

web

by Saeed Mohamed

Submission date: 31-Mar-2023 11:55PM (UTC+0100)

Submission ID: 201250200

File name: 42775_Saeed_Mohamed_web_488464_1752421090.docx (1.78M)

Word count: 2791

Character count: 15735

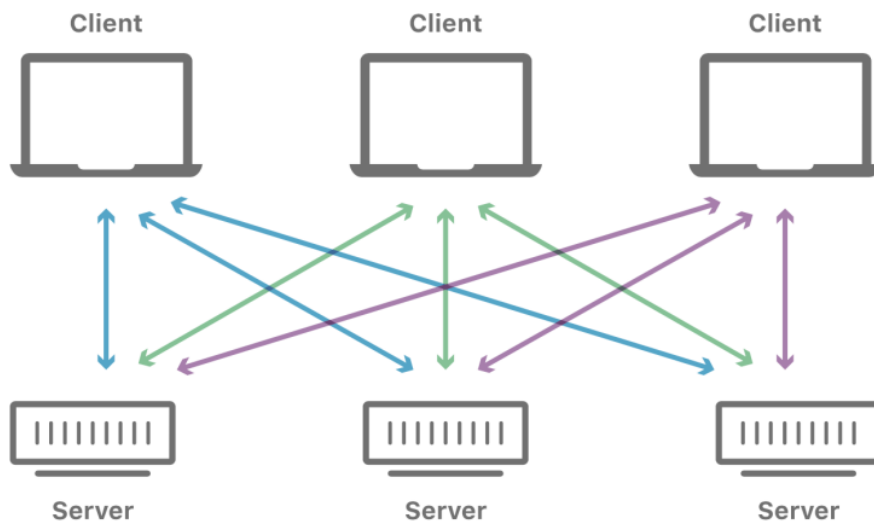
Client-side scripting.

Client-side scripting is executed by browsers. It is employed when browsers contain all code. The source code is utilized to be transmitted from the webserver to the customer's machine over the web and execute directly on the browser. It additionally serves for confirmation and user event functionality.

It permits increased interactivity. It typically executes multiple operations without contacting the user. It is incapable of connecting to databases on a web server. These scripts are unable to access the file system of the web browser. The pages are modified based on the user's preferences. Additionally, it can be used to create "cookies" that store information on the user's computer.

Each client device that accesses and browses the Internet transmits requests to multiple servers. Servers may accommodate numerous devices used by clients simultaneously.

Multiple consumers and servers engage in interaction:



Server-side scripting.

Web servers are utilised for server-side scripting execution. Essentially, they are used to generate dynamic documents. It is also able to access the webserver's file system. A web server is a server-side environment that utilises a scripting language.

Scripts can be composed in any of the available server-side scripting languages. It is used to retrieve and generate dynamic page content. It is used to necessitate plugin downloads. In general, loading times are speedier than client-side scripting. When you need to store and retrieve information, you will use a database to hold it. It can utilise extensive server resources. It reduces the overhead of client-side computation. The server transmits pages in response to the user/client's request.



Data travels back and forth among a client and server, and scripts can execute on both the client and server sides.

Differences between client and server?

We are aware that server side and client side are terms used in web development to characterise where code runs. But what exactly is scripting? Scripting refers to programming languages that automate tasks typically performed by humans. Consequently, client-side scripting executes code such as JavaScript on your mobile device or computer, whereas server-side scripting operates on the backend. This enables the delivery of dynamic content to web pages based on the client device. For instance, when you enter your username and password into a website, the request from the client side is sent to the server, where it is processed. The server responds to the client with the results upon completion.

Client-side scripting works with the following languages:

CSS

HTML

JavaScript

In addition, it transmits requests through web browsers such as Firefox, Google Chrome, Internet Explorer, and Safari. This enables you to obtain server information in a matter of seconds.

Server-side programming languages include PHP, JAVA and C#

2) Investigate the node.js framework and critically evaluate the benefits and drawbacks of using it.

Node.js is an open source, cross-platform runtime environment and library used to execute web applications outside of the client's web browser.

