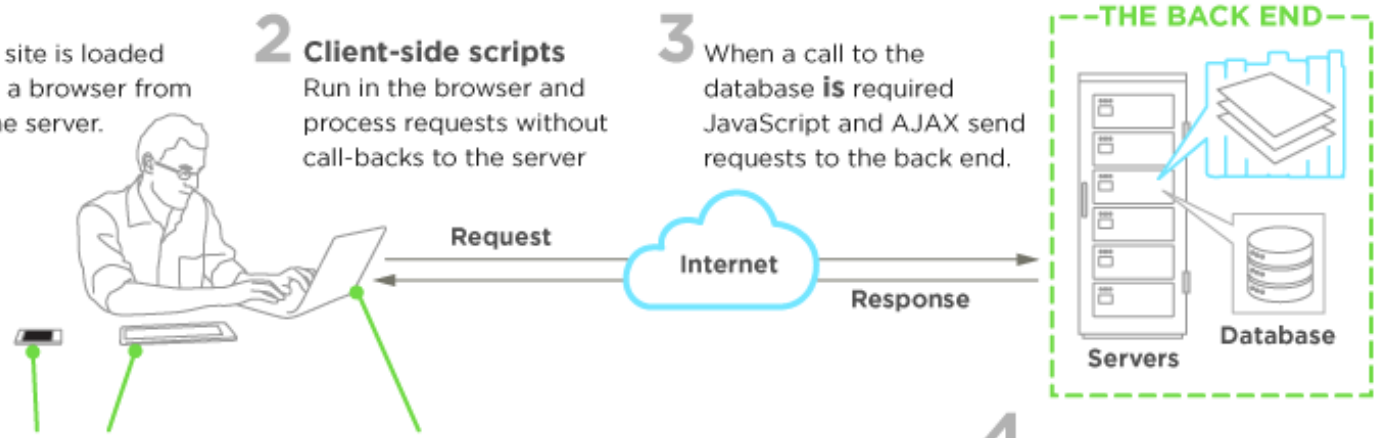
FRONT-END DEVELOPMENT

Upwork™

1 A site is loaded in a browser from the server.

2 **Client-side scripts** Run in the browser and process requests without call-backs to the server

3 When a call to the database **is** required JavaScript and AJAX send requests to the back end.



5 **Server-side scripts** process the data, then update the site—populating drop-down menus, loading products to a page, updating a user profile, and more.

Responsive front-end design allows a site to adapt to a user's device.

Everything a user sees in the browser is a mix of **HTML**, **CSS**, and **JavaScript**.

4 The **back-end server-side scripts** process the request, pull what they need from the database then send it back.

INTRODUCTION TO JAVASCRIPT

JAVASCRIPT

- JavaScript: An object oriented scripting language that designed primarily for people who are building web pages using HTML.
- JavaScript programs are embedded within HTML documents in source code form
- The script is platform independent and it is interpreted by the browser.

WHAT IS JAVASCRIPT?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language (a scripting language is a lightweight programming language)
- A JavaScript consists of lines of executable computer code
- A JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

How to Put a JavaScript Into an HTML Page?

```
<html>
<body>
    <script language = "javascript" type = "text/javascript">
        document.write("Hello World!");
    </script>
</body>
</html>
```

Cont..

- The script tag takes two important attributes –
- **Language** – This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML have phased out the use of this attribute.
- **Type** – This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

Ending Statements With a Semicolon?

- With traditional programming languages, like C++ and Java, each code statement has to end with a semicolon (;).
- Many programmers continue this habit when writing JavaScript, but in general, semicolons are **optional**! However, semicolons are required if you want to put more than one statement on a single line.

Example

Optional

```
<script language = "javascript"
type = "text/javascript">
   <!--
      var1 = 10
      var2 = 20
   //-->
</script>
```

Mandatory

```
<script language = "javascript"
type = "text/javascript">
   <!--
      var1 = 10; var2 = 20;
   //-->
</script>
```

Case Sensitivity

- JavaScript is a case-sensitive language.
- This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.
- So the identifiers ‘A’ and ‘a’ will convey different meanings in JavaScript.

