

# 삼성청년 SW·AI아카데미

JavaScript

## <알림>

본 강의는 삼성청년SW·AI아카데미의 콘텐츠로  
보안서약서에 의거하여  
강의 내용을 어떠한 사유로도 임의로 복사, 촬영,  
녹음, 복제, 보관, 전송하거나  
허가 받지 않은 저장매체를  
이용한 보관, 제3자에게 누설, 공개,  
또는 사용하는 등의 행위를 금합니다.

# Day2-1. Front-end 프로젝트

## 챕터의 포인트

- classList 살펴보기
- chrome://newtab 프로젝트

## classList 살펴보기

## class를 추가하거나 제어를 쉽게 하는 방법

- 문자열 파싱 없이, class 속성에 값을 추가하거나 제거하기 쉽다.

A guide to  
the new  
classList API

## classList.add(String)

- class의 속성 값을 추가한다.
- 만약 이미 추가 된 속성값이라면, 이를 무시한다.

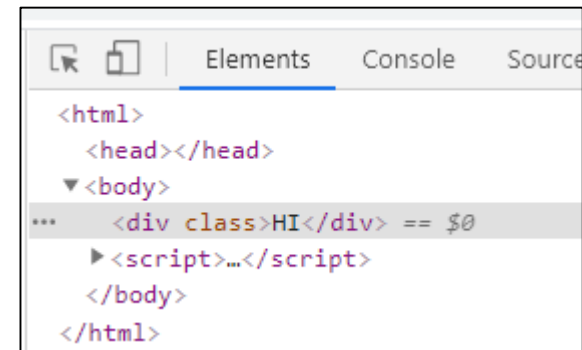
```
<> test.html > script
1  <div>HI</div>
2
3  <script>
4      const div = document.querySelector("div");
5
6      div.classList.add("test");
7
8  </script>
9
```

```
<html>
  <head></head>
  ▼<body>
**  <div class="test">HI</div> == $0
    ▶<script>...</script>
  </body>
</html>
```

## classList.remove(String)

- class 속성 값을 제거한다.
- 이미 제거되어 있다면, 이를 무시한다.

```
test.html > ...  
1  <div>HI</div>  
2  
3  <script>  
4      const div = document.querySelector("div");  
5  
6      div.classList.add("test");  
7      div.classList.remove("test");  
8  
9  </script>  
0
```





## classList.toggle(String)

- 특정 class 속성 값이 없다면, 추가한다.
- 특정 class 속성 값이 있다면, 제거한다.

```
<> test.html •
<> test.html > ...
1  <div>HI</div>
2
3  <script>
4      const div = document.querySelector("div");
5
6      div.classList.toggle("test");
7
8  </script>
9
```

class = "test" 추가

```
<> test.html ×
<> test.html > ...
1  <div>HI</div>
2
3  <script>
4      const div = document.querySelector("div");
5
6      div.classList.toggle("test");
7      div.classList.toggle("test");
8
9  </script>
10
```

class = "test" 제거

## classList.contains(String)

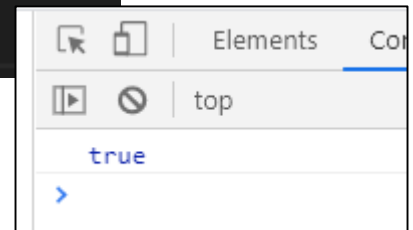
- 특정 class 명이 추가 되어있는지 확인 한다.
- true / false 를 리턴한다.

```
<div>HI</div>

<script>
  const div = document.querySelector("div");

  div.classList.toggle("test");

  let a = div.classList.contains("test");
  console.log(a);
</script>
```



## classList.replace(old, new)

- 특정 클래스 속성 값을 새로운 속성 값으로 교체한다.

```
<> test.html X
<> test.html > script
1 <div>HI</div>
2
3 <script>
4   const div = document.querySelector("div");
5
6   div.classList.toggle("test");
7
8   div.classList.replace("test", "KFC");|
9
10 </script>
11
```

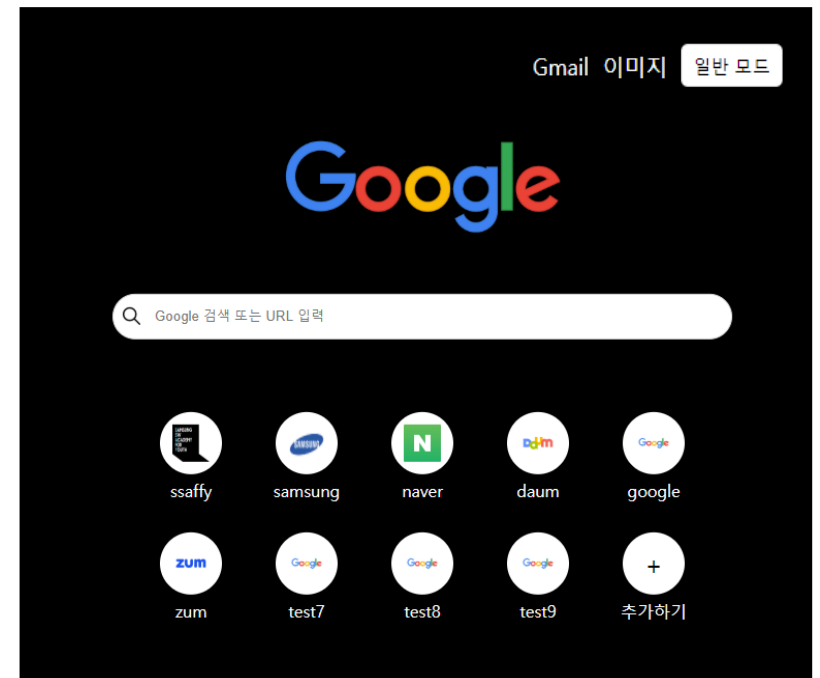
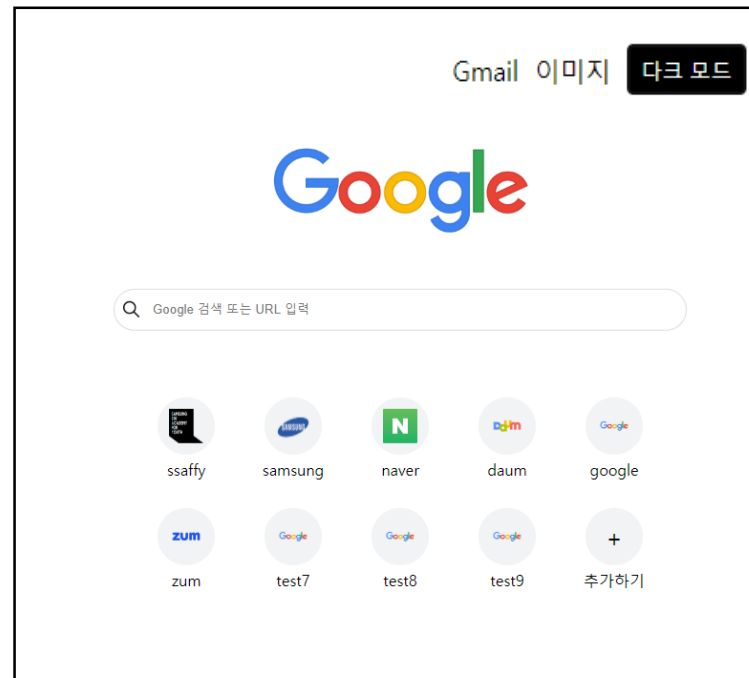
```
Elements Console Sources
<html>
  <head></head>
  ▼<body>
... <div class="KFC">HI</div> == $0
    ▶<script>...</script>
  </body>
</html>
```

- 버튼 세 개 만들어서 버튼을 눌렀을때 bg-yellow 관련 추가해주기
  - 첫번째 버튼은 class 추가 버튼
  - 두번째 버튼은 class 제거 버튼
  - 세번째 버튼은 class 토글 버튼
  - class “bg-yellow”는 css(#bg-yellow)에서 배경색을 노란색으로 설정.

**chrome://newtab 프로젝트**

## 요구사항

- 다크 모드 구현하기
- 디테일한 CSS : input, button, layout 등 CSS 작업
- 검색 기능 구현 : 실제로 구글 검색이 되어야 함



# Day2-2. jQuery 알아보기

## 챕터의 포인트

- jQuery 개요
- jQuery 설치
- jQuery 활용



# jQuery 개요

# jQuery, 웹 개발자의 고민을 덜어줌

## 당시

- ES5 때까지 각 웹브라우저 별로 웹페이지 호환성을 맞추어야 했음
- 개발자가 브라우저마다 JavaScript, CSS 를 따로 맞추어주는 작업을 해야 함
- 모든 웹브라우저에서 테스트 및 디버깅은 필수 작업

## jQuery 등장이후

- jQuery Library만 쓰면  
모든 웹에서 동일한 화면이 보여짐
- jQuery Library만 배우면  
JavaScript 고유 문법을 안쓰고 프로그래밍 가능
- 심지어  
JavaScript 보다 jQuery 문법 위주로 공부하기도 했었음



# jQuery 설치

## 1. Document 접속하기

## 2. CDN Link를 Copy 하여 소스코드에 붙여넣기

- URL 링크 거는 방식이라 다운로드가 필요없음
  - CSS는 <head> 안에
  - JS는 </body> 바로 위에 넣기

jQuery

3.x snippet:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

2.x snippet:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
```

1.x snippet:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
```

site:

[jquery.com](https://jquery.com)

versions:

