SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

Javascript

<알림>

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

6장. 문자열 Parsing

챕터의 포인트

- Parsing을 위한 준비단계
- Parsing 기본 메서드 1
- Parsing 기본 메서드 2
- URL Parsing

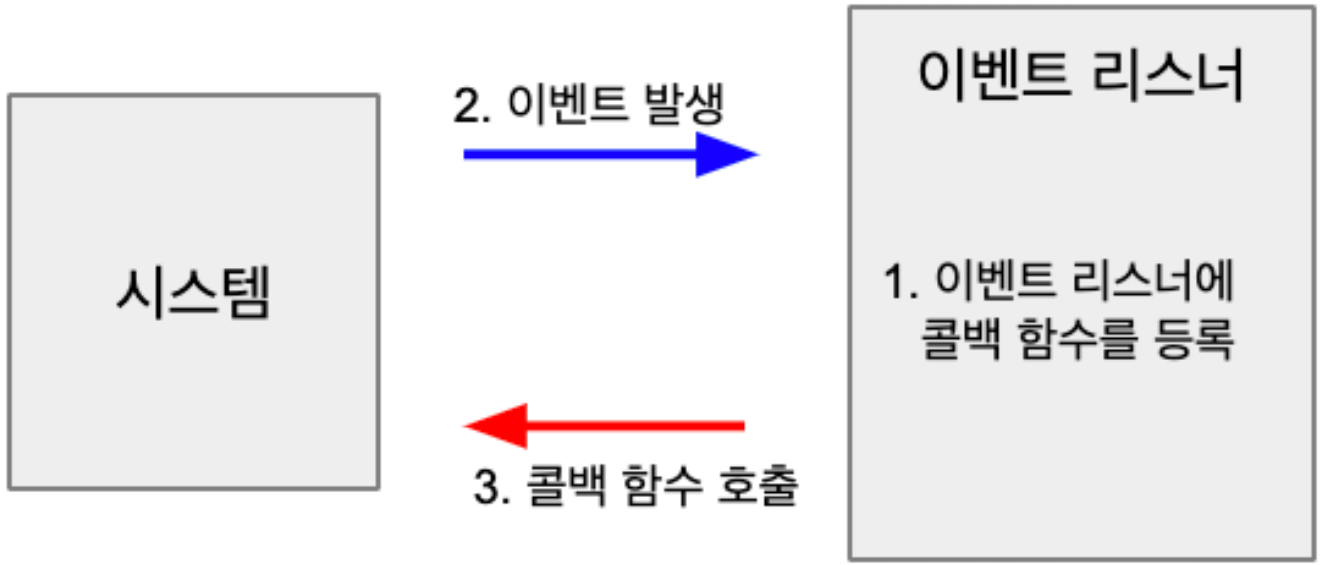
DOM으로 특정 노드를 선택하는 다양한 방법 중 Best

- QuerySelector : CSS Selector를 이용하여, 특정 DOM 영역 1개 선택하기
- QuerySelectorAll : CSS Selector를 이용하여, 해당하는 모든 DOM 영역 Array로 가져오기

이벤트 발생 동작 원리

이벤트 리스너에 함수를 등록한다.

- addEventListener 메서드를 사용하여 이벤트 핸들러 (콜백함수)를 등록한다.
- addEventListener('이벤트이름', '핸들러이름');



복습 - 버튼에 이벤트 추가하기

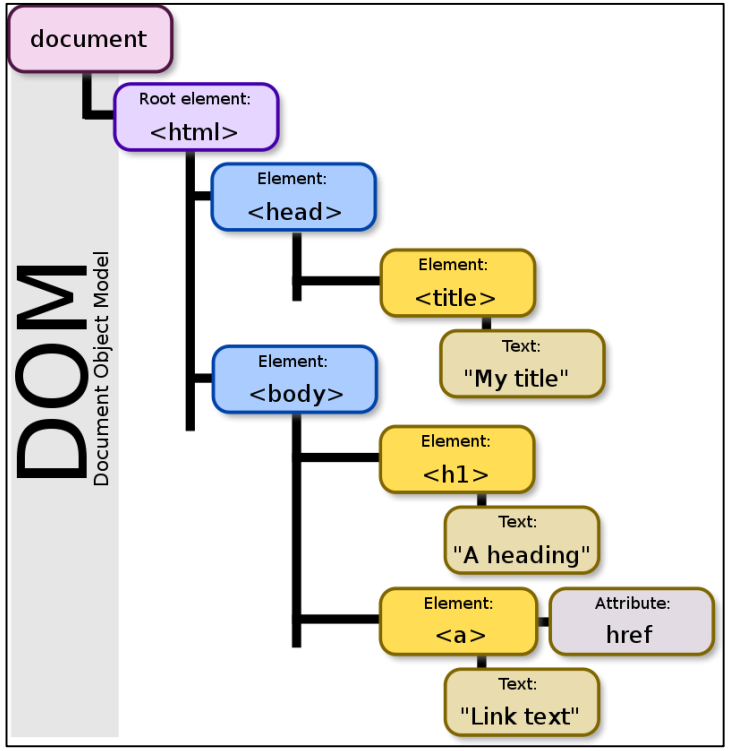
html 태그로 button 생성

- button 클릭시 alert로 안녕! 이 나오는 이벤트 추가하기

document.createElement('Tag 이름')

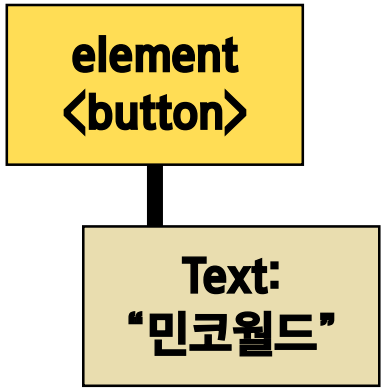
- Element Node를 생성한다.
- ex) let btn = document.createElement('button');

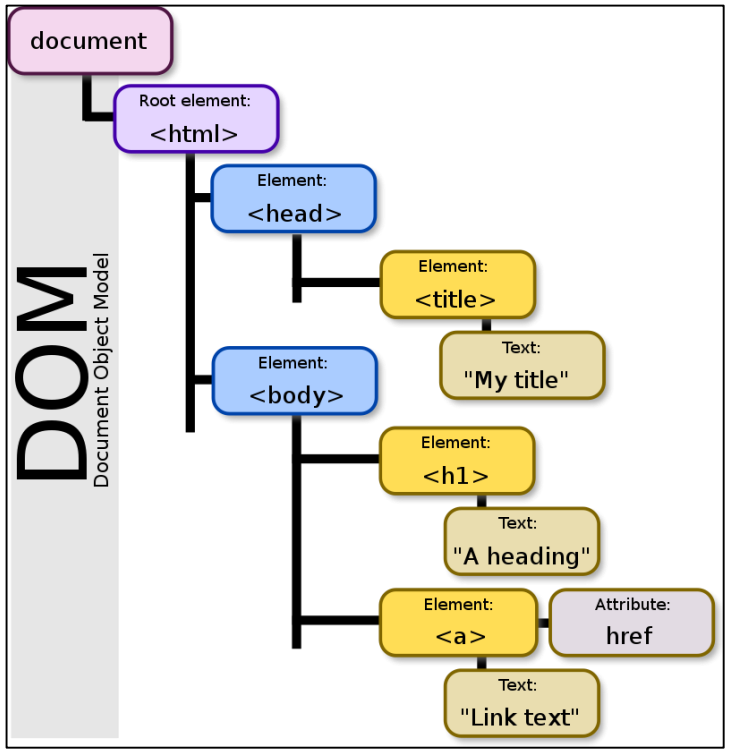
element
<button>



객체.append(Target노드)

- 객체 밑으로 자식 노드를 추가한다.
- ex) btn.append(txt);

