Front-end Bootcamp

CSS 방법론과 sass

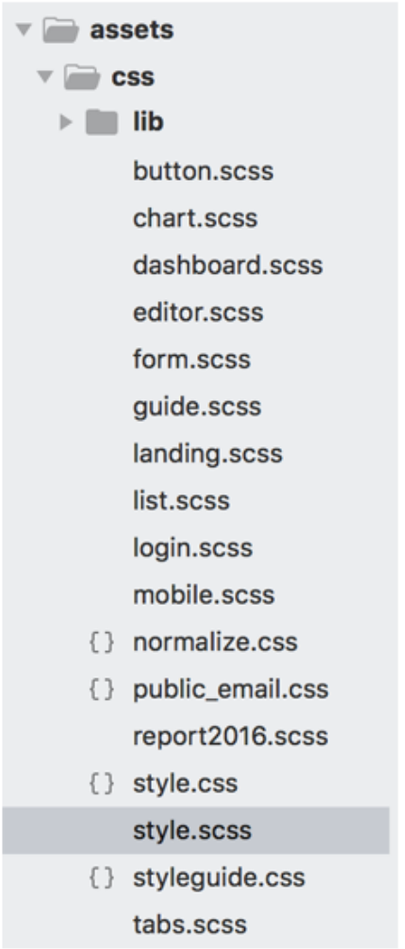
CSS설계

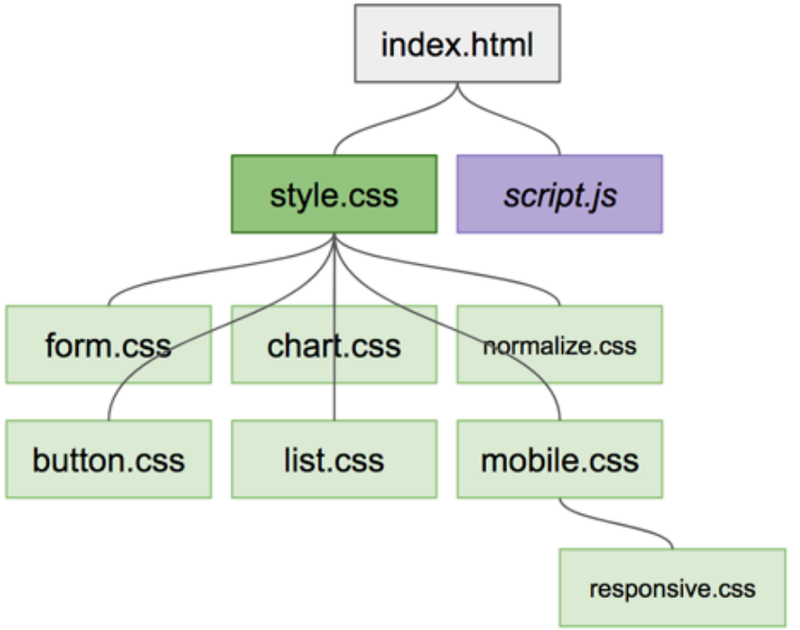
"빠른 작성, 유연성, 확장성, 쉬운 유지보수"

- 모듈화
  - 파일 단위 모듈화 → `@import url()`
  - 블럭 단위 모듈화 → BEM, OOCSS, Sass
- 네이밍
  - 명확한 class, id 네이밍

@import와 모듈화

```
@import url(reset.css);
@import url(button.css);
```





대표적인 CSS 네이밍/설계 방법론

- BEM
- OOCSS
- SMACSS

BEM

- html블럭와 css 선택자 네이밍의 어려움을 해결
- 모듈 **단위 분리** (=재사용)
- css 선택자 이름을 가능한 **명확하게**
- 조합은 **하이픈**(`-`)으로