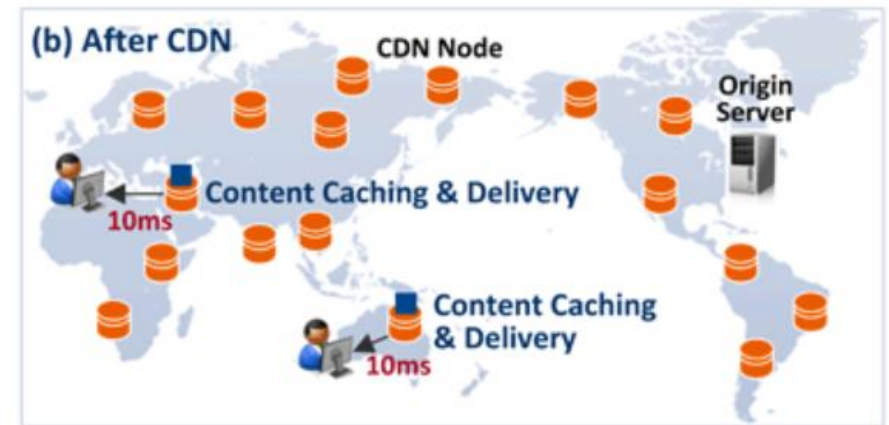
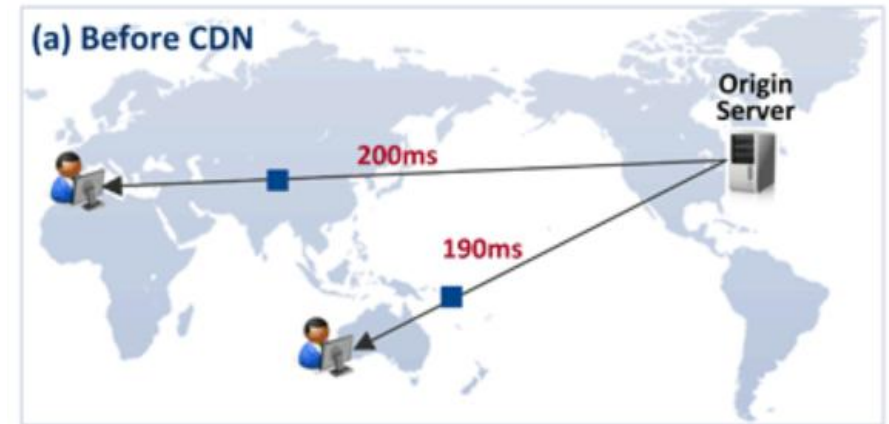
3.5.1, 3.5.0, 3.4.1, 3.4.0, 3.3.1, 3.2.1, 3.2.0, 3.1.1, 3.1.0, 3.0.0, 2.2.4, 2.2.3, 2.2.2, 2.2.1, 2.2.0, 2.1.4, 2.1.3, 2.1.1, 2.1.0, 2.0.3, 2.0.2, 2.0.1, 2.0.0, 1.12.4, 1.12.3, 1.12.2, 1.12.1, 1.12.0, 1.11.3, 1.11.2, 1.11.1, 1.11.0, 1.10.2, 1.10.1, 1.10.0, 1.9.1, 1.9.0, 1.8.3, 1.8.2, 1.8.1, 1.8.0, 1.7.2, 1.7.1, 1.7.0, 1.6.4, 1.6.3, 1.6.2, 1.6.1, 1.6.0, 1.5.2, 1.5.1, 1.5.0, 1.4.4, 1.4.3, 1.4.2, 1.4.1, 1.4.0, 1.3.2, 1.3.1, 1.3.0, 1.2.6, 1.2.3

note:

3.3.0, 2.1.2, 1.2.5 and 1.2.4 are not hosted due to their short and unstable lives in the wild.

## 빠른 해외 웹사이트를 접속하기 위한 기술

- 해외 접속자들이 빠르게 서버를 이용할 수 있도록 세계 각국에 Caching Server를 두어 가장 가까운 서버에 접속 할 수 있도록 하는 서비스



CDN (Content Delivery Network)

## 1. JavaScript 링크

- jquery.min.js

## xxx.min.js 파일

- 가독성을 포기하고, 공백 / 줄바꿈을 제거하여 파일 용량을 줄인 파일
- 변경할 예정이 없는 Library에 사용한다.



# jQuery 활용

## \$(선택자).동작함수() 방식

- \$는 제이쿼리를 의미하며 제이쿼리에 접근 할 수 있게 해주는 식별자

```
7     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
8 </head>
9 <body>
10    <div class="hi">hello</div>
11    <script>
12        $(".hi");
13        console.log($(".hi"));
14    </script>
```



`$(선택자).on("이벤트이름", 함수);`

- `addEventListener` 대신 `on` 함수를 활용하여 이벤트를 등록한다.

```
<body>
  <div class="hi">hello</div>
  <button class="click-button">클릭 버튼 </button>
  <script>
    $(".click-button").on("click", function(){
      alert("hello");
    });
  </script>
</body>
```

## \$(선택자).css(name, value);

- \$는 제이쿼리를 의미하며 제이쿼리에 접근 할 수 있게 해주는 식별자

```
<body>
  <div class="hi">hello</div>
  <button class="click-button">클릭 버튼 </button>
  <script>
    $("body").css("backgroundColor", "gray");
```

# Day2-3. Bootstrap 5 이해

## 챕터의 포인트

- jQuery를 걷어낸 Bootstrap 5
- Bootstrap 5 설치
- Components와 Forms 다루기

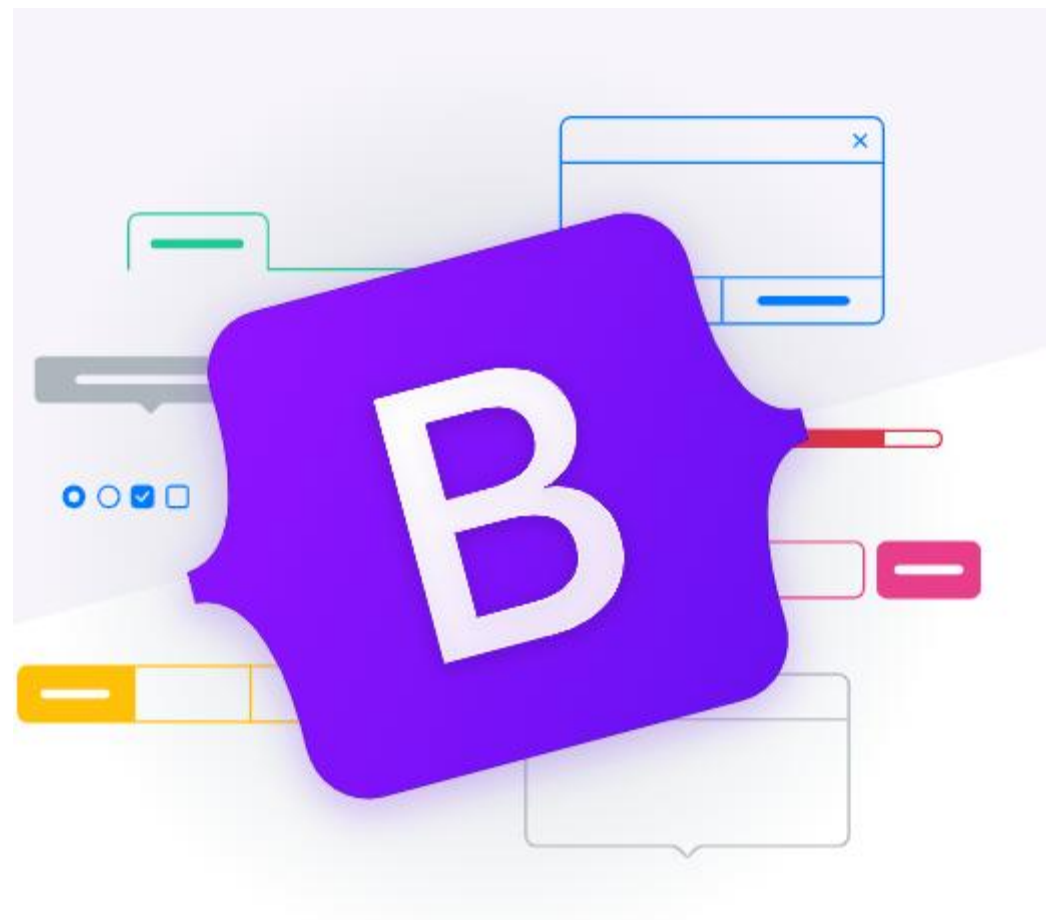
## jQuery를 건너낸 Bootstrap 5

## 손쉽게 고급 UI를 만들 수 있는 Front-end Library

- CSS와 JavaScript를 사용하여 만들어진 Library
- 빠른 속도로 멋진 UI의 웹 페이지를 개발할 수 있다.

## 장점

- 멋진 홈페이지를 빠르게 만들 수 있다.
- 반응형 웹을 쉽게 만들 수 있다.



## Bootstrap 5

- jQuery 걷어낸 뒤, 바닐라 자바 스크립트로 전환

jQuery를 더 이상 쓰지 않아도 됨

## ES6 이후로, jQuery의 장점이 사라짐

- 웹 브라우저마다 동일한 화면을 보여주기 시작함

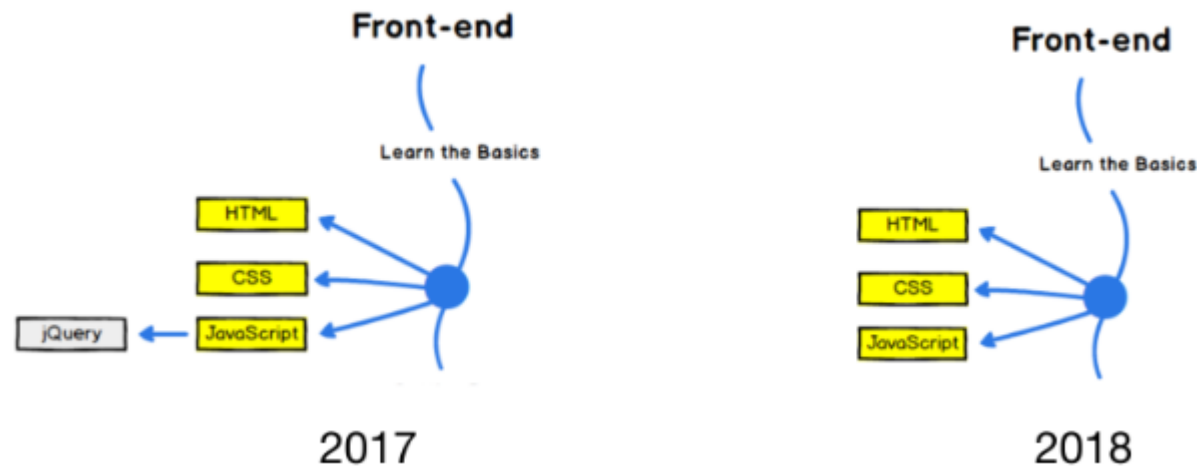
## jQuery의 단점이 부각되다.

