Sigurros Inga Helgadottir

10-08-2020

Projekt dokumentation – S. Inga Helgadottir

Teknologier

Gulp - Jeg valgte at bruge gulp, fordi man bruger nav og footer mange steder og jeg er ikke god nok til React eller Gatspy. Gulp gør det meget nemmere at bruge ting mange steder uden at skrive det mange gange. Gulp-imagemin minimerer også billeder, som gør siden hurtigere, specielt i en opgave med så mange billeder som denne opgave.

Flexbox of Grids - Jeg brugte flexbox mange steder, for eksempet i nav baren, fordi det gør stylingen meget nemmere og hurtigere. Jeg brugte også grids mange steder, for eksempel i events of the month. Det er den nemmeste måde at sætte ting oven på hinanden (med z-index).

Javascript - Javascript har jeg brugt for alt funktionaliteten, fordi det har jeg godt styr på og man kan lave så meget med Javascript. Javascript har jeg brugt i gennem hele hovedforløvet og det meste af grundforløbet.

Teknisk dokumentation

Jeg starter med at vise gulpfile, men den er meget lang så jeg klipper noget ud og forklarer resten.

```
const gulp = require("gulp");
function html(done){
    gulp.src("./src/html/templates/*.ejs")
        .pipe(ejs())
        .pipe(rename(function(path){
            path.extname = ".html"
        }))
        .pipe(gulp.dest("./dist"))
        .pipe(connect.reload());
    done();
}
function watchHtml(done){
    gulp.watch("./src/html/**/*.ejs", { ignoreInitial: false }, html);
}
gulp.task("dev", function(done){
    watchHtml();
 ({
        livereload: true,
```

```
        root: "dist"
    });
    done();
});
gulp.task("build", function(done){
    html(done);
    done();
});
```

Min gulpfile starter med require som siger hvilke npm pakker jeg skal bruge her (jeg viser kun en af dem men der er mange jeg har slettet i denne forklaring. For hver function er der for eksempel en html og en watch html. Html siger at den skal tage de ejs filer i mappen som hedder templates og ændre dem til html og sætte dem inden i dist mappen. Watchhtml funktionen kigger efter ændringer i min ejs filer og opdatederer dem da de har ændret sig. Gulp.task kører alle functionerne.

Det her er min kode for fetch i events sektionen

```
    fetch("http://localhost:4000/events", {
        "method": "GET",
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded'
        }
      })
      .then (function(response){
          return response.json();
      })
      .then (function(result){
          result.forEach(event => {
            let date = new Date(event.eventDate);
            let day = (new Date(event.eventDate)).getDate();
            let month = (new Date(event.eventDate)).getMonth();
            if(month === 0){
                month = "Jan"
            }
            let hours = (new Date(event.eventDate)).getHours();
            let min = (new Date(event.eventDate)).getMinutes();
            let location = event.location;
            let description = event.eventDescription.slice(0, 100) + "...";
            let name = event.eventName;
            inputData(location, month, day, hours, min, description, name);
          });
      });
```

Det starter med en normal fetch og så gemmer jeg det jeg har fetched og sender det med da jeg kalder på min inputData funktion, hvor jeg sætter det jeg har fetched ind i min ejs fil som ændres til en html med gulp. Datoen ind på API`et var i ISO så jeg

skulle oversætte den til normal tid, derfor bruger jeg new date, get hours og minutes. If bruger jeg fordi måneden kommer i tal, alle månederne er de samme så jeg bruger kun Jan, men hvis der var flere ville jeg have lavet en array med månederne og givet dem en index.

Jeg lavede min post code i html, den åbner en ny side med det information du har sendt til API`et.

Resten af min Javascript kode er meget simpel, eventlisteners, click funktioner, setInterval og fetch. Jeg lavede formvalidering i html med pattern, og jeg havde aldrig behøvet at lave en for websider så jeg skulle finde ud af det og det virkede med pattern=”www.+\.[a-z]{2,}” som betyder, det skal starte på www. så skal der være tekst så et punktum og så 2 bogstaver.

