CSS 방법론

SI2팀 엄예지

CSS 방법론

01 사용목적

02 SMACSS

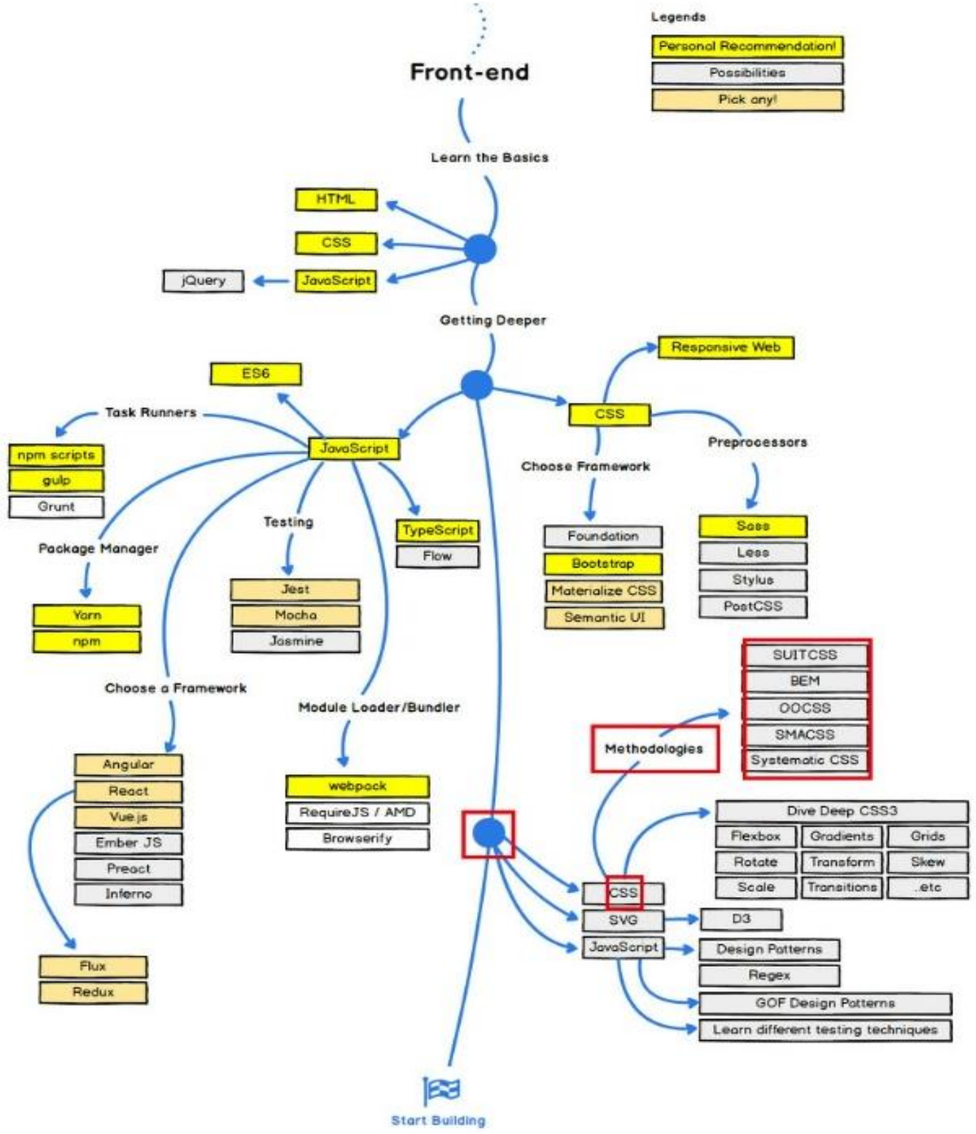
03 BEM

04 OOCSS

05 토론

CHAPTER 01 사용목적

사용목적



- CSS에 이름을 붙이는 대표적인 방법
- SMACSS
- BEM
- OOCSS

사용목적

```
<div class="background-yellow">안녕 안녕~~~</div>

<style>
.background-yellow{
        Background-color:yellow;
}
</style>
```

- Class의 개수가 100개, 1000개를 넘어간다면?
- 전에 작성한 클래스 이름과 새로운 클래스의 이름이 겹치지 않는다고 확신할수 있는가?
- 이전에 썼던 코드를 또 쓰고 있지는 않은가?
- 협업시 그 사람의 코드와 내 코드가 겹치지는 않을까?
- 클래스의 이름만 보고 무엇을 위한 스타일인지 유추할수 있을까?