- DOM Tree Selector가 느리다.
- jQuery만의 문법을 새로 배워야 한다.
- jQuery Ajax 사용시, 서버 부담이 크다.



## 웹 개발자 필수 Stack에서 jQuery가 빠짐

- GitHub의 웹 개발자 Stack Roadmap에서 jQuery가 필수 Stack 목록에서 제거 됨
- <https://github.com/kamranahmedse/developer-roadmap>
- Bootstrap 5 부터 트렌드에 맞추어 jQuery를 제거 했다.



## Bootstrap 5 설치

구글 검색 : “bootstrap”



## 1. Document 접속하기

## 2. CDN Link를 Copy 하여 소스코드에 붙여넣기

- URL 링크 거는 방식이라 다운로드가 필요없음
  - CSS는 <head> 안에
  - JS는 </body> 바로 위에 넣기

### Quick start

Looking to quickly add Bootstrap to your project? Use BootstrapCDN, provided for free by the folks at StackPath. Using a package manager or need to download the source files? [Head to the downloads page.](#)

### CSS

Copy-paste the stylesheet `<link>` into your `<head>` before all other stylesheets to load our CSS.

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/5.0.0-alpha1/css/bootstrap.min.css">
```

### JS

Many of our components require the use of JavaScript to function. Specifically, they require our own JavaScript plugins and [Popper.js](#). Place the following `<script>`s near the end of your pages, right before the closing `</body>` tag, to enable them. Popper.js must come first, and then our JavaScript plugins.

```
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Cp3ckJia" data-cs="3" data-kind="parent"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/5.0.0-alpha1/js/bootstrap.min.js" integrity="sha384-Cp3ckJia" data-cs="3" data-kind="parent"></script>
```

## 1. CSS 링크 (Head 내부)

- bootstrap.min.css : bootstrap css 파일

## 2. JavaScript 링크 (Body 하단)

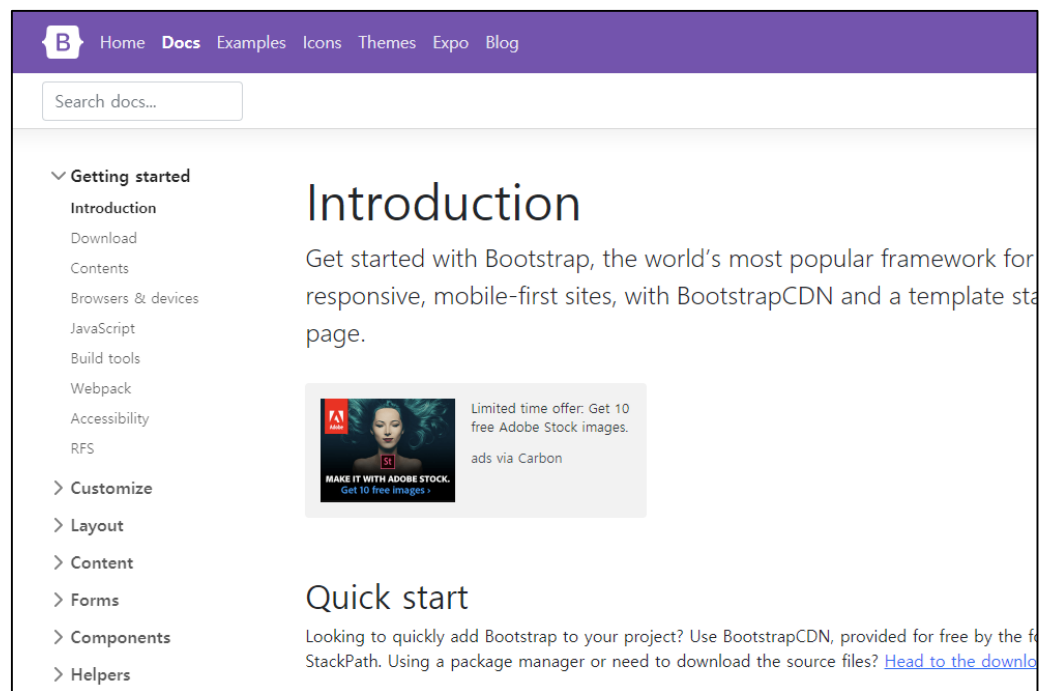
- popper.min.js : 툴팁 메시지 외부 Library
- bootstrap.min.js : bootstrap js 파일

## xxx.min.css 파일

- 가독성을 포기하고, 공백 / 줄바꿈을 제거하여 파일 용량을 줄인 파일
- 변경할 예정이 없는 Library에 사용한다.

Bootstrap은 규모가 있는 Library 인 만큼  
필요할 때 마다 Document 참고하여 구현

- 이 중 자주 쓰는 기능을 함께 실습 할 예정
- 실습하면서 해당 Page를 참고하면서 구현 할 것



## Button 2개를 추가해보자

- 설치가 잘 되어있는지 확인차 버튼 2개 추가
- Bootstrap 사용방법
  - 정해진 class 속성 값을 추가하는 방식으로 Bootstrap을 사용할 수 있다.

```
hello.html
hello.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-sc
6      <title>Document</title>
7      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.c
8  </head>
9  <body>
10
11      <button class="btn btn-primary">B.B.Q</button>
12      <button class="btn btn-success">K.F.C</button>
13
14      <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/di
15      <script src="https://stackpath.bootstrapcdn.com/bootstrap/5.0
16  </body>
17  </html>
18
```

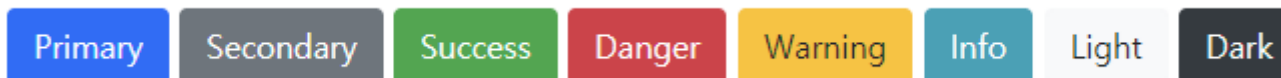
## Components와 Forms 다루기



## 버튼을 추가하고, 기능 구현하기

1. Document에서 마음에 드는 버튼 2개 추가한다.
2. 버튼의 사이즈를 조절한다. (큰 버튼 1개 / 작은 버튼 1개)
3. 큰 버튼 클릭시 : alert (“HI”);
4. 작은 버튼 클릭시 : 버튼 크기가 달라진다.

class = “btn btn-xxxx”

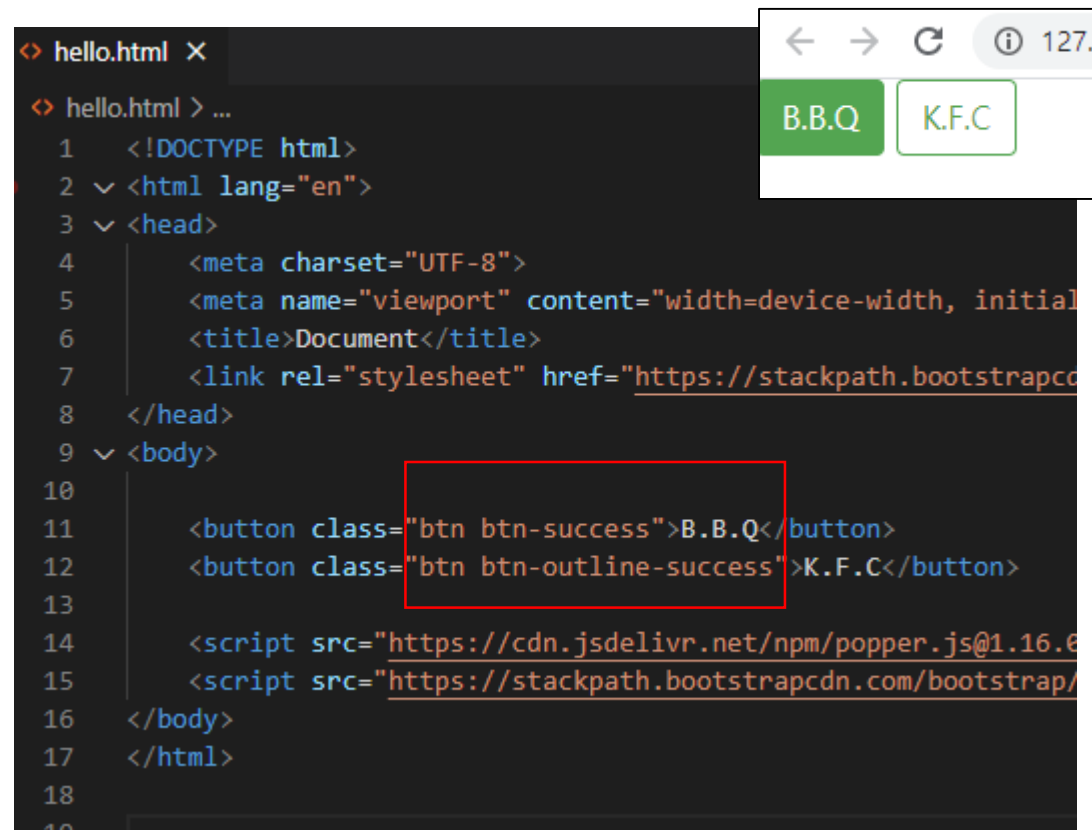


class = “btn btn-outline-xxxx”



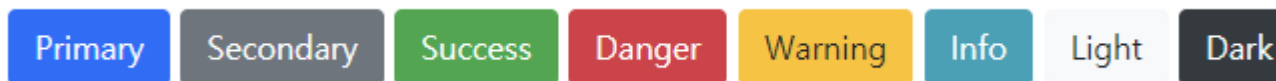
## 버튼 2개 추가하기

- 일반적인 btn과 outline btn 하나씩 고른다.
- class 속성 값은 모두 소문자로 기입한다.

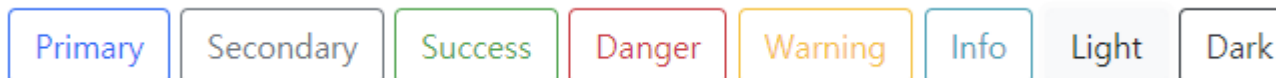


