Andreas Frick Toresand

This is my report on my React Portfolio

While developing my website in React, which is a JavaScript library for building user interfaces, I decided not to use a CSS (Cascading Style Sheets, this is how you style web pages/applications) framework, mainly because I wanted to practice implementing styled-components (which is a React library for writing CSS) and give my website my own set of CSS values and styling. I am however interested in trying a framework out since it makes it easier and gives it a good standard.

Olika CSS-ramverk:

- Bootstrap
- SemanticUI
- Materialize (uses Material Design, which is a design language that Google developed in 2014)

Eventually, I will definitely try Bootstrap, since it is considered the best responsive CSS framework. Because it is so popular, and designed for front-end development specifically which is what I am studying, there are a lot of resources available (such as tutorials, extra plugins, etc.) that will make the work way easier.

Another one that seems interesting is Semantic UI, a framework that is aimed at making website building process more semantic(logical). I like that idea because you get to style your website and also practise good semantics. Semantic UI also has readily available information and guides on how to get started which always is a good thing of you get stuck or need inspiration.

SASS eller LESS:

Since styling isn't my strong suit, I will definitely dive into and try all ways that are easier and more helpful in that department, such as a preprocessor. A preprocessor is a program that takes input data, processes it to produce output that is then used as input to another program. A CSS preprocessor does this with CSS. SASS and LESS are both CSS preprocessors. SASS stands for ***syntactically awesome style sheets*** and LESS stands for ***Leaner Style Sheets.***
Sass was designed to both simplify and extend CSS. Less was designed to be as close to CSS as possible, and as a result existing CSS can be used as valid Less code so I am already close! Advantages of using a preprocessor is a cleaner code with reusable pieces and variables and it saves you time and is easier to maintain. Just what I am looking for!

Javascript, grundläggande syntax:

```
function isManInSweden(personnumber) {
 const persNumbString = personnumber.toString();
 if ((persNumbString.length === 10) && (persNumbString.slice(7, 9) % 2 !== 0)) {
   return true;
 }
 return false;
}
```

This is a function that checks if a person comes from sweden by checking the social security number, more precisely of the number next to last is uneven (as they are in sweden).

We create a constant variable called persNumbString and set that to combine the arguments that gets put in to the function to the toString-method.

The toString-method turns every argument it receives into a string, and we need to turn the argument into a string in order to use the slice-method later on.

Then there is an if-loop that first checks that the argument passed in is exactly 10 numbers long by using the .length property and using three strict equality operators (equal signs) pointing to the number 10. Then we use the logical AND operators (&&) to use another method: the slice()-method.

The slice() method extracts a section of a string, from the beginning to the end (end not included)based on the values put in to it and returns it as a new string, without modifying the original string. In this case it cuts out the number next to last (000000000) (since the index always starts at zero) and the modulo operator(the % sign), you can call it the "remainder operator", checks if the number the slice-method cut out is an uneven number by using the exclamation point, because it reverses the value. It then returns true if these conditions are met. Otherwise the function returns false and the person is not from sweden.

(However, the function does not check if the argument passed in is actually all numbers)

Hur Javascript körs i webbläsaren:

A JavaScript engine is a computer program that executes JavaScript code. A popular example of a JavaScript Engine is Google's V8 engine.

The Engine consists of two main components:

- Memory Heap — this is where the memory allocation happens
- Call Stack — JavaScript is a single-threaded programming language, which means it has a single Call Stack. The Call Stack is a data structure which records basically where in the program we are. Therefore it can do one thing at a time.

We also need Web APIs, which more or less are functions provided by the browsers. And while I am working with React I also get React's APIs.

Javascript-bibliotek:

A Javascript library is a collection of resources in a computer memory that can retrieve stored information even if the computer has been shut off and on, with pre-written Javascript. Two examples I have worked with in this project is Node.js and React.js.

Node.js is a runtime environment (node's package system, npm, is the largest ecosystem of open source libraries in the world) that executes JavaScript code outside of a browser, meaning something you could run on your machine as a standalone application, and it also uses Google's V8 engine. So what was I using when I was using React?

