

09. BEM

웹은 덩어리, 요소, 변형으로 구성되어 있다.



Naming things

"컴퓨터 과학에는 두 가지 난제가 있다.
캐시를 무효로 만드는 것과 작명."

– Phil Karlton

작명 규칙을 잘못 관리한 사례

의미를 파악할 수 없는 작명: 😞

```
.bx { ... }
```

```
.cnt { ... }
```

```
.mt { ... }
```

전역 공간을 선점한 흔한 이름: 😞

```
.content { ... }
```

```
.button { ... }
```

```
.top { ... }
```

선택 규칙을 잘못 관리한 사례

reset.css

```
a { text-decoration: none; }
```

local.css

```
.module a { text-decoration: underline; }
```

```
#special.module a { text-decoration: none; } 😬
```

```
#another#special.module a { text-decoration: underline; } 😬
```

CSS selector specificity(선택자 우선순위 규칙)

id	class, [attr], :class	type, ::element
0	0	0

a	0, 0, 1 → 001
.a	0, 1, 0 → 010
#a	1, 0, 0 → 100
#a a	1, 0, 1 → 101
#a.a a	1, 1, 1 → 111
#a#b[href]::before	2, 1, 1 → 211

cssstats.com

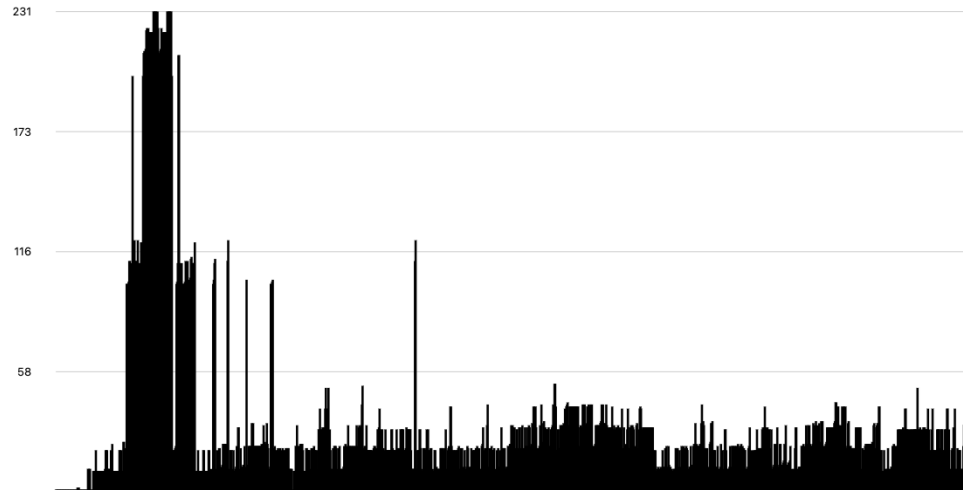
A site

Specificity

Average score
32

Max score
231

Base 10 specificity score for each selector by source order. Generally, lower scores and flatter curves are better for maintainability. [Learn More](#)