Comments in JavaScript

- JavaScript supports both C-style and C++-style comments,
- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

Example

```
<script language = "javascript" type = "text/javascript">
  <!--
    // This is a comment. It is similar to comments in C++
    /*
    * This is a multi-line comment in JavaScript
    * It is very similar to comments in C Programming
    */
  //-->
</script>
```

JavaScript - Placement in HTML File

- There is a flexibility given to include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTML file are as follows −
  - *Script in <head>...</head> section.*
  - *Script in <body>...</body> section.*
  - *Script in <body>...</body> and <head>...</head> sections.*
  - *Script in an external file and then include in <head>...</head> section.*

JavaScript in External File

Example:

```
<html>
  <head>
    <script type = "text/javascript" src = "filename.js" > </script>
  </head>
  <body>
    .......
  </body>
</html>
```

JavaScript Variables

- In javascript variable declaration is implicit.
- Declare a JavaScript variable with the var keyword
- Syntax: var variable_name;
  - *var x = 5;*
  - *var y = 6;*
  - *var z = x + y;*

JavaScript Variable Scope

- The scope of a variable is the region of your program in which it is defined.
- **Global Variables** – A global variable has global scope which means it can be defined anywhere in your JavaScript code.
- **Local Variables** – A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

```
<html>
  <body onload = checkscope();>
    <script type = "text/javascript">
      <!--
        var myVar = "global";
// Declare a global variable
        function checkscope( ) {
          var myVar = "local";
// Declare a local variable
          document.write(myVar);
        }
      //-->
    </script>
  </body>
</html>
```

JavaScript Identifiers

- All JavaScript variables must be identified with unique names.
- These unique names are called identifiers.
- Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).
- The general rules for constructing names for variables (unique identifiers) are:
  - *Names can contain letters, digits, underscores, and dollar signs.*
  - *Names must begin with a letter*
  - *Names can also begin with $ and _ (but we will not use it in this tutorial)*
  - *Names are case sensitive (y and Y are different variables)*
  - *Reserved words (like JavaScript keywords) cannot be used as names*

Data Types

- Primitive Data Types
  - *Numbers*
  - *Strings*
  - *Boolean (True, False)*
- Composite Data Types
  - *Arrays*
  - *Objects*

JavaScript Datatypes

- One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.
- JavaScript allows you to work with three primitive data types –
  - *Numbers, eg. 123, 120.50 etc.*
  - *Strings of text e.g. "This text string" etc.*
  - *Boolean e.g. true or false.*
- JavaScript also defines two trivial data types,
  - *Null* - something that doesn't exist. In JavaScript, the data type of null is an object.
  - *Undefined* - variable without a value
- In addition to these primitive data types, JavaScript supports a composite data type known as **object**.

JavaScript Types are Dynamic

- JavaScript has dynamic types. This means that the same variable can be used to hold different data types
- ```
  var x;           // Now x is undefined
  x = 5;           // Now x is a Number
  x = "John";      // Now x is a String
  ```
  ```

JavaScript Strings

- A string (or a text string) is a series of characters
- Strings are written with quotes.
- You can use single or double quotes
- var carName1 = “MIT"; // Using double quotes
  var carName2 = ’MIT'; // Using single quotes

JavaScript Numbers

- JavaScript has only one type of numbers.
- Numbers can be written with, or without decimals.
- var x1 = 34.00; // Written with decimals
  var x2 = 34; // Written without decimals

JavaScript Booleans

- Booleans can only have two values: true or false.
- ```
  var x = 5;
  var y = 5;
  var z = 6;
  (x == y)       // Returns true
  (x == z)       // Returns false
  ```
  ```

JavaScript Arrays

- JavaScript arrays are written with square brackets.
- Array items are separated by commas.
- The following code declares (creates) an array called cars, containing three items (car names):
- var cars = [“Honda", "Volvo", “Ford"];
- Array indexes are zero-based, which means the first item is [0], second is [1], and so on.

Arrays

- An array is a compound data type that stores numbered pieces of data
- Each numbered datum is called an *element* of the array and the number assigned to it is called an *index*.
- The elements of an array may be of any type. A single array can even store elements of different type.

Creating an Array

- There are several different ways to create an array in JavaScript
- Using the `Array()` constructor:
  - *var a = new Array(1, 2, 3, 4, 5);*
  - *var b = new Array(10);*
- Using array literals:
  - var c = [1, 2, 3, 4, 5];

Accessing Array Elements

- Array elements are accessed using the [ ] operator
- Example:
  - *var colors = [“red”, “green”, “blue”];*
  - *colors[0] => red*
  - *colors[1] => green*

Adding Elements

- To add a new element to an array, simply assign a value to it
- Example:

```
var a = new Array(10);
a[50] = 17;
```

Array Length

- All arrays created in JavaScript have a special length property that specifies how many elements the array contains
- Example:
  - *var colors = [“red”, “green”, “blue”];*
  - *colors.length => 3*

Statements

- In a programming language, these programming instructions are called **statements**.
- A **JavaScript program** is a list of programming **statements**.
- A statement is a section of JavaScript that can be evaluated by a Web browser
- A script is simply a collection of statements
- JavaScript statements are composed of: Values, Operators, Expressions, Keywords, and Comments.