React.js is basically a frontend library but is considered by many developers to be a framework. It focuses on composable user interface (UI) based on component-based architecture. One component was, for example, my "Home" page. Another one "Contact" etc. All of them combined. It makes dom-manipulation much faster and easier and is quite easy to learn thanks to its JSX syntax (which, put simply, means it allows us to write HTML in React.) You tell React what state you want the UI to be in, and it makes sure the DOM matches that state via a library such as ReactDOM. For instance, to see what the website would look like in the browser when I was working on it, I would give the command "npm start" (from Node.js) and my text editor, using React, connected me with the browser. All the changes I made in Visual Studio Code(the text editor I used) was instantly upgraded to the browser. I am pretty sure you have used or at least seen React, since It was developed and maintained by Facebook, and Facebook and Instagram use it.

Javascript-ramverk:

I said that React is viewed as a framework though it really is a library, but it is used pretty much in the same way. A framework serve as a skeleton for single page apps, and makes it easier and faster to write code and takes care of the structure with prebuilt patterns and functions which makes it more efficient. This leads into less room for mistakes. And since it is supported by large communities, it's safer and often open source and free.
But there are some cons. Firstly : You're not "really" learning how to code. There is a danger to just learning the framework and not learning how to develop. If you're not already an expert, it is important to understand the code which powers the framework.
Also, updates can introduce issues and lastly if any new developers are introduced to the project they will need to understand the framework in order to contribute, which can take time.

Two examples of frameworks are Angular.js and Vue.js. Angular is often compared to React and is known for quick code production, easy testing of any app part and two-way data binding (changes in the backend are immediately reflected on the UI).

Vue.js (vue.2.0) takes features from other frameworks such as Ember, React and Angular. It is proved to be faster and leaner, comparing to React and Angular 2.0.

DOM:

To build a website, you create a HTML document. You use CSS to style that document with different styling-commands. In order to view that styled HTML document you need a browser (Chrome, Firefox etc) which is a program that renders or generates all the content and structure into a webpage. This representation of the HTML document is called a Document Object Model, or the DOM.

In React, for every DOM object, there is a "virtual DOM object." A virtual DOM object is a lightweight representation of a DOM object and not a "real" DOM object.
When you render a element all the virtual DOM objects gets updated, and this happens really quickly.
Before each update, React takes a "snapshot" of the virtual DOM, then compares it with the updated version, and figures out exactly which virtual DOM objects have changed.
Then React updates those objects, and only those objects, on the real DOM.
This process is called "diffing." This is what makes it so fast and efficient! This is exactly what i was describing when I was talking about instantly seeing the changes I made in VS Code in the browser window.

Some argue that virtual DOM actually is slower than carefully crafted manual updates. That it's not really about performance but about developer convenience.

Vad är AJAX? Vad används det till? För-, och nackdelar?

AJAX is is a client-side script, which means that it is source code that is executed on the client's browser instead of the web-server. It is a method that can exchange data with a server, and updating parts of a web page – without reloading the entire page. In a similar way to the way that virtual DOM only updates the necessary object on the real DOM, instead of all the objects.
This allows for the creation of faster and more responsive web applications.

Javascript is implemented differently depending on what browser you have, and since AJAX highly depends on JavaScript this can cause problems. Browsers that does not support Javascript can't use the functions. No problem for me though, since JS is the only programming language is use and even want to use! (A tattoo is being considered, but maybe a few more years of practice so I can back it up…)

HTTP1.1 / 2.0:

In order to use any of this, you need to be able to access the web and your device needs to communicate with other devices: you need HTTP!
The Hypertext Transfer Protocol (HTTP) is an application protocol for , and the foundation of data communication for the World Wide Web. A protocol is a set of rules over the data communication mechanisms between clients (for example web browsers used by internet users to request information) and servers (the machines containing the requested information).
For the last 15-16 years the HTTP1.1 has been the working version. In February 2015, the Internet Engineering Task Force (IETF) HTTP Working Group revised HTTP and developed the second major version of the application protocol in the form of HTTP/2.
Basically, HTTP/2 is a better, more advanced version of HTTP1.1 and focuses on flow control, upgrade and error handling work for developers to ensure high performance web-based applications.

"…we are not replacing all of HTTP – the methods, status codes, and most of the headers you use today will be the same. Instead, we're re-defining how it gets used "on the wire" so it's more efficient, and so that it is more gentle to the internet itself…"Mark Nottingham, Chair the IETF HTTP Working Group and member of the W3C TAG.

This is a natural progression considering that the use of the internet and web-based applications is growing steadily by the minute.

http://addefreak.surge.sh/