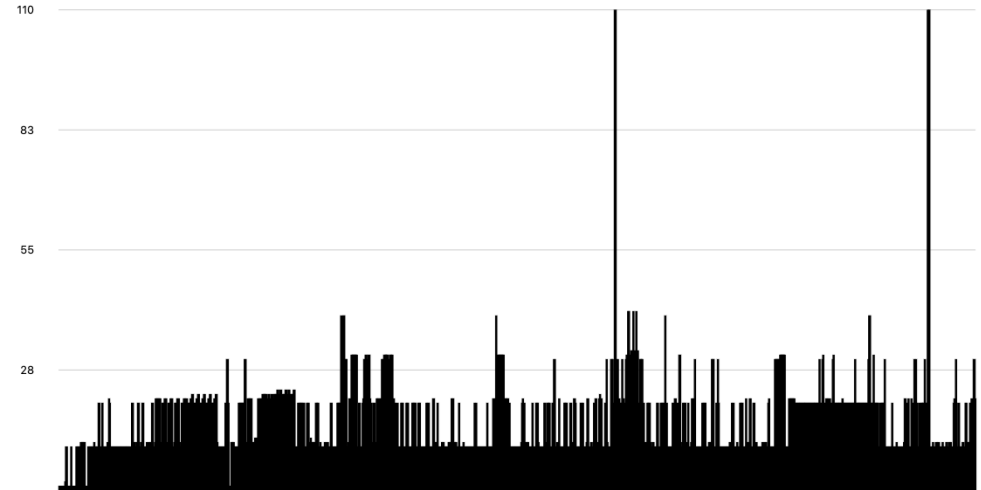
B site

Specificity

Average score
16

Max score
110

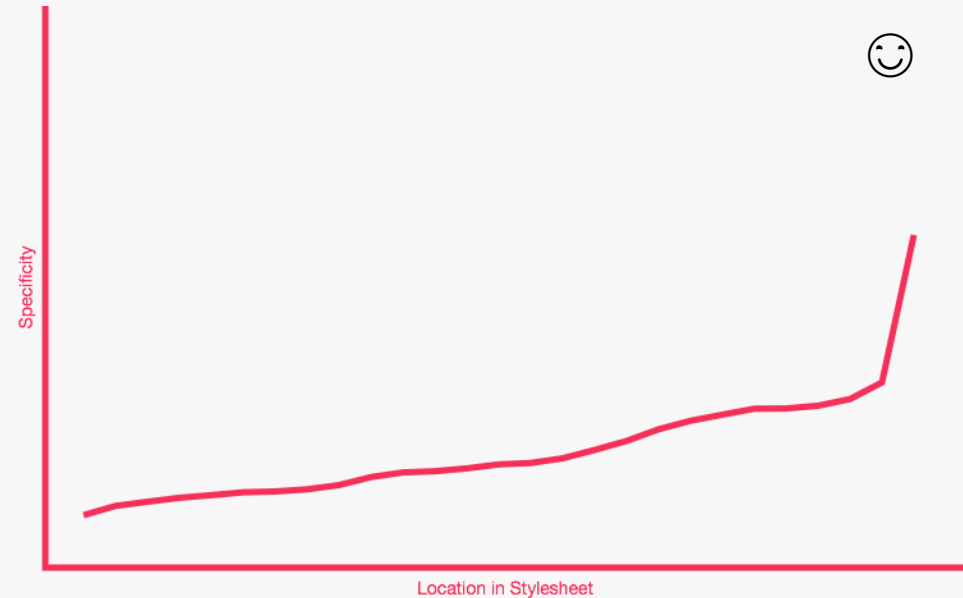
Base 10 specificity score for each selector by source order. Generally, lower scores and flatter curves are better for maintainability. [Learn More](#)



