Introduction to HTML & CSS

By Mikaila Akeredolu
Email: mikaila@zipcodewilmington.com
LinkedIn: https://www.linkedin.com/in/mikailaakeredolu

Course Objectives

- Making a web page with HTML, CSS and JavaScript
- Writing HTML tags and CSS rules
- Laying out a web page with multiple sections
- Working with Images on the web
- Links / URL
- Navigation
- Customizing fonts
- CSS3 animations
- Intro to JavaScript
- Assesment

What is HTML?

HTML is the language of the web.

HTML stands for **H**yper **T**ext **M**arkup **L**anguage

Simple definition:
It's a way to markup a document to specify attributes like different font sizes, list and links on web pages...

HTML is written in text files and end with the extension .html (eg: index.html)

Web browsers are typically used to display HTML such as safari, Chrome & I.E

DOCUMENT STRUCTURE

DOCTYPE

HTML

TITLE

BODY

```
<!DOCTYPE html>
<html>
  <head>
    <title>Lake Tahoe</title>
  </head>
  <body>
    <header>
      <span>Journey through the Sierra Nevada
Mountains</span>
      <h1>Lake Tahoe, California</h1>
    </header>
    <p>
      Lake Tahoe is one of the most breathtaking attractions
located in California. It's home to a number of ski resorts,
summer outdoor recreation, and tourist attractions. Snow and
skiing are a significant part of the area's reputation.
    </p>
    <a href="#">Find out more</a>
  </body>
</html>
```

HTML SYNTAX

Let’s take a look at some HTML elements and Tags!

Let’s start zip coding...

To get started visit https://www.w3schools.com/html

Next visit https://thimble.mozilla.org

What is CSS?

CSS Stands for Cascading Style Sheet.

We use CSS to make our HTML markup presentable. {add some swag}

By adding colors to fonts/text, background colors

Adding spaces between elements, selecting multiple elements

Laying out and arranging images and videos on our websites and more....

CSS SYNTAX

The Syntax / Rule

We select elements with tags, ID’s and Classes...

h1 {color : yellow;}

PropertyName : Value

Selector{declaration block}

JavaScript

Javascript is a programming language that enables you to create dynamic content, control multimedia and more. It can be used on the client (frontend code that runs on the user's computer) and server side (backend) of your application to send user's info to the frontend or store user info in a database.
An example is using node.js on the server and mongoDB as your database.
We will briefly go over JavaScript so that we can use it in our webpage to affect the frontend. We will also get introduced to programming with javascript to learn basic procedural programming skills.

JavaScript Variables and data types

Variable are like labelled boxes, they hold things (data types)
Data types in Javascript consist of strings, numbers, arrays, booleans, objects etc..
Strings are just text inside quotes like this “zipcodewilmington” or “1978”
Let’s say we want a variable that stores a name. We can declare it like this var name;
then we can store a name inside like this var name = “craig mack”
We can also store a number data type in a variable like var number = 1000.
Arrays are a type of list data type to hold things like a grocery list or list of favorites
Eg: var songsInAlbum = [“Get Down”, “Flava in Ya Ear”, “MainLine”, “That Y’all”];
Objects are variables too but they can store different data types with keys and: “values”
For example: var car = {brand: “Honda”, “model”:”CRV”, “year”: 2017”, “color”:”black”};

Data Types

String: used to define text or letters

Number: used to defined variables that must be numbers or decimals

Boolean: used to hold data that will be True or False

Array: used to hold a list of data with indexes starting at 0

Objects: are variables too but they can hold many values using keys

Arithmetic Operators

Addition Plus sign (+)

Subtraction Minus sign (-)

Multiplication Multiplication sign (*)

Division Division sign (/)

Modulus Remainder(%)

PostFix Increment and Decrement (x++) and x(--)

Prefix Increment and Decrement (++x) & (--x)

Comparison Operators

```
x  > 10      = false

x  <  10     = true

x  >= 5      = true

x  <= 100   = true

x == "5"     = true  // Type coercion

x === "5"   = false // No type coercion (Checks for type and equality)

x != b       = true

x !== "5"   =  true // (Checks if operands are the same type but Not equal)
```

Logical Operators

&& AND both sides need to be true -

|| OR One side needs to be true

! NOT if something was true it makes it false (vice versa)

&& and || both benefit from short circuit. As soon as the statement is answered starting from left to right.

