Introduction to HTML, CSS, & JavaScript

Table of Contents

[Module 1 HTML Overview 1](#_Toc190594355)

[Glossary - HTML Overview 1](#_Toc190594356)

[Module 2 CSS Overview & HTML5 Elements 2](#_Toc190594357)

[Module Summary 2](#_Toc190594358)

[Glossary - CSS Overview & HTML5 Elements 2](#_Toc190594359)

[Module 3 JavaScript Programming for Web Applications 3](#_Toc190594360)

[Glossary - JavaScript Programming for Web Applications 3](#_Toc190594361)

# Module 1 HTML Overview

In this module, you learned that:

* HTML provides the basic structure and content for a website using tags.
* Tags represent the elements of an HTML page.
* The HTML DOM Tree describes how a website is structured.
* HTML uses APIs to enhance the user experience, providing features for advanced animation, audio, and video.
* Scripting provides a more interactive user experience when browsing websites.
* It is recommended to not rely on scripting as it can be disabled.
* HTML5 sandboxes help manage iframe mashups.
* HTML5 Browser Support Tables describe which browsers support which HTML5 features.
* JavaScript is used to check if an element is supported by a browser.
* CSS provides consistent style and design throughout the website.
* There are two types of CSS layouts to design websites: fluid and fixed.

## Glossary - HTML Overview

| **Term** | **Definition** |
| --- | --- |
| **DOM Tree** | "Document Object Model" is the data representation of the objects that comprise the structure and content of a document on the web. |
| **XML** | An "eXtensible Markup Language" Designed to store and transport data allowing users to define their own markup languages, primarily to display documents on the web. |
| **XHTML** | An "eXtensible Hypertext Markup Language" similar to HTML but with stricter formatting rules. |
| **Web Storage APIs** | APIs that allow data storage in a browser. |

# Module 2 CSS Overview & HTML5 Elements

## Module Summary

In this module, you learned that:

* CSS creates a uniform look throughout each element of each page of the website.
* CSS is usually coded in external style sheets and creates base styles for a website.
* CSS frameworks assists in implementing UI elements and creating dynamic web pages.
* CSS has two types of frameworks:
* Utility-first frameworks, which provide utility classes to help in building one's own styles and layouts.
* Component frameworks, which provide a wide selection of pre-styled components and templates that can be implemented onto a website.
* Plain (Vanilla) CSS lets developers write the styles and layouts of a website.
* HTML5 elements provide structure and function to websites.
* HTML5 uses the <input> tag to allow users to input information.

## Glossary - CSS Overview & HTML5 Elements

| **Term** | **Definition** |
| --- | --- |
| **Component Framework** | Component frameworks provide pre-styled components and templates which are easy to add to any website. |
| **CSS** | "Cascading Style Sheet"s is a style sheet language that describes how HTML elements are displayed​. It is the design that is layered over the top of an HTML web page​. |
| **Fixed Layout** | A fixed layout is a layout where ​you specify the height and width of elements, and those values remain the same regardless of which operating system or browser you use to access the website. |
| **Fluid Layout** | A fluid layout is a layout in which ​the height and width of elements is flexible ​and can expand or contract based on the browser window, the operating system, and other user preferences. |
| **Utility Framework** | The utility framework provides utility classes that are scoped to individual CSS properties, which helps in building custom designs in HTML files. |

# Module 3 JavaScript Programming for Web Applications

**Module Summary**

In this module, you learned that:

* JavaScript is a scripting language that enables developers to add dynamic content to web pages.
* JavaScript variables are declared using the 'let' or 'const' keywords and take their type from the value assigned.
* Program execution is controlled by statements like If…Then…Else, Switch, For loops, and While loops.
* JavaScript uses blocks of code, called functions, that can be called from anywhere in the script.
* New methods and properties can be added to an object by modifying the prototype for that object.
* Prototypes allow you to define properties and methods for all instances of a specific object.
* Client-side scripts are programs that accompany HTML documents and are used by developers to incorporate more interactive elements.
* The script tag can incorporate a script within an HTML document or call a script from an external file.
* The Document Object Model (DOM) is the programming interface between HTML or XHTML and JavaScript.
* Developers can access HTML DOM elements from JavaScript scripts using the correct DOM notation.
* APIs are often used to access HTML DOM elements in web pages.

