UNIT 1: HTML, CSS & Client Side Scripting

JavaScript

JavaScript is a scripting language used to create and control dynamic website content, i.e. anything that moves, refreshes, or otherwise changes on your screen without requiring you to manually reload a web page. Features like:

- animated graphics
- photo slideshows
- auto complete text suggestions
- interactive forms

The three elements together form the backbone of web development.

1. HTML is the structure of your page—the headers, the body text, any images you want to include.
2. CSS controls how that page looks (it's what you'll use to customize fonts, background colors, etc.)
3. JavaScript is the magic third element. Once you've created your structure (HTML) and your aesthetic vibe (CSS), JavaScript makes your site or project dynamic.

JavaScript is a "scripting language." Scripting languages are coding languages used to automate processes that users would otherwise need to execute on their own, step-by-step. Short of scripting, any changes on web pages you visit would require either manually reloading the page, or navigating a series of static menus to get to the content you're after.

JavaScript Used For the following:

- Adding interactivity to websites—Used to make website to be more than a static page of text, you'll need to do some Java Scripting
- Developing mobile applications—JavaScript isn't just for websites…it's used to create those apps you have on your phone and tablet as well
- Creating web browser based games—Used for game to play directly from your web browser. JavaScript probably helped make that happen
- Back end web development—JavaScript is mostly used on the front end of things, but it's a versatile enough scripting language to be used on back end infrastructure, too.

JavaScript is either embedded into a web page or else it's included in a .js file. JavaScript is also a "client-side" language. JavaScript can be added directly to a page's code using <script> tags and giving them the type attribute text/javascript.

Including JavaScript in an HTML Page

```
CSS:
<style>
CSS goes here
</style>

JavaScript:
<script type="text/javascript">
JavaScript code goes here
</script>
```

Call an External JavaScript File

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

<code></code>

Pros and Cons of JavaScript

• Pros: –

- Allows more dynamic HTML pages,
- even complete web applications

• Cons: –

- Requires a JavaScript-enabled browser.
- Requires a client who trusts the server enough to run the code the server provides.
- JavaScript has some protection in place but can still cause security problems for clients.
- Remember JavaScript was invented in 1995 and web browsers have changed a lot since then.

Including Comments

Comments are important because they help other people understand what is going on in the code or remind if forgot something. In JavaScript there are have two different options:

- *Single-line comments* — To include a comment that is limited to a single line, precede it with //
- *Multi-line comments* — In case you want to write longer comments between several lines, wrap it in /* and */ to avoid it from being executed

Variables in JavaScript

Variables are stand-in values that you can use to perform operations. You should be familiar with them from math class.

var, const, let

You have three different possibilities for declaring a variable in JavaScript, each

with their own specialties:

1. var — The most common variable. It can be reassigned but only accessed within a function. Variables defined with var move to the top when the code is executed.

2. const — Cannot be reassigned and not accessible before they appear within the code.
3. let — Similar to const, the let variable can be reassigned but not re-declared.

Data Types

Variables can contain different types of values and data types. Use = to assign them:

- Numbers — var age = 23
- Variables — var x
- Text (strings) — var a = "init"
- Operations — var b = 1 + 2 + 3
- True or false statements — var c = true
- Constant numbers — const PI = 3.14
- Objects — var name = {firstName:"John", lastName:"Doe"}

There are more possibilities. Note that variables are case sensitive. That means *lastname* and *lastName* will be handled as two different variables.

Objects

Objects are certain kinds of variables. They are variables that can have their own values and methods. The latter are actions that you can perform on objects.

Objects are created in a number of ways, although the most common is by the

use of the new operator. This is a unary, prefix operator, which means it is placed before a single operand. The new operator should be followed by a constructor function.

```
new Object();
var accountNumbers = new Array( 5 );
var firstName = new String( "Jose" );
```

Sufficient free memory is located in which to create the new object. The new operator returns a reference to the location in memory where the new object has been created. Variables do not store objects, they store a reference to the object located elsewhere in memory.

Creating String objects

Although the following way of creating String objects is perfectly acceptable JavaScript:

var firstName = new String("Jose");

it is usually much more convenient to used the special syntax provided for creating String literal objects. The same result could be achieved with the following statement:

var firstName = "Jose";

String objects can be created in two different ways because the creation of String objects is such a common feature of programming that the developers of the JavaScript language provided this second, simpler syntax especially for Strings.

Object properties and methods

Objects have both data properties and methods (function properties). Often a

property of one object is an object in its own right or perhaps another kind of collection, such as an array.

Properties

Variables (data) that "belong" to objects are called properties. Example for array's length property:

```
var myArray = new Array( 5 );

alert( "length of array is: " + myArray.length );
```

In the example above the length property of the myArray object is accessed by the dot notation: **myArray.length**

Methods

Methods are object properties containing functions. In earlier units you have seen examples of methods, for example the reverse method of an array:

```
var myArray = new Array( 3 );

myArray[0] = 11; myArray[1] = 22; myArray[2] = 33;

 alert( myArray.reverse() );
```

In the example above the reverse() method of the myArray object is invoked by the dot notation: **myArray.reverse()**

As can be seen, the only difference from (data) properties is when methods are invoked with the use of the () (parentheses) operator — this is a special operator expecting a variable holding a Function object to the left, and optional operands (the function arguments) between the parentheses.

It is important to realise that methods are properties, but ones which can also be invoked since they are properties that hold Function objects.

The 'dot' syntax

The dot notation is used to indicate which properties an object owns. Ownership of properties can exist to any number of levels deep, and the dot notation can be used at every level. fine.

The use of the full stop (dot) can be thought of as meaning "the thing on the right belongs to the object (named collection) on the left". The general form for properties and methods to be invoked is as follows:

<object>.<property>

<object>.<method>()

Common examples include:

```
var myArray = new Array( 3 ) ;

myArray.length;

 var images = new Image();

images.src = "images.jpg";
```

Operators

If you have variables, you can use them to perform different kinds of operations. To do so, you need operators.

Basic Operators

1. + Addition
2. — Subtraction
3. * Multiplication
4. / Division
5. (...) — Grouping operator, operations within brackets are executed earlier than those outside
6. % Modulus (remainder)
7. ++ Increment numbers
8. -- Decrement numbers

Comparison Operators

1. == Equal to
2. === Equal value and equal type
3. != Not equal
4. !== Not equal value or not equal type
5. > Greater than
6. < Less than
7. >= Greater than or equal to
8. <= Less than or equal to
9. ? Ternary operator

Logical Operators

10. && Logical and
11. || Logical or
12. ! Logical not

Bitwise Operators

13. & AND statement
14. | OR statement
15. ~ NOT
16. ^ XOR
17. << Left shift
18. >> Right shift
19. >>> Zero fill right shift

Outputting Data

A common application for functions is the output of data. For the output, you have the following options:

1. alert() — Output data in an alert box in the browser window

2. confirm() — Opens up a yes/no dialog and returns true/false depending on user click
3. console.log() — Writes information to the browser console, good for debugging purposes
4. document.write() — Write directly to the HTML document
5. prompt() — Creates a dialogue for user input