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Statements</h2>
<p>In HTML, JavaScript statements are executed by the browser.</p>
<p id="demo"></p>
    <script>
    document.getElementById("demo").innerHTML = "Hello World.";
    </script>
</body>
</html>
```

This statement tells the browser to write "Hello World." inside an HTML element with id="demo":

JavaScript Operators

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 y=2 x+y	4
-	Subtraction	x=5 y=2 x-y	3
*	Multiplication	x=5 y=4 x*y	20
/	Division	15/5 5/2	3 2,5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Assignment Operator

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Compare Operator

Operator	Description	Example
==	is equal to	5==8 returns false
===	is equal to (checks for both value and type)	x=5 y="5" x==y returns true x===y returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operator

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

The typeof Operator

- You can use the JavaScript typeof operator to find the type of a JavaScript variable.
- The typeof operator returns the type of a variable or an expression:

- ```
  typeof ""             // Returns "string"
  typeof "John"         // Returns "string"
  typeof "John Doe"     // Returns "string"
  ```
  ```

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
    <script>
    document.getElementById("demo").innerHTML =
    typeof "" + "<br>" +
    typeof "John" + "<br>" +
    typeof 24;
    </script>
</body>
</html>
```

Escape sequences

- \b Backspace
- \f Form Feed
- \n New Line
- \r Carriage Return
- \t Horizontal Tabulator
- \v Vertical Tabulator
- \' Single quote
- \" Double quote
- \\ Backslash

Control Structures

- There are three basic types of control structures in JavaScript: the `if` statement, the `while` loop, and the `for` loop
- Each control structure manipulates a block of JavaScript expressions beginning with { and ending with }

JavaScript - if...else Statement

- JavaScript supports the following forms of
  - *if statement*
  - *if...else statement*
  - *if...else if... statement.*

if statement

```
if (expression) {
   Statement(s) to be executed if expression is true
}
```

If... else... statement

```
if (expression) {
   Statement(s) to be executed if expression is true
} else {
   Statement(s) to be executed if expression is false
}
```

if...else if... statement

```
if (expression 1) {
   Statement(s) to be executed if expression 1 is true
} else if (expression 2) {
   Statement(s) to be executed if expression 2 is true
} else {
   Statement(s) to be executed if no expression is true
}
```

Examples

```
<script type = "text/javascript">
        var age = 15;
if( age > 18 ) {
            document.write("<b>Qualifies for driving</b>");
        }
else {
            document.write("<b>Does not qualify for driving</b>");
        }
</script>
```

```
<script type = "text/javascript">
        var book = "maths";
         if( book == "history" ) {
            document.write("<b>History Book</b>");
         } else if( book == "maths" ) {
            document.write("<b>Maths Book</b>");
         } else if( book == "economics" ) {
            document.write("<b>Economics Book</b>");
         } else {
            document.write("<b>Unknown Book</b>");
         }
</script>
```

Repeat Loops

- A repeat loop is a group of statements that is repeated until a specified condition is met
- Repeat loops are very powerful programming tools; They allow for more efficient program design and are ideally suited for working with arrays

The While Loop

- The while loop is used to execute a block of code while a certain **condition** is true

```
count = 0;
while (count <= 10) {
    document.write(count);
    count++;
}
```

The For Loop

- The for loop is used when there is a need to have a **counter** of some kind
- The counter is initialized before the loop starts, tested after each iteration to see if it is below a target value, and finally updated at the end of the loop

Example: For Loop

```
// Print the numbers 1
     through 10

for (i=1; i<= 10; i++)
     document.write(i);
```

i=1 initializes the counter

i<=10 is the target value

i++ updates the counter at the end of the loop

Switch Case

```
switch (expression) {
   case condition 1: statement(s)
   break;
   case condition 2: statement(s)
   break;
   ...
   case condition n: statement(s)
   break;
   default: statement(s)
}
```

JavaScript - Functions

- A function is a group of reusable code which can be called anywhere in your program.

Function Definition

- *The most common way to define a function in JavaScript is by using the function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.*

Syntax:

```
<script type = "text/javascript">
function functionname(parameter-list) {
    statements
}
</script>
```

Example

```
<script type = "text/javascript">
function sayHello() {
        alert("Hello there");
    }
</script>
```

Calling a Function

To invoke a function somewhere later in the script, you would simply need to write the name of that function.