1.쉬운 유지보수

- Css방법론들이 공통적으로 지향하는 목표
- 대부분 css 선택자로 id와 type(태그)를 사용하지 말것을 권장

2.코드의 재사용

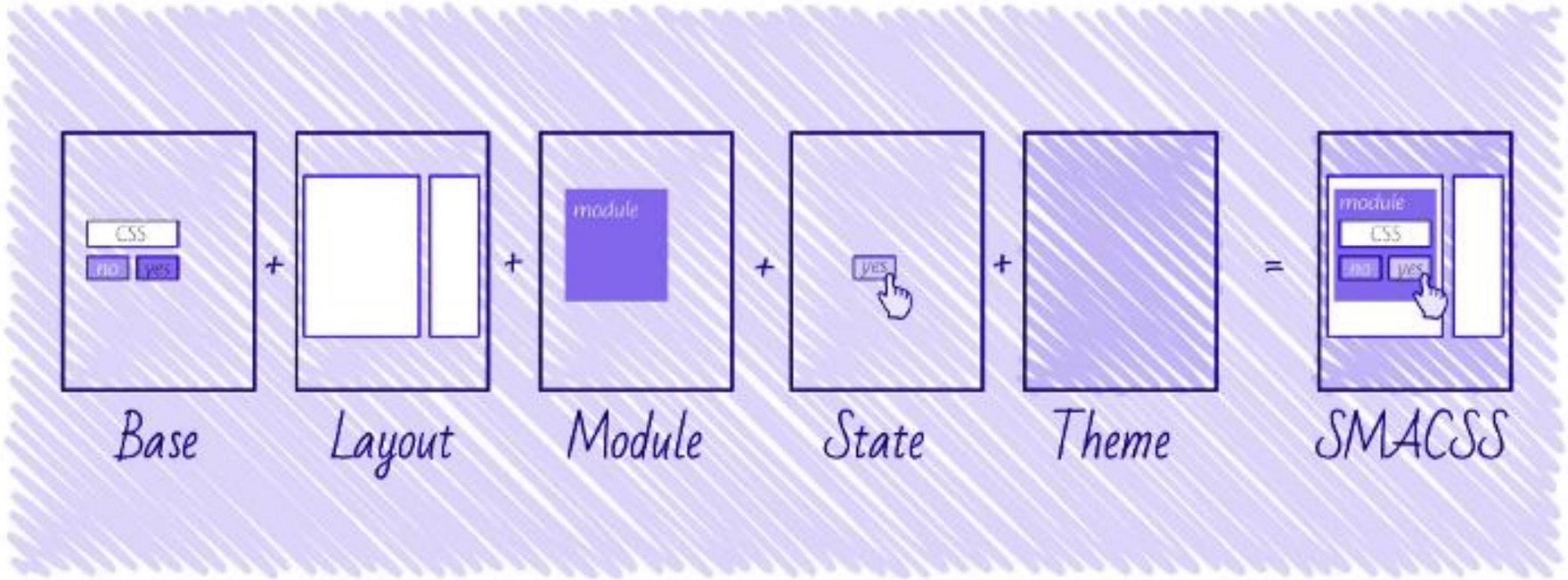
3.확장 가능

4.직관적인 네이밍

CHAPTER 02 SMACSS

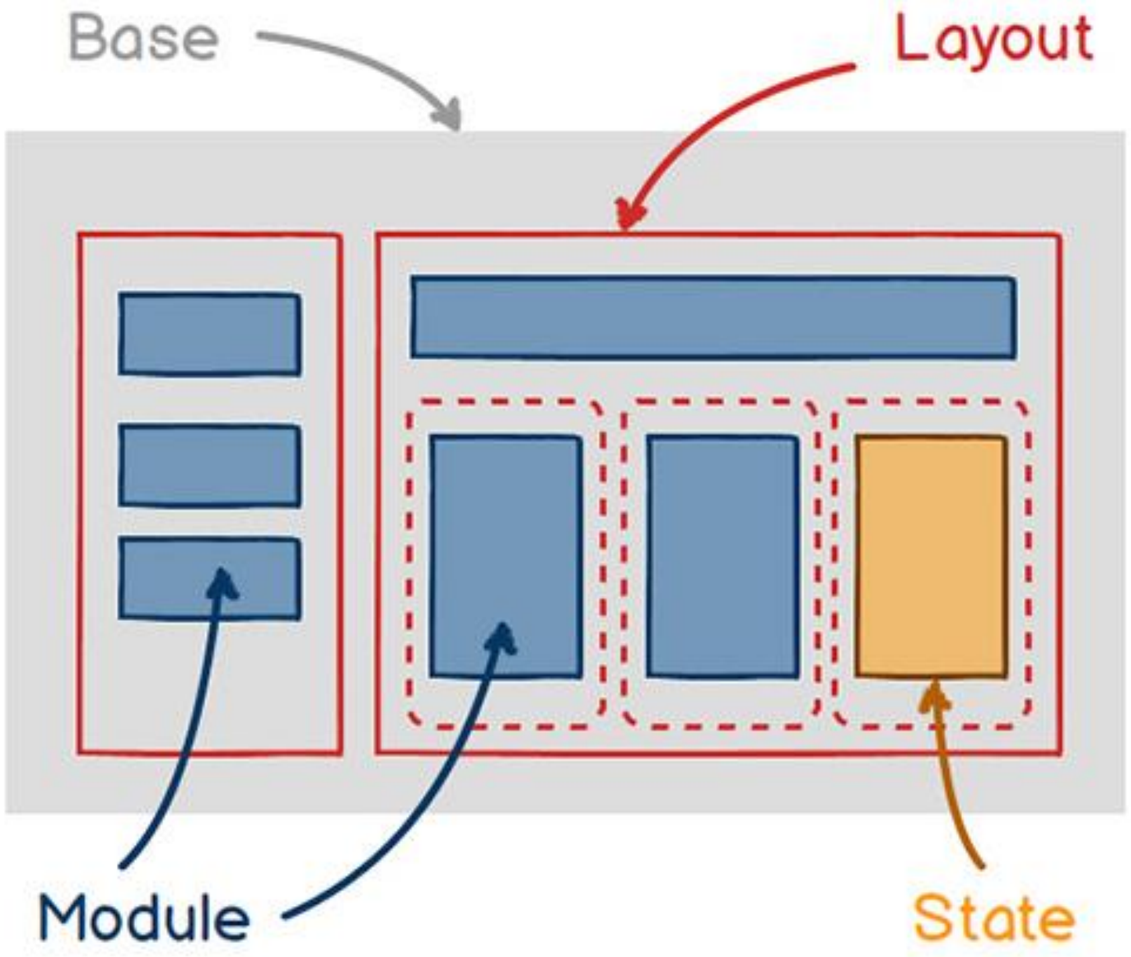
Scalable and Modular Architecture for CSS

- 점점 복잡해지고 관리하기 어려워지는 CSS를 효과적으로 쓸수있도록 돕는 스타일 가이드
- 조나단 스눅(Jonathan Snook)이 제안한 CSS 작성 규칙
- 재사용과 유지보수가 쉽고 확장 가능한 모듈식으로 CSS코드를 작성할수 있다



Scalable and Modular Architecture for CSS

SMACSS



- SMACSS의 핵심은,
CSS스타일을 몇 가지 범주로 묶는것!!
- 기본(Base)
- 레이아웃(Layout)
- 모듈 (Module)
- 상태(State)
- 테마(Theme)

- **범주화**로 패턴화 하고자 하는 방법론

1.기본 스타일(Base)

- 초기값 설정 (reset.css)
- 대부분 하나의 요소(elements)로 이루어져 있음
- 클래스, 아이디 선택자를 포함하지 않음
- !important는 허용하지 않음
- Ex) 기본 링크 스타일, 기본 폰트 스타일, 바디 배경 선언 등

```
html, body, form {margin:0;padding:0;}
input[type=text] {border:1px solid #999;}
a {color:#000;}
a:hover {color:#666;}
```

2. 레이아웃 스타일(Layout)

- 큰 틀의 레이아웃, 페이지의 다양한 요소를 배치하고 구별하는데 사용
- 주요 요소(컴포넌트 – id / header, footer, aside 등),
- 하위 요소(컴포넌트 – class / nav, item list, form 등)를 구분
- 클래스명을 사용할 때는 접두사로 l-, layout- 를 사용

```
// 주요 요소 ()
#header {
    width: 400px;
}
#aside {
    width: 30px;
}
```

```
// 하위 요소
.l-width #header {
    width: 600px;
    padding: 10px;
}
.l-width #aside {
    width: 100px
}
```