It is used for server-side programming and is predominantly deployed for non-blocking, event-driven servers, such as traditional web sites and back-end API services, but it was originally intended for real-time, push-based architectures. Each browser has its own implementation of the JS engine and node.js is developed on the V8 JavaScript engine of Google Chrome. This means that entire sites can be run using a unified 'stack,' making development and maintenance straightforward and providing you to concentrate on meeting the business objectives of the project. Important to note about Node.js is that, unlike conventional application software, it is not a framework nor a library, but rather a runtime environment.

History of Node.js.

Ryan Dahl originally developed Node.js in 2009. Previously, Dahl had criticised the limited options offered by prevalent web servers and coding practises. At the time, servers battled to manage many continuous links, and codes either halted the entire process or suggested the need for multiple stacks. All these obstacles hindered companies' ability to create diverse goods that fulfilled the needs of large numbers of users.

In reply, Dahl developed Node.js to enable devs to use JavaScript for server-side scripting and simplify the creation of web applications around just one programming language.

Nodejs architecture.

Node.js's mechanics are what contribute to its prominence among developers. In contrast to many alternative runtime environments, which employ multi-threaded processing models, Node.js only uses a single thread. In multithreaded processor environments, each server is entitled to a limited task array. Therefore, whenever a server receives a request, it pulls a thread from the pool and allocates it to that request in order to handle its processing. In this instance, processing is synchronous and sequential, meaning that a single operation is executed at a time.

Theoretically, a single-thread architecture can perform and scale more rapidly and efficiently than multiple-thread configurations. This was Ryan Dahl's intention when he created Node.js, and it's a major reason why it's so popular with web application developers.

Node.js work environment.

What exactly is JavaScript?

JavaScript is the underlying programming language for Node.js. It is a multi-paradigm, high-level programming language with syntax, dynamic typing, prototype-based object orientations, and first-class functions.

These capabilities allow JavaScript to transform a static web page into an interactive one by adding features such as search windows, embedded videos, and news feed refresh tools. These features seek to enhance the user experience by encouraging intuitive interaction.

Node.js modules.

Multiple 'modules' are contained within their own contexts in Node.js so that they are not in conflict with additional modules or contaminate the broad reach of node.js. This is indispensable for open-source software.

A module in Node.js is a basic or complex feature that is organised into JavaScript files and is repeatable through the entire application. Within Node.js, there are a trio of components: Core Modules, Local Modules, and Third-Party Modules.

Core Modules include Node.js's fundamental, bare-bones functionality. They form a comprised of Node.js's binary distribution and start instantly when a node operation begins.

Local Modules are Node.js application-specific modules. They include distinct and additional functionality in separate files and folders in addition to the package of primary functionality. Locally modules can also be packaged and disseminated to the Node.js community at large.

A third-party module is pre-existing code written by a third party that can be integrated into a Node.js application in order to widen or introduce new features or functions.

Worker threads of Node.js

Node.js's worker threads are beneficial for executing intensive JavaScript operations. The worker threads module enables parallel execution of JavaScript algorithms, speeding up and improving the overall process. It accomplishes this by utilising a pool of existing 'workers' to manage new and incoming tasks.

Node.js event loop.

Flexible and compartmentalized, worker threads assist with CPU-intensive tasks without disturbing the main Node.js thread. Even though JavaScript is single-threaded, the event loop enables Node.js to perform non-blocking I/O operations by transferring tasks to the system processor if possible. Since many current processors are multithreaded, they are capable of handling multiple background processes.

In general, the event loop is a mechanism that waits for and dispatches programme events or messages. Event loops are the fundamental control flow constructs in Node.js. When a request is about to be handled, for instance, it is placed on the event loop and processed as soon as it is available to be processed.

Instead of managing the system on its own, Node delegated the responsibility. Due to this behaviour, Node does not actively wait for this task to complete and can process other requests in the interim. The event loop makes Node.js more efficient and effective than competing technologies

Advantages and disadvantages of NODE.JS

Real time collaboration.

Collaboration in real time activities provide numerous software options for co-browsing, project management, video and audio conferencing, application sharing, and collaborative document editing, among others. Slack, Trello, and Google Docs are well-known and successful examples.