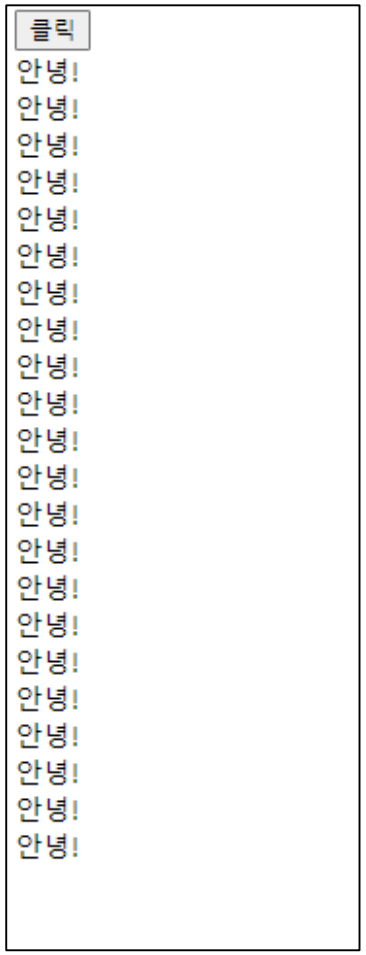




복습 - 버튼에 이벤트 추가하기

html 태그로 button 생성

- **button 클릭마다 안녕! text를 가진 div를 추가하기**



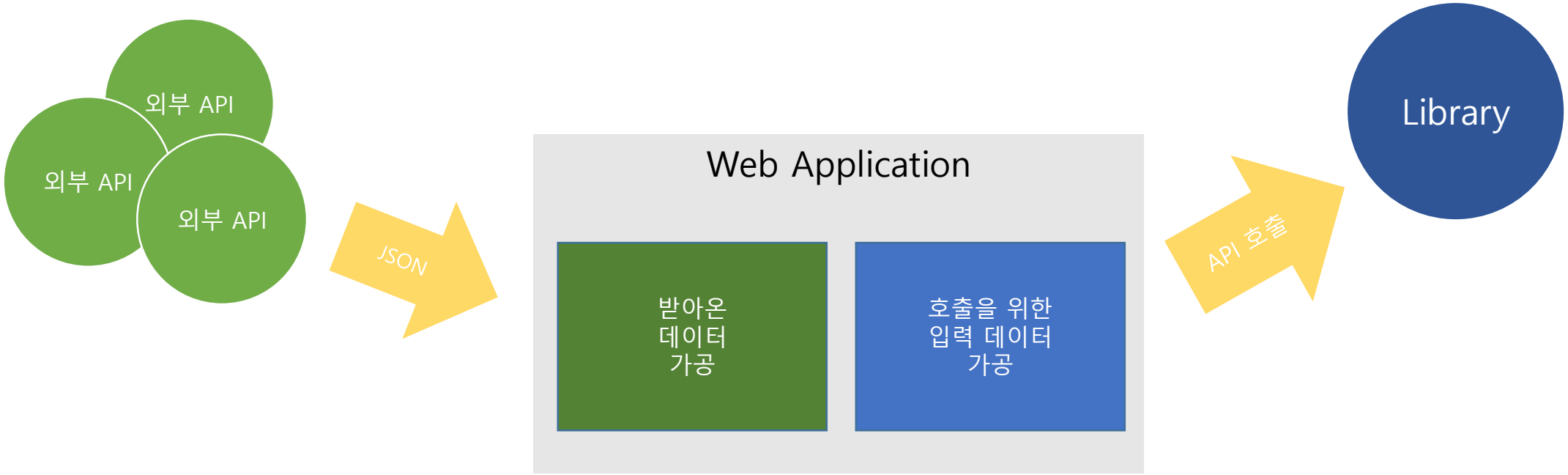
SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Parsing을 위한 준비단계

API를 통해 가져온 데이터를 가공

- 사용할 수 있도록 데이터 변경
- 가공 후 다른 API에 넣을 수 있도록 가공
- Node.js / Vue.js 수업시 활용 예정



2020년 상반기 기준 코딩테스트 Parsing 출제

- 현대카드 신입 : 4 문제 중 3 문제 Parsing
- 쿠팡 : 4 문제 중 1 문제 Parsing
- CJ 올리브 네트웍스 : 2 문제
- Toss (Server 경력직) : 1 문제

Parsing 특징

프로그래밍 언어별 Parsing 에 사용하는 메서드 및 원리가 같다.

- **indexof / substring**
- **insert / erase**
- **replace / split**

Web API 활용 및 코딩테스트 준비를 위해
JavaScript 메서드를 이용하여 Parsing을 연습하는 것을 목표

프로그래밍 언어마다 Parsing을 위한 메서드가 비슷

- **C++ / JavaScript 메서드 차이**

종류	C++	JavaScript	비고
index 검색	find	indexOf	a.indexOf("BBQ");
구간 자르기	substr	substr	a.substr(3, 2);
문자열 붙이기	+ 연산자	+ 연산자	let a = "ABC " + "BBQ"
특정 위치 문자열 추가	insert	**없음**	직접 구현 필요
특정 위치 문자열 제거	erase	**없음**	직접 구현 필요
replace	없음	replace	a.replace(/문자열/g, "MC");
split	없음	split	a.split('|');
문자열을 숫자로	stoi	Number	Number("ABC");
숫자를 문자열로	to_string	String	String("123");

[실습] 문자열 입력 받기

세 문자열을 입력 받는 방법

- 띄어쓰기를 구분기호로 입력 받기
- 배열에 하나씩 넣고 alert 출력

String 문자열 3 개 처리하기 (2분)

- 3 개 문자열이 모두 동일하다면 “Very Good” 출력 (Alert)
- 2 개 문자열이 모두 동일하다면 “Two” 출력
- 모두 다르다면 “Only One” 출력

```
2   <script>
3
4       let a = prompt("세 문자열 입력");
5
6       let b = a.split(' ');
7
8       let result = [];
9       for (let i = 0; i < b.length; i++) {
10          result.push(b[i]);
11      }
12
13      alert(result);
14
15  </script>
```

예시) 세 문자열 입력받는 예제

하드코딩 하여 실습하기

앞으로 입력처리가 아닌 하드코딩으로 진행

- 빠른 실습을 위해, 하드코딩 처리
- 실제 코딩테스트 에서도 추천하는 방식

```
2   <script>
3
4       let a = prompt("세 문자열 입력");
5
6       let b = a.split(' ');
7
8       let result = [];
9       for (let i = 0; i < b.length; i++) {
10          result.push(b[i]);
11      }
12
13      alert(result);
14
15  </script>
```



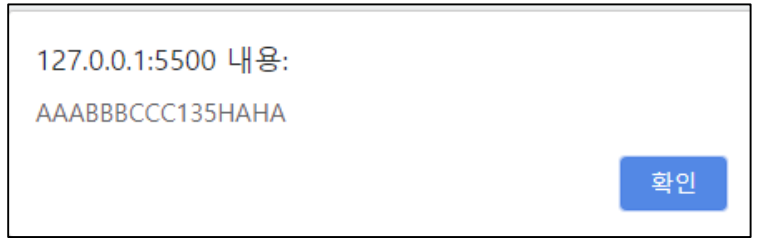
```
2   <script>
3
4       let a = "ASDA";
5       let b = "QQWE";
6       let c = "EWRWERW";
7
8   </script>
```

'+' 연산자

문자열을 붙일 때 사용함

- a.concat 메서드와 동일

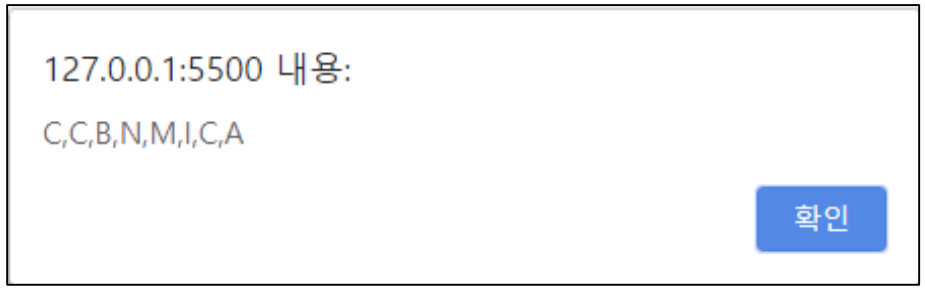
```
2   <script>
3
4       let a = "AAA";
5       let b = "BBB";
6       let c = "CCC";
7
8       let d = a + b + c + String(135) + "HAHA";
9
10      alert(d);
11
12  </script>
```

[도전] Last Index부터 출력

짝수번째 index만 출력하기 (3분)

