Javascript Essential Training v5

Trainer: Fritz Lim

Website:www.tertiarycourses.com.sg
Email: enquiry@tertiaryinfotech.com

About the Trainer

With more than 10 years of experience teaching at a local polytechnic, Fritz is passionate about imparting knowledge to teens, adults, and children. He believes that an education in technology and in how things work is essential for everyone, so that they can harness and invent the technologies of the future.

He is excited about exploring anything related to computers and IT, with a keen interest in electronics and native cross-platform mobile app development so that our ubiquitous mobile phones can be conveniently used to control and interact with devices wirelessly and over the Internet.

Fritz is ACTA-certified, as well as a registered MOE instructor, and a graduate of the 2018 ConsenSys Blockchain Developer Program, with a Bachelor in Electrical and Electronic Engineering (Computer Specialisation) from Nanyang Technological University.

Let's Know Each Other...

Say a bit about yourself

- Name
- What Industry you are from?
- Do you have any prior knowledge in Javascript?
- Why do you want to learn Javascript?
- What would you like to learn from this course?

Ground Rules

- Set your mobile phone to silent mode
- Actively participate in the class. No question is stupid.
- Respect each other view
- Exit the class silently if you need to step out for phone call, toilet break

Ground Rules for Virtual Training

- Upon entering, mute your mic and turn on the video. Use a headset if you can
- Use the 'raise hand' function to indicate when you want to speak
- Participant actively. Feel free to ask questions on the chat whenever.
- Facilitators can use breakout rooms for private sessions.

Guidelines for Facilitators

1. Once all the participants are in and introduce themselves
2. Goto gallery mode, take a snapshot of the class photo - makes sure capture the date and time
3. Start the video recording (only for WSQ courses)
4. Continue the class
5. Before the class end on that day, take another snapshot of the class photo - makes sure capture the date and time
6. For NRIC verification, facilitator to create breakout room for individual participant to check (only for WSQ courses)
7. Before the assessment start, take another snapshot of the class photo - makes sure capture the date and time (only for WSQ courses)
8. For Oral Questioning assessment, facilitator to create breakout room for individual participant to OQ (only for WSQ courses)
9. End the video recording and upload to cloud (only for WSQ courses)
10. Assessor to send all the assessment records, assessment plan and photo and video to the staff (only for WSQ courses).

Prerequisite

The following knowledge is assumed:

1. Basic HTML
2. Basic CSS

Agenda

Topic 1 Basic Javascript Programming

- Overview of Javascript Programming
- Javascript Output
- Data Types & Variables
- Control Structures
- For and While Loop

Topic 2 Javascript Functions

- Javascript Function
- Function Parameters
- Arrow Function
- Function Scope
- Javascript Event
- Potential Javascript Security Issues

Agenda

Topic 3: DOM

- Access DOM Elements
- Modify DOM Elements
- DOM Events

Topic 4. Error Handling and Debugging

- Error Handling with Try-Catch-Throw
- Debugging Techniques

Google Classroom

- The resources can be found on the Google classroom
- Goto https://classroom.google.com and enter the class code below
- If you have problem to access the Google Classroom, please inform trainer or the staff

j6gvyo3

Exercise Files

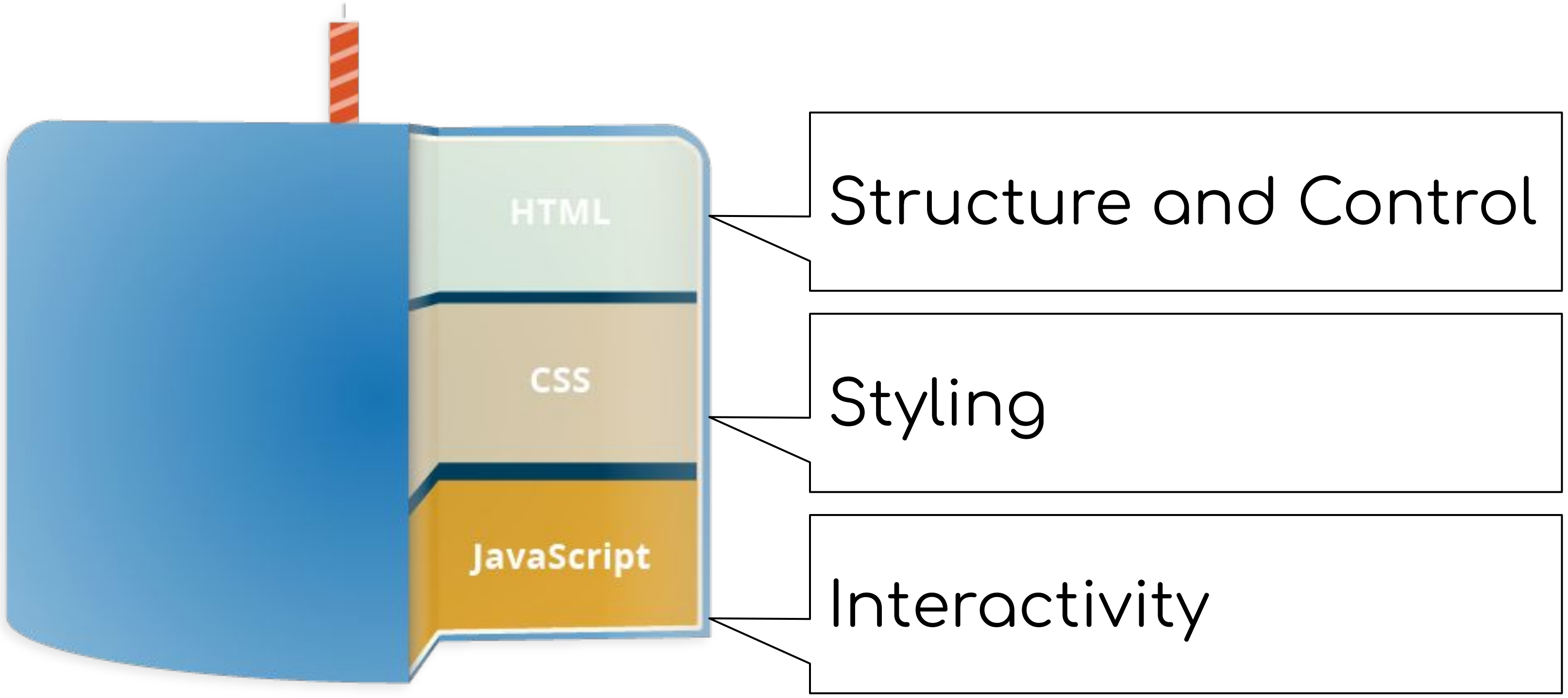
- You can download the exercise files from

