# Individuell Projektuppgift 2 FE18 - Redogörelser.

William Nordqvist.

## **CSS-ramverk**:

Som Front-End utvecklare idag är det populärt att använda sig av olika former av CSS-ramverk. Utan den hjälp vi får från dem olika ramverken hade vårt jobb som utvecklare varit mer tidskrävande och problematiskt. Precis som dem flesta Front-End ramverk/bibliotek är CSS-ramverk i ständig förnyelse och utveckling.

Några av dem populäraste CSS-ramverk som finns i dagsläget är:

1. **Bootstrap** - Anses vara det populäraste och bästa CSS-ramverket för Front-End utvecklare idag. Då det underlättar arbetet betydligt med att bygga hemsidor i mobile first med responsiv design. En anledning till att Bootstrap blivit så pass stort är pga av den delning som finns på internet, Samt att det är väldigt lätt att implementera gridsystem och färdiga knappar mm. Det finns extremt många tutorials, plugins och färdig kod för Bootstrap. Implementationen är bara en google sökning ifrån. Detta är det ramverk som jag personligen har kommit i kontakt med mest. Det finns många fördelar med det. Dock använde jag mig av det mer i början av utbildningen, då det var lättare och effektivare att bygga upp kod. I dagens läge tycker jag att det är opersonligt och vill därför undvika det.

2. **Bulma** - Är ett öppet och gratis CSS-ramverk. Precis som Bootstrap. Det är ett ramverk som har blivit känt för vara väldigt tidseffektivt och ett lätt ramverk att lära sig. Första och främst använder sig Bulma av bra User interface komponenter som tex. Navbars, dropdowns och Panels. Om du vill titta mer på Bulma komponenter så rekommenderar jag deras sida - https://bulma.io/documentation/components/. Som man ser på sidan är deras komponenter modulära vilket gör det väldigt lätt att importera dem komponenter du vill använda. Sist men inte minst är det också ett väldigt lättläst ramverk. Av det jag sett är jag imponerad av Bulma då jag gillar det modulära tänket. Tyvärr har jag inte hunnit testa ramverket ännu.

3. **Materialize** - Är ett populärt CSS-ramverk som använder sig utav “Material Design”. Material Design introducerades av Google 2014 som deras designspråk och har använts bland annat till att utveckla och designa Gmail, Google Docs och Google Drive. Om man är intresserad av att börja använda Material Design är just Materialize ett rekommenderat CSS-ramverk att börja med och dem beskriver sig själva att vara “Ett modernt responsivt Front-End Ramverk baserat på Material Design. Även detta ramverk samt Material Design är nytt för mig och har därmed inte haft möjlighet att testa.

## **SASS/LESS:**

Att skriva i CSS kan i mångas ögon anses som långtråkigt ,upprepande och tidskrävande. Där utav har SASS och LESS skapats som en extension till CSS. SASS och LESS sammanfattningsvist två olika sätt att skriva CSS på ett effektivare sätt där du slipper återupprepning. Det blir med andra ord ett mer CSS DRY. Vilket i sin tur resulterar i att koden blir enklare att läsa, tidseffektivare att skriva koden och lättare att organisera. Både SASS och LESS är “backward compatible” vilket betyder att du lätt kan konvertera koden till vanlig CSS igen bara genom att döpa filen till .css istället för .less eller .scss. I det stora hela är SASS och LESS lika varandra, det som skiljer dem åt är att LESS är baserat på Javascript medans SASS baseras på Ruby. Detta tror jag är en av anledningar till att LESS är så pass mycket större idag än vad SASS är . Jag har inte arbetat så pass mycket i varken SASS eller LESS, men är också en av dem (få) som tycker det är roligt att sitta med detaljer i CSS. Dock ser jag att det är väldigt populärt att använda LESS på furom som Codepen.io eller Github och känner att det är något som jag kommer behöva sätta mig in i för att kunna jobba med andra i framtiden. Den främsta anledningen till varför skulle välja LESS eller SASS är pga av att du har en möjlighet att döpa kod in till olika variabler som du

senare kan återanvända istället för att skriva ut all kod på nytt. På samma sätt som du kan skapa variabler i Javascript. Här nedan följer ett exempel.