- **문자열 하드코딩**
  - a = “ABCMINMIN”;
  - b = “BBQCOCO”;
- **두 문자열을 + 로 붙이기**
- **가장 마지막 글자부터 첫 글자까지 짝수 index 문자만 배열에 쌓기**
- **배열 출력하기**



출력결과

SAMSUNG SW ACADEMY FOR YOUTH

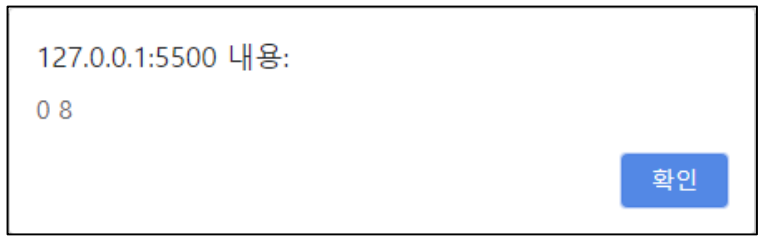
함께가요 미래로!
Enabling People

Parsing 기본 메서드 1

문자열 또는 문자의 index를 찾아주는 메서드

- indexOf(“찾을 문자열”, 검색시작 index);
- 못 찾는 경우는 -1을 리턴한다.
- 대소문자 구분 필요

```
<script>

    let a = "BTSWORLDBTS";

    let g1 = a.indexOf("BTS");
    let g2 = a.indexOf("BTS", 4);

    alert(String(g1) + " " + String(g2));

</script>
```

[도전] 'C' 출력하기

1, 2 단계 문제 indexOf를 활용하여 풀기 (5 분)

- **문자열 "KFCOCOBBQMINMINC" 하드코딩**
  - 1 단계 : 'C' 가 처음 발견되는 index 출력하기 → 정답 : 2
  - 2 단계 : 'C' 가 존재하는 모든 index 출력하기 → 정답 : 2, 4, 15

해당 방식은 Parsing에서 자주 사용되는 기법으로 반드시 마스터 필요

[도전] “ABC” 초콜릿 찾기

다음 문자열에서 ABC가 몇 개인지 Counting (5분)

- 반드시 indexOf를 활용하여 문제를 풀 것.
- 문자열 “ABCABCMCABCABCMCBBQABC”

[도전] 찾아라 드래곤볼

5 개의 문자열을 찾아, 가장 많이 등장하는 문자열 출력 (7 분)

- **문자열 “BTSMINBTSMINBTSKKOPSM” 하드코딩**
- **찾아야 하는 5 개의 문자열**
  - SM
  - MIN
  - OP
  - VO
  - TSM
- **가장 많이 등장하는 문자열 출력 (정답 : SM)**

substring, slice, splice

substring(시작 index, 끝 index)

- 특정 index 부터 끝 index 전 까지 문자열을 얻는 메서드
- a = str.substring(2, 5) ; 2 ~ 4번 index 까지 세 글자 parsing

slice(시작 index, 끝 index)

- substring 과 일반적인 상황에서 완벽히 동일하게 동작
- 그러나, 두 가지 차이점
  - 인덱스를 역전시켰을 경우 결과가 다름
  - 배열에서도 사용 가능

splice(시작 index, 끝 index)

- 배열에서만 사용 가능
- 끝 index 까지 리턴
- 원래 배열을 바꿔버림.

[도전] 일부 뽑아내기

123456789123456789 문자열 파싱 (5 분)

- “123456789123456789” 문자열을 for문과 + 기호를 이용하여 만들기
- 숫자 2 개 하드코딩 (input1 / input2)
- input1 ~ input2 까지 문자열 뽑아내서 출력하기

[도전] 특정 영역 내용 Parsing 하기

대괄호로 덮어 쌓인 하나의 영역 Parsing (5 분)

- 문자열 “ABCDEF[1599]AAQ” 하드코딩
- 괄호는 하나만 있으며, 항상 정상적인 데이터임을 가정
- 괄호 안에 있는 숫자 Parsing 후 1 더한 값 출력하기 (정답 : 1600)

[도전] 여러개의 대괄호 Parsing

대괄호로 덮어 쌓인 모든 숫자들의 합 구하기 (10 분)

- 문자열 “AB[100]T[41]ABS[1][5]BTS” 하드코딩
- 정상적인 데이터임을 가정
- 괄호 안 숫자들 모두 Parsing 후 총 합을 출력 (정답 : 147)

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

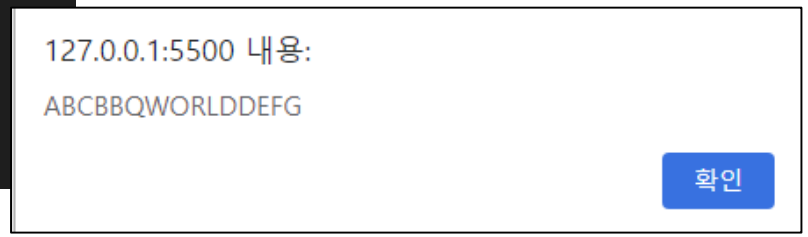
Parsing 기본 메서드 2

[실습] insert 구현하기

특정 index에 문자열을 넣어보자

- 특정 지점에 문자열을 넣는 함수 구현하기
- JavaScript에서 제공되지 않음
- insert(전체문자열, 넣을 index, 넣을 문자열);

```
<script>

    function insert(str, index, target) {
        let a = str.substring(0, index);
        let b = str.substring(index, str.length);

        return a + target + b;
    }

    let str = "ABCDEFG";

    let result = insert(str, 3, "BBQWORLD");
    alert(result);

</script>
```



[도전] erase 구현하기

erase 함수 구현 (5 분)

- 특정 index에서 n 개의 글자를 제거하는 함수 구현하기
- erase(전체 문자열, Start Index, 지울 글자수);

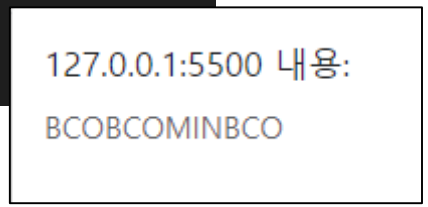
[실습] replace 메서드

특정 문자열을 모두 다른 문자열로 교체하는 메서드

- replace 자체는 특정 문자열 하나를 다른 문자열로 교체하는 메서드
- 정규식을 이용하면 특정 문자열 모두를 다른 문자열로 교체 가능

- str.replace(/문자열/g, “바꿀 문자열”);

```
1   <script>
2
3       str = "BTSBTSMINBTS";
4
5       let result = str.replace(/TS/g, "CO");
6       alert(result);
7
8   </script>
```

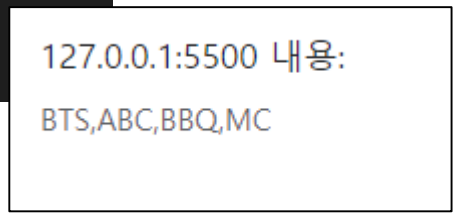


[실습] split 메서드

특정 문자 / 문자열로 구분하여 Array로 만드는 메서드

- split(“ ”) : 한 글자씩 Array로 저장
- split(“|”) : Bar를 구분 기호로 Array로 저장

```
1   <script>
2
3       str = "BTS ABC BBQ MC";
4       let a = str.split(" ");
5       alert(a);
6
7   </script>
```



[도전] replace & split

e-mail 주소를 파싱하여 다음 미션을 수행하시오.