https://github.com/tertiarycourses/JavascriptTraining/tree/master/exercises

Topic 1
Basic Javascript Programming

What is Javascript

- JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else



Javascript IDE

You can use the following IDE for Javascript coding in this course

- https://playcode.io/
- https://codepen.io/
- https://repl.it/
- Visual Studio Code
- Sublime Text 3
- Atom

First Javascript Program

```
<!DOCTYPE html>
<html>
<body>

<h2>My First JavaScript</h2>

<button type="button"
onclick="document.getElementById('demo').innerHTML = Date()">
Click me to display Date and Time.</button>

<p id="demo"></p>

</body>
</html>
```

Internal Javascript Script

- In HTML, JavaScript code is inserted between <script> and </script> tags.
- You can place any number of scripts in an HTML document.
- Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.
- Example

```
<script>
document.getElementById("demo").innerHTML = "My
First JavaScript";
</script>
```

External Javascript Script

- Scripts can also be placed in external files.
- JavaScript files have the file extension .js.
- To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

```
<script src="script.js"></script>
```

Javascript Statements

- A JavaScript program is a list of programming statements.
- JavaScript statements are composed of Values, Operators, Expressions, Keywords, and Comments.
- The statements are executed, one by one, in the same order as they are written
- Semicolons separate JavaScript statements.
- JavaScript ignores multiple spaces.
- JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.
- JavaScript statements often start with a keyword to identify the JavaScript action to be performed.

Comments

- JavaScript comments can be used to explain JavaScript code, and to make it more readable.
- JavaScript comments can also be used to prevent execution, when testing alternative code.
- Single line comments start with //.
- Multi-line comments start with /* and end with */.

Javascript Output

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

Inner HTML Method

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

Document Write Method

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
document.write(5 + 6);
</script>

</body>
</html>
```

Window Alert

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
alert(5 + 6);
</script>

</body>
</html>
```

Console Log Method

```
<!DOCTYPE html>
<html>
<body>

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

JavaScript Values

- The JavaScript syntax defines two types of values: Fixed values and variable values.
- Fixed values are called literals. Variable values are called variables.
- The most important rules for writing fixed values are:
  - Numbers are written with or without decimals
  - Strings are text, written within double or single quotes:

JavaScript Variables

- In a programming language, variables are used to store data values.
- JavaScript uses the var keyword to declare variables.
- An equal sign is used to assign values to variables.
- All JavaScript variables must be identified with unique names. These unique names are called identifiers.
- Example:

```
    var x = 5;
    var y = 6;
    var z = x + y;
```

JavaScript Data Types

- JavaScript variables can hold many data types: numbers, strings, objects and more:
- In programming, data types is an important concept.
- To be able to operate on variables, it is important to know something about the type.
- Without data types, a computer cannot safely solve this

```
var x = 16 + "Volvo";
```

JavaScript Data Types

- In JavaScript there are 5 different data types that can contain values:
  - string
  - number
  - boolean
  - object
  - function
- There are 6 types of objects:
  - Object
  - Date
  - Array
  - String
  - Number
  - Boolean
- And 2 data types that cannot contain values:
  - null
  - undefined

JavaScript Numbers

- JavaScript has only one type of number. Numbers can be written with or without decimals.
- Example:

