

OJordan - Assignment1

by Oliver Jordan

Submission date: 30-Mar-2023 11:28AM (UTC+0100)

Submission ID: 200995647

File name: 2911_Oliver_Jordan_OJordan_-_Assignment1_488464_1133291594.docx (33.46K)

Word count: 2911

Character count: 16329

Assignment 1

Oliver Jordan

MSc Software Engineering (Web Development)

Web Development (ACCA7004)

Michael Dacey

Table of Contents

1	<i>Comparison of Client- and Server-Side Scripting</i>	<i>3</i>
1.1	Client-Side Scripting.....	3
1.2	Server-Side Scripting.....	3
1.3	Comparison	4
2	<i>Investigation into the Node.js framework</i>	<i>4</i>
3	<i>Discussion of Ajax and React.....</i>	<i>5</i>
4	<i>Virtualisation</i>	<i>5</i>
5	<i>Conclusion.....</i>	<i>6</i>
6	<i>Bibliography</i>	<i>7</i>

1 Comparison of Client- and Server-Side Scripting

In web development, websites can be designed to run code on either the server's own architecture or send code to the user's client to execute locally. This introduces several differences and behaviours between the two designs [1].

1.1 Client-Side Scripting

Client-side scripting refers to anything that is executed or interpreted by the user's own client. This can range from simply interpreting html or CSS files sent by the server to running entire JavaScript files. This does require the correct software to be being used to interact with the server, however, and this is commonly a modern web browser, which has evolved over the years to include more sophisticated tools to be used in client-side scripting [2].

One issue observed with client-side scripting is the security vulnerabilities it could introduce. Cross-Site Scripting (XSS) was one of these issues, and was originally thought to be a DOM (Document Object Model) specific issue [3], but is nowadays not restricted to requiring any DOM functionality to be executed [2]. XSS allows for a malicious user to be able to execute scripts on the client-side to cause a web page to perform unintended behaviour, often to the detriment of another user or the provider [4]. Another large security issue was present in the form of clickjacking, where an attacker could mask a site element by manipulating the opacity of it using CSS attributes, allowing for a user to click on an element without realising it was there [2]. This led to the introduction of new headers in HTTP responses to protect against these attacks [5].

Some of the scripting languages used for client-side scripting include JavaScript, jQuery, and Flash (although support for the latter has been discontinued by Adobe in 2020 [6]) with some far lesser used languages being used previously such as Silverlight and Java [2]. Using jQuery as a main example, it can be demonstrated that scripting languages provide the framework to turn static html pages into interactive pages, allowing for DOM traversal, and make HTTP requests through a script [7]. As part of this assignment, a project containing examples of code is being submitted, and contains an example of jQuery code showing some of these features – these can be found under the folder “jQueryExamples”. The page created under this directory has two example pages, showing how jQuery can manipulate the html of the page, as well as make an HTTP request to a dummy API.

1.2 Server-Side Scripting

On the opposite end of the scale, server-side scripting refers to servers running the scripts on their own hardware, executing code when a client makes a request to various endpoints. This allows for the source code to remain hidden, ensuring security and control. Often, an HTTP request is made to servers operating in this fashion, passing a request to the server, which then executes the required commands and returns some response, often including a status code and some data or an html page.

Since the scripts are being executed by the server, this introduces computational load onto server hardware, and as such, server-side scripting languages need to consider computational performance for optimal operation [8]. Due to this nature, one of the vulnerabilities of server-side scripting is a distributed denial of service attack (DDOS) – as the server has to process every request on its own hardware, if many request are made simultaneously with malicious intent, this can slow or even stop the server entirely [9]. Server-side scripting is also vulnerable to its own form of XSS – if user inputs are being stored in a database, without proper precautions, another client retrieving the data could be exposed to unintended code [10], as if the data is being displayed as plaintext in the browser in an html page, a malicious user could make use of script tags to execute code on other user's browsers.

Running scripts on the server side also allows for code to securely connect to a database as part of the service, as it can be made safe to do so as the user cannot see any of the process. This allows for websites and web services to store data securely and perform more actions than a client-side script would be able to.