```
hello.html x
hello.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial
6   <title>Document</title>
7   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com
8 </head>
9 <body>
10
11   <button class="btn btn-success">B.B.Q</button>
12   <button class="btn btn-outline-success">K.F.C</button>
13
14   <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0
15   <script src="https://stackpath.bootstrapcdn.com/bootstrap/
16 </body>
17 </html>
18
19
```

class = "btn btn-xxxx"



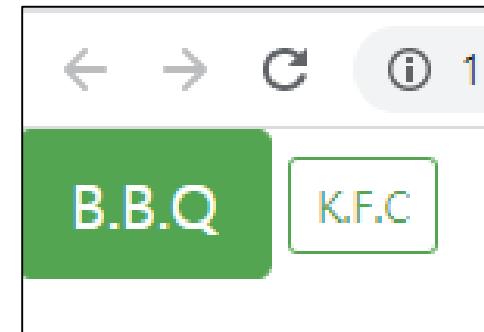
class = "btn btn-outline-xxxx"



## 버튼 사이즈 조절하기

- btn-lg : 빅 사이즈, large 속성 값을 추가한다.
- btn-sm : 스몰 사이즈, small 속성 값을 추가한다.

```
hello.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, ini
6   <title>Document</title>
7   <link rel="stylesheet" href="https://stackpath.bootstr
8 </head>
9 <body>
10
11   <button class="btn btn-success btn-lg">B.B.Q</button>
12   <button class="btn btn-outline-success btn-sm">K.F.C</
13
14   <script src="https://cdn.jsdelivr.net/npm/popper.js@1.
15   <script src="https://stackpath.bootstrapcdn.com/bootst
16 </body>
17 </html>
18
```



## 큰 버튼 클릭시 click event 처리

- 큰 버튼과 작은 버튼에 id 속성값을 부여한다. (ex. id="bbq", id="kfc")
- alert 메시지를 출력한다.

```
<button id="bbq" class="btn btn-success btn-lg">B.B.Q</button>
<button id="kfc" class="btn btn-outline-success btn-sm">K.F.C</button>

<script>
  const btn1 = document.querySelector('#bbq');
  const btn2 = document.querySelector('#kfc');

  btn1.addEventListener('click', gogo1);

  function gogo1() {
    alert("HI");
  }
</script>
```

## 두 번째 버튼 크기 조절

- 작은 버튼 상태에서 클릭하면 큰 버튼으로 변경
- 큰 버튼 상태에서 클릭하면 작은 버튼으로 변경

## 사용한 메서드

- `str = str.replace( 단어1, 단어2 )` : 단어 교체하기
- `getAttribute( '속성이름' )` : 속성 값 가져오기
- `setAttribute( '속성이름', '속성값' )` : 속성 값 변경

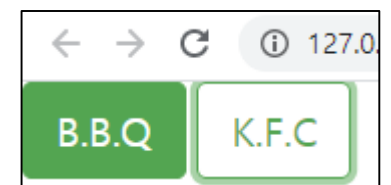
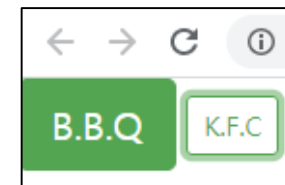
위와 같이 파싱하지 않고,  
`classList`로도 구현 가능

```
<script>
const btn1 = document.querySelector('#bbq');
const btn2 = document.querySelector('#kfc');

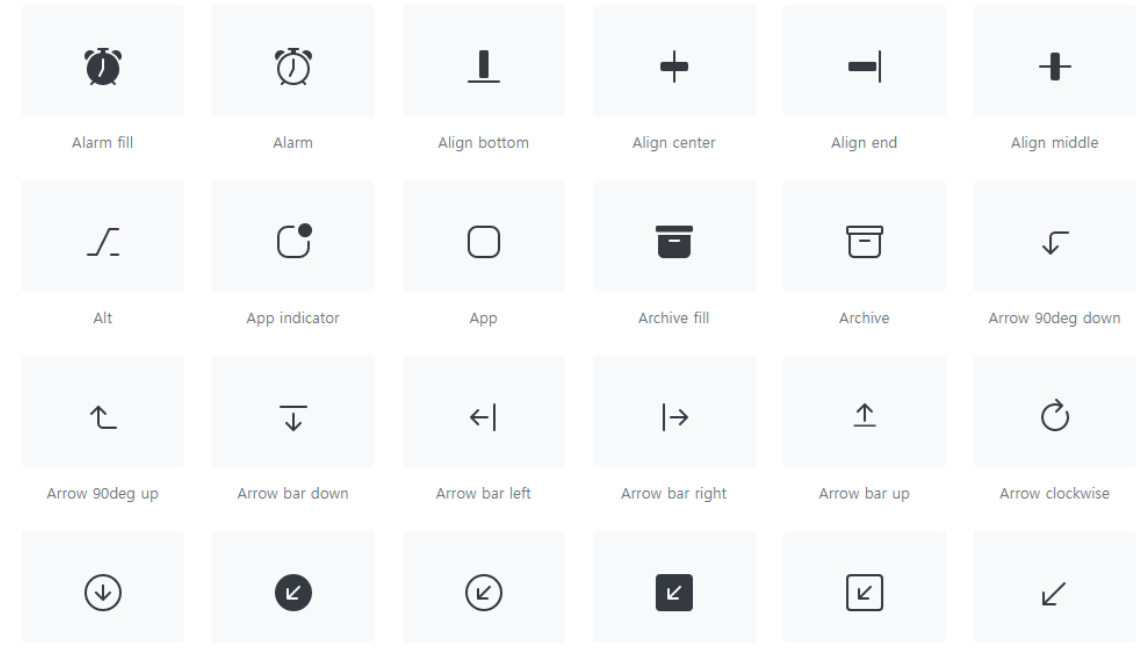
btn1.addEventListener('click', gogo1);
btn2.addEventListener('click', gogo2);

function gogo1() {
    alert("HI");
}

function gogo2() {
    let str = btn2.getAttribute('class');
    if (str.indexOf("btn-lg") !== -1) {
        str = str.replace("btn-lg", "btn-sm");
    }
    else if (str.indexOf("btn-sm") !== -1) {
        str = str.replace("btn-sm", "btn-lg");
    }
    btn2.setAttribute('class', str);
}
</script>
```

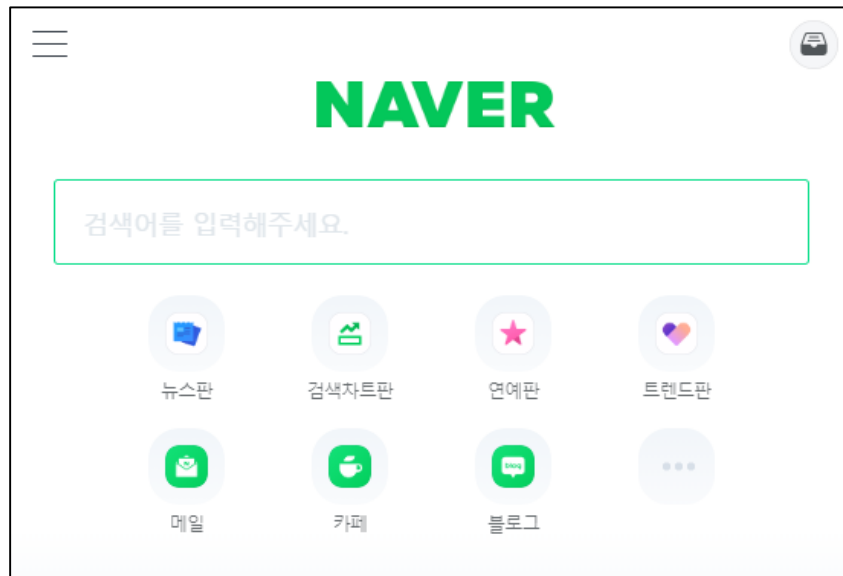


Form Control : input  
From Select : select  
icon : bootstrap icon

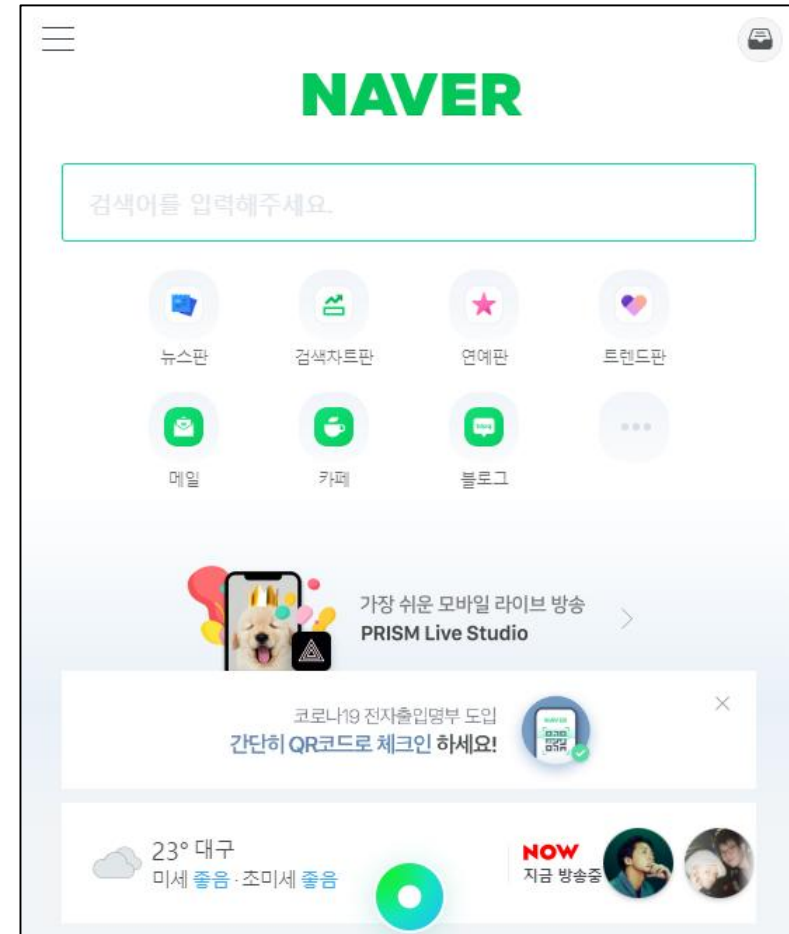
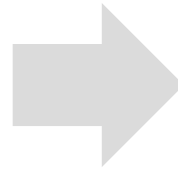


## 네이버 모바일 만들기

- 비슷하게 만들기
- HTML / JavaScript / CSS 사용
- 다크모드는 선택사항



Normal 버전 도전 과제



웹고수를 위한 도전 과제

## Day2-4. 자바스크립트 심화



## 챕터의 포인트

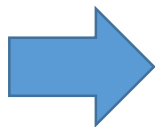
- ES6 문법 정리
- 화살표 함수
- 긍정 부정

## ES6 문법 정리

## 백틱 ``

- 키보드 왼쪽 Tab 버튼위에 위치
- “”와 다르게 Enter를 인식한다.
- 문자열과 변수를 같이 사용할 수 있다.
  - 변수를 사용하는 경우 \${변수} 형태로 사용한다