```
var x = 3.14;    // A number with decimals
var y = 3;       // A number without decimals
```

JavaScript Operators

- Arithmetic operators are used to perform arithmetic on numbers

+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

JavaScript Strings

- A JavaScript string is zero or more characters written inside quotes
- To find the length of a string, use the built-in length property.
- The backslash (\) escape character turns special characters into string characters
  - \' Single quote
  - \" Double quote
  - \\ Backslash
  - \b Backspace
  - \f Form Feed
  - \n New Line
  - \r Carriage Return
  - \t Horizontal Tabulator
  - \v Vertical Tabulator

String Methods

- String methods help you to work with strings.
- The length property returns the length of a string. Example
  var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  var sln = txt.length;
- The **indexOf()** method returns the index of (the position of) the first occurrence of a specified text in a string. Example
  var str = "Please locate where 'locate' occurs!";
  var pos = str.indexOf("locate");
- The **slice()** extracts a part of a string and returns the extracted part in a new string. Example
  var str = "Apple, Banana, Kiwi";
  var res = str.slice(7, 13);
- The **replace()** method replaces a specified value with another value in a string. Example
  str = "Please visit Microsoft!";
  var n = str.replace("Microsoft", "W3Schools");
- A string is converted to upper case with toUpperCase(). Example
  - var text1 = "Hello World!";
  - var text2 = text1.toUpperCase();
- The **concat()** joins two or more strings:
- The **trim() method** removes whitespace from both sides of a string:

JavaScript Regular Expressions

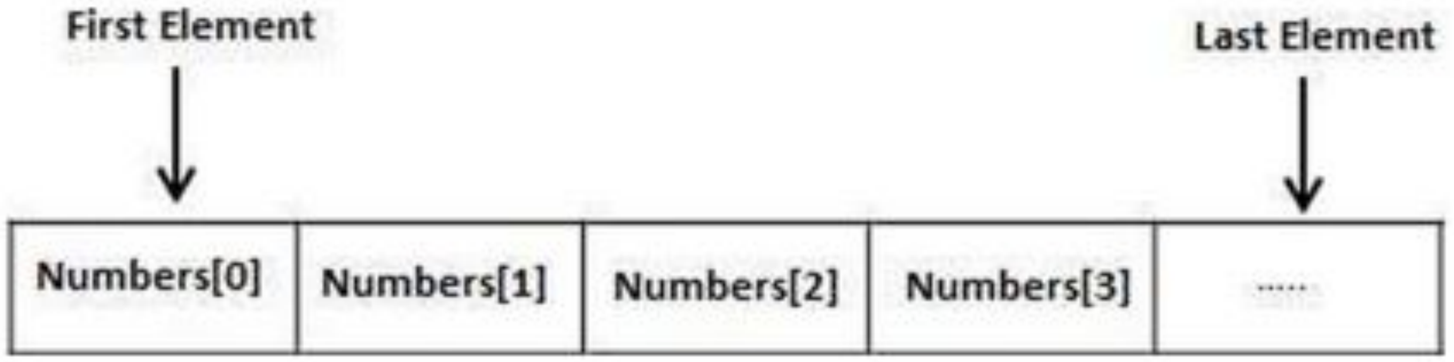
- A regular expression is a sequence of characters that forms a search pattern.
- The search pattern can be used for text search and text replace operations
- Syntax:

  /pattern/modifiers;
- Example:

```
<script>
var str = "Visit W3Schools!";
var n = str.search("W3Schools");
document.getElementById("demo").innerHTML = n;
</script>
```

JavaScript Arrays

- JavaScript arrays are used to store multiple values in a single variable.
- Using an array literal is the easiest way to create a JavaScript Array. Example
  var array_name = [item1, item2, ...];



Array Methods

- The JavaScript method **toString()** converts an array to a string of (comma separated) array values.
- The **join()** method also joins all array elements into a string.
- The **pop()** method removes the last element from an array:
- The **push()** method adds a new element to an array (at the end):
- The **shift()** method removes the first array element and "shifts" all other elements to a lower index
- The **unshift()** method adds a new element to an array (at the beginning), and "unshifts" older elements
- The **splice()** method can be used to add new items to an array.
- The **concat()** method creates a new array by merging (concatenating) existing arrays
- The **slice()** method slices out a piece of an array into a new array.

Sorting Arrays

- The **sort()** method sorts an array alphabetically. Example
  var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.sort();

- The **reverse()** method reverses the elements in an array. Example
  var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.sort();
  fruits.reverse();

Javascript Date

- By default, JavaScript will use the browser's time zone and display a date as a full text string:
- Date objects are created with the new Date() constructor.
- There are 4 ways to create a new date object:
  - new Date()
  - new Date(year, month, day, hours, minutes, seconds, milliseconds)
  - new Date(milliseconds)
  - new Date(date string)

Date Format

- Independent of input format, JavaScript will (by default) output dates in full text string format
  var d = new Date("2015-03-25");
- When setting a date, without specifying the time zone, JavaScript will use the browser's time zone.
- Short dates are written with an "MM/DD/YYYY" syntax like this:
  var d = new Date("03/25/2015");

Javascript Boolean

- A JavaScript Boolean represents one of two values: true or false.
- You can use the Boolean() function to find out if an expression (or a variable) is true
- Example

```
var x = false;
var y = new Boolean(false);
```

Javascript Object

- You define (and create) a JavaScript object with an object literal.
- The name:values pairs in JavaScript objects are called properties. Example

```
var person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

Activity: Javascript Object

Given a Javascript Object

```
var person = {name:"Ally",height:160};
```

display "Ally's height is 160cm" on the HTML

Time: 10 mins

Javascript Math

- The JavaScript Math object allows you to perform mathematical tasks on numbers. Example
  - Math.min(0, 150, 30, 20, -8, -200)
  - Math.max(0, 150, 30, 20, -8, -200)
  - Math.random()
  - Math.round(4.7)
  - Math.ceil(4.4)
  - Math.floor(4.7)
  - Math.sin()
  - Math.PI
  - Math.E

Activity: Javascript Math

Given a circle of radius 5, compute the area of the circle.

Hint: Area = πr^2

Time: 10 mins

Conditional Statements

- Conditional statements are used to perform different actions based on different conditions.
- In JavaScript we have the following conditional statements:
  - Use if to specify a block of code to be executed, if a specified condition is true
  - Use else to specify a block of code to be executed, if the same condition is false
  - Use else if to specify a new condition to test, if the first condition is false
  - Use switch to specify many alternative blocks of code to be executed

Operators

equal	==
greater	>
smaller	<
greater or equal	>=
smaller or equal	<=
not equal	!=
and	&&
or	||
not	!