1. 문자열 하드코딩 : bts@macdonald.co.kr || inho@mincoding.co.kr || jason@jyp.com
2. co.kr / com 을 모두 .net으로 변경하기
3. bar || 구분 기호로 분류하기
4. id만 추출하여 배열에 넣기
5. 배열 출력하기

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

URL Parsing

HTTP 프로토콜의 대표적인 Request용 Method

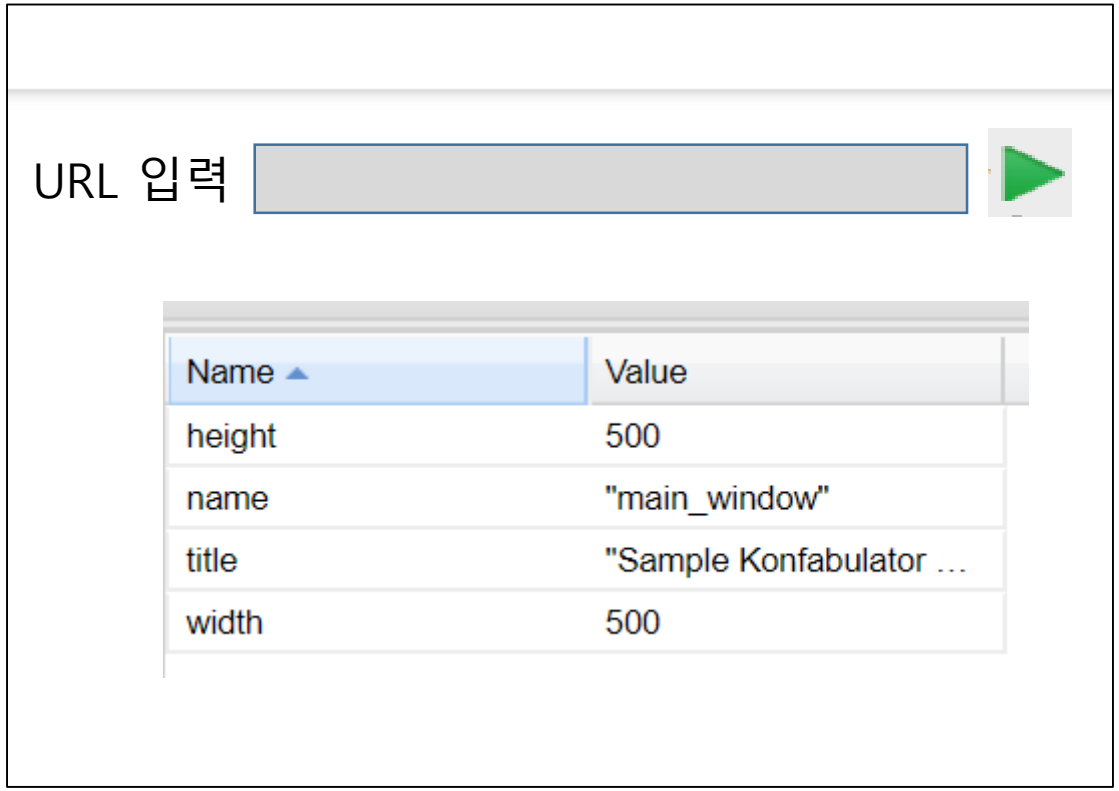
- 서버에서 문서를 가져올때 사용하는 명령어
- 특정 문서에 접속하기 위해 URL을 적으면 GET Method를 사용하는 것
- Key, Value 값을 URL에 **포함하는지 상관없이**, URL로 인터넷 접속시 GET Method를 사용하는 것

- Get Method 예시
  - GET /bbq/page.html
  - GET /bbq/page.html?id=1004&page=152&num=119

[도전] URL Parsing 사이트 제작

URL을 입력하면 Key와 Value를 구분해주는 웹사이트 제작

- **화면 구성하기 (Bootstrap)**
- **Run 버튼 누르면 특정 메서드 호출**
- **Parsing 후 결과 출력**
  - ex) https://www.google.com/search?q=치킨
    - ? 뒷부분을 가져오기



SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Arrow Function

복습. 함수는 객체이다.

함수를 정의하는 방법

- **함수 이름이 곧, 객체 이름이다.**

함수 객체 이름 : a

함수 객체 이름 : b

```
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 : Arrow Function

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

화살표 함수는 항상 무명메서드 이다.

```
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>
```

Arrow Function 은 항상 무명이다.

- **값을 전달하고, 리턴 가능**

```
<script>

    let a = (a, b) => {
        alert(a + b);
    }

    a(3, 4);

</script>
```

a, b, 받고 합 출력

```
<script>

    let a = (a, b) => {
        return a + b;
    }

    let sum = a(3, 4);
    alert(sum);

</script>
```

a, b 받고 합 return 하기

[도전] 화살표 함수 사용하기

3을 전달 하여 제곱을 리턴하는 화살표 함수 만들기

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

화살표 함수의 더 간략한 표현

- Argument가 1 개 일때, 괄호 생략 가능
- 다른 코드 없이 Return만 하는 화살표 함수는 { } 생략 가능
- 한 줄만 수행하는 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

[도전] 화살표 함수 코드 이해

다음 소스코드를 보고 실행 결과를 예측하시오

```
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

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

7장. Object와 Class

챕터의 포인트

- JavaScript의 Object
- JavaScript에서 Class 사용
- 피자 프로젝트
- Prototype의 이해

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

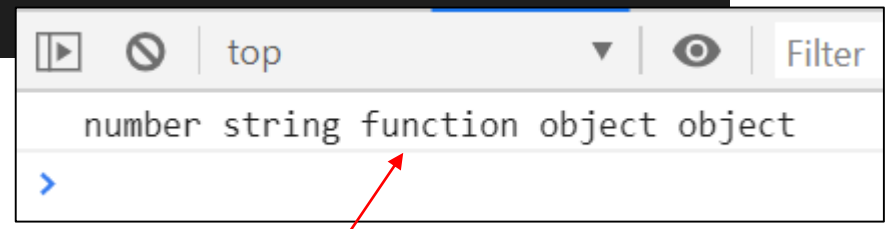
JavaScript의 Object

변수는 총 여덟가지 타입이 존재

- **원시 타입(7 개)**
  - number
  - string
  - Boolean
  - null
  - undefined
- **객체 타입 (1 개)**
  - 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>
```



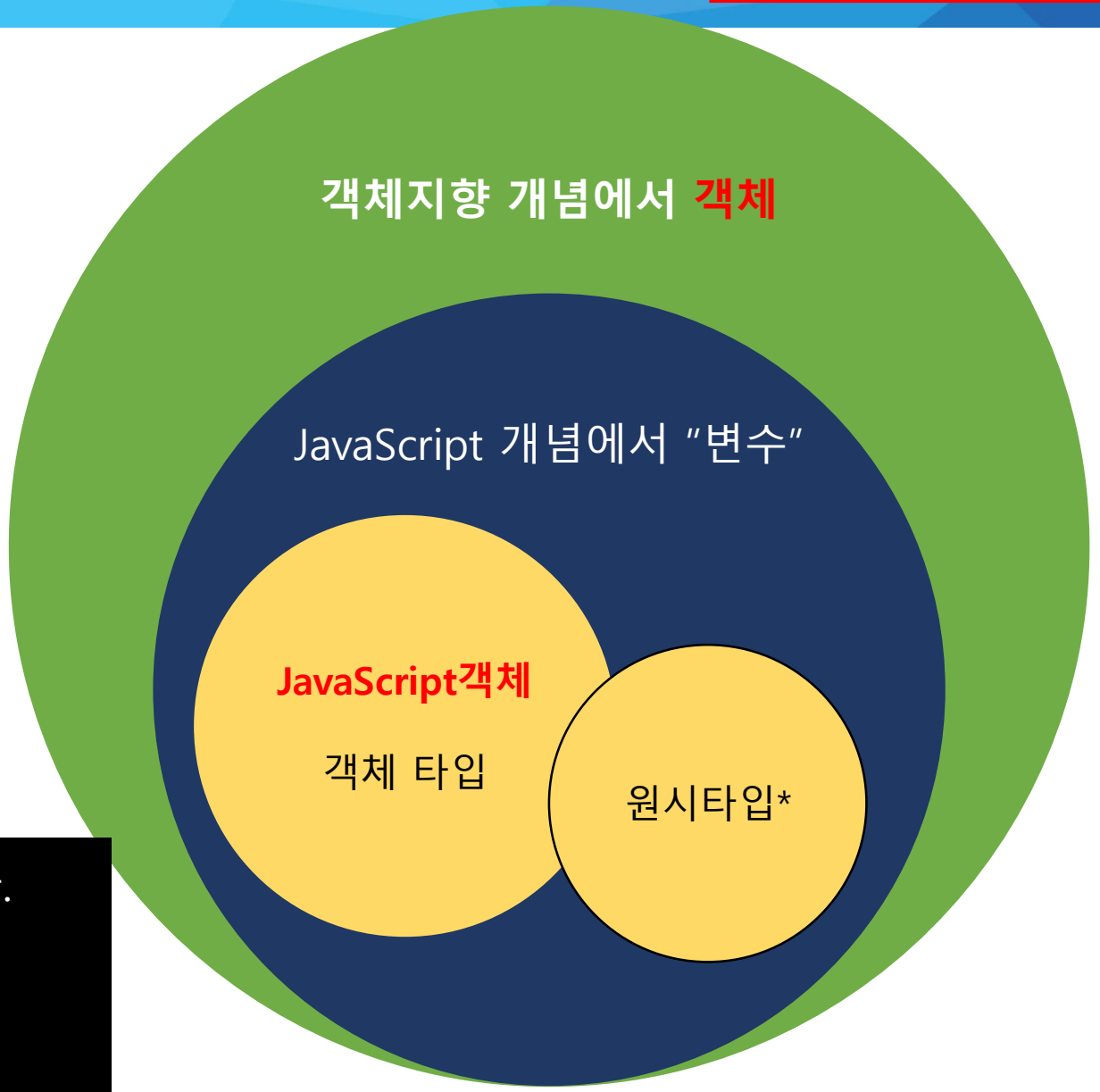
실제로 함수 Type은 존재하지 않음
호환성을 위해 function으로 표기

Object의 범위

객체는 다음과 같은 의미를 갖는다.

- **OOP 개념에서 모든 변수는 객체이다.**
  (내장 속성 or 메서드를 가짐)

- **JavaScript 개념에서 Object Type으로**
  만들어진 변수는 객체 이다.



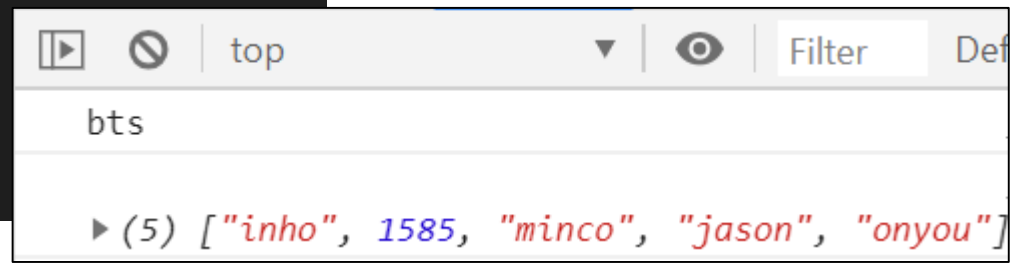
따라서 JavaScript 객체는 다음과 같은 것을 의미한다.
0. 키와 값으로 이루어진 프로퍼티의 모음
1. Array
2. Function
3. new + 모든 Types

Object Type으로 만든 변수 = "객체"

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