CSS selector specificity

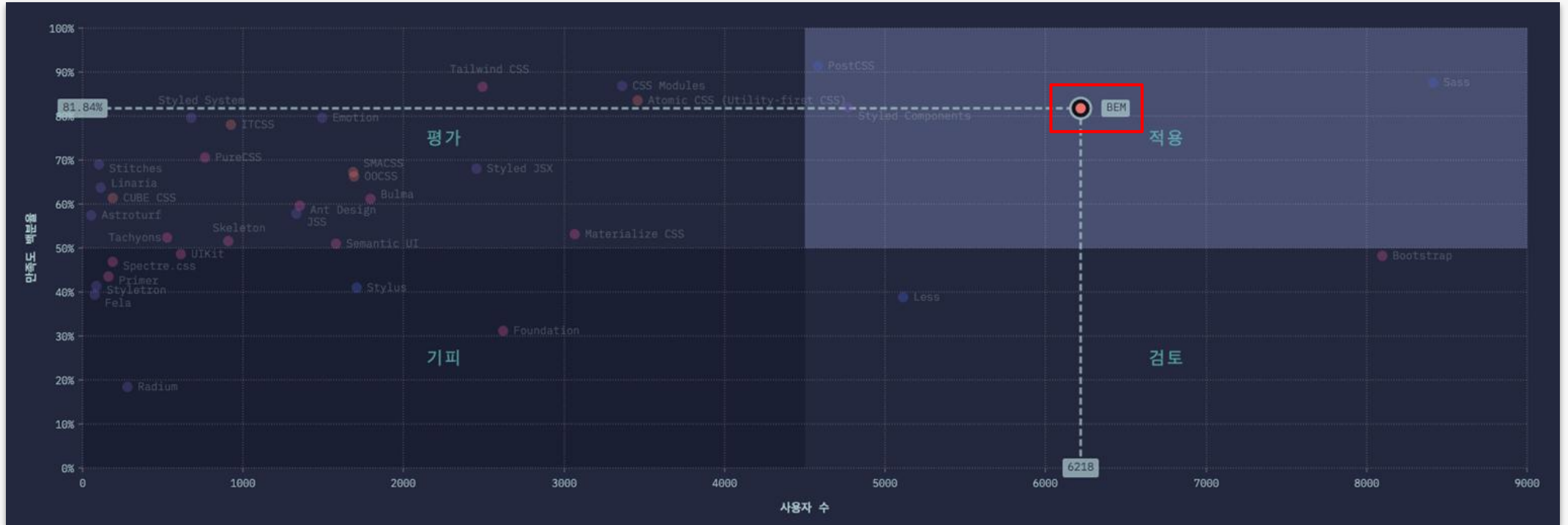
	A site	Specificity Average score Max score 32 231	B site	Specificity Average score Max score 16 110
Average	32		16	
	<code>.a .b .c { ... }</code> <code>.a .b .c .d { ... }</code>		<code>.a { ... }</code> <code>.a .b { ... }</code>	
Max ^(1~999)	231		110	
	<code>#a #b .a .b .c a { ... }</code>		<code>#a .a { ... }</code>	