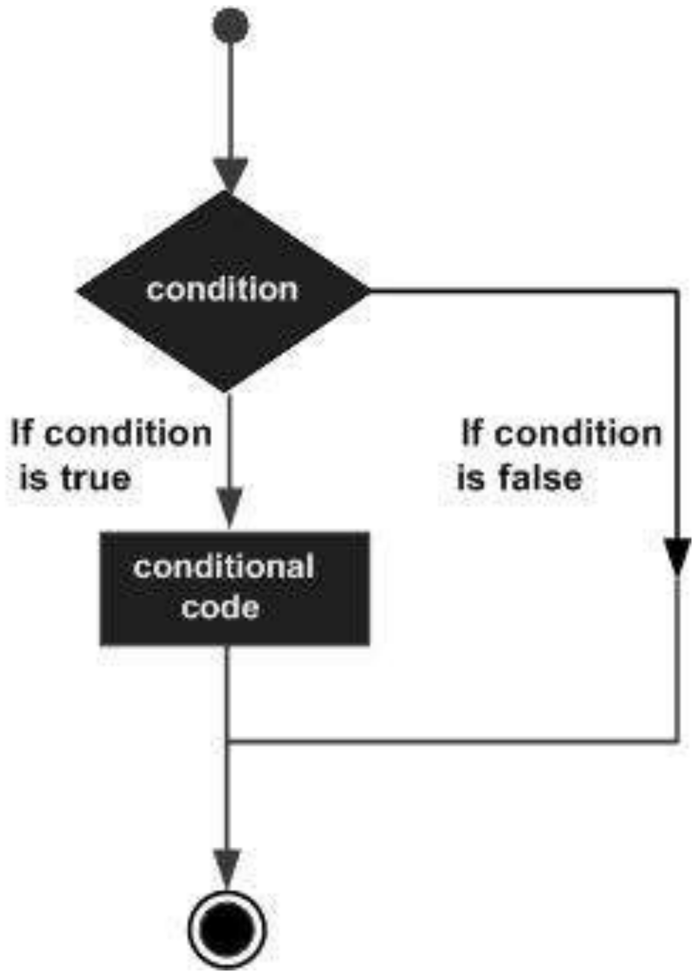
The If Statement

- Use the if statement to specify a block of JavaScript code to be executed if a condition is true.
- Syntax:

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

- Example:

```
if (hour < 18) {
  greeting = "Good day";
}
```



Activity: If Statement

- Simulate throwing a dice: 1 to 6 using Math.floor((Math.random() * 6) + 1);
- Use if to distinguish is the number is even or odd Eg "Even", "Odd"

Hint
```
var dice = Math.floor((Math.random() * 6) + 1);
```

Time: 10 mins

The else if Statement

- Use the else if statement to specify a new condition if the first condition is false.
- Syntax

```
if (condition1) {
  //  block of code to be executed if condition1 is true
} else if (condition2) {
  //  block of code to be executed if the condition1 is false and
condition2 is true
} else {
  //  block of code to be executed if the condition1 is false and
condition2 is false
}
```

- Example:

```
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

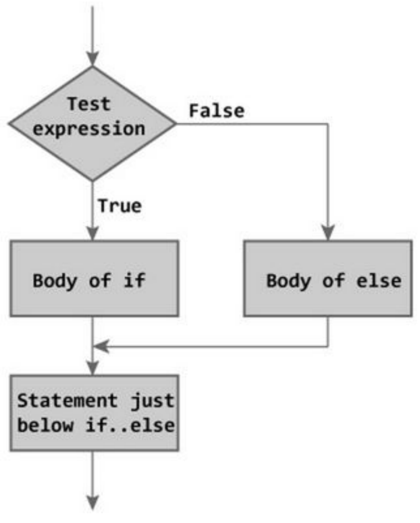


Figure: Flowchart of if...else Statement

Activity: Else If Statement

- Simulate throwing a dice: 1 to 6 using Math.floor((Math.random() * 6) + 1);
- Use elseif to print out the number in text eg "One", "Two" ...

Hint

```
var dice = Math.floor((Math.random() * 6) + 1);
```

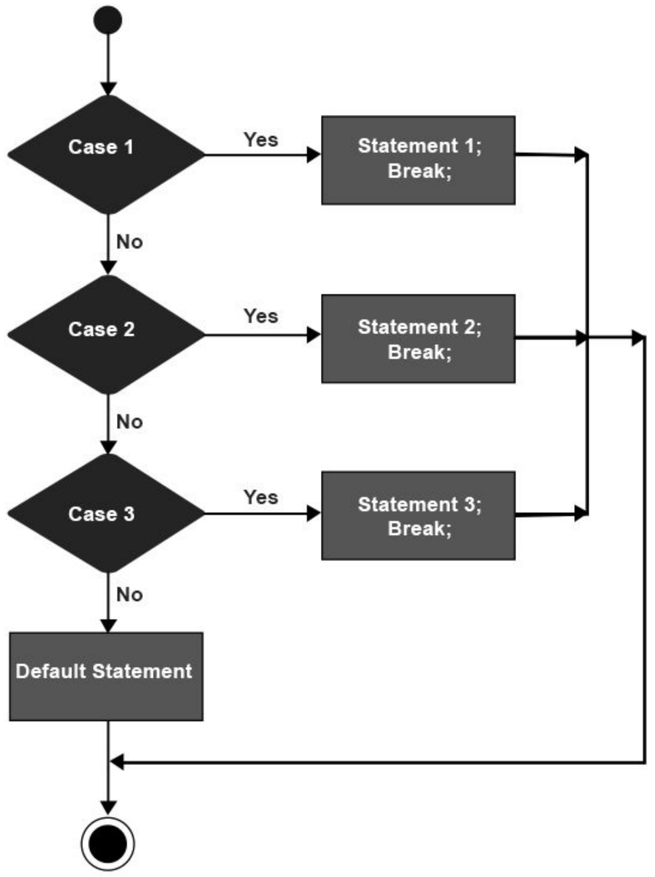
Time: 10 mins

Ternary Operator

condition ? true : false

Eg

```
var a = 8;
var b = 8;
var c = (a>b) ? a : b
alert("the larger number is " + c);
```



Switch Syntax

- The switch statement is used to perform different actions based on different conditions.
- Syntax:

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

- Example:

```
var grade = 'C';
switch (grade) {
       case 'A': alert("Well done!");
       break;
       case 'B': alert('Good job!');
       break;
       case 'C': alert('Work harder next
time');
       break;
       default: alert('I am not sure
your grade');
}
```

Activity: Switch Statement

- Simulate throwing a dice: 1 to 6 using Math.floor((Math.random() * 6) + 1);
- Use switch to print out the number in text eg "One", "Two"...

Hint