As with real-time conversations, Node's asynchronous and event-driven architecture is well-suited for collaboration applications. In these applications, numerous requests and events occur simultaneously. Multiple users can, for instance, modify the same paragraph, make comments, post messages, and attach media from various devices. Changes to a piece of content may be implemented only after a series of events, where each phase is dependent on the one preceding

NODE.JS on streaming.

Node.js's native Stream API is superb for building of streaming applications. It has a visible and written stream protocol that can be easily observed and handled.

Stream instances are essentially Unix pipelines that permit portions of an application's executable code to be transmitted to the local machine, maintaining an open connection for new components to download on demand.

NODE.JS MICROSERVICES ARCHITECTURE.

Node.js is an outstanding platform for creating microservices and establishing simple APIs for interacting with them. Specifically, the Node.js repository contains the Express and Koa frameworks, which make it simple to deploy multiple server instances for each microservice and design their routing addresses.

Since 2013, PayPal has utilised Node.js to fuel its microservices architecture, thereby successfully capitalising on these advantages.

Why should we use NODE.JS.

JavaScript has become one of the most utilized languages for front-end programming, and almost every web developer is conversant with it. Therefore, it is significantly simpler for developers to transition to Node.js on the backend. It requires less time and effort to understand and work. Node.js is entirely neutral, so you can create anything from beginning and customise every aspect. It can perform fundamental duties, but a fresh installation provides only the bare minimum, allowing you to add features with fewer restrictions. Node.js gives you a free I/O system that enables the concurrent processing of multiple queries. Incoming requests are queued and executed in rapid succession. As a result, your application will consume significantly less system RAM, achieve high scalability, and run quicker.

Disadvantages of Node.js

Inconsistency API

Node.js's lack of consistency is one of its most significant drawbacks. The Node.js API undergoes frequent and frequently incompatible modifications. When this occurs, developers must modify the existing code base to make it compatible with the most recent version of the Node.js API. Evidently, this may result in additional work that detracts from your application improvement objective.

Lacking tools

Even though Node.js is stable at its core, many packages in the Node package manager (npm) registry are still of poor quality or are not adequately documented. As most of the ecosystem is comprised of open-source software, certain applications have not been screened for quality and may not adhere to coding standards. The structure of npm makes it more difficult to identify reliable programmes. Therefore, you may need more experienced developers who can identify trustworthy tools.

AJAX.

AJAX stands for Asynchronous JavaScript and XML.

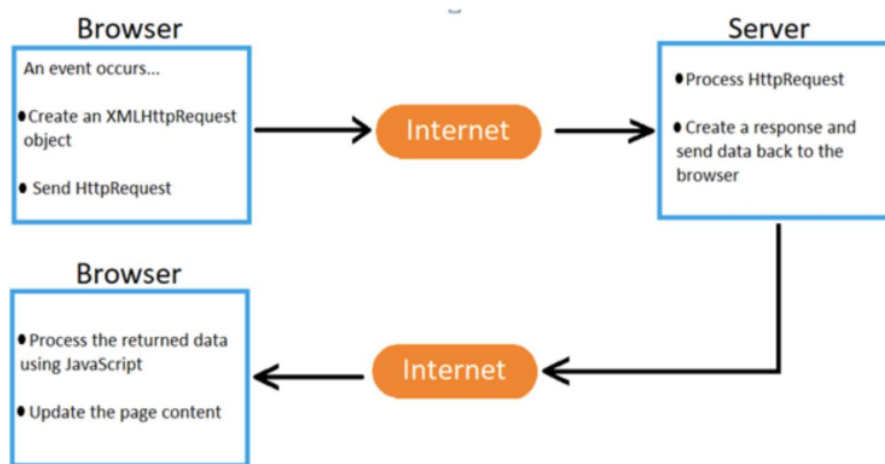
Ajax is a concept, not a programming language or a tool. Ajax is a client-side script that communicates with a server/database without requiring a postback (the process of transmitting a web page to the server for processing) or a page refresh. Ajax is "the method for exchanging data with a server and updating portions of a web page without having to reload the entire page. "AJAX enables asynchronous page updates by exchanging data with a web server in the background. This means it is possible to alter portions of a web page without having to reload the entire page.

AJAX developers can:

- Update a website without page reloading
- Obtain information from a server after the website has loaded
- Obtain information from a server after the page has loaded.
- Send information to a server in the background.