Some example languages of server-side scripting languages are PHP, JHP and Node.js. Using a variant of Node.js, Express, some examples have been provided in the project submitted alongside this report. These can be found under the “expressJsExamples” directory. This code demonstrates the capability of server-side code to perform logic with the browser receiving minimal code to execute. The server is also capable of connecting to a database and performing CRUD actions. The server injects required parameters into an html template view before sending it to the client, resulting in the client only receiving an html file from the server.

1.3 Comparison

Both client-side and server-side scripting have their own strengths and uses, with server-side scripts being required for performing operations on systems hidden from the client’s view, and client-side scripting being useful for taking responsibility for the computational load of an interactive webpage. Both styles of scripting are vulnerable to various malicious attacks, but these have been researched and defended against in many cases [2] [3] [9].

2 Investigation into the Node.js framework

Node.js is a framework for developing web applications, that has proven to be lightweight and efficient [11]. Node.js is built using the JavaScript programming language and allows for efficient building of web applications through the help of the many tools found in the NPM package repository [12]. Node.js works with an event loop, allowing for operations to be completed without blocking the running of new processes [13]. When these processes are finished, a call back function is then executed and added to the queue [13]. Node.js has been used by 2.0% of sites [14], including sites such as Twitter and Netflix, showing that is capable of being used for large and complex sites.

Node.js, when used for backend development, provides the benefits of JavaScript development [13]. As JavaScript is a widely used language, this allows access to a huge number of resources – easier code sharing, larger community knowledge base and many free tools are just some of these benefits. As many front ends also use JavaScript for their development, code can be reused in both server and client-side scripts [13].

As previously mentioned, Node.js is fast and efficient, partly in due to the asynchronous nature of its event queue. However, while many low CPU-intensive tasks can be handled efficiently, a single large CPU-intensive task can slow a server running Node.js down [13]. These kinds of tasks can be rare for web applications, however. In 2013, PayPal began adopting Node.js for some of their site pages, and reported a significant decrease in response times [15].

Node.js also lends itself well to the use of microservices, a style of architecture that provides a web app as a suite of smaller apps performing their own tasks [16]. By breaking down the business logic required on a server into smaller microservices, developing better flexibility and allowing for much smoother scaling of a product [13].

Some of the drawbacks of Node.js include, as previously mentioned, the bottleneck introduced by CPU-intensive tasks. This is a holdover from the language Node.js is built on, as JavaScript only uses a single thread for CPU tasks. Node.js has introduced a feature called worker threads in 2018 in an effort to combat this [17], allowing for some level of parallelisation.

Care must also be taken around the Node.js framework's fondness for call back functions. As the framework utilises call back functions to be used with tasks, those call backs may also include their own call backs, resulting in several layers of confusing call back functions that can become unreadable and difficult to maintain [13].

3 Discussion of Ajax and React

Ajax is an approach to using several technologies in order to allow a web page to handle requests to a server without reloading a web page [18]. Ajax is a shorthand for "Asynchronous JavaScript and XML" and though it has XML in the name, JSON is more often used for its lightweight nature. By making use of HTTP requests, the data can be retrieved from a server, and a function can then handle the data retrieved from the server's response. This data can then be used with a dynamic web page to update some element that the user is viewing [19].

Ajax improves the performance of web pages by only transmitting the necessary data to the server, allowing for quicker response times, and the asynchronous nature of it allows for many commands to be processed by a web app at once, and elements on a page to update simultaneously. The limited data being sent can also reduce bandwidth, pages remain efficient and load faster.

Ajax also provides a secure way to communicate between a client and a server, ensuring a page only sends to required and correct data. By using JSON web tokens (JWT), these communications can be kept secure and only accessible to the correct users. JWTs is an open standard and describes a method for transferring information securely as a JSON object [20]. By signing a JWT with a secret, or with a public or private key, the data can be verified as being secure. This allows web applications to be able to store user specific data without allowing unauthorised users to be able to view it, and still provide a fast experience without the need to constantly verify who a user is manually.

React JS is a framework targeted at providing high performance web apps, while allowing code to be developed in an easily maintainable, modular format [21]. This allows complicated web applications to be built while maintaining good object-orientated programming practices. React allows for custom component creation through the use of JSX and TSX programming languages, opening up the possibility of unique, rich interfaces. React was created by Meta (formerly Facebook) for use on the site Facebook for this reason. It comes with many considerations to code stability, such as downward data flow, and towards faster rendering with the introduction of its virtual DOM feature [22].