Soft dokumentation

Siden er lavet i firefox developers edition med max width 1440px (fordi jeg googlede den mest brugte skærm størrelse), jeg havde ikke tid til at ordne siden i andre browsere og jeg lavede kun lidt media queries, jeg havde ikke tid til at få det til at se helt rigtigt ud i mobil, men i hver felt er det ikke helt forfærdeligt på mindre skærme. Men jeg laver næsten altid min widths og grids i % så at det ser okay ud, men det hjælper ikke i mobil størrelse. For at se opgaven skal man åbne terminalen og skrive npm run dev, så kommer siden på http://localhost:8080.

Øverst er der Hero headeren, hvor billederne skifter mellem de to header billeder og loader billedet. Derefter kommer de tre billeder som ved hover viser text. Så kommer events, som er et billede galleri som bliver styret af de tre kasser som ligger under og hvis du sætter musen over vises en tekst og en knap, men de gør intet. I en pink kasse under billederne er der en dato, tid og lokation som kommer fra API`et, billederne er statiske fordi der var ingen billeder på det sted i API`et.

Så kommer billede galleriet, men den henter kun det rette antal billeder fra API`et. Så kommer night club track men det er kun et audio tag. Så kommer video galleriet, der var kun en video i mappen så det andet video er kun et billede som viser at der ikke er en video. Hvis man klikker på pilene som er under videoen skifter man mellem videoen og billedet.

Så kommer man til testiomonials, hvor text bliver hentet fra API`et, og igen var der ingen billeder så jeg henter billederne fra images mappen. De tre kasser under fører

dig igennem de tre testimonials. Så kommer blog posts hvor billederne er statiske og testen kommer fra API`et. Men der sker intet da du kliker på dem.

Så kommer newsletter delen, hvor man kan sætte ind en email og klikke på submit. Man skal skrive en email som ser rigtigt ud, den email skulle have blive sat på API`et men jeg havde nogle problemer med det, i stedet sætter den et email: null på API`et.

Linkerne i footeren ledder dig til de sider de viser. Så kan man trykke på contact us i toppen og man kommer ind på en side med et form som er valideret men sender intet til API`et fordi jeg kunne ikke få det til at virke i newsletteren, så kan jeg ikke få det til at virke her.

Arbejdsprocessen

Jeg startede med at opsætte gulp, der havde jeg et problem med imagemin, men skulle kun skifte til en gammel version for at det virkede. Derefter startede jeg med at lave alle mapper og så startede jeg med at lave mine ejs filer. Jeg begyndte på at lave de ejs filer som jeg bruger alle steder (head, nav, footer, last). Så lavede jeg index siden i ejs og scss, som gulp ændrer til html og css. Jeg lavede ejs og scss en sektion ad gangen, og startede med at gøre den statisk og jeg hoppede over dem der var sværest. Fordi da man er til en eksamen er det bedre at lave det nemmeste først, så bliver du lidt mere optimistisk og det gør det nemmere at lave de svære ting senere. Da jeg havde lavet index siden startede jeg på at lave funktionaliteten, så at jeg har tideligt lidt af hvert (html, css, javascript). Jeg hoppede over animationerne fordi jeg har glemt det meste om hvordan man laver dem og det ville tage lang tid at genfriske det, så derfor tænkte jeg at det var bedre at lave dem med hover eller i javascript. Men animationen i events sektionen lavede jeg med hover og lod være med at få dem til at flytte sig som de skulle fordi jeg havde ikke tid til at genfriske det nu, det gør jeg dog før eksamen. Da jeg havde fået lidt funktionalitet flyttede jeg mig til at hente fra API`et. Så begyndte jeg at lave de svære ting som POST, det virkede til sidst da jeg lavede det i html, men det var efter jeg havde taget pause og started på media queries og burger menu.

Til sidst prøvede jeg at færdiggøre denne her opgavebeskrivelse og da jeg er færdig prøver jeg at færdiggøre media queries.