B**lock

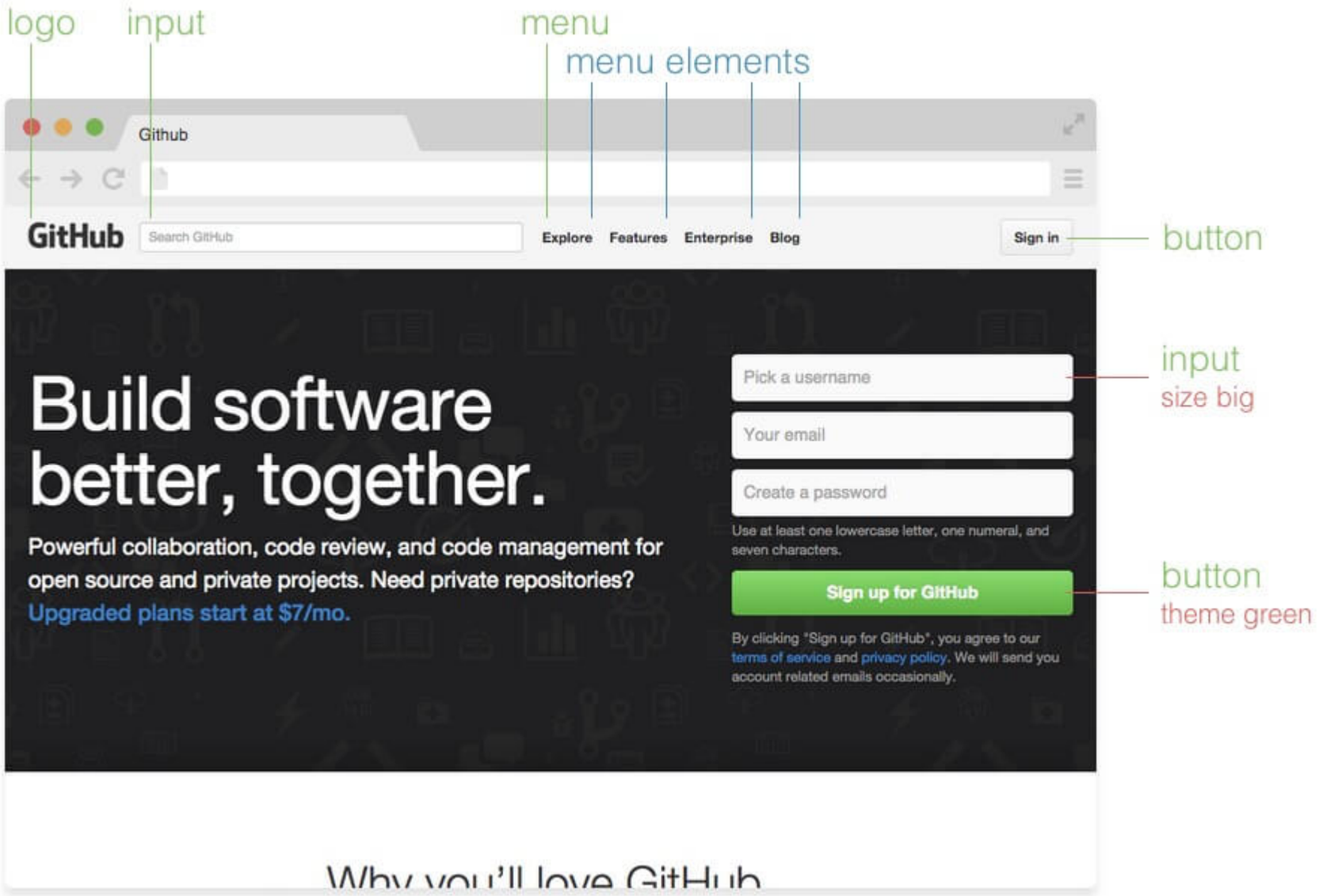
- 독립된 구성 요소
- = 재사용이 가능하다
- = 다른 곳에 가져다 붙여도 잘 보인다
- 형태가 아닌 목적 중심으로 네이밍
  - 형태: blue, red, big
  - 목적: nav, menu, button
- 예시: header, nav, footer, search-form