```
var dice = Math.floor((Math.random() * 6) + 1);
switch (dice) {
    case 1:console.log('One');
    break;
.....
```

Time: 10 mins

Loop

- JavaScript supports different kinds of loops:
  - for - loops through a block of code a number of times
  - for/in - loops through the properties of an object
  - for/of - loops through the values of an iterable object
  - while - loops through a block of code while a specified condition is true
  - do/while - also loops through a block of code while a specified condition is true

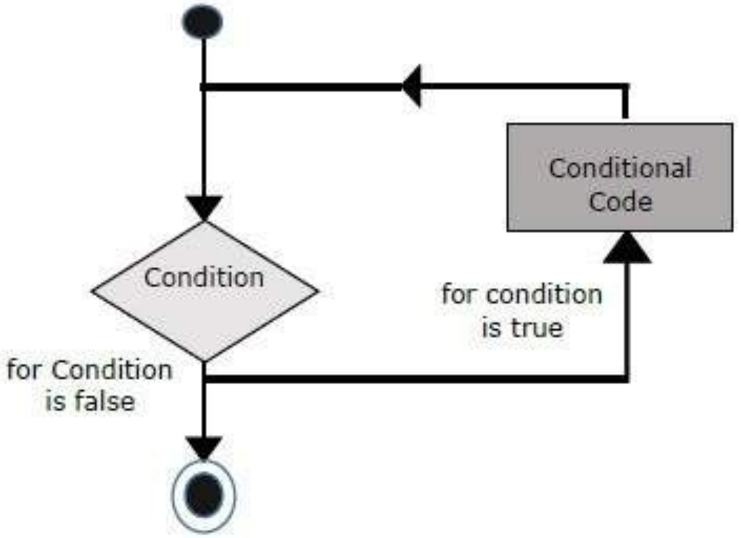
For Loops

- The for loop has the following syntax:

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

- Example:

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



Activity : For Loop

Given the following array
cars = ["BMW", "Volvo", "Saab", "Ford"];

Print out all the elements in the HTML

Time: 10 mins

For-In Loop

- The for/in statement loops through the properties of an object:
- Syntax:
```
for (variablename in sequence){
   statement or block to execute
}
```
- Example:
```
var person = {fname:"John", lname:"Doe", age:25};
var text = "";
var x;
for (x in person) {
  text += person[x];
}
```

For/Of Loop

- The JavaScript for/of statement loops through the values of an iterable objects.
- It loop over data structures that are iterable such as Arrays, Strings, Maps, NodeLists, and more
- Syntax:

```
for (variable of iterable) {
  // code block to be executed
}
```

- Example:

```
var cars = ['BMW', 'Volvo', 'Mini'];
var x;

for (x of cars) {
  document.write(x + "<br >");
}
```

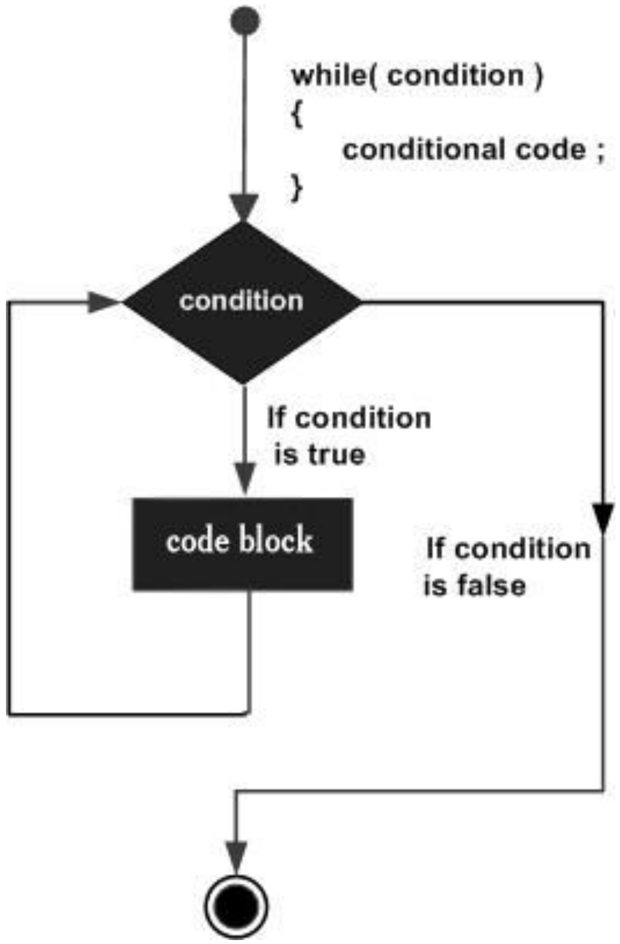
While Loop

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

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

- Example:

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



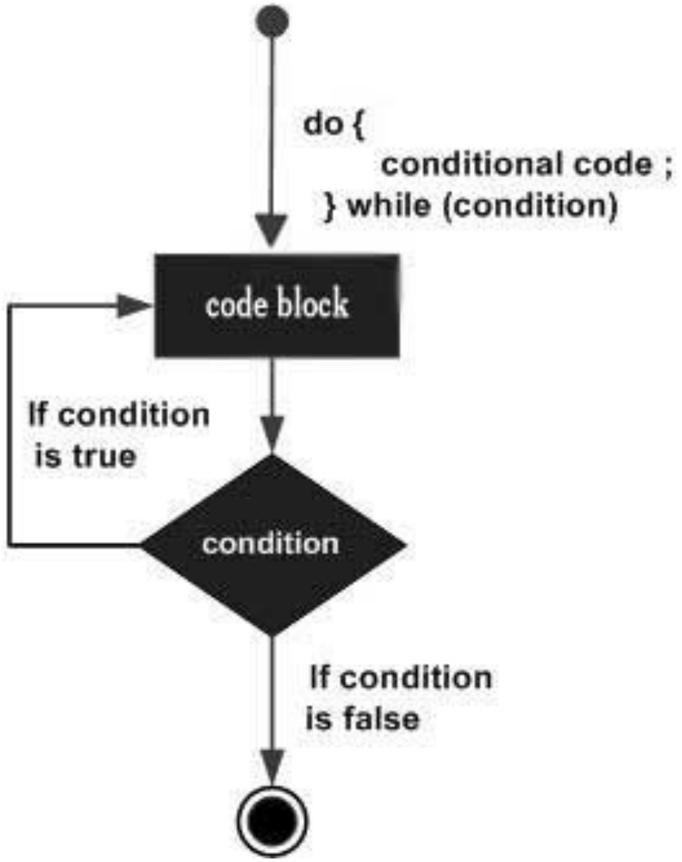
Do While Loop

- The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.
- Syntax

