

Class

Class란?

비슷한 부류의 함수를 한군데 모아둔 단위

```
class Calculator {  
    constructor() {  
        // 생성자  
    }  
  
    sum(a, b) {  
        return a + b;  
    }  
  
    subtract(a, b) {  
        return a - b;  
    }  
}
```

Class 사용 이유

비슷한 함수를 한군데서 관리하여 높은 모듈화 제공
ex) 계산기 class에 더하기, 빼기 함수 구현

```
class Calculator {  
    constructor() {  
        // 생성자  
    }  
  
    sum(a, b) {  
        return a + b;  
    }  
  
    subtract(a, b) {  
        return a - b;  
    }  
}
```

Class 사용 이유

코드의 구조화

- Class내의 함수끼리만 공유할 수 있는 변수 지정 가능
- 여러개의 데이터와 함수가 하나의 논리적 단위로 묶음
- 유지보수 하기 좋음

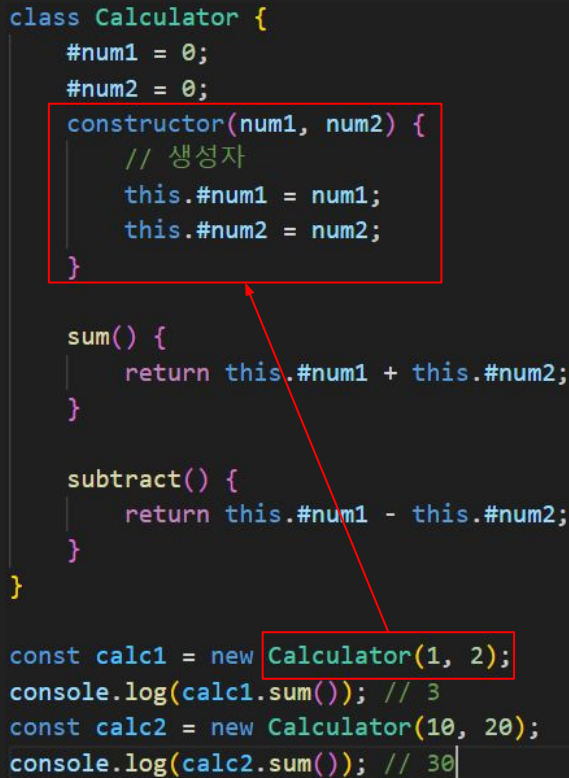
```
class Calculator {  
  #num1 = 0;  
  #num2 = 0;  
  constructor(num1, num2) {  
    // 생성자  
    this.#num1 = num1;  
    this.#num2 = num2;  
  }  
  
  sum() {  
    return this.#num1 + this.#num2;  
  }  
  
  subtract() {  
    return this.#num1 - this.#num2;  
  }  
}  
  
const calc1 = new Calculator(1, 2);  
console.log(calc1.sum()); // 3  
const calc2 = new Calculator(10, 20);  
console.log(calc2.sum()); // 30
```

Class 기본 문법

Class Constructor(생성자)

class가 만들어질 때(new 클래스이름) 딱 한번만 실행됨
보통 Class내의 변수의 초기화에 사용

```
class Calculator {  
  #num1 = 0;  
  #num2 = 0;  
  constructor(num1, num2) {  
    // 생성자  
    this.#num1 = num1;  
    this.#num2 = num2;  
  }  
  
  sum() {  
    return this.#num1 + this.#num2;  
  }  
  
  subtract() {  
    return this.#num1 - this.#num2;  
  }  
}  
  
const calc1 = new Calculator(1, 2);  
console.log(calc1.sum()); // 3  
const calc2 = new Calculator(10, 20);  
console.log(calc2.sum()); // 30
```

A red box highlights the constructor function within the Calculator class. Another red box highlights the 'new' keyword in the line 'const calc1 = new Calculator(1, 2);'. A red arrow points from the 'new' keyword to the constructor function, indicating that the constructor is called when a new instance is created.

Class Public

Class 내부, 외부 모두 호출가능한 함수, 변수

- 단, 내부에서 호출시 `this` 키워드 사용

```
class Calculator {  
    #num1 = 0;  
    #num2 = 0;  
    constructor(num1, num2) {  
        // 생성자  
        this.#num1 = num1;  
        this.#num2 = num2;  
    }  
  
    sum() {  
        // public : Class 안, 밖에서 호출 가능  
        return this.#num1 + this.#num2;  
    }  
  
    sumPrint() {  
        console.log(this.sum()); // 내부 호출  
    }  
}  
  
const calc1 = new Calculator(1, 2);  
console.log(calc1.sum()); // 외부 호출
```

Class Private

Class 내부에서만 호출가능한 함수, 변수

-함수, 변수 이름 앞에 '#' 을 붙이면
private로 선언됨

```
1  class Calculator {
2      #num1 = 0;
3      #num2 = 0;
4      constructor(num1, num2) {
5          // 생성자
6          this.#num1 = num1;
7          this.#num2 = num2;
8      }
9
10     #sum() {
11         // public : Class 안, 밖에서 호출 가능
12         return this.#num1 + this.#num2;
13     }
14
15     sumPrint() {
16         console.log(this.sum()); // 내부 호출
17     }
18 }
19
20 const calc1 = new Calculator(1, 2);
21 console.log(calc1.sum()); // 외부 호출 -> 에러발생
22
23
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

C:\Program Files\nodejs\node.exe .\Array\CustomArray.js

Process exited with code 1

Uncaught TypeError TypeError: calc1.sum is not a function

at <anonymous> (c:\Users\하승우\Desktop\test\Array\CustomArray.js:21:19)

Class의 그 외 기능

상속, 캡슐화, 다형성, **Static** 등등 **Class**를 사용하는 많은 이유가 있으나

현재 **React**는 함수형 컴포넌트를 사용을 권장하고 있으므로

이 이상의 **Class** 개념은 **AngularJs** 혹은 백엔드의 **NestJs**를 사용하게 될 때 다시 공부해 보자.

일단 여기까지~!