CSS selector specificity

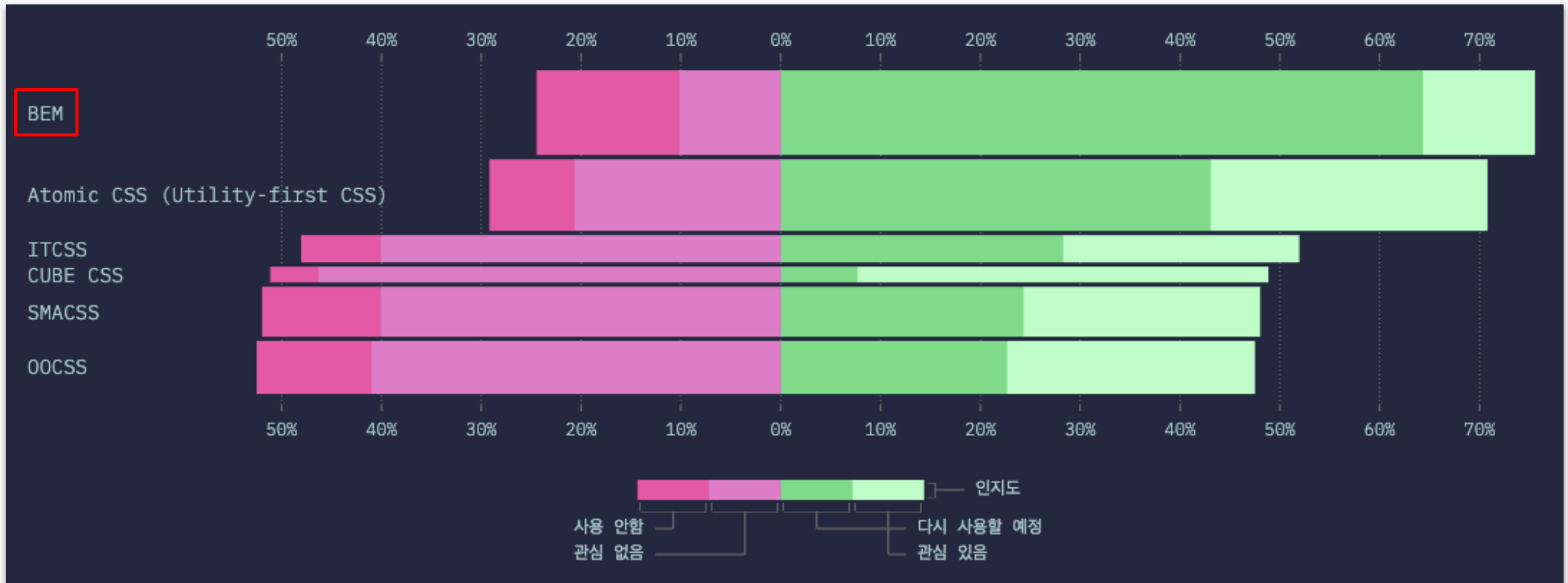


<https://csswizardry.com/2014/10/the-specificity-graph/>

BEM state 2020 (Satisfaction : 81.84%)



BEM state 2020 (Positive : 75%)



“

BEM BASIC



BEM : 명명 규칙

Block

재사용 가능한 독립적인 블록.

Element

블록을 구성하는 종속적인 하위 요소.

Modifier

블록 또는 요소의 변형(모양, 상태, 동작).

BEM : 특징 (엄격한 의미론)

1. 의미론적 클래스 선택자 작명 규칙.
2. 다른 형식의 선택자 사용을 제한.
3. 전역에서 유일한 이름 권장.
4. 낮은 선택자 특이성 유지.
5. HTML/CSS 연결이 느슨. 병렬 개발 가능

BEM : 명명 규칙

1. 두 개의 언더바(__*)는 하위 요소를 의미.
2. 두 개의 하이픈(--*)은 상태 변형을 의미.
3. 하나의 이름에 요소, 변형은 각 한 번만 허용.

BEM : 명명 규칙

```
.block {...}  
.block__element {...}  
.block__element--modifier {...}  
.block--modifier {...}
```

구분자(__, --)로 분리한 1~3개의 설명자 형식 외 다른 형식을 허용하지 않음.

BEM : 선택 사항

선택 사항: 키워드 연결 방법

1. PascalCase
2. camelCase ☆
3. kebab-case
4. snake_case

Bem : 응용 예제

이름 공간을 위한 접두어 사용 추천, camelCase 사용 예시.

.IzModal { ... }


.IzModal__title { ... }

.IzBtn { ... }

.IzBtn--small { ... }

다른 라이브러리와 공존 가능.

<https://getbootstrap.com/docs/5.0/components/buttons/>



```
class="btn btn-primary">I
class="btn btn-secondary"
class="btn btn-success">S
class="btn btn-danger">D
class="btn btn-warning">W
class="btn btn-info">Inf
class="btn btn-light">Li
class="btn btn-dark">Dar
```

“

BEM EXAMPLE



BEM Example

'블록'이 요소 또는 변형을 반드시 요구하는 것은 아니다.

```
// 단순 블록 ○  
<button class="btn">
```

'변형'은 블록 또는 요소의 스타일을 확장한다.

```
// 변형 추가 ○  
<button class="btn btn--submit">  
<em class="info__label info__label--warning">
```

BEM Example

'변형' 클래스 단독 사용 불가. 항상 블록 또는 요소와 함께 사용.

// ✕

```
<button class="btn--submit">
```

// ○

```
<button class="btn btn--submit">
```

```
<em class="info__label info__label--warning">
```

BEM Example

'선택자 특이성'이 높아지는 중첩 구조, 타입 선택자는 안티 패턴.