```
<html>
  <head>
    <script type = "text/javascript">
      function sayHello() {
        document.write ("Hello there!");
      }
    </script>
  </head>

  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Say Hello">
    </form>
    <p>Use different text in write method and then try...</p>
  </body>
</html>
```

Function Parameters

- Till now, we have seen functions without parameters.
- We can pass different parameters while calling a function.
- These passed parameters can be captured inside the function and any manipulation can be done over those parameters.
- A function can take multiple parameters separated by comma.

```
<html>
  <head>
    <script type = "text/javascript">
      function sayHello(name, age) {
        document.write (name + " is " + age + " years old.");
      }
    </script>
  </head>

  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello('ABC', 7)" value = "Say
Hello">
    </form>
    <p>Use different parameters inside the function and then try...</p>
  </body>
</html>
```

return Statement

- A JavaScript function can have an optional return statement.
- This is required if you want to return a value from a function.
- This statement should be the last statement in a function.
- For example, you can pass two numbers in a function and then you can expect the function to return their multiplication in your calling program.

```
<html>
  <head>
    <script type = "text/javascript">
      function concatenate(first, last) {
        var full;
        full = first + last;
        return full;
      }
      function secondFunction() {
        var result;
        result = concatenate('Zara', 'Ali');
        document.write (result );
      }
    </script>
  </head>
  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "secondFunction()" value = "Call Function">
    </form>
    <p>Use different parameters inside the function and then try...</p>
  </body>
</html>
```

JavaScript - Events

- JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.
- When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.
- Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

```
<html>
  <head>
    <script type = "text/javascript">
      function sayHello() {
          alert("Hello World")
        }
     </script>
  </head>

  <body>
    <p>Click the following button and see result</p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Say
Hello" />
    </form>
  </body>
</html>
```

Events

- onclick - This is the most frequently used event type which occurs when a user clicks the left button of his mouse.
- onsubmit: is an event that occurs when you try to submit a form. You can put your form validation against this event type.
- onmouseover and onmouseout: onmouseover event triggers when you bring your mouse over any element and the onmouseout triggers when you move your mouse out from that element.

JavaScript Basic Examples

```
<script>
document.write("Hello World!")
</script>
```
⇒ format text with HTML code - heading

```
<script>
alert("Hello World!")
</script>
```

Example

```
<script>
x="Hello World!"
document.write(x)
</script>

<script>
x="Manipal"
document.write("Welcome to" +x)
</script>
```

⇒ use line break html code

JavaScript - Dialog Boxes

- JavaScript supports three important types of dialog boxes. These dialog boxes can be used to raise and alert, or to get confirmation on any input or to have a kind of input from the users.

Cont..

- Alert Box
  - *An alert box is often used if you want to make sure information comes through to the user.*
  - *When an alert box pops up, the user will have to click "OK" to proceed.*

```
<script>

alert("Hello World!")

</script>
```

Cont..

- Confirm Box
  - *A confirm box is often 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.*

Cont..

- Prompt Box
  - *A prompt box is often used if you want the user to input a value 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.*

Prompt Box Example

```
<script>
        var x=prompt ("Manipal", " ");
        alert("Welcome to\n" +x);
</script>
```

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Variables</h2>
<p>Create a variable, assign a value to it, and display it:</p>
<p id="demo"></p>
    <script>
    var name = "Manipal";
    document.getElementById("demo").innerHTML = name;
    </script>
</body>
</html>
```

It's a good programming practice to declare all variables at the beginning of a script.

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Variables</h2>
<p>You can declare many variables in one statement.</p>
<p id="demo"></p>
    <script>
    var person = "Manipal", carName = "Institute of", price =
    "technology";
    document.getElementById("demo").innerHTML = person +
    carName + price;
    </script>
</body>
</html>
```

Value = undefined

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Variables</h2>
<p>You can declare many variables in one statement.</p>
<p id="demo"></p>
    <script>
    var person;
    document.getElementById("demo").innerHTML = person;
    </script>
</body>
</html>
```

JavaScript Objects

- JavaScript objects are written with curly braces {}.
- Object properties are written as name:value pairs, separated by commas.
- var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
        <script>
        var person = {
          firstName : "John",
          lastName  : "Doe",
          age     : 50,
          eyeColor  : "blue"
        };
        document.getElementById("demo").innerHTML =
        person.firstName + " is " + person.age + " years old.";
        </script>
</body>
</html>
```