Ajax is a method for developing better, quicker, and more intelligent web applications using XML, HTML, CSS, and JavaScript. Early in the 21st century, web pages contained only page information. Now, however, the developer employs the AJAX technique, which enables the page to automatically reload with the most recent updates. So that the user can readily incorporate new information. Ajax provides a one-to-one communication among client and server for a secure connection. To help secure user data and transmit it to the server in a safe way. Therefore, we can conclude that Ajax not only secures the data, but also improves efficacy. Ajax is an open-source web development framework. Anyone can modify the Ajax code to implement improved and advanced functionality in their application. The programmer may update the Ajax software to provide their application with the most recent functionality and flexibility.

HOW AJAX WORKS.



- 1-An event takes place on a web page.
- 2-JavaScript generates an XMLHttpRequest object.
- 3-The XMLHttpRequest object requests a web server.
- 4-The server processes the request. 5-The server transmits the web page a response.
- 6-JavaScript parses the response
- 7-The correct action (such as page update) is carried out by the JavaScriptAJAX Web Application Model.

REACT.

React.js is an open-source JavaScript library used to develop user interfaces for single-page web applications. It manages the view layer for web and mobile applications. React also allows us to design UI components that are reusable. React was initially developed by Facebook software engineer Jordan Walke. React was implemented for the first time on Facebook's newsfeed in 2011 and on Instagram.com in 2012.

React enables developers to construct large, data-changing web applications without having to reload the page. React's primary purpose is to be quick, scalable, and uncomplicated. Only application user interfaces are supported. This matches the view within the MVC template. It is compatible with a variety of JavaScript libraries and frameworks, including Angular JS in MVC.



- Included in React.js properties are the following:
- React.js is a declarative language.
- React.js is basic.
- React.js is founded on components.
- React.js supports server-side rendering.
- React.js is comprehensive.
- React.js is quick.
- React.js is simple to master.

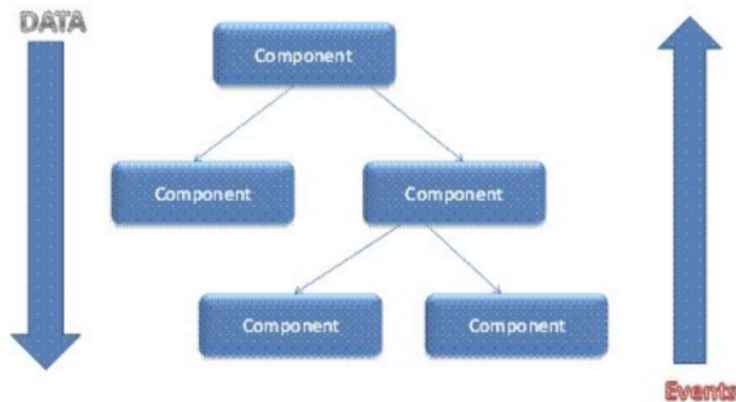
JSX

Instead of ordinary JavaScript, JSX is used for templating in React. JSX is an easy JavaScript that permits HTML quoting and renders subcomponents using HTML tag syntax. HTML syntax is converted into React Framework JavaScript commands. We are also able to write in unadulterated JavaScript.



WAY DATA FLOW.

In React, a set of fixed variables are given as HTML element properties to the component renderer. The component are unable to explicitly change any properties, but it can transmit a callback function through which modifications can be made. This entire process is referred to as "properties flow downward; actions flow upward."



WHY REACT?

Now, the primary query before us is why one should utilise React. There are numerous open-source platforms that simplify the development of front-end web applications, such as Angular. Let's examine the advantages of React over other competing technologies and frameworks.

1. simpleness

ReactJS is easier to comprehend immediately. The component-based strategy, clear lifecycle, and use of plain JavaScript make React extremely easy to understand, construct and support professional web (and mobile) applications. React employs a special syntax called JSX that enables the mixing of HTML and JavaScript. This is not required; developers may continue to write in ordinary JavaScript, but JSX is significantly simpler to use.

2. Simple to learn

React is readily understood by anyone with a fundamental understanding of programming, whereas Angular and Ember are referred to as "Domain-specific Language," implying that they are difficult to learn. To react, only rudimentary knowledge of CSS and HTML is required.