```
const name = "이온유";  
undefined  
const introduce = "제 이름은 " + name + "입니다" ;  
undefined  
introduce  
'제 이름은 이온유입니다'
```



```
const name = "이온유";  
const introduce = `제 이름은 ${name} 입니다` ;  
introduce  
'제 이름은 이온유 입니다'
```

## insertAdjacentHTML(position, '문자열 형태의 태그')

- 문자열 형식의 태그를 집어 넣을 때 사용
- position
  - beforebegin
    - 요소 바로 이전에 삽입
  - afterbegin
    - 요소 첫 번째 자식 앞에 삽입
  - beforeend
    - 요소 마지막 자식 뒤에 삽입
  - afterend
    - 요소 바로 다음에 삽입

```
<body>

  <script>

    const htmlTag = `
      <button>
        안녕하세요
      </button>
    `;

    const body = document.querySelector("body");

    //HTML에 텍스트 형식의 HTML 태그 넣을때 사용
    body.insertAdjacentHTML('beforeend', htmlTag);
    body.insertAdjacentHTML('beforeend', htmlTag);

  </script>
</body>
```

## Destructuring - 객체

- 객체 안에 있는 값을 추출해서 변수/상수로 선언하는 방식

```
const abc = {  
  name: "치킨",  
  type: "후라이드"  
}
```

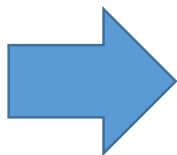
undefined

```
abc['name']
```

'치킨'

```
abc.name
```

'치킨'



```
const abc = {  
  name: "치킨",  
  type: "후라이드"  
}
```

undefined

```
const { name, type } = abc;
```

undefined

name

'치킨'

type

'후라이드'



```
const abc = {  
  name: "치킨",  
  type: "후라이드"  
}
```

undefined

```
const {  
  name: name,  
  type: type  
} = abc;
```

undefined

name

'치킨'

type

'후라이드'

## Destructuring - 배열

- 객체와 다르게 배열은 순서(index)에 따라 값이 할당되므로, 변수명은 자유롭게 지정할 수 있다.

```
const array = [1, 2];
```

```
undefined
```

```
const [one, two] = array;
```

```
undefined
```

```
one
```

```
1
```

```
two
```

```
2
```

# Spread

- 객체/배열을 통째로 가지고 와서 펼치는 형태

```
const square = {
  width: 200,
  height: 200,
}

const colorSquare = {
  ...square,
  color: 'red'
}

console.log(colorSquare)
▶ {width: 200, height: 200, color: 'red'}
```

객체 Spread

```
const catTypeAnimals = ['고양이', '호랑이'];
const dogTypeAnimals = ['개', '늑대'];

const allTypeAnimals = [...catTypeAnimals, ...dogTypeAnimals, '비버', ];
undefined

allTypeAnimals
▶ (5) ['고양이', '호랑이', '개', '늑대', '비버']
```

배열 Spread

# Rest

- 나머지 객체, 배열을 Rest에 담아서 추출하는 방법

```
const chicken = {  
  type: '양념',  
  drumsticks: 2,  
  wing: 2  
}  
  
const {type, ...another} = chicken;  
console.log(type);  
양념  
undefined  
console.log(another)  
▶ {drumsticks: 2, wing: 2}
```

객체의 Rest

```
const numberArray = [0, 1, 2, 3, 4, 5, 6];  
const [one, two, ...another] = numberArray;  
undefined  
one  
0  
two  
1  
another  
▶ (5) [2, 3, 4, 5, 6]
```

배열의 Rest



## 화살표 함수

## 함수를 정의하는 방법

```
1  <script>
2
3      let a = function () {
4          alert("AA");
5      }
6
7      function b() {
8          alert("BB");
9      }
10
11     a();
12     b();
13
14 </script>
```

## 함수를 정의하는 방법

- 일반 함수 : `function () {}`
- 화살표 함수 : `() => {}`

화살표 함수 : 익명 함수

```
1  <script>
2
3      let a = function () {
4          alert("AA");
5      }
6
7      let b = () => {
8          alert("BB");
9      }
10
11     a();
12     b();
13
14 </script>
15
```

## 화살표 함수는 익명 함수이다.

- 값을 전달하고, 리턴 가능

```
1  <script>
2
3      let a = (a, b) => {
4          alert(a + b);
5      }
6
7      a(3, 4);
8
9  </script>
```

a, b, 받고 합 출력

```
1  <script>
2
3      let a = (a, b) => {
4          return a + b;
5      }
6
7      let sum = a(3, 4);
8      alert(sum);
9
10 </script>
```

a, b 받고 합 return 하기

## 3을 전달 하여 제곱을 반환하는 화살표 함수 만들기

- ex. `go(3)` → 9 출력
- ex. `go(10)` → 100 출력

## 화살표 함수의 더 간략한 표현

- Argument가 1 개 일 때, 괄호 생략 가능
- 다른 코드 없이 Return만 하는 화살표 함수는 {} 생략 가능
- 한 줄짜리 단일 실행문 함수도 {} 생략 가능

```
1 <script>
2
3   let a = v1=> { alert(v1); };
4   //let a = (v1)=> { alert(v1); };
5
6   a(15);
7
8 </script>
```

전달값이 1개이기에  
( ) 소괄호 생략 가능

```
1 <script>
2
3   let a = v1=> v1 + 15;
4   //let a = (v1)=> { return v1 + 15 };
5
6   let ret = a(15);
7   alert(ret);
8
9 </script>
```

return 만 수행하는 코드이기에  
{ } 중괄호 생략 가능

다음 소스코드를 보고 실행 결과를 예측해보자.

```
const n = 10;  
const sampleFunc = (n) => n * 10;  
const a = n * 20;  
  
const result = sampleFunc(10) + a;  
console.log(result);
```

1

```
const func1 = (v1, v2) => v1 * v2;  
const a = func1(4, 5) + func1(5, 6);  
console.log(func1(1, 10) + a);
```

2

```
const run = (func1, func2) => func1(1, 2) * func2(3, 4);  
const result = run(  
  (a, b) => a + b,  
  (a, b) => a * b  
);  
console.log(result);
```

3

## 긍정 부정



## • 부정의 의미

- undefined
- null
- 0
- ""(빈 문자열)
- false

## • 긍정의 의미

- 부정의 의미를 제외한 모든 것
- 빈 배열과 빈 객체는 긍정의 의미를 가지고 있다.

```
console.log(!!undefined)
console.log(!!null)
console.log(!!0)
console.log(!!'')
console.log(!!false)
```

false

false

false

false

false

!! -> 부정의 부정 -> 긍정

# Day2-5. Object와 Class

## 챕터의 포인트

- JavaScript의 Object
- Prototype의 이해
- JavaScript에서 Class

# JavaScript의 Object

## 변수의 타입

### • 원시 타입

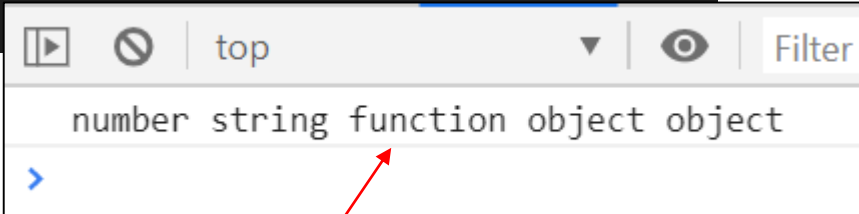
- number
- string
- Boolean
- null
- undefined

### • 객체 타입

- array
- **function**
- object

ex. let a = new Number(10);