```
<script>

  let a = { };

  let b = {
    teamName : "bts",
    teamCount : 7,
    member : ["inho", 1585, "minco", "jason", "onyou"],
    other : {
      averAge : 20,
      averHeight : 200
    }
  };

  console.log(b.teamName);
  console.log(b.member);

</script>
```

JSON (JavaSciprt 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.

객체 정의

- 객체는 0개이상의 프로퍼티 & 메서드로 구성된 집합이다.

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

Object 에 함수 추가하기

Object(객체)의 속성으로 함수도 추가 가능

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

```
<script>

  let a = { };

  let b = {
    teamName : "bts",
    dance : function() {
      alert("HI");
    }
  };

  b.dance();

</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

```
const ssafy = new Object();

ssafy.name = "embedded"
ssafy.study = function(){
    console.log("study network programming")
}
console.log(ssafy)
ssafy.study()
```

절차지향 프로그래밍

- **코드를 위부터 아래로 내려가며 실행하는 방식**
- **같은 코드를 사용할 경우 코드를 카피해서 넣거나, 새로 코딩을 했다.**
  - 코드의 양이 길어짐
  - 같은 중복의 코드가 너무나도 많아짐
  - 유지보수가 힘들어짐

객체지향 프로그래밍이란?

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.키값의 형식으로 접근

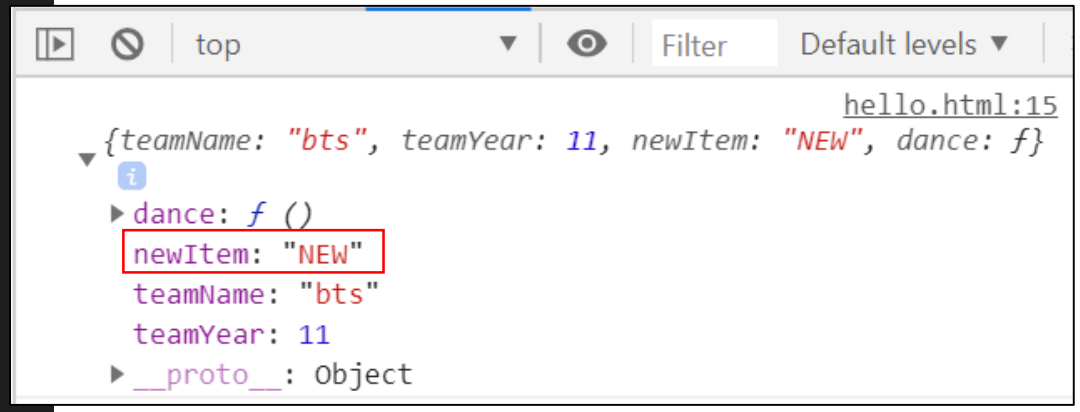
```
const ssafy = {
    weight: 90,
    eat(){
        return this.weight + 5;
    },
    run(){
        return this.weight -3;
    }
}

console.log(ssafy);
console.log(ssafy['eat']())
console.log(ssafy['run']())
```

객체['키값']의 형식으로 접근

객체의 속성 추가하기

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
13    b.newItem = "NEW";
14
15    console.log(b);
16
17  </script>
```

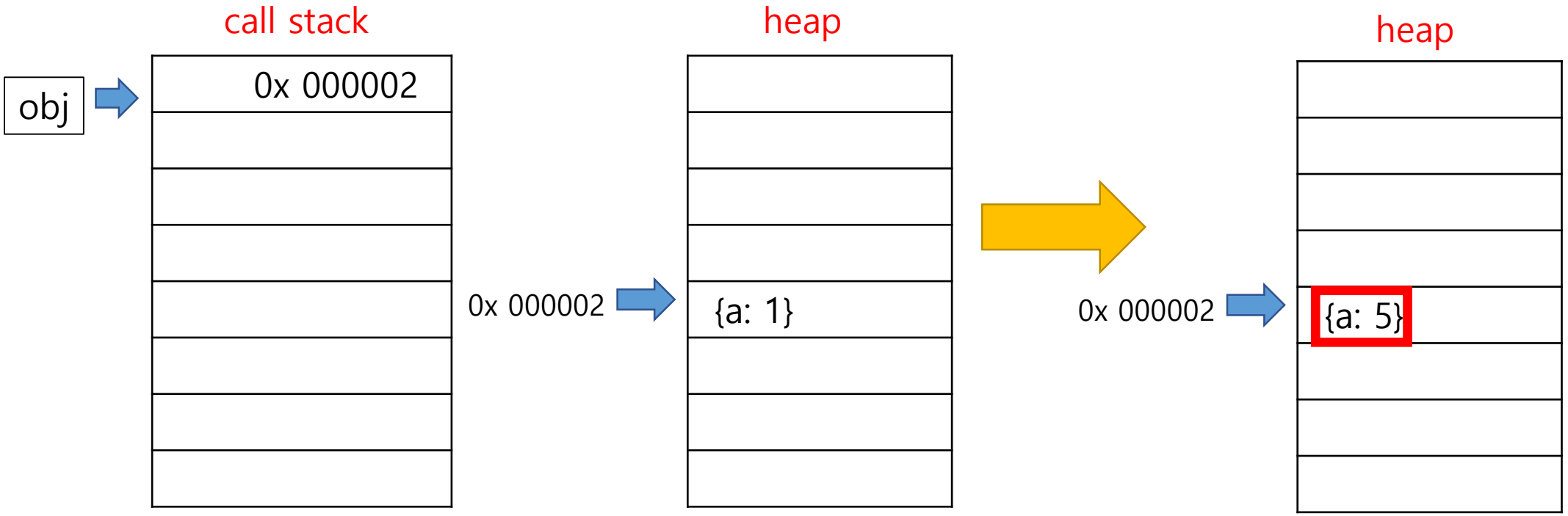


참조 자료형에서 값 바꾸기

const 로 선언된 참조 자료형에서 값 바꾸기

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

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



SAMSUNG SW ACADEMY FOR YOUTH

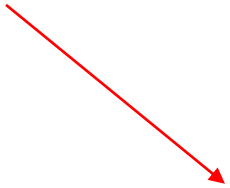
함께가요 미래로!
Enabling People

Prototype의 이해

함수를 만드는 두 가지 방법

- **함수 이름은 변수 (객체)로 취급된다.**

myFunction 은 함수 이름이자
객체 이름이다.



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

myFunction();
```