3. Native Method

React can be employed to build mobile applications. (React Native). And react is a fervent advocate of code reuse, so extensive code reuse is supported. Thus, we can simultaneously develop IOS, Android, and Web applications.

HOW REACT IS USED TO CREATE MOBILE APPLICATION.

1. Creating simple HTML l file

```
01. <!-- Load React Libraries -->
02. <!--
    - Note: when deploying, replace "development.js" with "production.min.js". --
    >
03. <script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin>
    </script>
04. <script src="https://unpkg.com/react-dom@17/umd/react-
    dom.development.js" crossorigin></script>
```

import react library

```
01. <html>
02.   <head>
03.     <title>Let's React with npm</title>
04.   </head>
05.   <body>
06.   </body>
07. </html>
```

html documents look like this.

```
01. <html>
02.   <head>
03.     <title>Let's React with npm</title>
04.     <!-- Load React Libraries -->
05.     <!--
    - Note: when deploying, replace "development.js" with "production.min.js". --
    >
06.     <script src="https://unpkg.com/react@17/umd/react.development.js" cross
    </script>
07.     <script src="https://unpkg.com/react-dom@17/umd/react-
    dom.development.js" crossorigin></script>
08.   </head>
09.   <body>
10.   </body>
11. </html>
```

Place holder

```
01. <body>
02.   <div id="root"></div>
03. </body>
```

creating react component

```
01. class HelloClass extends React.Component
02. {
03.     render()
04.     {
05.         return React.createElement('div', null, 'React without npm');
06.     }
07. }
```

calling react component

```
01. ReactDOM.render(
02.     React.createElement(HelloClass, null, null),
03.     document.getElementById('root')
04. );
```

complete code

```
01. <html>
02.   <head>
03.     <title>React's React</title>
04.     <!-- Load React. -->
05.     <!--
06.     - Note: when deploying, replace "development.js" with "production.min.js". --
07.   >
08.     <script src="https://unpkg.com/react@17/umd/react.development.js" cross
09.   </script>
10.     <script src="https://unpkg.com/react-dom@17/umd/react-
11.   dom.development.js" crossorigin></script>
12.   </head>
13.   <body>
14.     <div id="root"></div>
15.   <!--
16.   - This is embedded JavaScript. You can even place this in separate .js file --
17.   >
18.     <script>
19.       window.onload = function()
20.       {
21.         class HelloClass extends React.Component
22.         {
23.           render()
24.           {
25.             return React.createElement('div', null, 'React without
26.           }
27.         }
28.         ReactDOM.render(
29.           React.createElement(HelloClass, null, null),
30.           document.getElementById('root')
31.         );
32.       };
33.     </script>
34.   </body>
35. </html>
```

importance of JSON Web Tokens.

JavaScript Object Notation (JSON) is a text-based format for transmitting data between web applications. It maintains information in an easily accessible format for both users and machines. It can be used by any programming language as a data format and is rapidly superseding XML as the preferred syntax for APIs.

Now that you are familiar with JSON as a data text format, you may be wondering: What is a token? Simply stated, a token is a string of data that represents another entity, such as an identity. In the context of authentication, a non-JWT-based token is a string of characters that permits the recipient to verify the identity of the originator. The key distinction here is the absence of significance within the characters themselves.

JWTs can use a public/private key pair in the form of an X.509 certificate for signature, making them more secure. Using the HMAC algorithm, a JWT can also be symmetrically signed by a shared secret. And while SAML tokens can utilise public/private key pairs like JWT, signing XML with XML Digital Signature without introducing obscure security flaws is much more complex than signing JSON.

JWT is utilised on an internet-wide extent because it is simpler to process. This means that it is simpler to process on the devices of users, particularly mobile devices. JWT main flaw is security that is contained within JSON.

FINAL GRADE

53 / 100

GENERAL COMMENTS

Instructor

CSvsSS (25):14

node (25):19

AJAX/React/JWT (25):20

Docker (25): 0

Feedback:

=====

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

The discussion of node.js was good but the analysis was a little unstructured and needed a bit more depth.

AJAX/React discussion was quite good, should have expanded upon JWT.

No discussion of Docker provided.

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13