```
1  <script>
2
3    let a = 153;
4    let b = "ABC";
5    let c = function BTS() { };
6
7    let d = { };
8    let e = [3, 4, 5];
9
10   console.log(typeof a, typeof b, typeof c, typeof d, typeof e);
11
12 </script>
```

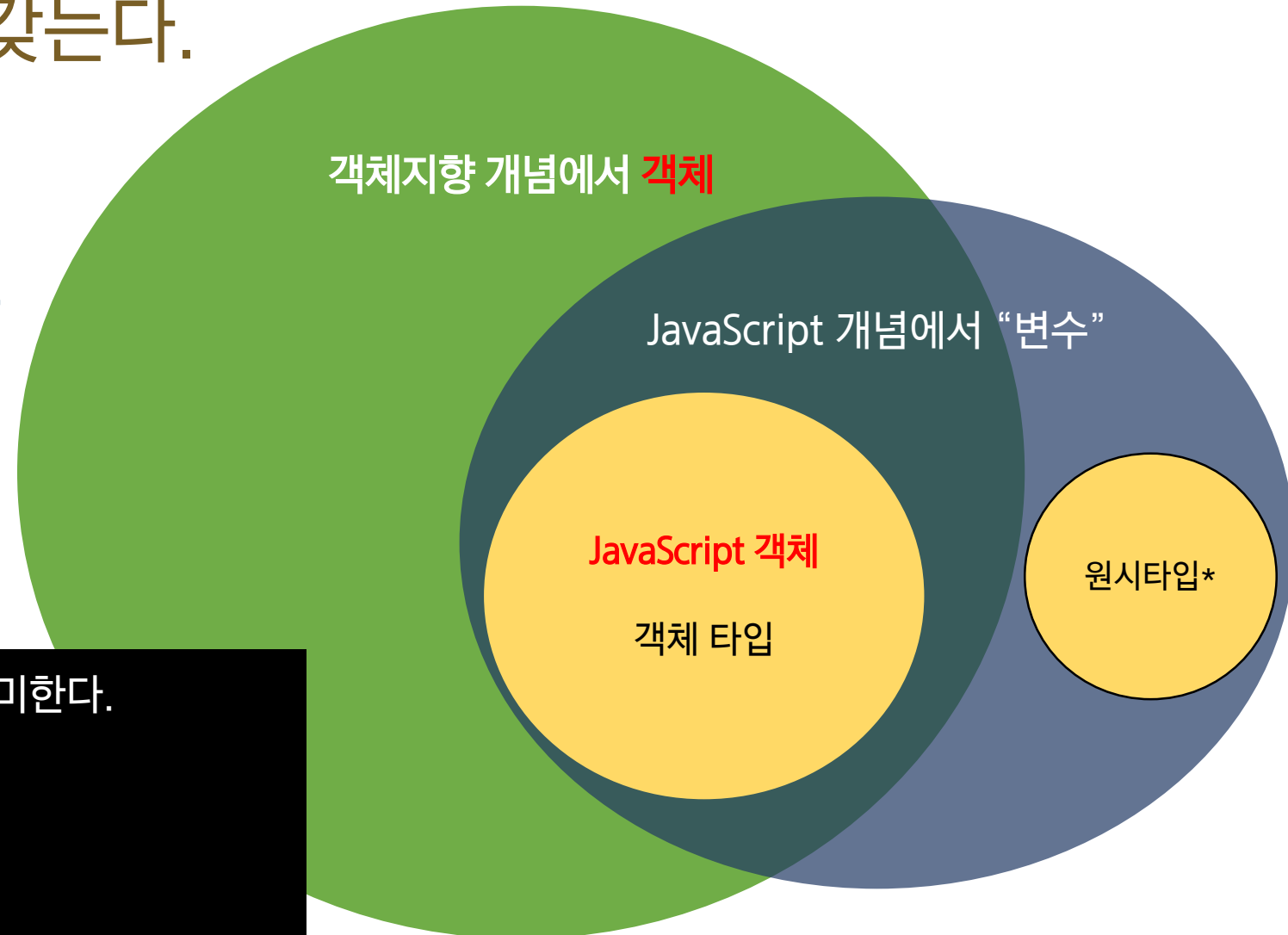


number string function object object

실제로 함수 Type은 존재하지 않음  
호환성을 위해 function으로 표기

## 객체는 다음과 같은 의미를 갖는다.

- OOP 개념에서 모든 변수는 객체이다.  
(내장 속성 or 메서드를 가짐)
- JavaScript 개념에서 Object Type으로 만들어진 변수는 **객체**이다.



따라서 JavaScript 객체는 다음과 같은 것을 의미한다.

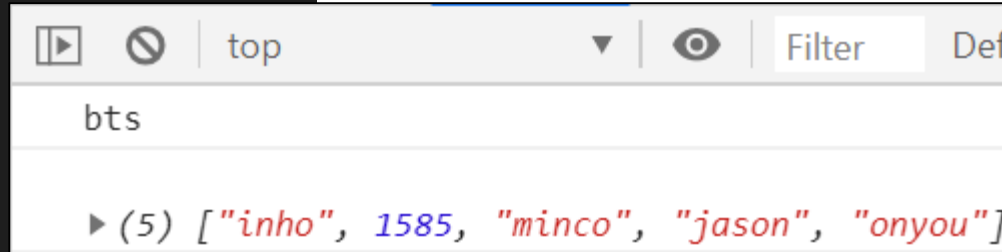
1. 키와 값으로 이루어진 프로퍼티의 모음
2. Array
3. Function
4. new + 모든 Types

## Object Type으로 만든 변수 = “객체”

- 여러 Type들을 넣을 수 있는 집합 ( 타 언어에서 class와 비슷한 개념 )
- 속성값으로 number, string, function 등 가능

```
1 <script>
2
3   let a = { };
4
5   let b = {
6     teamName : "bts",
7     teamCount : 7,
8     member : ["inho", 1585, "minco", "jason", "onyou"],
9     other : {
10      averAge : 20,
11      averHeight : 200
12    }
13  };
14
15  console.log(b.teamName);
16  console.log(b.member);
17
18 </script>
```

JSON (JavaScript Object Notion)  
문법과 비슷하다.



변수에 1이라는 number 데이터타입을 할당해보자.

- 변수에 'ssafy'라는 문자열 데이터타입을 할당해보자.
- 변수에 bbq라는 함수를 할당해보자.
- 변수에, 1과 'ssafy'와, 함수를 모두 할당해보자.



객체는 0개 이상의 프로퍼티 & 메서드로 구성된 집합이다.

```
const obj = {  
  age : 1,  
  belong : 'ssafy',  
  bbq: function(){  
    this.age = this.age + 1;  
    console.log(this.age)  
  }  
}
```

## Object(객체)의 속성으로 함수도 추가 가능

- 객체 안에 있는 요소 : **프로퍼티** (속성)
- 객체 안에 있는 함수 : **메서드**

```
1  <script>
2
3      let a = { };
4
5      let b = {
6          teamName : "bts",
7          dance : function() {
8              alert("HI");
9          }
10     };
11
12     b.dance();
13
14 </script>
```

## 객체 리터럴 방식

```
const ssafy = {  
  name: "embedded",  
  study: function () {  
    console.log("study system programming");  
  }  
};  
  
console.log(typeof ssafy);  
console.log(ssafy);
```

## 생성자 함수에 의한 객체 생성

```
function Ssafy() {  
    this.name = "embedded";  
    this.study = function () {  
        console.log("study network programming");  
    };  
}  
  
const ssafy = new Ssafy();
```

## 절차지향 프로그래밍

- 코드를 위부터 아래로 내려가며 실행하는 방식
- 함수를 사용해 기능을 나눌 수 있지만, 데이터와 함수가 분리되어 있기 때문에 구조적 재사용에 한계가 있음
- 프로그램 규모가 커질수록 중복 코드와 전역 상태 관리 문제가 발생하기 쉬움
- 변경이 발생하면 여러 위치를 함께 수정해야 하므로 유지보수가 어려움

## 객체의 집합으로 프로그램을 표현하려는 프로그래밍 패러다임

- 기존 절차지향을 통해 느꼈던 단점들을 보완해서 탄생
  - 재사용성을 높여 코드의 재사용성 향상
  - 유지보수 우수

## 객체지향프로그래밍의 특징

- 추상화
- 캡슐화
- 상속
- 다형성

## 객체로 만든 함수를 호출하는 두가지 방법

- 객체에서 객체내부 값을 참조할 경우에는 this로 접근 가능

```
const ssafy = {  
  weight: 90,  
  eat(){  
    return this.weight + 5;  
  },  
  run(){  
    return this.weight - 3;  
  }  
}  
  
console.log(ssafy);  
console.log(ssafy.eat());  
console.log(ssafy.run());
```

객체.키값 의 형식으로 접근

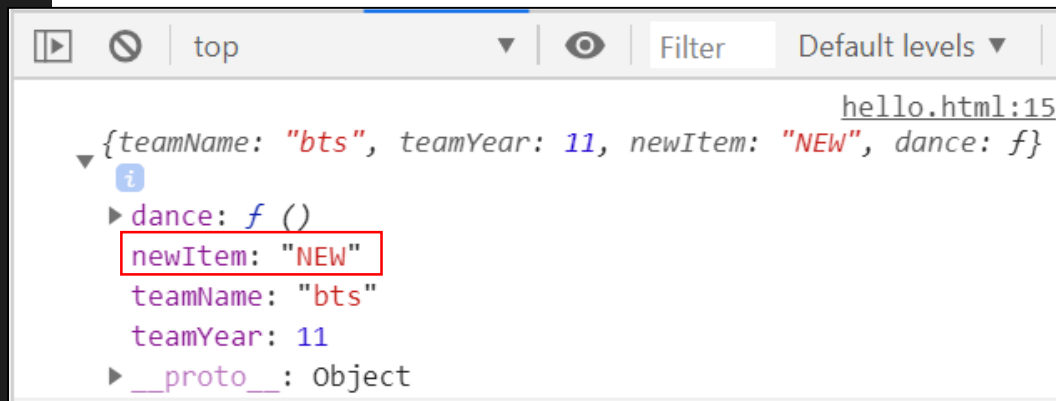
```
const ssafy = {  
  weight: 90,  
  eat(){  
    return this.weight + 5;  
  },  
  run(){  
    return this.weight - 3;  
  }  
}  
  