3. 모듈 스타일(Module)

- 버튼, 위젯, 배너 등과 같이 페이지에서 재사용을 위한 스타일
- 네비게이션, 슬라이드 위젯 등
- 레이아웃이나 다른 모듈 안에 위치
- 각 모듈은 독립적으로 존재할 수 있도록 설계
- 재사용을 위해 CSS선택자로 id, 엘리먼트 선택자를 피하고 클래스를 사용

```
<div class="box">
    <span class="box-name"> ... </span>
    <span class="box-items"> ... </span>
</div>
```

```
.box { ... }
.box-name { ... }
.box-items { ... }
```

4. 상태 스타일(State)

- 요소의 상태 변화를 표현하는 스타일
- 모듈과 레이아웃 둘 다에 적용할수 있음
- 주로 JS에서 조작되는 클래스에 지정, 클래스명 접두사로 is-, s- 사용
- EX) is-hidden, is-collapsed, is-tab-active

```
<!-- 레이아웃 요소, 접힌 상태 -->
<div id="header" class="is-collapsed">
  <form>
    <!-- 모듈, 오류 상태 -->
    <div class="msg is-error">
      There is an error!
    </div>
    <!-- 연관된 라벨이 숨겨진 상태 -->
    <label for="search" class="is-hidden">Search</label>
    <input type="text" id="search">
  </form>
</div>
```

- #header : 레이아웃 요소 라는것
- Is-collapsed : 접힌상태 라는것
- .msg : 모듈 이라는것
- Is-error : 오류 상태 라는것
- Search 연관 라벨은 숨겨져 있다

5. 테마 스타일(Theme)

- 전반적인 표현과 느낌을 결정하는 색깔이나 이미지를 정의
- 테마 스타일 규칙만 따로 모아서 분리하면 테마를 쉽게 교체하고 재정의 가능
- 적용 범위가 넓은 테마의 경우 접두사로 theme- 사용
- 사용자가 선택 가능하도록 스타일을 재선언하여 사용

- Base.css

```
.background-yellow{
        Background-color:yellow;
}
```

- Theme.css (base.css 뒤에서 읽도록 적용)

```
.background-yellow{
        Background-color:blue;
}
```

SMACSS의 장단점

장점
- 범주화를 통한 패턴화
- 모듈 및 요소의 상태 확인
- 확장의 용이성

단점
- 규칙에 대한 종속성
- 카테고리를 나누는 기준이 작성자에 따라 불분명
- 코드를 나누어서 작성해야 하기 때문에 css를 사용하기 번거로움

CHAPTER 03 BEM

BEM

Block Elemnet Modifier

- Block (전체를 감싸고 있는 블록요소)
- Element (내부 요소)
- Modifier (기능/수정)
- ID에는 사용할 수 없고, 오직 클래스 명에만 활용 가능 (태그명으로도 사용 X)
- 모든 이름은 간단하고 직관적, 명료, 유니크한 클래스명을 사용할것
- 엄격한 네이밍 규칙을 가짐
- Selector 만으로도 무엇을 하는지 어디에 사용하는지 유추 가능하게 작성

1. Blockd (전체를 감싸고 있는 블럭요소)

- 문단 전체에 적용된 요소
- 요소를 담고 있는 컨테이너 (그루핑 요소의 가장 최상위 요소)
- Header, Footer, Sidebar 등
- 클래스의 어근 형성
- 항상 맨 앞에 위치

```
.header{
    ...
}
.footer{
    ...
}
```

2. Element (요소)

- 블록이 포함하고 있는 한 조각, 블록을 구성하는 한 요소 단위
- 블록 요소에 의존적인 형태
- 블록(전체) __ 요소(조각), 두개의 밑줄표시로 연결(__)

```
.header__logo{
    property : value; }
.header__tagline{
    property : value; }
.header__tap{
    property : value; }
.header__tap__item{
    property : value; }
```