## LESS Color Example

```
@nice-blue: #5B83AD;
@light-blue: @nice-blue + #111;

#header {
    color: @light-blue;
}
```

### Javascript:

Även fast jag lärt mig betydligt mycket mer inom Javascript sen senaste Redogörelse är det fortfarande det språket jag har mest skräckblandad förtjusning till. Då komplexiteten är så pass stor så får jag en känsla av att jag aldrig kommer känna mig fullärd. I min projektuppgift 2 har jag inte använt mig av så mycket ren Javascript kod. Dock är sidan uppbyggd med React vilket inkluderar Javascript i många komponenter. Den del jag känner varit mest utmanande är inkluderandet av väder API som du kan se nedan.

```
const API = 'https://api.openweathermap.org/data/2.5/weather?q=stockholm,se&APPID=f8384513fad5f91ea04d07a2cbf916ec';

export class Nav extends Component {
    state = {}

    componentDidMount() {
        fetch(API)
            .then(res => res.json())
            .then((json) => {
                this.setState({ weather: (json.main.temp - 273.15).toFixed(2) });
            });
    }

    render() {
        return (
            <nav>
                <p className="weather">
                    <FontAwesomeIcon icon={faCloud} />
                    {this.state.weather ? this.state.weather : '0'}
                </p>
```

1. Här har jag döpt API länken med en const variabel men namn API.
2. Sedan har jag lagt in ett tomt state objekt i början av Nav klassen.
3. Efter det använder jag mig av en componentDidMount funktion som kommer bli kallad på när min komponent blir “mounted” till DOM:en.
4. I funktioner binder jag API adressen med fetch och använder .then promises för att få ut Json data från API:et.
5. Efter detta använder jag this.setState funktion för att definiera mitt tomma state.
6. Jag gör det genom att plocka ut datan jag vill komma åt från json.main.temp.
7. Subtrahera värdet med 273.15 för att få fram celsius.
8. Och använder mig av toFixed för att avrunda talet till två decimaler.
9. Sist kallar jag på mitt State i min P-tag. Om det finns ett värde i mitt State kommer det att visas.
10. Om det inte finns ett värde kommer strängen “0” att visas.

Tre senaste Javascriptsversionerna och kompatibilitet är följande:

1.Browser Support for ES5 (2009)

| Browser | Version | From Date |
|---|---|---|
| Chrome | 23 | sep 2012 |
| Firefox | 21 | apr 2013 |
| IE | 9* | mar 2011 |
| IE / Edge | 10 | sep 2012 |
| Safari | 6 | jul 2012 |
| Opera | 15 | jul 2013 |

Browser Support for ES6 (ECMAScript 2015)

| Browser | Version | Date |
|---|---|---|
| Chrome | 58 | apr 2017 |
| Firefox | 54 | jun 2017 |
| Edge | 14 | aug 2016 |
| Safari | 10 | sep 2016 |
| Opera | 55 | aug 2017 |

Browser Support for ES7 (ECMAScript 2016)

| Browser | Version | Date |
|---|---|---|
| Chrome | 68 | May 2018 |
| Opera | 47 | jul 2018 |

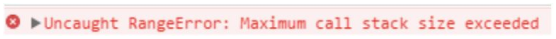
## Javascript i webbläsaren:

Det finns antagligen många Front-End utvecklare idag som använder Javascript utan att ha någon aning om vad som egentligen händer under skalet när deras kod körs. Jag är utan tvekan en utav dem men skyller på att jag är student och lär mig  Många anser att Javascript är ett utvecklings språk som beter underligt under skalet i förhållandevis till dem andra språken.

Javascript är uppbyggd med en Javascript-motor . Vi har bla Google's V8 engine som används i Chrome och Node.js. Självaste motorn i sig är uppdelad i "Memory Heap" och "Call Stack". Kort sagt kan man säga att det är i Memory Heap minnes fördelning och Call Stack är den delen som hanterar den skriva koden.
Utöver motorn så använder sig Javascript även sig av browsers API:er som tex DOM och AJAX.