Function Type 변수 생성 후
Function 호출

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

myFunction();
```

Function 생성 후 호출

함수에 속성 추가하기

함수는 객체 이므로, 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", "HI" 가 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 언어으로, 과거 class가 없었으나 ES6부터 Prototype 문법을 이용하여 class 문법이 추가 되었다.**

- **JavaScript의 핵심 원리 중 하나인 “프로토타입” 을 이해하는 챕터**

프토토타입을 사용하는 이유

프토토타입을 활용해서 중복을 최소화

- **같은 메모리의 객체를 사용할수 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
```

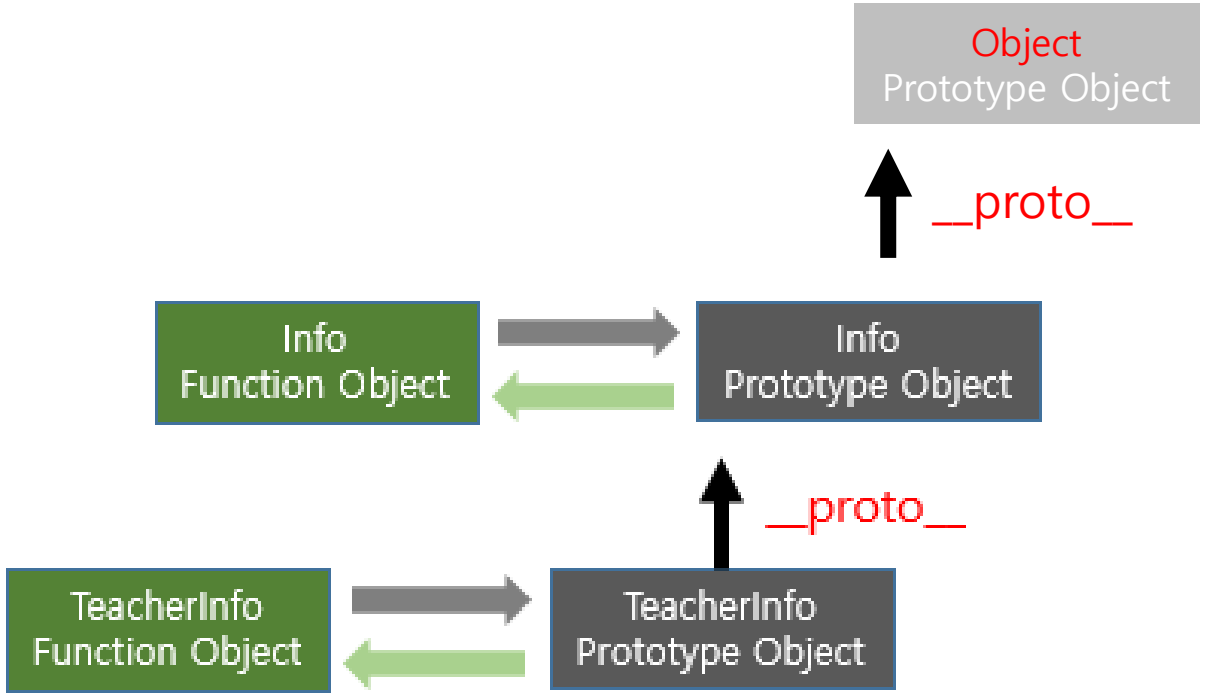
일반 함수에 객체를 선언한 경우

```
function Chicken(){

}

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


const bbq = new Chicken();

const bhc = new Chicken();

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

프토토 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.**

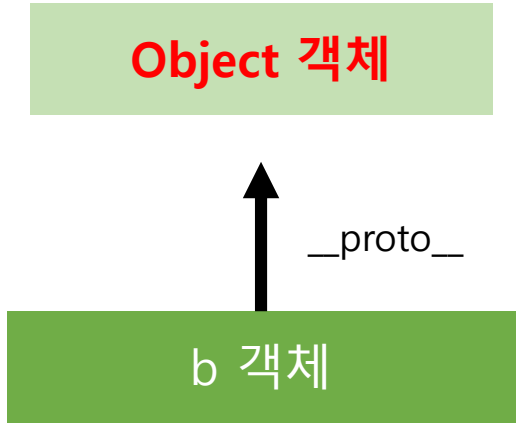


모든 객체들의 부모 객체 "Object 객체"

"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

```
<script>

  let a = { };

  let b = {
    teamName : "bts",
    dance : function() {
      alert("HI");
    }
  };

  b.dance();

</script>
```



SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

JavaScript에서 Class

class 기본 사용 예제

JavaScript에서도 Class 사용 가능

- **OOP 프로젝트에 사용**

```
class Human {

    constructor(name){
        this.name = name;

    }

    sayMyName(){
        console.log("이름: " + this.name);

    }

}

const person1 = new Human("이온유");

person1.sayMyName();

const person2 = new Human("허범성");

person2.sayMyName();


const person3 = new Human("이자룡");

person3.sayMyName();
```

[도전] class 활용 - 자기소개하기

Lawyer class 생성

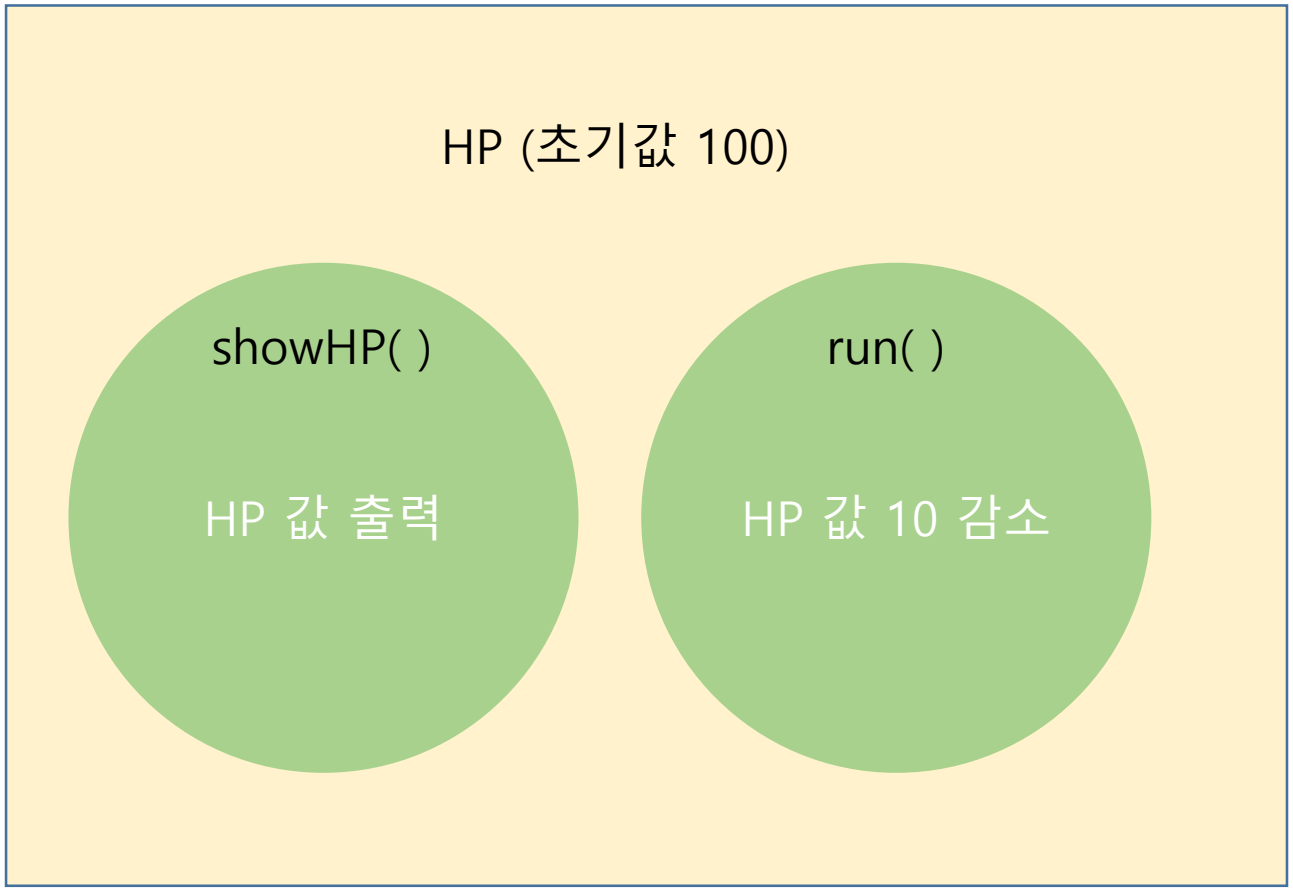
- 생성자에는 이름, 나이 , 자기소개가 들어갈수 있도록 만든다.
- intoduce 호출시 이름, 나이, 자기소개 console.log로 출력

