# Task 1.1 Explain how JavaScript is used as a web programming language.

## What is JavaScript

JavaScript is a scripting language that can be used to implement complex features on web pages or can be run independently using an execution engine, such as Node.js.

Originally called Mocha, or LiveScript, JavaScript was first implemented in Netscape Navigator 2 during 1995. It began to be adopted by more web clients and proved more popular than competing standards such as Microsoft’s JScript. It became a de facto standard before being officially standardized as ECMAScript (as Oracle Systems hold a copyright on ‘JavaScript’) in 1997.

There have been multiple versions of ECMAScript over the years, and some web browsers have incomplete or quirky implementation of the standard. Developers had to code to specific browser versions due to the differences between them, but this has greatly improved over the years requiring less hacks or workarounds.

In web browsers JavaScript interacts with both data and the user interface to allow a wide range of interactions. The Document Object Model (DOM) allows scripts to modify the view rendered in a web browser for simple effects (such as roll overs animations) to complex data modelling, creation and removal of UI elements and much more.

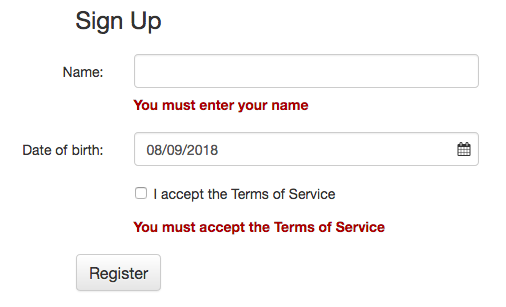
Web applications use the trinity of HTML, CSS and JavaScript to build web applications to view and modify data. JavaScript does not interact directly with HTML or CSS (as these are just the languages used to describe the UI data initially loaded into the DOM) but can effectively achieve this by manipulating its model stored in the DOM.

## Event Driven Language

JavaScript is an event driven language, where the host of the JavaScript engine (eg web browser) raises event when something happens such as a UI click, a timer fires or a system event is raised (such as close the window). To react to this event, JavaScript must declare an event handler in the DOM. These are functions are then called by the JavaScript engine when it receives a notification for a matching event.

When no events are raised, no code will be executed.

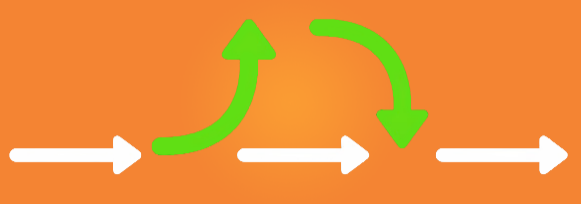
These interactions with the UI (through the DOM) allow JavaScript to perform a range of activities. Amongst the activities it can perform it is widely used for Form Validation and UI animation effects.

**Form Validation** is the function of ensuring that any data inputted by the user is correct and suitable for use. Common examples include check a text input conforms to the format used by telephone numbers, dates or credit card numbers, or that data does not exceed minimal or maximal lengths. It can also be used to show/hide or create UI elements dynamically – a user option could change the language shown on a page without a reload, or change the visual theme on a page by requesting new data from a server and incorporating it into the DOM.

A screenshot of a video game

AI-generated content may be incorrect.**Animation effects** can be simple changes to CSS (make buttons green to indicate success or disabled/grey when data is incomplete or unacceptable) or more complicated drawings or graphs generated on the fly. Aside from looking pretty, they can improve the user experience by providing feedback during long running event.

## Asynchronous Operations

JavaScript engines are simple beasts and operate using a simple loop to process events that have been raised to it, known as the Event Loop. As it runs in a single thread, the event loop can only process one event at a time. A long running event, such as accessing a remote server, can stall the event loop which will prevent other events from running.

JavaScript combats this by allowing asynchronous operations, giving the appearance of multi-threaded operations. An example of this is when that while a server request is being processed (potentially taking a long time) over JavaScript events can be processed, so a ‘submit’ button would not stop the processing over animations.

Asynchronous programming can be thought of as similar to event driven programming, where the asynchronous code is called when the long running event raises a ‘I have finished’ event, although the actual implementation is more complicated.

## Server-side Execution

Most JavaScript is runs in a web browser (or client) and this is known as client-side code and only effects the local browser. The client-side code cannot directly access resources such as databases and files stored on a server and must ask a server to modify them for it. Server-side code can be written in many different languages including JavaScript.

To run JavaScript on a server (as opposed to in a browser) it must be run by inside an engine application as JavaScript is not compiled and cannot be ran on its own. The most popular engine is node.js which can run on most servers irrespective of the operating system used. These scripts process requests for different clients and can be used to generate data return it (generate an image or a report perhaps), or to modify data stored in a file system or database depending on validated user requests.

The security issues on server-side execution are more arduous that web client interfaces but are far more important to get correct. Even though a web client applies validation of any inputs, the server still needs to validate its inputs separately as requests can be made from any client – they may have not used your carefully crafted website and can wreak havoc on your systems without proper validation.



## Single Page Applications

More traditional web applications used a submit-and- redirect structure. Under this methodology, a browser sends a request to a server. The server processes the request and then sends a signal to the browser to redirect to show a different page. So, for example, an Edit page would save any changes and instruct the browser to show the List page.

JavaScript allows a different paradigm called Single Page Applications (SPA), where after the server returns only the modifications it has made. The JavaScript running the browser simply updates the elements that have been changed without having to refresh the entire DOM from server files. An example of this is chat window, where changes from the server update the chat content dynamically and across multiple clients simultaneously and no clients need to reload the entire page, which would ruin the chat experience.

SPA application are quicker to load and more responsive to use as less data needs to be transferred. SPA’s can also be quicker to develop as they allow multiple programmers to work on the different elements at the same time – both the different sections within a page and the server-side elements.

Of course, this adds complexity to the development of SPA sites, but use of a pre-written framework such as React, Angular or Vue greatly reduces the work and learning curve involved.