Javascripts är ett så kallat "single-threaded programming language". Vilket menas att det endast kan köras ett Call Stack åt gången. Som nämnt tidigare är Call Stack den delen av motorn som hanterar datastrukturen och vilken del/vart i  koden som körs. Varje del i Call Stack kallas för Stack Frame.

När du använder för mycket kod som sker samtidigt så finns risken att Call Stack i motorn fyller upp sin maximala kapacitet. När det händer så kommer webbläsare bli lång som och hänga sig och du kommer antagligen se felmeddelande som ser på följande vis:


## Javascript-bibliotek:

I kodnings-världen brukar man tala om att undvika onödigt jobb och inte återuppfinna hjulet på nytt. Med det sagt är det uppmuntrande till och med rekommenderat att återanvända någon annans kod. Ett smart sätt att göra detta är att använda bibliotek. Ett Javascript bibliotek är helt enkelt en samling av kod/funktioner som kan utför en funktionalitet. Tex kan du använda dig av ett bibliotek om du vill ha en “Slideshow” till dina bilder.

Jag själv använder mig att ett bibliotek som heter “Particles” för att få till min bakgrund på ett snyggt sätt. Jag har har också använt mig av Font Awesome bibliotek för att få tillgång till deras ikoner.

För att få tillgång till bibliotek behöver du först installera det genom NPM i din terminal. Sedan är det bara att importerade bibliotek där du vill ha det och plocka ut dem delar i biblioteket. Så här såg det ut när jag använde Font Awesome biloteket.

```
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faEnvelope } from '@fortawesome/free-solid-svg-icons';
import { faFacebookSquare, faInstagram, faGithub, faLinkedin } from '@fortawesome/free-brands-svg-icons';
```

## Javascript-ramverk:

När jag väl fick höra om Javascript-ramverk tog det ett tag för mig att få en förståelse vad ett Javascript-ramverk var och dess innebörd. Men efter några minuters google så skulle jag vilja sammanfatta det att ett Javascript-ramverk är kod skriven och delad av utvecklare som gör att vi sedan slipper skriva onödig kod. Det gör också att vi blir mer tidseffektiva. Tex är det lättare att skriva en funktion som vi kan återanvända. I det stora hela kan man säga att ett ramverk är kod som gör vår kod lättare att skriva.
Dock så är det rekommenderat att förstå hur vanilla javascript fungerar innan du ger dig på olika ramverk.

Det finns flera bra Javascript-Ramverket att välja på idag och valet kan variera mellan utvecklare baserat på smak, vad man känner sig bekväm i eller vad för krav som finns på projektet/företaget.
Här nedan följer en lista med några av dem populärast Javascript-ramverken idag med dess för och nackdelar.

**Angular.js** - är ett open-source ramverk som kommer från Google. Idag ser vi att ramverket är populärast hos startup bolag till medelstora företag då går snabbt att arbete i och är lätt testat. Det verkar också vara hypat med dess “two-way data binding“ feature.

Fördelarna:

- Det har en väl dokumenterad arkitektur.
- Two-way data binding med ett populärt DOM interface.
- Ett stort community med bra support.
- Modulerna ät uppbyggt i vanilla javascript vilket gör det lätt att testa/ändra koden.

Nackdelar:

- Inlärningskurvan sägs vara svår.
- Tar längre tid att köras i webbläsaren pga av DOM elements.
- Det sägs vara svårt att förstå hur scopes funkar.
- För att kunna använda ramverket krävs det att man har en bra kunskap grund inom programmering.

**React.js** - Vad jag förstått är det en ständig diskussion om huruvida React ska räknas som ett ramverk eller bibliotek. React kommer från Facebook som utvecklade det 2013 och inte helt otippad är Facebook och Instagram uppbygt med React. Det används främst idag för att skapa webb och mobilanpassat plattformar.

Fördelar:

- Anses fortfarande som väldigt nytt och utveckling sker väldigt fort.
- Vi kan enkelt använda många verktyg tillsammans med React.
- Den virtuella DOM körs snabbt och effektivt.
- Återanvändning med komponenter anses väldigt effektiv.