Element

- 블럭 안의 특정 요소
- `block__element`의 형태(더블 언더바)
- 요소는 부모 블럭에 종속, 다른 곳으로 옮겨질 수 없다

Modifier

- 모양과 표현 요소(color, size, checked, disabled)
- `block__element--modifier`(더블 하이픈)
- 타입
  - 불리언타입 `form__button--checked`의 `--cheked`
  - 키-밸류 타입 `content__title--color-red`의 `--color-red`



- BEM real world examples

OOCSS

- Object Oriented CSS
- CSS안에서도 구조와 표현을 분리
- 표현(색상, 넓이, 여백) 요소를 별도의 스킨 class로 분리
- 요소의 위치가 어디라도 똑같이 표현된다
- = 요소의 자유로운 이동이 가능하다

네이밍

- 최대한 짧고 간결하게
- 일반적인 단어
- 형태가 아닌 목적 중심으로 네이밍
  - 형태: blue, red, big
  - 목적: nav, menu, button

```
// 기존 방식
<a class="facebook_btn">Facebook</a>
<a class="twitter_btn">Twitter</a>

// OOCSS 적용
<a class="btn facebook">Facebook</a>
<a class="btn twitter">Twitter</a>

.btn { 공통 버튼 스타일 정의 }
```

장점

- 간결하과 축약적인 작성 가능
- 지금까지 공부해 온 방식
- 상대적으로 작은 프로젝트에서 빛을 발한다

단점

- 프로젝트가 커지면 관리가 어렵다.
- inspect하기 전까지 정확한 의미 파악이 어렵다.

airbnb의 사례

- Airbnb CSS / Sass Styleguide
- BEM과 OOCSS를 함께 사용

```
/* ListingCard.css */
.ListingCard { }
.ListingCard--featured { }
.ListingCard__title { }
.ListingCard__content { }
```

```
<button class="btn btn-primary js-request-to-book">Request to Book</button>
```

SASS

- CSS preprocessor(전처리기)
- 쉬운 작성과 모듈화
- 크로스브라우징
- 압축
- `.sass`, `.scss`의 확장자

sass 작성 환경

gulpfile.js 수정

```
gulp.task('sass', function () {
  return gulp.src('assets/css/style.scss')
    .pipe(sass({outputStyle: 'compressed'}).on('error', sass.logError))
    .pipe(gulp.dest('assets/css'));
});

gulp.task('browser-stream', ['sass'], function () {
  return gulp.src([
      'assets/css/*.css'
    ])
    .pipe(browserSync.stream());
});
```

기초 문법

variables(변수 사용)

```
$blue : #3e81f6;
$green : #00d2bc;
$bg : #F4F4F5;
$lg : #e6e6eb;
$dg : #606060;

h1.title {color:$blue;}
```

nesting(감싸기)

- BEM 방법론 코딩
- 모듈화에 유용

```
.navigation {
  ul {
    li {
      a {
        display:block;

        &:hover { // a:hover로 선택됨
          color:$green
        }
      }
    }
  }

  &-colored { // .navigation-colored로 선택됨
    background:$yellow;
  }
}
```

import(가져오기)

- import한 파일까지 모두 모아서 압치기, 압축
- 1개의 css파일로 서비스 가능

```
@import 'reset.css';
```

mixin(함수)

- 중복된 코드 재사용

```
@mixin clearfix {
  &:before, &:after {content: " ";display: table;}
  &:after { clear: both;}
}
```

```
@mixin ani($duration) {
        -webkit-transition: all ease-in-out $duration;
        -o-transition: all ease-in-out $duration;
        transition: all ease-in-out $duration;
}

.box {
  @include cf;
  @include ani(.3s);
}
```

operator(계산하기)

```
section {
  width: 600px / 960px * 100%; // 62.5%로 계산하여 표현
}
```