Facebook uses react in their Facebook Ads Manager, and on their mobile platform versions of the site. By using react, individual parts of the site can update based on user inputs – for example, a search bar can fill with results as a user begins to type, without reloading the whole page [23].

These various tools and methodologies allow for fast and secure webpages to be created, allowing the web to develop into an efficient communication tool.

4 Virtualisation

Virtualisation refers to the process of splitting computational resources to provide one or more execution environments. One way of achieving this is with the use of a virtual machine, which is an isolated instance of a runtime environment. Virtual machines are used when the tasks required to be executed may be risky to a host machine, and by isolating an environment, ensures the risk is mitigated [24]. Virtual machines often need a long boot time, and needs additional software to create and run, called a hypervisor. Virtual machines can be useful in cloud computing as one server

can be running many virtual machines each with their own virtual CPU and resources, meaning less hardware is required to create more systems.

Docker is a piece of software aimed at easing development by using containers, a standalone package that has everything needed to execute the software contained within it. This allows the software to run exactly the same no matter what system it is being run on [24]. It does this without the need for additional environments, tying in closer with the host OS and effectively reducing the overhead needed to create and run a virtual machine. This closer link to the host OS does mean it has some more security risks than a virtual machine, however [25].

The main reason for the creation and adoption of containerisation is to allow for greater automation of development and deployment tasks [25]. Docker can be used to launch several applications simultaneously, meaning large complex applications can be deployed with ease. Continuous integration tools can also make use of containerisation, such as Drone.io, in order to automatically run testing on applications when commits are made to a repository.

While containerisation can be useful for this, and allow for fast and efficient development pipelines, the security concerns [25] mean virtual machines still have a place in cloud computing, offering a secure environment for individual server software to run without posing a risk to others.

5 Conclusion

This report investigated the differences between server-side and client-side scripting, providing brief examples of languages that could be used and submitting some code alongside the report as an example. The strengths of each style of scripting are presented and discussed.

The next section investigated the Node.js framework, discussing the efficiency of the framework and widespread adoption. The use of an event queue is discussed, concluding that the asynchronous nature of this queue allows for more responses to be processed by servers running this framework. Various issues, such as CPU-intensive tasks, are discussed as weaknesses of the framework.

A discussion of Ajax and React is presented, discussing the importance of fast and secure web communication in the modern internet. The use of JWTs is discussed, alongside React's capabilities to deliver highly responsive and interactive web pages.

Finally virtual machines and Docker are discussed and compared. While Docker allows for efficient development pipelines, virtual machines are still capable of providing secure environments for software to run in and have their uses.

These investigations have shown the importance of speed and robustness in an ever-developing internet, as more and more activities are being performed through the internet. Secure communication and instant responsiveness have proven to be key features of developed technologies and will likely continue to be targeted points of research and improvement as time goes on.

6 Bibliography

- [1] Cloudflare, "What do client side and server side mean? | Client side vs Server side," Cloudflare, 1 January 2023. [Online]. Available: <https://www.cloudflare.com/en-gb/learning/serverless/glossary/client-side-vs-server-side/>. [Accessed 27 March 2023].
- [2] B. Stock and E. Al, "How the Web Tangled Itself: Uncovering the History of Client-Side Web (In)Security," Saarland University, Saarland, 2017.
- [3] A. Klein, "DOM based cross site scripting or XSS of the third kind," *Web Application Security Consortium*, vol. 4, no. 4, pp. 365-372, 2005.
- [4] OWASP Foundation, "Cross Site Scripting (XSS)," OWASP Foundation, 1 January 2023. [Online]. Available: <https://owasp.org/www-community/attacks/xss/>. [Accessed 27 March 2023].
- [5] Mozilla, "X-Frame-Options - HTTP | MDN," Mozilla, 1 January 2023. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options>. [Accessed 27 March 2023].
- [6] Adobe, "Adobe Flash Player EOL Enterprise Information Page," Adobe, 31 December 2020. [Online]. Available: <https://www.adobe.com/uk/products/flashplayer/enterprise-end-of-life.html>. [Accessed 28 March 2023].
- [7] jQuery, "jQuery," jQuery, 1 January 2023. [Online]. Available: <https://jquery.com/>. [Accessed 27 March 2023].
- [8] S. Trnat and E. Al, "Performance Comparison of PHP and JSP as Server-Side Scripting Languages," in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, Leuven, 2008.
- [9] Cloudflare, "What is a ddos attack," Cloudflare, 1 January 2023. [Online]. Available: <https://www.cloudflare.com/en-gb/learning/ddos/what-is-a-ddos-attack/>. [Accessed 27 March 2023].
- [10] OWASP, "Types of Cross Site Scripting," OWASP, 1 January 2023. [Online]. Available: https://owasp.org/www-community/Types_of_Cross-Site_Scripting. [Accessed March 27 2023].
- [11] K. Lei, Y. Ma and Z. Tan, "Performance Comparison and Evaluation of Web Development Technologies in PHP, Python and Node.js," in *IEEE 17th International Conference on Computational Science and Engineering*, Chengdu, 2014.
- [12] H. Sun and E. Al., "Efficient Dynamic Analysis for Node.js," in *Proceedings of 27th International Conference on Compiler Construction*, New York, 2018.
- [13] Altexsoft, "Pros and Cons of Node.js Web App Development," Altexsoft, 1 January 2023. [Online]. Available: <https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/>. [Accessed 28 March 2023].

