**React Overview**

Over the last few years, the options for creating fast interactive front ends for applications have exploded. Technology big ons like Facebook and Google have been vying to become the developer's choice from creating massive and rich Application Interfaces, React JS is one of these technologies.

Developed by Facebook, and used extensively in their Facebook and Instagram applications. In fact, react JS was first used on the Facebook newsfeed back in 2011, and has evolved into one of the most popular ecosystems in JavaScript development, Companies such as Twitter, Airbnb, Netflix, Uber, and a whole host of other household names use react JS in their applications.

So what is it? Many assume that it's a framework, it isn't. It's a JavaScript library to help developers create reusable, interactive and stateful UI components. It can be seen as the V or the view part of any application, including MVC architecture. React JS makes no assumptions about the rest of the stack that serves the application. It simply takes data and displays it.

So what does react JS do? As I've already stated, it simply takes data and displays it. I say simply, but as you've guessed, it's not that simple. At a more technical level, most applications have react JS performing work on the client computer to update what the user sees as they interact and use the application. New views and updated data are seamlessly created and displayed with very few overheads, like additional network calls. A concept called the virtual DOM is used to do this.

React JS can also work on the server, rendering the page before delivering it to the client. It's possible to have an application, having some parts rendering from the server and other parts rendering in the client. When a normal application renders in a browser, it creates the Document Object Model, a hierarchy of the webpage structure that can be manipulated with JavaScript. Facebook found that, if they used the copy of the DOM kept in memory, called the virtual DOM, they could make fast and fluid UI's. The idea is quite simple.

Take the react JS code and the user interactions to make updates to the virtual DOM. Then use processes called diffing and reconciliation to make the actual DOM reflect the changes in the virtual DOM saving the developer the trouble of having to code these interactions, attribute manipulations and events handling. Another benefit of this, is that objects and pages don't need to refresh when something changes, which is more efficient.

Thankfully, we don't need to get into too much detail about how all that happens. We just need to be aware of the process that is happening. And just so you know, the reconciliation engine is called fiber and has been used since react version 16. The build process for deploying a react JS application produces the files needed to deploy it. As you would expect, this is the HTML file to create the initial DOM and its associated CSS and JavaScript files. The JavaScript is mostly the react JS code that has been transpiled into finale JavaScript.

So when the user requests the URL, the HTML is delivered with these associated files. The JavaScript file runs, producing the virtual DOM and the render for the initial view. As the user interacts with the application, various parts of the JavaScript code execute, updating the virtual DOM and therefore the actual DOM through the diffing and reconciliation process. If you're familiar with the OWASP Top 10 Vulnerabilities, then you're probably already thinking about security issues with react JS.

Most obvious of these is using components with known vulnerabilities. React JS application has hundreds of dependencies, and it's important to ensure that these are kept up to date. Npm to audit command can then be used to help identify what is known to be vulnerable. and the audit fix command can be used to safely update them.

However, if an update would introduce breaking changes, then a bigger pill rebuild may be required. At the time of recording, there were no known cross-site scripting vulnerabilities in react JS's core code and developers should always be aware of the dangers of incoming data and whether it should be trusted.

**Introduction to React  
  
Development Environment Set-Up**

So what needs to be installed to use React JS? First off, you'll need a text editor. Our preference is for VS Code as it is open source, and it has lots of features that make a developer's life easy, including syntax highlighting, linting and auto complete. As you've probably guessed, there's a high reliance on dependencies. So our package manager, in our case, NPM, is needed.

Therefore you'll need to download and install NodeJS, which includes Node package manager. To set up your application, there are two options. The first is the Zero-config option, and uses node package runner called create-react-app, to create a skeleton application, pack full of features and tooling for testing which is much easier than the other option.

That other option is to set up manually. This requires to set up a bundling tool such as Webpack, a translation tool, such as Babel, testing tools, and all the other desirable stuff that we need to build a good application. Node JS, which includes NPM, the Node Package Manager, can be downloaded from nodejs.org, and has versions for all the major operating systems.

Having downloaded node and installed it, I then have opened Visual Studio code and gone to the terminal to check that node and NPM are both recognized commands on my system. I've done this by using the minus V switch, which displays the version number. Having done this, I can now use the NPX command. This allows me to run a node package runner called create-react-app.

And I'm going to use my first app as the name of my project. When I press Enter here, it will install all of the files and dependencies that are needed for me to get my first app, react application up and running. So now the installation is complete. I'll follow the last piece of advise that it gives to CD to my first app and type NPM start. This as you can see, opens up my browser and displays the default react app. And there we go. This comes from react application, as delivered by create-react-app.

**Development Tools**

Accessing the Developer tools in my browser allows me to see the Domino has been built and also see the references to the bundle file a Webpack is created to render my app. These are shown in the script tux here. I can also get access to the files that are being used to create those bundles. I can do this by switching to the Sources tab and clicking through the folder structure.

You can see here that I can see the App.js file and I could set breakpoints and watches, and then reload the application and debug it as I would do a normal JavaScript application. If I switched to the Console tab, you can see that I'm being told that there is a development tool available. If I follow the link here, it takes me to the React page, about the development tools.

And as I'm using Chrome it gives me a link to the Chrome web store. I can add this to Chrome and then go back to my developer tools. These won't show until I close the tools down and reopen them. Although you can see that there's an icon appeared in the top right-hand corner next to the address bar.

This time, you can see that I've got now a Components tab and a Profiler tab. We'll go into these into more detail later but this just lets me access the actual react components that are being rendered as part of my application, rather than the raw JavaScript. It also lets me access, things like seeing the matching DOM element.

So clicking on it, opens up the elements and shows me exactly where the app component is rendered in my DOM. I can also ask it to output my components to the Console. If I click on this button here and then go to the Console and then expand this, it allows me to access some of the properties that are attached to my app components. The final button takes me to the part of the files that actually create that component. And as you can see, this is same as the App.js file that I showed earlier in this video.

React Project Anatomy

React and the DOM