```
  do {
      do Something;
  } while (condition)
```

- Example

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



Activity : While Loop

Create a loop that runs as long as i is less than 10.

Time: 10 mins

Break & Continue

- The break statement "jumps out" of a loop.
- The continue statement "jumps over" one iteration in the loop.
- Example

```
for (i = 0; i < 10; i++) {
  if (i === 3) { break; }
  text += "The number is " + i + "<br>";
}
for (i = 0; i < 10; i++) {
  if (i === 3) { continue; }
  text += "The number is " + i + "<br>";
}
```

Ex: Break or Continue

For {var i=0;i<10;i++}, make the loop skip at 5 and stop at 8 .

Time: 5 mins

Topic 2
Javascript Functions

JavaScript Function

- JavaScript functions are defined with the function keyword.
- You can use a function declaration or a function expression.
- Syntax:

```
function functionName(parameters) {
  // code to be executed
}
```

- Example

```
function myFunction(a, b) {
  return a * b;
}
```

Activity: JavaScript Function

Create a function to compute the area of a circle with radius r

Hint: Area = πr^2

Time: 5 mins

Function Expressions

- A JavaScript function can also be defined using an expression.
- A function expression can be stored in a variable
- Example:

```
var x = function (a, b) {return a * b}
```

- After a function expression has been stored in a variable, the variable can be used as a function

```
var x = function (a, b) {return a * b};
var z = x(4, 3);
```

Parameter Defaults

- If a function is called with missing arguments (less than declared), the missing values are set to: undefined
- Sometimes this is acceptable, but sometimes it is better to assign a default value to the parameter
- Example

```
function (a=1, b=1) {
  // function code
}
```

JavaScript this Keyword

- The JavaScript this keyword refers to the object it belongs to.
- Example

```
var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

Arrow Function

- Arrow functions were introduced in ES6.
- Arrow functions allow us to write shorter function
- Syntax:

```
hello = function() {
  return "Hello World!";
}
hello = () => {
  return "Hello World!";
}
```

- If the function has only one statement, and the statement returns a value, you can remove the brackets and the return keyword:

```
hello = () => "Hello World!";
```

Arrow Function with Parameter

- If you have parameters, you pass them inside the parentheses
  hello = (val) => "Hello " + val;
- In fact, if you have only one parameter, you can skip the parentheses as well
  hello = val => "Hello " + val;

Activity: Arrow Function

Create a arrow function to compute the area of a circle with radius r

Hint: Area = πr^2

Time: 5 mins

JavaScript Function Scope

- In JavaScript there are two types of scope:
  - Local scope
  - Global scope
- JavaScript has function scope: Each function creates a new scope.
- Scope determines the accessibility (visibility) of these variables.
- Variables declared within a JavaScript function, become **LOCAL** to the function.
- Local variables have Function scope: They can only be accessed from within the function.
- A variable declared outside a function, becomes **GLOBAL.**
- A global variable has global scope: All scripts and functions on a web page can access it.

Local Scope Demo

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Scope</h2>

<p>Outside myFunction() carName is undefined.</p>

<p id="demo1"></p>

<p id="demo2"></p>

<script>
myFunction();

function myFunction() {
  var carName = "Volvo";
  document.getElementById("demo1").innerHTML = typeof carName + " " + carName;
}
document.getElementById("demo2").innerHTML = typeof carName;
</script>

</body>
</html>
```

Global Scope Demo

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Scope</h2>

<p>A GLOBAL variable can be accessed from any script or function.</p>

<p id="demo"></p>

<script>
var carName = "Volvo";
myFunction();

function myFunction() {
  document.getElementById("demo").innerHTML = "I can display " + carName;
}
</script>

</body>
</html>
```

Block Scope

- ES2015 introduced two important new JavaScript keywords: **let** and **const**.
- These two keywords provide Block Scope variables (and constants) in JavaScript.
- Before ES2015, JavaScript had only two types of scope: Global Scope and Function Scope.
- Variables declared with the var keyword cannot have Block Scope.
- Variables declared inside a block {} can be accessed from outside the block.