Using Variables & Operators for Calculations

The grouping operator ( ) controls the precedence of evaluation of expressions

```
var  x = 500;
var y = 400;
var z = 10;
var a = 2;
var answer =  x - y + z  * a ;   // How to fix ?
Console.log(answer);
```

Arrays

let fruits= ["Banana", "Orange", "Apple", "Mango"];

toString() converts an array to a string of comma separated array values

console.log(fruits.toString());

join() behaves like toString but you can specify a separator

console.log(fruits.join(' * '));

pop() removes the last element from an array

console.log(fruits.pop ());

push() adds a new element to the end of the array and also returns the array length

console.log(fruits.push (‘ kiwi ‘));

Arrays cont..

shift() removes the first element from an array and returns the element

```
console.log( fruits.shift () );
```

unshift() adds a new element at the beginning of the array

```
console.log( fruits.unshift ( ' lemon ' ) );
```

.length Used to get the length of the array

```
console.log(' The length of the array is  ' +  fruits.length);
```

For further array methods see w3schools for methods such as slice, splice, sort, reverse

Methods (In JS Functions that belong to Objects)

Math.round (x) returns the value of x rounded to the nearest integer //4 . 7

Math.PI returns 3.141592653589793

Math.pow(x, y); returns the value of x to the power of y //8 , 2

Math.sqrt(x) returns the square root of x // 64

Math.abs(-x) returns the absolute positive value // -4 . 7

Math.ceil(x) returns the value of x rounded up to the nearest integer //4 . 4

Math.floor(x) returns the value of x rounded down to its nearest integer //4.7

Math.min(1,2,3)and Math.max(1,2,3) used to find the lowest and highest values

Math.random returns a random number between 0 and 1

For further array methods see w3schools

Functions in JavaScript

A function is a block of code designed to perform a particular task. Functions are executed when called or invoked. For Example: This function multiplies the values of number1 and number2 and returns the answer.

```
Function multiplyTwoNumbers(number1, number2){
        return  number1 * number2;
 }
```

To call or invoke the function we simply do - **multiplyTwoNumbers**(2, 20); number1 and number2 are known as parameters while 2 and 20 are known as arguments The return keyword-- means that this function returns a number.

Return Keyword

return (expression) ; - The return statement ends function executions and specifies a value to be returned to the caller of the function.
It is affected by the semicolon therefore always end your return statements with a semicolon **;**

```
 function myStringFunction(){
        return 'Craig Mack' ;
}

console.log(myStringFunction());
```

Notice that this function has no parameters and takes no arguments

Mixing Global Variables with Function Parameters

```
var z = 100;

function addToZ(x, y) {
 return x + y + z;
}

console.log(' The result of adding to z is ' + addToZ(3, 17));
```

Anonymous Function & Function as variables values

An anonymous function has no name. Often used as callback functions. For example You can assign a function with no name to a variable.

```
var xFunc = function (p1 , p2 ){
return p1 * p2;
}

console.log(xFunc(20,6));
```

Arrow Functions

JavaScript also allows for Arrow Functions which is an es6 Javascript feature..

```
const add = (num1, num2) =>  num1 + num2;
console.log(add);

const greet = () => {
 console.log("Craig Mack - 1000 degrees")
};

greet();
```

Ternary Condition Operator

It takes three operands such as condition ? val1 : val2

```
var currentAge = 20;
var drinkingStatus  = ( currentAge >= 21)  ?  ‘can drink’  :  ‘cannot drink’;

console.log(drinkingStatus);
```

Conditional Statements (If Statement)

We have two main conditional statements in JavaScript.
The if else statement and switch statements

IF STATEMENT SYNTAX

```
if (true) {
console.log(true);
} else{
console.log(false);
}
```

If Statement Practice

```
var nameOfArtist = " Gucci Mane";

if(nameOfArtist === "Craig Mack" ){

  console.log("Craig kicked a brand new Flava in ya ear");

 }else{
          console.log("Get the bag");

}
```

Else If ...Continuation of If Statement

We use the Else If to test a new condition, if the first condition is false or not met

```
var condition1 = 1;     var condition2 = 2;
if(condition1 > condition2){
console.log('One is greater than Two');
}else if(condition1 == condition2){          // change to Not equal
console.log('One is equal to Two');
}else{
console.log('None of the above is true');
}
```

Else If Practice

```
var nameOfArtist = " Beyonce";

If (nameOfArtist === "Craig Mack" ){
console.log("Craig kicked a brand new Flava in ya ear");
}else if (nameOfArtist === " Beyonce" ){
        console.log("Run the world");
}else{
  console.log("Get the bag");
}
```

Switch Statement

```
switch(value){
        case 0:
        console.log(print something......);
        break;

        case 1:
        console.log(print something else......);
        break;


        default:
        console.log(print something else......);
        break;
}
```

Let’s code one!