```
이름: 우영우
나이: 27
자기소개: 기러기 토마토 스위스 역삼역
이름: 최수연
나이: 27
자기소개: 봄날의 햇살
```

[도전] Class 사용해보기

Hero class 만들기

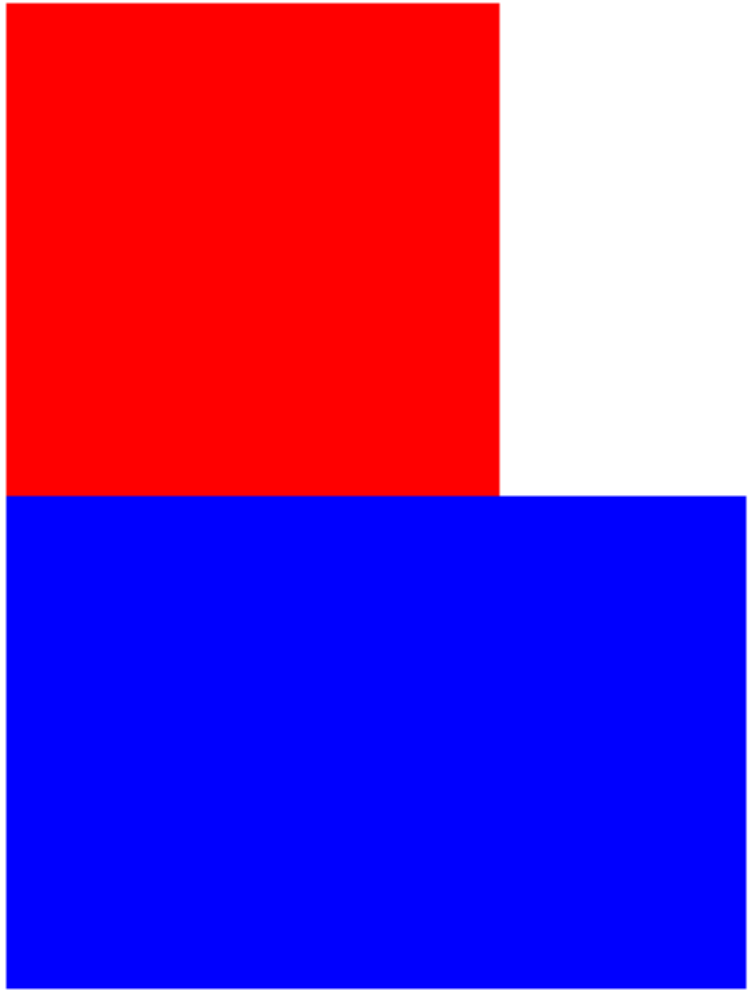
- “batman” instance 만들기
- 다음 코드를 수행
  - batman.showHP( );
  - batman.run( );
  - batman.run( );
  - batman.showHP( );



[실습] class 를 응용한 element 생성하기

요구사항

- 클릭시 해당 element 크기가 늘어나도록 class 생성하기
- 이를 통해 this 이슈가 발생할때 대처법 알아보기



this로 객체 접근할때 유의사항

this 를통해 class 객체 접근

- 접근이 안되는 이슈가 발생

```
clickEvent() {
    console.log(this);

    this.element.addEventListener('click', function (e) {
        // e와 this는 모두 지금 이벤트 함수를 가리킨다.
        console.log(e);
        console.log(this);
        this.element.style.width = "300px";
    })
}
```

```
▶ PointerEvent {isTrusted: true, pointerId: 1, width: 1, height: 1, pressure: 0, …}
  <div style="background-color: red; width: 200px; height: 200px;"></div>
▶ Uncaught TypeError: Cannot read properties of undefined (reading 'style')
    at HTMLDivElement.<anonymous> (026.this 유의사항.html:45:34)
```

this로 객체 접근할때 해결 방법 1

해결방법1

- this로 원하는곳에 접근이 안되는 경우 상단에 this를 다른 변수로 선언한다.

```
clickEvent() {
    console.log(this);
    const THIS = this;
    this.element.addEventListener('click', function (e) {
        // e와 this는 모두 지금 이벤트 함수를 가리킨다.
        console.log(e);
        console.log(this);
        console.log(THIS)
        THIS.element.style.width = "300px";
    })
}
```

```
▶ PointerEvent {isTrusted: true, pointerId: 1, width: 1, height: 1, pressure: 0, …}
  <div style="background-color: blue; width: 300px; height: 200px;"></div>
▶ Element {backgroundColor: 'blue', width: '200px', height: '200px', element: div}
```

this로 객체 접근할때 해결 방법 2

해결방법2

- 화살표함수는 this가 본인객체가 아닌 본인보다 상위 객체를 가리킨다.

```
clickEvent(){
    this.element.addEventListener('click', (e) => {
        // e와 this는 모두 지금 이벤트 함수를 가리킨다.
        console.log(e);
        console.log(this);

        this.element.style.width = "300px";
    })
}
```

```
▶ PointerEvent {isTrusted: true, pointerId: 1, width: 1, height: 1, pressure: 0, …}
▶ Element {backgroundColor: 'red', width: '200px', height: '200px', element: div}
```

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();
```