- [14] W3Techs, "Usage Statistics and Market Share of Node.js March 2023," W3Techs, 1 March 2023. [Online]. Available: <https://w3techs.com/technologies/details/ws-nodejs>. [Accessed 28 March 2023].
- [15] PayPal, "Node.js at PayPal," PayPal, 22 November 2013. [Online]. Available: <https://medium.com/paypal-tech/node-js-at-paypal-4e2d1d08ce4f>. [Accessed 28 March 2023].
- [16] N. Dragoni and E. Al, *Microservices: Yesterday, Today and Tomorrow*, Manhattan: Springer Charm, 2017.
- [17] Node.js, "Node v10.5.0 (Current) | Node.js," Node.js, 20 June 2018. [Online]. Available: <https://nodejs.org/en/blog/release/v10.5.0>. [Accessed 28 March 2023].
- [18] Mozilla, "Ajax - Developer Guides," Mozilla, 1 January 2023. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX>. [Accessed 28 March 2023].
- [19] Mozilla, "Getting Started - Developer Guides | MDN," Mozilla, 1 January 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started. [Accessed 28 March 2023].
- [20] Auth0, "JSON Web Token Introduction," Auth0, 1 January 2023. [Online]. Available: <https://jwt.io/introduction>. [Accessed 28 January 2023].
- [21] Meta, "React," MEta, 1 January 2023. [Online]. Available: <https://react.dev/>. [Accessed 28 March 2023].
- [22] Meta, "Virtual DOM and Internals - React," Meta, 1 January 2023. [Online]. Available: <https://legacy.reactjs.org/docs/faq-internals.html>. [Accessed 2023 March 2023].
- [23] M. Pruciak, "21 Amazing Examples Of React Web Development," Ideamotive, 5 January 2023. [Online]. Available: <https://www.ideamotive.co/blog/amazing-examples-of-react-web-development>. [Accessed 28 March 2023].
- [24] Cloud Acadamey, "Docker vs Virtual Machines: Differences you should know," Cloud Academy, 5 October 2022. [Online]. Available: <https://cloudacademy.com/blog/docker-vs-virtual-machines-differences-you-should-know>. [Accessed 28 March 2023].
- [25] T. Combe and E. Al., "To Docker or Not To Docker: A Security Perspective," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 54 - 62, 2016.

OJordan - Assignment1

GRADEMARK REPORT

FINAL GRADE

70/100

GENERAL COMMENTS

Instructor

CSvsSS (25):19

node (25):18

AJAX/React/JWT (25):15

Docker (25):18

Feedback:

=====

Some good discussion of security issues. You could have provided more discussion of client side scripting on dynamic UI, offloading processing to the client and provided some examples of scripting.

The discussion of node was a bit brief and there were some benefits that should have been considered such as package management.

AJAX/React was bit brief but the basic concepts are explained, would have liked an example.

Virtualisation needed to provide more analysis of the benefits and drawbacks.

PAGE 1

PAGE 2

PAGE 3

PAGE 4