Block Scope Demo

```
<!DOCTYPE html>
<html>
<body>

<h2>Declaring a Variable Using let</h2>

<p id="demo"></p>

<script>
var  x = 10;
// Here x is 10
{
  let x = 2;
  // Here x is 2
}
// Here x is 10
document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```

JavaScript Events

- HTML events are "things" that happen to HTML elements.
- When JavaScript is used in HTML pages, JavaScript can "react" on these events.
- Example

```
<button onclick="alert('Hello World')">Run</button>
```

Common HTML Events

- Here is a list of some common HTML events:

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

Activity: JavaScript Events

Given by the following image in the HTML

```
<p><img src="https://i.ibb.co/h709QDx/merlion.jpg" id="merlion" alt="Merlion" width="350"  />.Click on the image </p>
```

Click on the image and alert "This is merlion"

Time: 5 mins

Security of JavaScript Applications

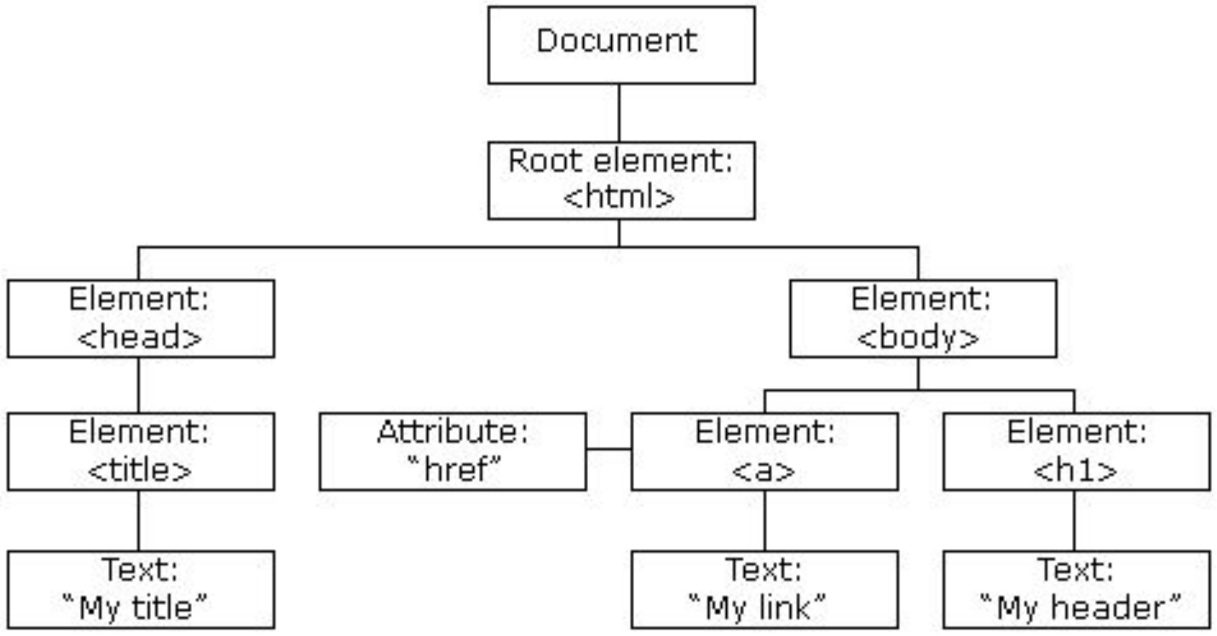
- When creating Javascript functions and events, note that your script could be vulnerable to following cyber threats:

Security	Description
Cross-Site Scripting (XSS)	● Cross-Site Scripting is a commonly used technique that allows running external JavaScript in the context of the attacked website ● To protect your users against XSS, make sure that you never inject unknown user input into the page
Cross-Site Request Forgery (CSRF)	● CSRF is an attack that exploits the mechanism of sending HTTP requests from the browser. ● A simple example of CSRF attack implies that a hacker finds an unprotected <form> element on the web page.
Server-side JavaScript Injection (SSJI)	● Server-side JavaScript Injection is one of the most widespread web app vulnerabilities on the web nowadays. ● You should avoid the use of the eval function to decrease the risk of such vulnerabilities.

Topic 3
DOM

JavaScript HTML DOM

- With the HTML DOM, JavaScript can access and change all the elements of an HTML document.
- When a web page is loaded, the browser creates a Document Object Model of the page.
- The HTML DOM model is constructed as a tree of Objects:



JavaScript HTML DOM

- With the object model, JavaScript gets all the power it needs to create dynamic HTML:
    - JavaScript can change all the HTML elements in the page
    - JavaScript can change all the HTML attributes in the page
    - JavaScript can change all the CSS styles in the page
    - JavaScript can remove existing HTML elements and attributes
    - JavaScript can add new HTML elements and attributes
    - JavaScript can react to all existing HTML events in the page
    - JavaScript can create new HTML events in the page

The DOM Programming Interface

- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as objects.
- The programming interface is the properties and methods of each object.
- A property is a value that you can get or set (like changing the content of an HTML element).
- A method is an action you can do (like add or deleting an HTML element).

Finding HTML Element by Id

- The most common way to access an HTML element is to use the id of the element.
- The easiest way to get the content of an element is by using the innerHTML property.
- Example

```
<html>
      <body>
      <p id="demo"></p>
      <script>
      document.getElementById("demo").innerHTML = "Hello World!";
      </script>
      </body>
</html>
```

Activity: getElementbyID

Popup the alert for the h1 content

```
<h1 id="intro">Hello World!</h1>
```

Time: 5mins

Finding HTML Elements by Tag Name

```
<!DOCTYPE html>
<html>
<body>

<h2>Finding HTML Elements by Tag Name</h2>

<p>Hello World!</p>
<p>This example demonstrates the <b>getElementsByTagName</b>
method.</p>
<p id="demo"></p>

    <script>
    var x = document.getElementsByTagName("p");
    document.getElementById("demo").innerHTML =
    'The text in first paragraph (index 0) is: ' + x[0].innerHTML;
    </script>

</body>
</html>
```

Finding HTML Elements by Class

```
<!DOCTYPE html>
<html>
<body>

<h2>Finding HTML Elements by Class Name</h2>

<p>Hello World!</p>
<p class="intro">The DOM is very useful.</p>
<p class="intro">This example demonstrates the
<b>getElementsByClassName</b> method.</p>