class 상속 - constuctor에 넣을 값이 있는 경우

super의 활용

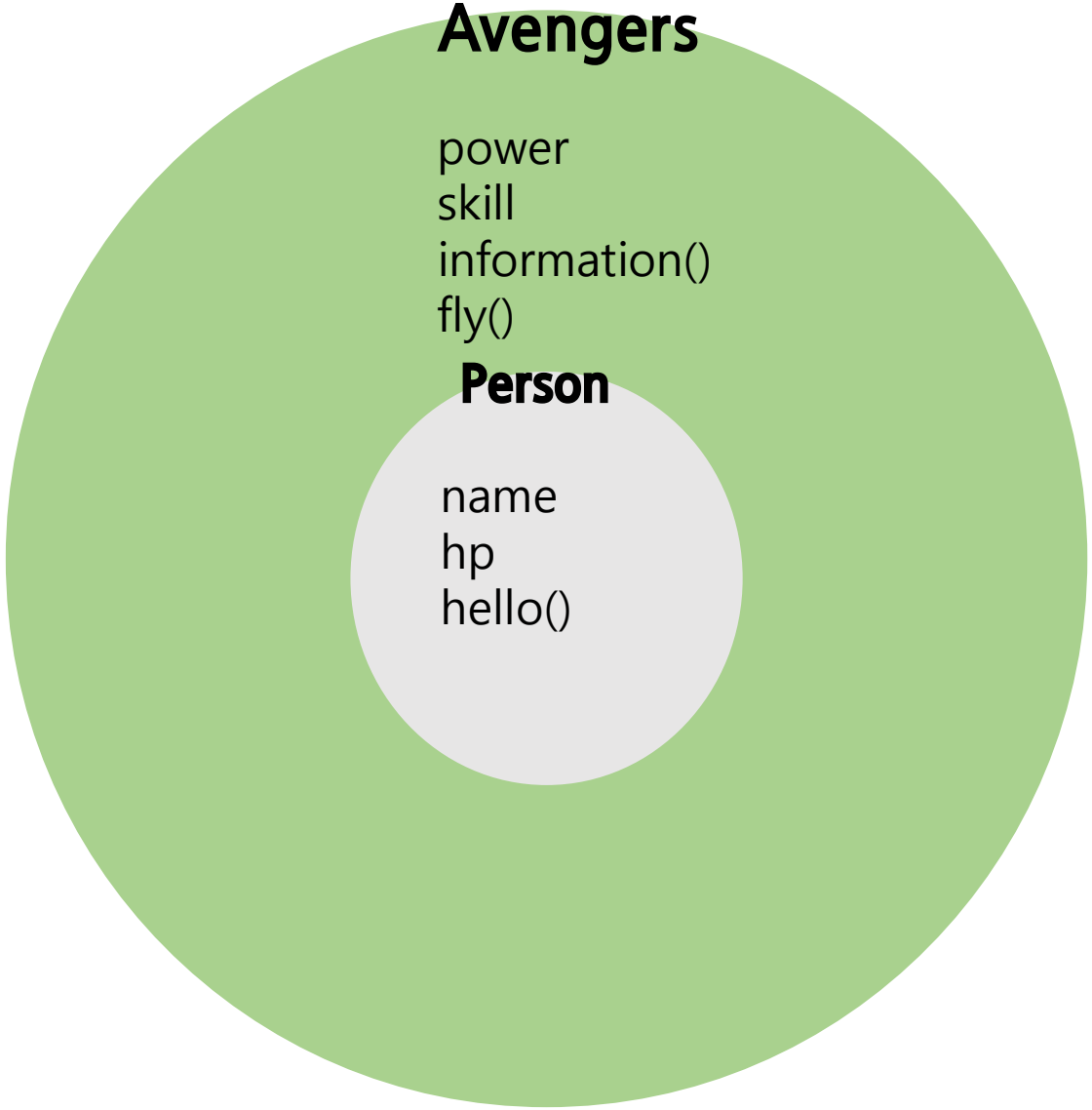
- **super의 두가지 특징**
  - 부모 클래스에 정의된 메서드에 접근이 가능
  - 부모 생성자를 호출 할때 사용

```
class Hero {
    constructor(hp) {
        this.hp = hp;
    }
    walk(){
        console.log("조금 빠르기 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();
```

[도전] Avengers 만들기

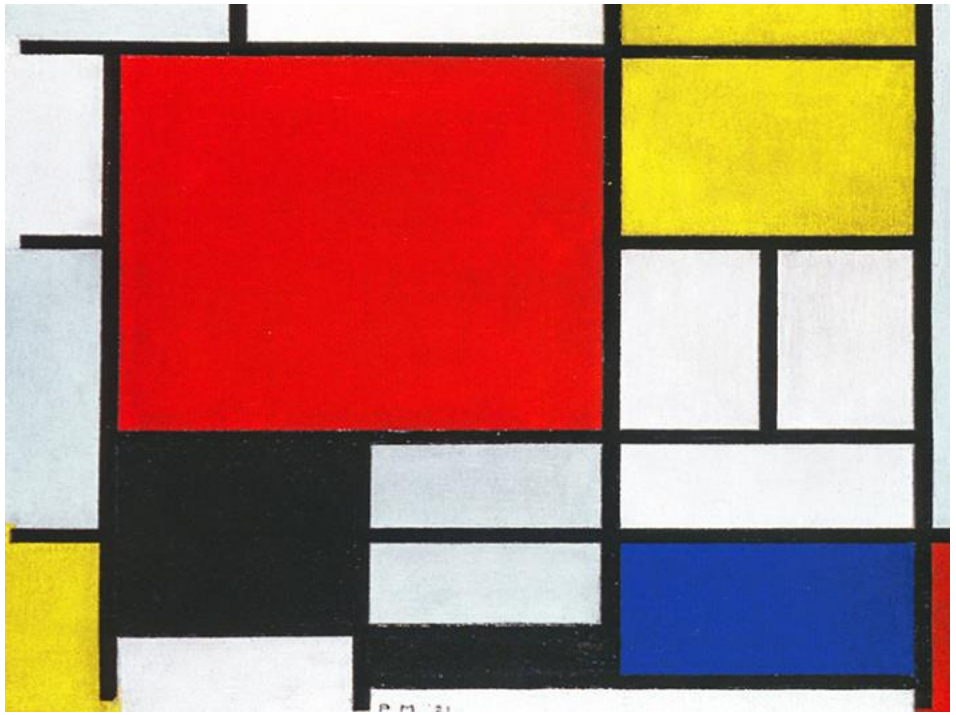
- Person Class
  - **name, hp**
  - **hello()**
    - hello()시 이름 출력
- Avengers Class
  - **Person Class**
  - **power**
  - **skill**
  - **information**
    - name, hp, power, skill 출력
  - **fly**
    - 비행중 출력

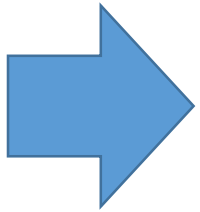


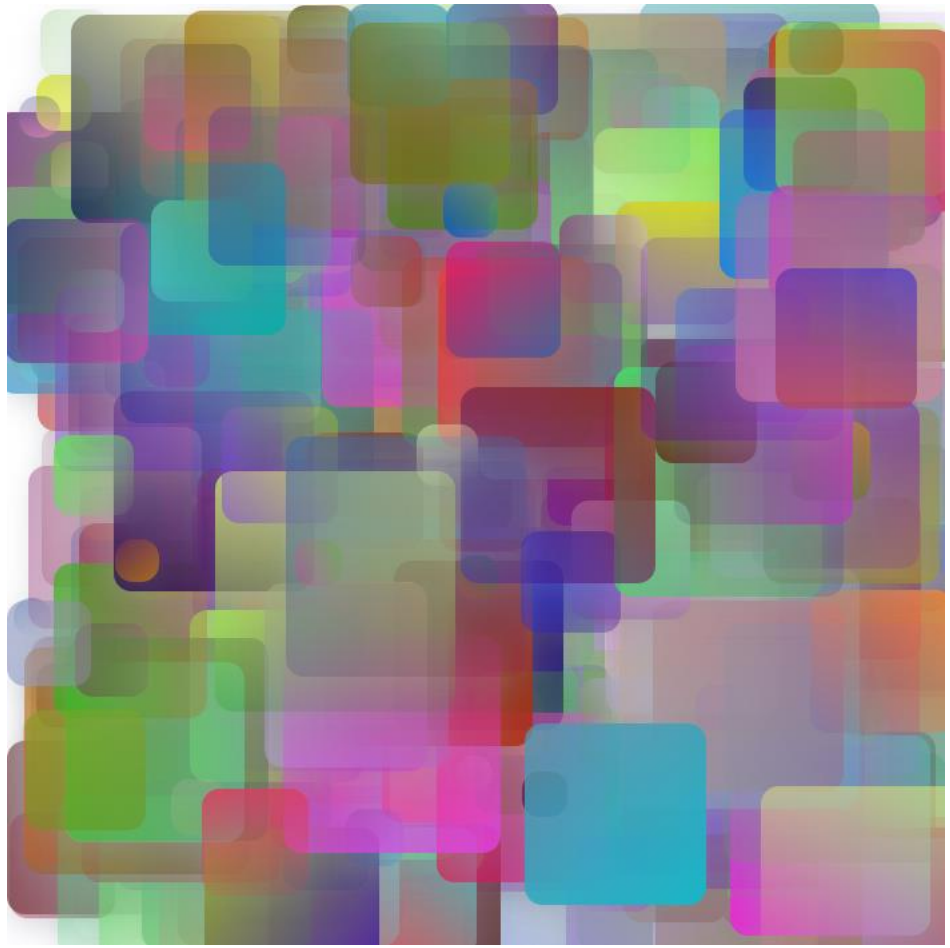
[실습] 예술 작품 만들기

Class 및 상속을 활용

- 나만의 예술 작품을 만들어보자







SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성 청년 SW 아카데미