3. Modifiers (속성/수정)

- 블록 또는 요소의 속성/수정
- 볼록 또는 요소의 외관이나 상태를 변화시키는 것
- 클래스 명을 지을때의 목적은 그 요소를 반복하여 재사용할수 있게 하기 위한 것!
- 블록, 또는 요소에 –를 추가하여 표시

```
.header--hide{
    property : value; }
.header__tap--big{
    property : value; }
.header__tap__item--big{
    property : value; }
```

Block Element Modifier

BEM

BEM의 장단점

장점
- 직관적인 클래스명으로 마크업 구조를 직접 보지 않아도 구조 파악이 쉬움

단점
- 클래스명이 길어질수밖에 없는 구조
- 기존 마크업에서 새롭게 기능이 추가되었을 경우 클래스명 재수정이 불편

CHAPTER 04 OOCSS

OOCSS

Object Oriented CSS

- 모듈 방식으로 코딩하여 중복을 최소화 하고 캡슐화를 원칙으로 하는 방법론
- Nicole Sullivan에 의해 개발된 프레임 워크
- 구조(structure)와 모양(skin)의 분리 (구조와 외형의 분리)
- 콘테이너와 콘텐츠의 분리

1. 구조와 외형의 분리

- Separate structure and skin
- 구조 : width, height, padding, margin, border
- 외형 : color, border-color, font-color, background-color

```
#button {
  width: 200px;
  height: 50px;
  padding: 10px;
  border: solid 1px #ccc;
  background: linear-gradient(#ccc, #222);
  box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
}

#box {
  width: 400px;
  overflow: hidden;
  border: solid 1px #ccc;
  background: linear-gradient(#ccc, #222);
  box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
}

#widget {
  width: 500px;
  min-height: 200px;
  overflow: auto;
  border: solid 1px #ccc;
  background: linear-gradient(#ccc, #222);
  box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
```

```
.button {
  width: 200px;
  height: 50px;
  padding: 10px;
}

.box {
  width: 400px;
  overflow: hidden;
}

.widget {
  width: 500px;
  min-height: 200px;
  overflow: auto;
}

.skin {
  border: solid 1px #ccc;
  background: linear-gradient(#ccc, #222);
  box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
}
```

2. 컨테이너와 콘텐츠의 분리

- Separate container and content
- 위치에 의존하지 않는 스타일 정의
- 구조적 상황에 관계 없이 문서 어느 곳에서나 재사용 가능
- 태그가 변경되어도 css를 바꿀 필요가 없음

```
<h2>...</h2>
h2{ font-size: 16px; }
```

```
<h2 class="title">...</h2>
<span class="title">...</span>
.title { font-size: 16px; }
```

OOCSS의 장단점

장점
- 더 빠른 웹사이트
- 유지보수가 쉬운 스타일시트

단점
- 복잡해지는 HTML(가독성 떨어짐)
  이를 보완하기 위해 OOSASS(OOCSS+SASS)와 같은 방법이 제시되기도 함

깜짝퀴즈 (어떤 css방법론을 적용한건지 맞춰보세요 ㅎㅎ)

```
<div class="header common-width">Header</div>
<div class="footer common-width">Footer</div>


.header{
    position: fixed;
    top: 0;
}

.footer{
    position: absolute;
    bottom: 0;
}

.common-width{
    width: 700px;
    margin: 0;
}
```

```
<div class="btn skin tel">tel</div>
<div class="btn skin">email</div>


.btn{공통 스타일 정의}
.skin{공통 스타일 정의}
```

```
body, p, table, form, fieldset, legend, input, button ... {
    margin: 0;
    padding: 0;
}
```

```
// 주요 요소 ()
#header {
    width: 400px;
}
#aside {
    width: 30px;
}

// 하위 요소
.l-width #header {
    width: 600px;
    padding: 10px;
}
.l-width #aside {
    width: 100px
}
```

```
.stick { ... }
.stick-name { ... }
.stick-number { ... }
```

```
.is-error { ... }
.is-hidden { ... }
.is-disabled { ... }
```

```
// base.css
.header {
    background-color: green;
}


// theme.css
.header {
    background-color: red;
}
```

```
.header { ... }
.block { ... }
```

```
.header { ... }
.header__link { ... }
.header__tap { ... }
.header__tap__item { ... }
```

```
.header--hide { ... }
.header__tap--big { ... }
.header__tap--big { ... }
```

특징을 한 단어로 표현하자면...

SMACSS

5형제

BEM

네이밍

OOCSS

재사용

CHAPTER 04 토론

THANK YOU!