Nackdelar:

- Skillnader mellan den virtuella DOM:en och vanliga DOM:en kan uppkomma.
- Du måste ha lite erfarenhet för att kunna använda ytterligare bibliotek.

**Vue.js** - Lanserad 2014 men vart inte riktigt populärt förens dem lanserade en mer stabil version Vue 2.0 år 2016. Då dem plockat populära funktioner som används hos React, Angular och Ember kombinerat.Då det har utformats för att vara förhållandevist anpassningsbar, kan den användas i projekt med olika JS-bibliotek. Liksom Angular är Vue kända för att använda “two-way data binding”.

Fördelar:

- Lätt ramverk att lära sig.
- Lätt dokumentation att förstå.
- Ett annat plus är dess storlek. Det är ett väldigt litet program som du kan enkelt ladda ner och komma igång med fort.

Nackdelar:

- Då det fortfarande är ganska nytt är communityt ganska lite och det finns begränsad information på internet.
- Då det är skrivet av “Chinese-American” kan det finnas språkbarriärer. Det säga tex att man ofta kan hitta kod skriven på kinesiska i många forum.

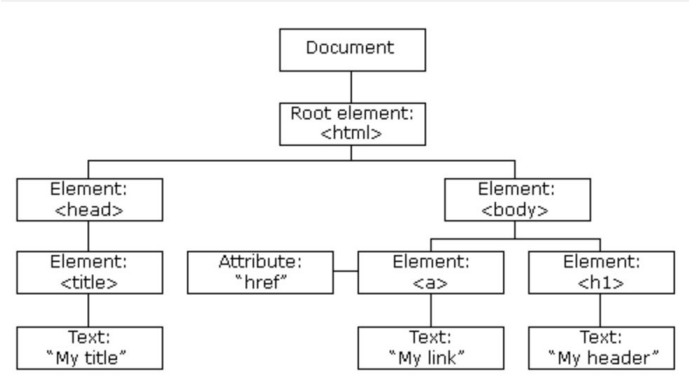
Personligen har jag endast erfarenhet ifrån React. Det var skrämmande i början då jag tyckte det särskilde sig väldigt mycket från vanilla javascript. Dock gick det ganska fort att uppskatta funktionaliteten med att jobba modulärt. Det största problemet jag hade med vanilla javascript var att det är väldigt svårt att jobba med när projektet växer. När jag skulle fixa en bugg resulterade det oftast i att jag hade skapat två nya. Detta är mycket

lättare med React då mer eller mindre varje komponent har sin egen fil. Av det jag läst är jag väldigt imponerad av Vue och vill gärna lära mig mer av det ramverket

## DOM:

DOM står för Document Object Model och är ett gränssnitt för HTML och XML. Kort sagt kan man sammanfattade det att DOM:en är självaste trädstrukturen som bygger upp en hemsida med styling och innehåll. Notera dock att DOM inte är ett programmeringsspråk som Javascript. DOM är strukturen som vi kan modifiera och göra sidan mer levande tack vare Javascript.
Varje element i DOM trädet kallas för node. För att kunna modifiera DOM:en behöver vi ta ett element (node) och modifiera det genom javascript. Tack vare DOM:en/Webbläsarens API: er kan vi använda verktyg som "getElementById" eller "removeChild".

Om man vill se hur DOM strukturen ser upp på en sida kan man enkelt göra det genom Dev tools. Om inte annat bifogar jag en bild nedan på hur trädstrukturen kan se ut.




Som nämnt tidigare jobbar du med modulärt med React. Därmed använder du också en annan typ av DOM i react som kallas för Virtual DOM. Med Virtual DOM kör du (Render) bara den del/komponent som är aktuell. Du behöver alltså inte köra hela DOM som du annars skulle göra i vanilla JS.

Istället för att skapa en DIV med ett UL i, kan du med React skapa en React.DIV objekt som har en React.ul objekt. Vi kan då manipulera react objektet lätt utan att röra den "riktiga" DOM:en eller utan att använda DOM API:er. När vi sedan kör komponenten så kollar DOM:en på den virtuella DOM:en för att lista ut vad som har ändrats.