<p id="demo"></p>

<script>
var x = document.getElementsByClassName("intro");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) with class="intro": ' + x[0].innerHTML;
</script>

</body>
</html>
```

Activity: Finding Element

- Use the getElementById method to find the <p> element, and change its text to "Hello".

  <p>Change the Text</p>

Time: 5min

Changing HTML Content

- The easiest way to modify the content of an HTML element is by using the innerHTML property.
- Example

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript can Change HTML</h2>
<p id="p1">Hello World!</p>
<script>
document.getElementById("p1").innerHTML = "New text!";
</script>
<p>The paragraph above was changed by a script.</p>
</body>
</html>
```

Changing the Value of an Attribute

- To change the value of an HTML attribute, use this syntax:

  document.getElementById(id).attribute = new value
- Example:

```
<!DOCTYPE html>
<html>
<body>
    <img id="image" src="smiley.gif" width="160" height="120">
    <script>
    document.getElementById("image").src = "landscape.jpg";
    </script>
    <p>The original image was smiley.gif, but the script changed
    it to landscape.jpg</p>
</body>
</html>
```

Changing HTML Style

- The HTML DOM allows JavaScript to change the style of HTML elements.
- To change the style of an HTML element, use this syntax:

```
document.getElementById(id).style.property = new style
```

- Example:

```
<script>
document.getElementById("p2").style.color = "blue";
document.getElementById("p2").style.fontFamily = "Arial";
document.getElementById("p2").style.fontSize = "larger";
</script>
```

Activity: Changing Content

- Use HTML DOM to change the value of the image's src attribute.

```
<img id="image" src="smiley.gif">
```

Time: 5min

JavaScript HTML DOM Events

- A JavaScript can be 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

```
<!DOCTYPE html>
<html>
<body>
    <h1 onclick="changeText(this)">Click on this text!</h1>
    <script>
    function changeText(id) {
      id.innerHTML = "Ooops!";
    }
    </script>
</body>
</html>
```

Activity: DOM Event

- Create a onclick button to display the time

Topic 4
Error Handling and Debugging

JavaScript Errors

- When executing JavaScript code, different errors can occur.
- When an error occurs, JavaScript will normally stop and generate an error message. The technical term for this is: JavaScript will throw an exception (throw an error).
- Javascript allows you to handle errors using the following 4 methods:
  - The try statement lets you test a block of code for errors.
  - The catch statement lets you handle the error.
  - The throw statement lets you create custom errors.
  - The finally statement lets you execute code, after try and catch, regardless of the result.

JavaScript Error Syntax

```
try {
  Block of code to try
}
catch(err) {
  Block of code to handle errors
}
finally {
  Block of code to be executed regardless of the try /
catch result
}
```

Error Names

Error Name	Description
EvalError	An error has occurred in the eval() function
RangeError	A number "out of range" has occurred
ReferenceError	An illegal reference has occurred
SyntaxError	A syntax error has occurred
TypeError	A type error has occurred
URIError	An error in encodeURI() has occurred

Try Catch Demo

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Error Handling</h2>
<p>This example demonstrates how to use <b>catch</b> to diplay an
error.</p>
<p id="demo"></p>
<script>
try {
  adddlert("Welcome guest!");
}
catch(err) {
  document.getElementById("demo").innerHTML = err.message;
}
</script>
</body>
</html>
```

Throw Demo

```
<script>
function myFunction() {
  var message, x;
  message = document.getElementById("p01");
  message.innerHTML = "";
  x = document.getElementById("demo").value;
  try {
    if(x == "")  throw "empty";
    if(isNaN(x)) throw "not a number";
    x = Number(x);
    if(x < 5)  throw "too low";
    if(x > 10)   throw "too high";
  }
  catch(err) {
    message.innerHTML = "Input is " + err;
  }
}
</script>
```

JavaScript Debugging

- Programming code might contain syntax errors, or logical errors.
- Many of these errors are difficult to diagnose.
- Often, when programming code contains errors, nothing will happen. There are no error messages, and you will get no indications where to search for errors.
- Searching for (and fixing) errors in programming code is called code debugging.
- All modern browsers have a built-in JavaScript debugger.
- Built-in debuggers can be turned on and off, forcing errors to be reported to the user.
- With a debugger, you can also set breakpoints (places where code execution can be stopped), and examine variables while the code is executing.

The Console Log Method

- If your browser supports debugging, you can use console.log() to display JavaScript values in the debugger window

```
<!DOCTYPE html>
<html>
<body>

<h2>My First Web Page</h2>
<p>Activate debugging in your browser (Chrome, IE, Firefox) with F12, and
select "Console" in the debugger menu.</p>
<script>
a = 5;
b = 6;
c = a + b;
console.log(c);
</script>

</body>
</html>
```

Setting Breakpoints

- In the debugger window, you can set breakpoints in the JavaScript code.
- At each breakpoint, JavaScript will stop executing, and let you examine JavaScript values.
- After examining values, you can resume the execution of code (typically with a play button).

The Debugger Keyword

- The debugger keyword stops the execution of JavaScript, and calls (if available) the debugging function.
- This has the same function as setting a breakpoint in the debugger.
- If no debugging is available, the debugger statement has no effect.
- With the debugger turned on, this code will stop executing before it executes the third line.
- Example

```
<script>
var x = 15 * 5;
debugger;
document.getElementById("demo").innerHTML = x;
</script>
```

Summary
Q&A

Feedback

https://goo.gl/R2eumq

TINFOTECH

CERTIFICATE

of ACCOMPLISHMENT

You will receive a digital certificate in your email
after the completion of the class
If you did not receive the digital certificate,
please send your request to
enquiry@tertiaryinfotech.com

Thank You!

Fritz Lim
Tel: 91836486
fritzlim@gmail.com