console.log(ssafy);  
console.log(ssafy['eat']());  
console.log(ssafy['run']());
```

객체['키값'] 의 형식으로 접근



객체는 따로 선언없이 속성을 추가할 수 있다.

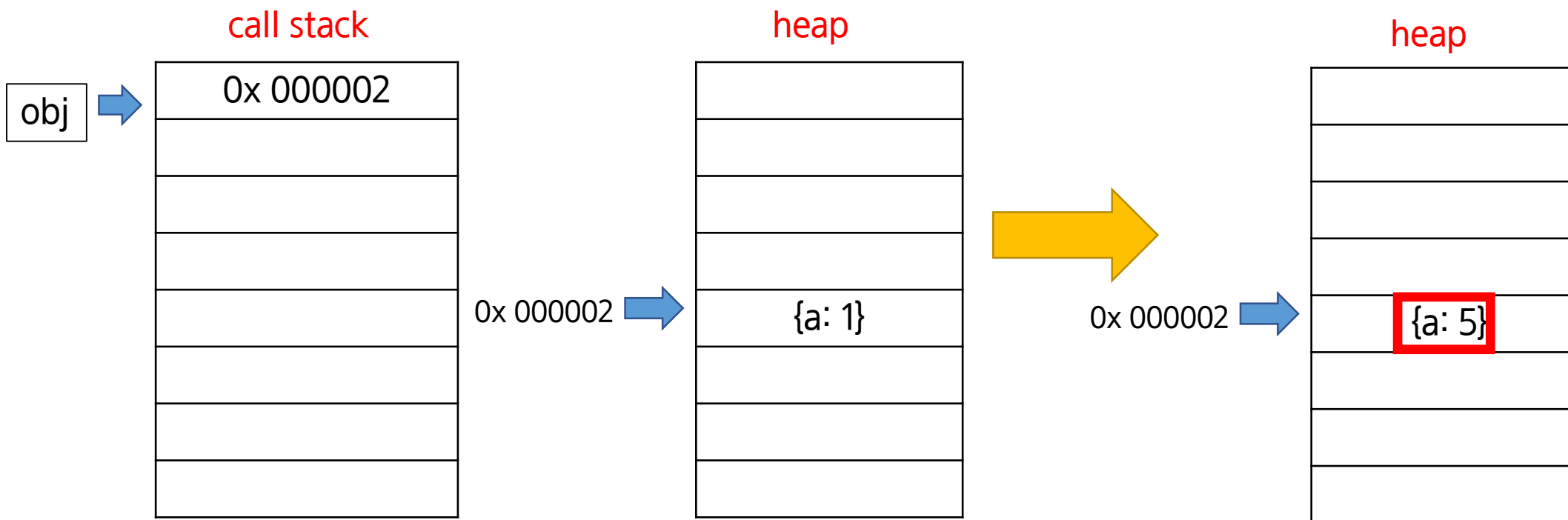
```
1 <script>
2
3   let a = { };
4
5   let b = {
6     teamName : "bts",
7     teamYear : 11,
8     dance : function()
9       alert("HI");
10    };
11
12    b.newItem = "NEW";
13
14    console.log(b);
15
16
17 </script>
```



## const 로 선언된 참조 자료형에서 값 바꾸기

- const는 기본적으로 call stack에 대한 변경을 막아 두었다.
- 객체의 값은 heap 메모리에 저장되기 때문에 변경 가능

```
> const obj = { a : 1};  
< undefined  
> obj.a = 5;  
< 5  
> console.log(obj);  
  ▶ {a: 5}
```



## Prototype의 이해

## 함수를 만드는 두 가지 방법

```
const myFunction = function() {  
  console.log("야호");  
};  
  
myFunction();
```

Function Type 변수 생성 후  
Function 호출

```
function myFunction() {  
  console.log("야호");  
}  
  
myFunction();
```

Function 생성 후 호출

함수는 객체이므로, 속성을 추가할 수 있다.

- a 속성을 추가하고 출력하는 예제

```
function myFunction() {  
  console.log("야호");  
}  
  
myFunction.a = 200;  
console.log(myFunction.a);  
myFunction();
```

```
const myFunction = function () {  
  console.log("야호");  
};  
  
myFunction.a = 200;  
console.log(myFunction.a);  
myFunction();
```

“200”이 출력된다.

JavaScript는 프로토타입 기반 객체지향 프로그래밍 언어이다.

- 객체지향 프로그래밍 언어는 class 기반과 prototype 기반이 있다.
  - class 기반 : C++ / Java / C# / Python
  - prototype 기반 : JavaScript / Lua / R / Perl
- JavaScript는 Prototype 기반 언어이므로, 과거 class가 없었으나 ES6부터 Prototype 문법을 이용하여 class 문법이 추가 되었다.
- JavaScript의 핵심 원리 중 하나인 “프로토타입” 을 이해하는 챕터

## 객체지향 프로그래밍 비교

	클래스 기반 프로그래밍	프로토타입 기반 프로그래밍
객체 생성 방법	클래스를 정의하고, 클래스로부터 객체를 생성	프로토타입 객체를 복사하여 객체 생성
클래스 정의	클래스로 객체의 속성과 메서드 정의	클래스 없이도 프로토타입 객체를 정의
상속	일반적으로 클래스 간 상속을 지원	프로토타입 체인을 이용한 객체 간 상속
언어 예시	Java, C++, Python 등	JavaScript
동적 객체 생성	클래스 기반으로 객체 생성 후 동적 변경 가능	프로토타입 객체를 통해 동적 변경 가능
유연성	상대적으로 덜 유연함	상대적으로 더 유연함
일반적인 사용 사례	정적인 객체 구조가 필요한 경우	동적인 객체 구조가 필요한 경우

## 프로토타입을 활용해서 중복을 최소화

- 같은 메모리의 객체를 사용할 수 있게 된다.

```
function Chicken(){
  this.information = {
    head : "머리",
    leg : "다리"
  }
}

const b = new Chicken();
const c = new Chicken();

console.log(b.information === c.information)
false
```

일반 함수에 객체를 선언한 경우

```
function Chicken(){
}

Chicken.prototype.information = {
  head : "머리",
  leg : "다리"
}

const bbq = new Chicken();
const bhc = new Chicken();

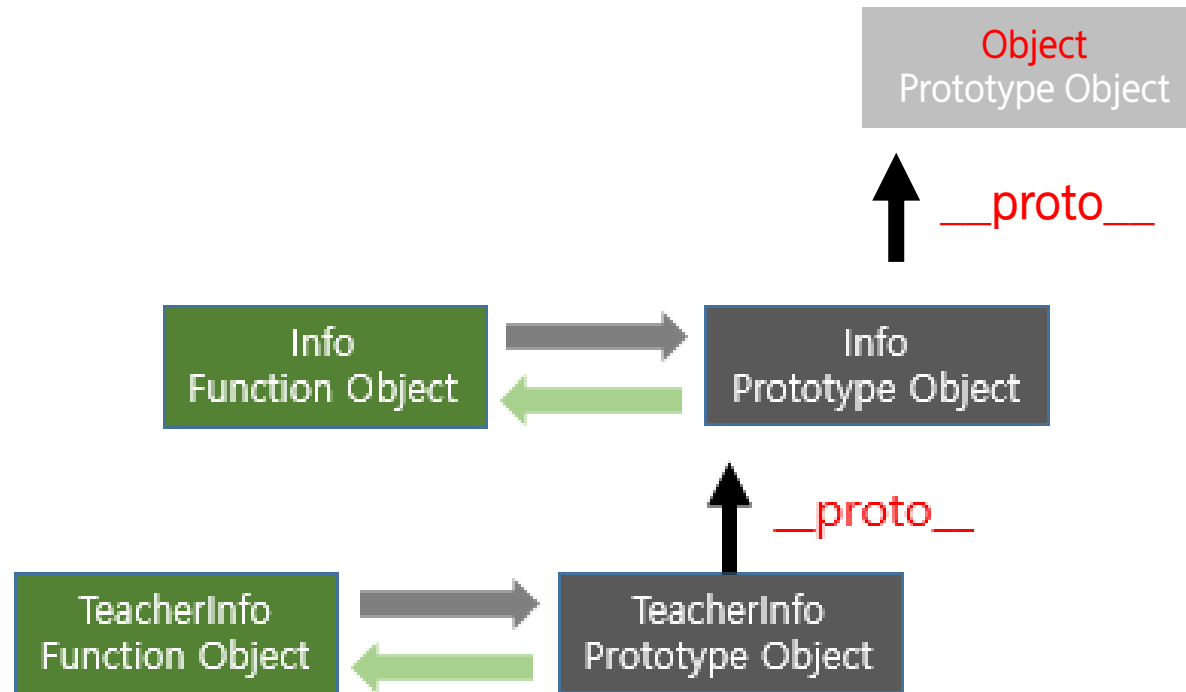
console.log(bbq.information === bhc.information);
true
```

프로토 타입을 활용한 경우



## JavaScript 모든 객체들의 조상 “Object” 객체

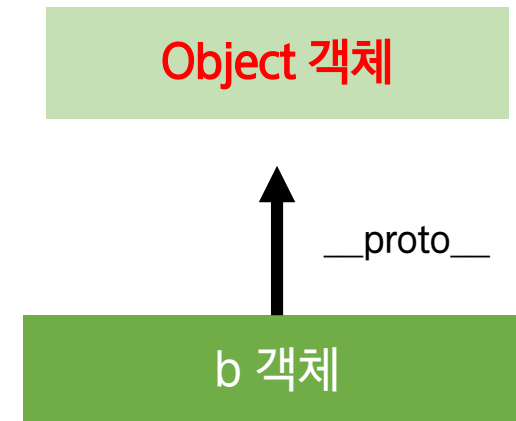
- 자바스크립트는 기본적으로 원시타입을 제외한 모든 타입은 객체
  - 함수, 객체, 배열 모두 객체
- 최상위 객체이며, 객체 생성시 기본적으로 Object를 상속한다.



“Object 객체”는 모든 객체의 부모이다.

- 객체를 만들면, Object 라는 객체가 자동으로 상속 받아진다.
- “Object 객체”의 메서드들을 사용할 수 있다.
- \_\_proto\_\_ 속성으로 부모 객체에 접근 가능