## Glossary - JavaScript Programming for Web Applications

| **Term** | **Definition** |
| --- | --- |
| **AJAX** | “Asynchronous JavaScript and XML” that encompasses more than asynchronous server calls through JavaScript and XML. It is not programming language or technology but rather a programming concept. Ajax represents a series of techniques that provide richer, interactive web applications through HTML, JavaScript, Cascading style sheets, and modifying the web page through the Document Object Model. The name is misleading though because nowadays, JSON is commonly used instead of XML. |
| **Anonymous Functions** | A type of function that has no name or we can say which is without any name. It is declared without any identifier and is often used as a parameter of another function. It is a common way to execute a function immediately after its declaration. |
| **Array** | A data structure that aids the programmer in the storage and retrieval of data by indexed keys. Arrays use a zero-based indexing scheme, meaning that the first element of an array has an index of zero. Arrays grow or shrink dynamically by adding or removing elements. |
| **Classes** | Classes act as a blueprint or template for building objects with similar characteristics and behaviours. A class encapsulates data (in the form of properties) and functions (in the form of methods) that work on that data. |
| **Client-Side Script** | A program that accompanies an HTML doc or embedded in HTML. Scripts run during load of a document or when an action is performed. They can be used to validate forms, process input, or dynamically create document elements. Embed a script in HTML, with the <script> tag in either of the following ways:   • <script> // JS code </script>   • <script src="path name"></script> Use <noscript> tag for browsers with JavaScript disabled or ones that don’t support JavaScript. |
| **Document Objects** | Document representing the main web page that gives access to all HTML elements on the page. When page is loaded HTML doc becomes a document. It is referred to with “document”. |
| **DOM** | “Document Object Model” is a programming interface (API) between HTML and JavaScript. It allows for dynamically accessing and updating content, structure, and style. JavaScript uses the DOM to access and modify web page elements in the browser. |
| **Element Nodes** | All HTML tags. |
| **Element Objects** | The most general base class that all element objects in a Document inherit. It only has methods and properties common to all elements. Everything in a HTML page is an element. And one element can have other elements nested within itself. |
| **Event** | An event is something either a browser or a user does that the JavaScript can react to such as a button click or when a user submits input on a form. |
| **Event Binding** | Refers to telling the browser that a function should be called whenever some ‘event’ occurs. |
| **Event Handlers** | A function that declares what to do when an action is performed such as the click of a button. Example: <button type="button" onclick="showAnswers()"> Show Solution  <script>   function showAnswers() {    //code    alert("A")   }  </script> </button> Note that showAnswers() is an event handler. |
| **Extend** | This keyword is used in class declarations or class expressions to create a class that **is** a child of another class. |
| **Functions** | Functions are modules of code that execute a particular task. They may take-in data, called arguments or parameters, and sometimes return data as well, called the return value. Functions are commonly defined with this syntax: function functionName() {  // function code;  // optional return statement; } |
| **IIFE** | “Immediately Invoked Function Expression” runs immediately after it is defined.  After the function executes it cannot be called again elsewhere in the program.  It is a type of self-executing function. |
| **Nodes** | The basis of all elements in the Document Object Model (DOM) structure. |
| **Objects** | Objects are instances created from a class. They are real-world entities that represent the characteristics defined by the class. Objects have a special set of properties that store data and methods that specify behaviours. These methods and properties can be accessed and changed to carry out specific tasks and communicate with other programs. |
| **Prototypes** | A function prototype lets you easily define and add properties or methods to an object. Prototypes exist for all objects that can be created with the keyword”new”. All object constructors create objects that inherit properties and methods that are defined by the prototype. At instantiation objects inherit the current state of the prototype. Note however, that scripts can override prototype properties and functions. Following is an example of using a prototype to add a method to the Car class: function Car(make, model, year) {  this.make = make;  this.model = model;  this.year = year; } Car.prototype.getName = function() {  return this.make + ‘ ’ + this.model + ‘ ’ + this.year; } |
| **Script** | Offers developers  means to modify and extend HTML documents in highly interactive ways. Scripts can be used to validate forms or to process input as it is typed. Scripts can be triggered by events that occur on a web page, such as the clicking of a button. Scripts can be used to dynamically create document elements on an HTML page. |
| **Self-Executing Functions** | Often used to initialize data or declare DOM elements.  These functions can be  anonymous. |
| **Text Nodes** | The nodes that contain actual text that go between an element start tag and end tag. |
| **this** | Keyword “this” refers to current instance of the object. The value of “this” can vary depending on how the object is called. |