För att göra detta ännu mer tydligt läste jag en förklaring på Stackoverflow som beskrev Virtual DOM på följande sätt.

"Tänk dig att du har ett väldigt stökigt rum som du behöver städa. Hur skulle du gå till väga för att lösa detta? Skulle du städa hela huset eller skulle du städa just det stökiga rummet? Med största sannolikhet skulle du bara städa just ditt rum som är stökigt. Och det är precis så virtual DOM funkar. Iställer för att köra hela DOM:en så körs endast den komponenten som behövs."

## AJAX:

Tack vara Ajax kan vi på ett effektivt sätt skicka och hämta data till en server Asynchronous . Detta betyder att vi kan köra flera funktioner samtidigt utan att vänta att alla ska renderas i turordning. Vi kan med andra ord köra en funktion och låta sidan fortsätta köras med funktionen i bakgrunden kommunicerar med servern. På så sätt kan vi skicka och hämta data till servern utan att hela webbläsaren behövs uppdateras.

På det sättet undviker vi att sidan hänger sig eller står och laddar. Dessutom kan användare som har en låg bandbredd använda sidan på ett bra sätt och det skapar en bra UE/UI.

Ett exempel där AJAX används är “Text hints” i sökfunktioner. När en användare skriver in den första bokstaven på ordet/meningen han/hon vill söka på kommer det upp förslag på sökord/meningar hämtat direkt från servern.

Den största nackdelen med AJAX är att den förlitar sig helt på Javascript. I de webbläsare som finns idag fungerar Javascript oftast olika på dem olika webbläsarna. Detta betyder också att AJAX funktionalitet kan fungera annorlunda på olika och i värsta fall inte alls på olika webbläsare.
Om du använder en webbläsare som inte kan hantera Javascript så kommer då inte heller AJAX att fungera.
Detta gör det svårt att skriva mobilanpassade applikationer med AJAX.

## HTTP1.1 / 2.0:

HTTP var i grunden skapat Tim Berners-Lee som utformade applikation protokollet med åtanke för att förbättra kommunikationsfunktioner mellan webbservrar och klienter.

HTTP står för Hypertext Transfer Protocol och är ett applikationsprotokoll som bestämmer hur filer som t ex bild och ljudfiler ska skickar på internet. Det är på detta sätt våra webbläsare pratar om andra server och kan skicka HTML,CSS,Javascript data. Så fort en användare öppnar en webbläsare kommunicerar webbläsaren men utomstående servrar med att skicka och ta emot HTTP Requests och Responses.

Om man ska förklara det stegvis så ser det ur såhär:

- Webbläsare skickar en HTTP-begäran till webben
- En webbserver tar emot begäran
- Servern kör en applikation för att behandla förfrågan
- Servern returnerar ett HTTP-svar (utdata) till webbläsaren
- Klienten (webbläsaren) får svaret

I maj 2015 standardiserades HTTP  2 officiellt av Googles SPDY-protokoll. Det relativt nya HTTP 2-protokollet snabbar upp sidan och stöds av alla det större webbläsare och servrar.
I jämförelse med HTTP 1.1 kunde det endast skicka en request per TCP connection . Vilket resulterade i att Browsern var tvungen att använda flera TCP connections för att hantera flertal request samtidigt.

Webbläsare som använder flera TCP connections tar upp större del av nätverket och därmed försämras prestandan och för andra användare.
HTTP / 2 har skapat en lösning för att hantera dessa problem. Där vi kan använda samma TCP connection för med flera parallella request samtidigt.

Med HTTP kom också något som heter “Server push” , vilket kan förklara som att server själv räknar ut vilken data som behövs skickas utan att den egentligen är begärd av klienten.
Om klienten tex begär efter dataresurs X och det är underförstått att även resurs Y behövs för att kunna rendera rusers X kommer servern att räkna ut det i förhand och även kludera resurs Y när resurn X skickar tillbaka.

http://wnordqvist.com/PortfolioREACT/#/