Alan Robb

  Glasgow Clyde College

client-side scripting report

Contents

[Introduction 2](#_Toc177646678)

[Editors 2](#_Toc177646679)

[Visual Studio Code 2](#_Toc177646680)

[Notepad++ 3](#_Toc177646681)

[Browsers 3](#_Toc177646682)

[Browser Icons 3](#_Toc177646683)

[Chrome Developer Tools 3](#_Toc177646684)

[Interpreted and Compiled Languages 4](#_Toc177646685)

[Client-Side and Server-Side Scripting Languages 4](#_Toc177646686)

[Document Object Model 5](#_Toc177646687)

[Document Object Model Example 5](#_Toc177646688)

[Client-Side Script Location 6](#_Toc177646689)

[References 6](#_Toc177646690)

# Introduction

There are many types of computer languages which serve infinite purposes within the world of Computer Science. One such type is Client-Side Scripts. Below are various features of Client-Side Scripts including the roles they play and some advantages and disadvantages of using them.

## Editors

Visual Studio Code is a powerful open-source scripting program and can be used for both client-side and server-side programming. Almost all scripting languages are supported by Visual Studio Code (for example HTML, CSS and Javascript) and extensions can be found for just about any purpose. Programmers can customize their working/coding environment to suit their preferred colour themes, profiles or settings. Visual Studio Code is very well equipped for integrating with external tools programmers may need to accomplish their goal for instance GitHub or Node.js.

### Visual Studio Code

A computer screen with many colorful text

Description automatically generated

Notepad++ is a free open-source code editor and can provide support for over 50 scripting languages. At first appearance Notepad looks like a simple word editor but has many useful features for programming including code folding, indentation and limited auto completion amongst other things. Notepad++ has support for plugins and macros. There are over 100 plugins developed for Notepad++.

### Notepad++

A screenshot of a computer

Description automatically generated

## Browsers

Browsers are software programs designed to display information over the internet or within an intranet. Most browsers of which there are many are mainly used for displaying web pages containing photographs, informational text/diagrams, multimedia (audio, video) popular browsers include Google Chrome, Firefox, Safari and Microsoft Edge to name but a few. A web browser takes information from other parts of the internet and displays it on your computer, mobile phone or any digital device that can run the software. The way it does this is to fetch data from a server and use a piece of software called a ‘rendering engine’ to translate the raw data into human readable text and images before displaying it on the user’s device. Many browsers use software called cookies to capture information about the user for instance - interests, browsing patterns or username and password information to provide more relevant content and a generally smoother experience to the user.

### Browser Icons

A group of circular logos

Description automatically generated

Programmers can utilise a variety of Developer Tools incorporated into browser software to identify errors in their code and change code to improve the appearance and accessibility of pages so everyone can easily view web pages. Browser developer tools display the structure of a web page’s code and allow programmers to make tweaks to their web pages to help display the content on all devices (it is important that web pages are dynamically coded to display correctly on mobile devices first and then larger devices like tablets or desktop computers).

### Chrome Developer Tools

A screenshot of a computer

Description automatically generated

## Interpreted and Compiled Languages

The main difference between compiled languages and interpreted languages is the way the two types of language execute code:

Compiled languages tend to run faster than interpreted language. C is a compiled language. A compiled language is converted into machine code therefore processors can execute it efficiently.

Interpreted languages are deemed less efficient than compiled languages. PHP, Javascript and Python are examples of interpreted languages. Interpreted languages execute code directly without converting it to machine code and therefore tend to be less efficient.

Both languages have advantages and disadvantages:

Compiled code is harder to reverse engineer so is more secure however due to its complexity is more time consuming to programmers.

Interpreted languages however tend to be easier to develop with due to not having to be compiled. The languages are easier for beginners to learn and have good cross-platform compatibility. Disadvantages can be higher resource usage and slower execution as well as late error detection.

## Client-Side and Server-Side Scripting Languages

Client-Side scripts like Javascript are executed within the web browser of the user whereas Server-Side languages such as PHP are mainly executed by the web server typically before data is passed to the browser. Server-side processes include handling user input, retrieving data from a database or identifying authentication as opposed to client-side scripts that deal with displaying the web page and handling user interactions with Javascript, CSS and HTML.

Server-side processes are executed on the web server and are less vulnerable to malicious attacks and tampering than client-side scripts. Client-side scripts are executed on the user’s device and are therefore potentially less secure.

Server-side scripts have access to the server’s resources such as its CPU, memory and storage as well as databases and any other servers that the web application uses. Client-side processes however only have access to the user device resources.

## Document Object Model

The Document Object Model (DOM) is the interface that contains a web page structure, style and content. It is an object-orientated representation of the web page and can be manipulated with scripting languages such as Javascript. The DOM itself is not a scripting language but represents a document as a whole for instance : the header, tables within the document, text within tables and all other elements in a document.

Javascript is commonly used to modify the DOM but other languages can be used for example Python or PHP to make web pages more responsive. Client-side scripts are used on web pages to add, remove or change content, respond to user actions such as upon the click of a button an event occurs, extract data from forms, validate user input and send data to a server for processing…. the list is endless.

Code to access the DOM can be embedded directly into the HTML of the document using <script> tags or linked to an external Javascript(.js) file containing relevant code. External code is better than too much scripting within the HTML as it makes code more difficult to read, is less secure and vulnerable to XSS (Cross-Site Scripting) attacks.

### Document Object Model Example

A diagram of a website

Description automatically generated

## Client-Side Script Location

Javascript can be located in different places to function in a web page. Javascript is traditionally embedded in a <script> tag at the bottom of the code just before the closing body tag to allow the HTML of the document to load before it reaches the Javascript. An interpreter reads code from top to bottom and this ensures page functionality. A more modern approach is to locate the <script> tag in the head of the HTML near the top of the code or linked to an external Javascript file using the .js extension. When Javascript is located in an external file it is accessed using the <link> tag in the head of the HTML code. External .js files separate your HTML and Javascript making code easier to read and maintain. It also allows the same code to be used across several pages and is better for very long scripts which will clutter HTML. Within the <script> tag of embedded or external Javascript the ‘async’ or ‘defer’ attribute is used to manipulate how the client-side script will affect the web page. Asynchronous loading (using the async attribute) means the Javascript will load in the background allowing other resources to load in parallel and reduces the risk of performance bottlenecks. Deferred loading (using the defer attribute) allows the HTML code to load before executing Javascript which results in a more predictable and controlled execution order. Using the ‘async’ or ‘defer’ attributes improves website performance and user experience by helping scripts load faster and not blocking rendering. Defer can be useful for scripts that need to interact with the DOM like event listeners and widgets whereas Async can be better suited to scripts that run independently and do not need to interact with the DOM.

# 

# References

(Anonymous, 2024)

(Collins, 2017)

(Kantor, 2007-2024)

(Point, 2024)

(Imms, 2022)