Loops

Loops are used to repeat certain task .

While Loops are used to loop through a block of code as long as a specified condition is true. (boolean)

```
        while (condition) {
        Code block to be executed
        }
```

```
var j = 0;    var counter  = 10;
        while( j < counter ) {    // meaning - while j is less than 10
        j++;                      //increase j by 1 by adding 1 to it
        console.log(j);           //Now value of j is 1
}
```

While Loop Practice

```
var nameOfArtist = " ";

while(nameOfArtist !== "craig mack"){

 nameOfArtist = prompt("which artist passed today?").toLowerCase( );

          if(nameOfArtist === "craig mack" ){

                alert("Craig kicked a brand new Flava in ya ear");

                 Break;
             }

}
```

Do While Loop

A variant of the while loop. It will execute the code at least once before checking if the condition is true, then it will repeat the loop as long as the condition is true.

```
do{

 Code block to be executed

}
while ( condition);
```

See example on the page next page.

Example

```
let i : number = 0;

do {

    console.log( " The number is " + i );

    i++;

} while(i < 5);
```

For Loop

For loops are used to loop through a block of code a number of times.

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

For example:

```
for( let x = 0 ;  x < 5 ;  x++ ) {
console.log( ' The value of x is ===> ' + x ) ;
}
```

Objects

An Object is a collection of named values. The named values are called properties.
Property is the key such as name
Value is the value of the key such as “ Becky”

To declare an object using an object literal we simply do something like this -

```
person = {
name: 'Becky',
age: 22,
sex: 'Female',
likesHipHop: true
};
```

Print out the Object’s contents

```
console.log("My name is " + person.name + "  I am " + person.age + " years old " +
person.sex + " and its " + person.likesHipHop + " That I like Hip Hop");
```

Try using the person Object to print out something unique. Make it your own!

Creating our own Methods

Methods are actions that can be performed on objects. An object method is a function definition. Let’s add a method to our person object and print it out as seen below!

```
person = {
name: 'Becky',
age: 22,
sex: 'Female',
likesHipHop: true
makeNoise: function(){ return "Arhhhhhhhh"; }
};console.log("My name is " + person.name + " " + " I am " + person.age + " years old " + person.sex + " and its " + person.likesHipHop + " That I like HipHop so" + " Now everybody make some noise " + person.makeNoise());
```

TypeOf

Sometimes you may come across data and need to know the type of data it is. That's when we use TypeOf as a mechanism to check

```
Var value = “1000”;

console.log(typeof(value));

will print  - String or Number ?
```

Practice with the other data types such as boolean, array, number etc...

Working with the DOM

Now that we have a basic understanding of objects , properties and methods . We will leverage the DOM Object and its methods to manipulate our website. In your script.js file let's create a variable and use the dom to manipulate our main-footer area.

```
var footer = document . getElementById ('main-footer') ;
footer. addEventListener ('mouseover', function() {
footer. style.backgroundColor = "black";
});
```

This changes the background color of the footer area once we mouse (hover) over it (Event). By invoking(calling the anonymous (no name) function)

Callback - A function that's passed to another function as an argument and used later.

More DOM methods - GetElementsByClassName

Another way to manipulate the DOM is by using the DOM class method - getElementsByClassName. You can select all classes at ones and target specific ones by their index. Eg: Let’s create a duplicate of the <span> with class=”title” like this <span class="title">Second Journey through the Mountains of code</span> . Next type

```
var theTitle = document.getElementsByClassName('title');
theTitle[1].addEventListener('click', function(){
theTitle[1].style.color = "purple";
});
```

Note that we are able to target a specific title by using its class name and index number

Triggering events from other elements

Finally, we will take a look at combining all we learned by triggering an event one one element from another element. Eg: We click a button and some other element is affected. Let’s create a button element below our <span> with class of title

`<button id="button">The button</button>` then add a class to our <ol> tag like this

`<ol class="orderedList">`

In our script.js file let’s target all the ordered list and make them disappear. (magic)

```
var button = document.getElementById('button');
var orderedList = document.getElementsByClassName('orderedList');
button.addEventListener('click', function(){
orderedList[0].style.display = "none";  });
```