```
1  <script>
2
3    let a = { };
4
5    let b = {
6      teamName : "bts",
7      dance : function() {
8        alert("HI");
9      }
10   };
11
12   b.dance();
13
14 </script>
```



# JavaScript에서 Class

## JavaScript에서도 Class 사용 가능

- 실제로는 프로토타입 기반을 편하게 사용하기 위해 class라는 명칭이 붙은 상태

```
class Human {  
    constructor(name){  
        this.name = name;  
    }  
  
    sayMyName(){  
        console.log("이름: " + this.name);  
    }  
}  
  
const person1 = new Human("이온유");  
person1.sayMyName();  
  
const person2 = new Human("변성은");  
person2.sayMyName();  
  
const person3 = new Human("이자룡");  
person3.sayMyName();
```

## Student class 생성

- 생성자에는 이름, 나이, 자기소개가 들어갈 수 있도록 만든다.
- introduce 호출 시 이름, 나이, 자기소개 console.log로 출력

이름: 이온유

나이: **30**

자기소개: 반갑습니다.

이름: 이자룡

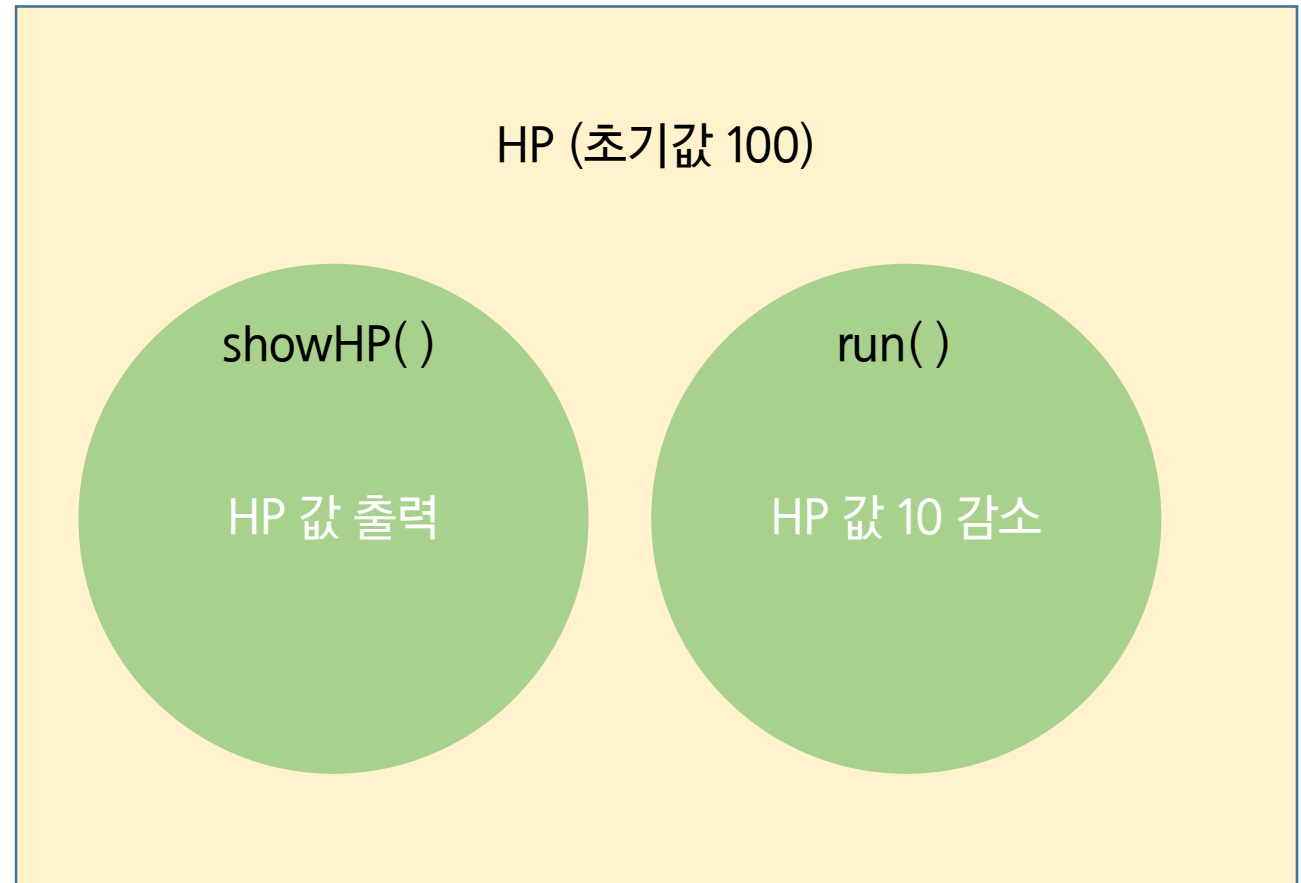
나이: **31**

자기소개: 반가워요.

## Hero class 만들기

- “batman” instance 만들기
- 다음 코드를 수행
  - `batman.showHP()`;
  - `batman.run()`;
  - `batman.run()`;
  - `batman.showHP()`;

### Hero class



## Class를 활용해 계산기 만들기

```
const calculator = new Calculator();

const result = calculator
  .add(5)
  .multiply(2)
  .subtract(3)
  .divide(2)
  .getResult();

console.log(result);
```

## extends 키워드를 이용하여 다른 클래스 상속

- 기존 클래스의 메서드들 모두 사용 가능

```
class Hero {  
    constructor() {  
        this.hp = 100;  
    }  
    run(){  
        this.hp = this.hp - 10;  
    }  
}  
  
class SuperMan extends Hero{  
    fly(){  
        console.log("fly");  
        console.log(this.hp);  
    }  
}  
  
let superman = new SuperMan();  
  
superman.run();  
superman.fly();
```



- Animal class

- name - 이름
- speak - 함수
  - name이 소리를 내며 짫습니다.

- Cat class

- Animal 상속
- speak
  - name이 야옹 소리를 냅니다.

- Dog class

- Animal 상속
- speak
  - name이 멍멍 소리를 냅니다.

## super의 활용

- super의 두가지 특징
  - 부모 클래스에 정의된 메서드에 접근이 가능
  - 부모 생성자를 호출 할 때 사용

```
class Hero {  
  constructor(hp) {  
    this.hp = hp;  
  }  
  walk(){  
    console.log("조금 빠르게 걷기")  
  }  
  
  run(){  
    this.hp = this.hp - 10;  
  }  
}
```

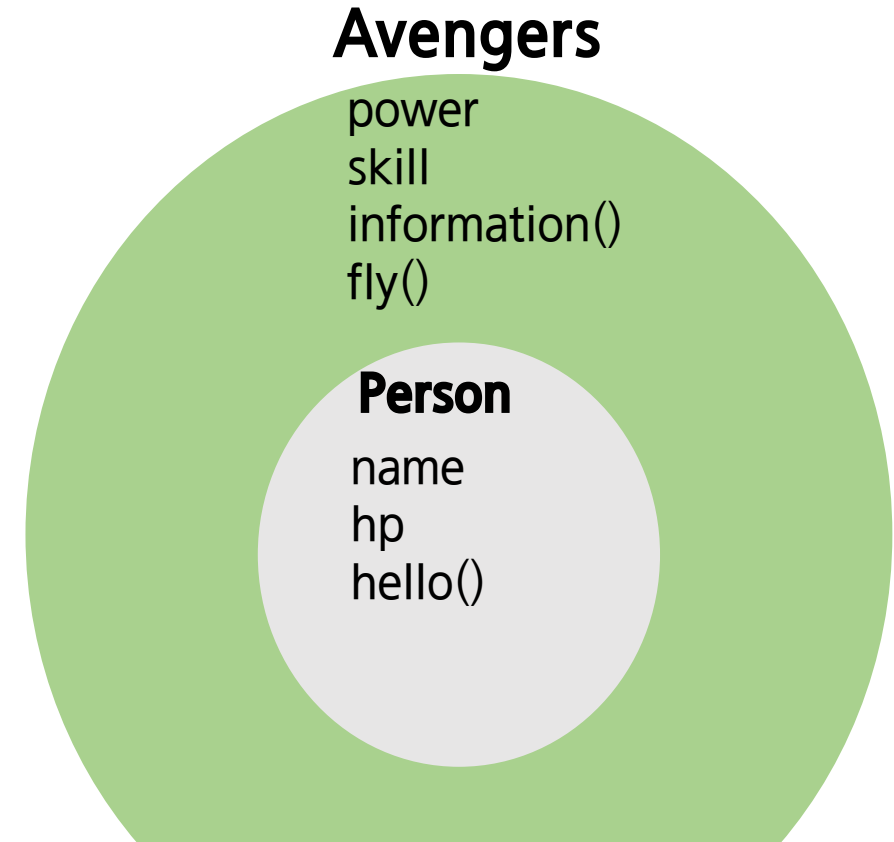
```
class SuperMan extends Hero{  
  
  constructor(hp, mp){  
    super(hp)  
    this.hp = hp;  
    this.mp = mp;  
  }  
  walk(){  
    //  
    super.walk();  
    console.log("완전 빠르게 걷기");  
  }  
  
  fly(){  
    console.log("fly");  
    console.log(this.hp);  
  }  
}  
  
let superman = new SuperMan(100, 100);  
  
superman.run();  
superman.fly();
```

## • Person Class

- name, hp
- hello()
  - hello()시 이름 출력

## • Avengers Class

- Person Class 상속
- power
- skill
- information
  - name, hp, power, skill 출력
- fly
  - 비행중 출력



```
const ironMan = new Avengers('아이언맨', 100, 10000, "미사일");  
ironMan.information();  
ironMan.fly();
```

# 내일 방송에서 만나요!

삼성청년SW·AI아카데미

과제1(12p)

classList

과제2(14p)

구글 newtab 만들기

과제3(47p)

네이버 모바일 만들기

과제4(58p)

화살표 함수 사용하기

과제5(94p)

class 사용해보기

과제6(97p)

동물 소리 구현

과제7(99p)

Avengers 만들기