// ✕

```
.photo {} /* 특이성 10 */
```

```
.photo img {} /* 특이성 11 */
```

```
.photo figcaption {} /* 특이성 11 */
```

BEM Example

제어하려는 모든 요소에 클래스 이름을 부여. 특이성을 관리한다.

```
// ○  
.photo {} /* 특이성 10 */  
.photo__img {} /* 특이성 10 */  
.photo__caption {} /* 특이성 10 */
```

BEM Example

블록/요소 이름 생략 금지. 요소/변형 이름 중복 금지.

```
// ✕  
.__elem { ... }  
.--modi { ... }  
.block__elem1__elem2 { ... }  
.block--modi1--modi2 { ... }
```

“

BEM SUMMARY



“ Summary

1. 의미론 작명법으로 읽고 이해하기 쉽다.
2. 생소한 이름에 약어를 사용하지 않는다.
3. 특이성을 '020' 보다 작게 유지한다.
4. 선택자 이름은 전역 공간에서 유일하다.
5. HTML/CSS 병렬 개발 가능.

“

Atomic / Utility First CSS



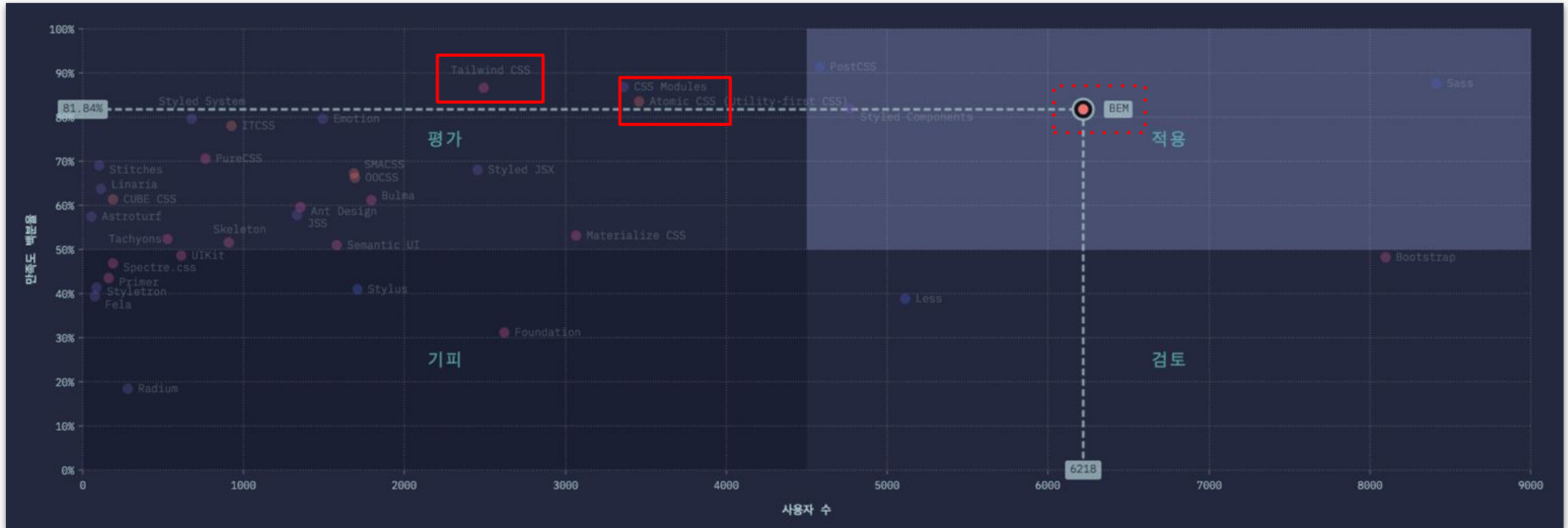
Atomic / Utility First CSS

```
<button class="w-1/2 flex items-center justify-center rounded-md bg-black text-white" >Buy now</button>
```

Atomic / Utility First CSS

1. 라이브러리 타입으로 빠른 스타일 구축 가능.
2. 다른 방법론과 함께 사용 가능.
3. 스타일 관점의 작명. 의미론을 사용하지 않음.□
4. HTML 코드에 스타일이 강하게 연결됨.□
5. HTML/CSS 병렬 개발 불가능.□ 소규모 팀 또는 단일 엔지니어 개발에 적합.

Atomic Css state 2020



“

실습 과제



실습 과제 해설

CSS BEM 퀴즈:

<https://t.ly/Bsnb>

Thank you !

