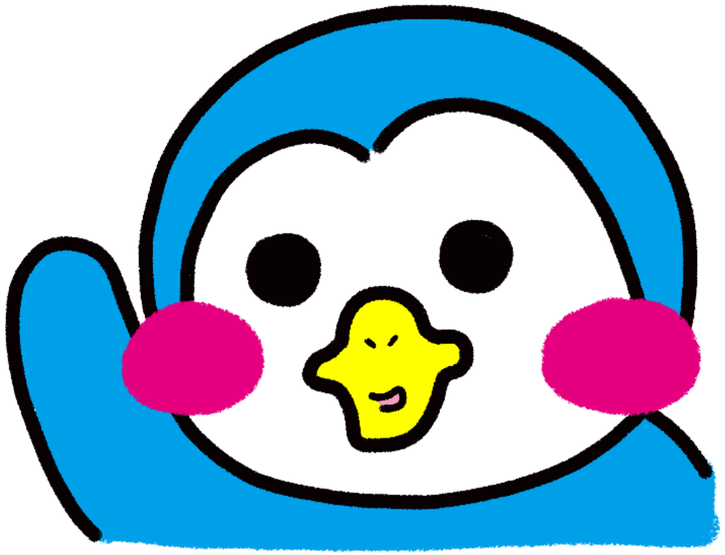
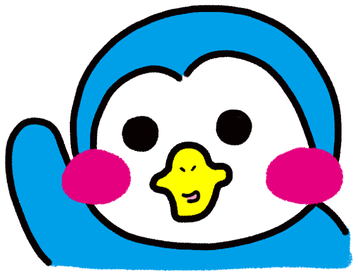
객체를 만드는 새로운(?) 방법



클래스

클래스

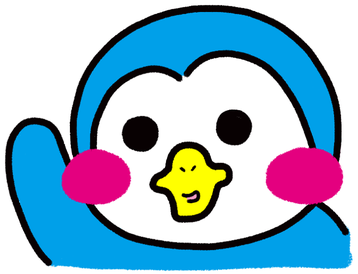


클래스란 오브젝트를 생성하기 위한 설계도이다. 기존 자바스크립트의 프로토타입을 대체할 수 있으며, 클래스를 통해 정의한 오브젝트는 Array, Date 등 내장 객체와 동일하게 취급할 수 있다.

```
1 | class 클래스명{
2 |     멤버변수;
3 |     멤버함수(메소드){}
4 | }
```

=> 클래스 선언문 기본형

클래스와 메소드

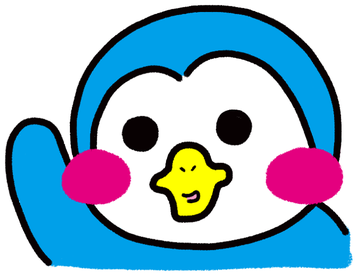


클래스 내부에 정의된 함수는, 객체의 멤버함수와 마찬가지로 메소드라 부른다. new 연산자에 의해 실제 사용 가능한 객체를 생성하며, 생성된 객체를 통해 메소드를 호출할 수 있다.

```
1  class Member{
2    getName(){
3      return "이름";
4    }
5  }
6
7  let memberObj = new Member();
8  console.log(memberObj.getName());
```

=> 메소드 정의 시에는 function 키워드를 생략할 수 있다!

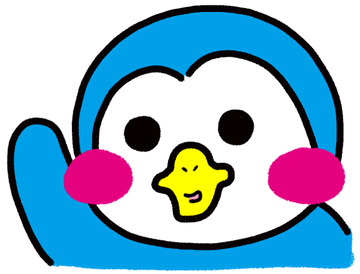
클래스 선언 시 참고사항



클래스를 선언할 때는, 이름의 첫 글자를 대문자로 작성하자.
이는 문법적으로 필수 사항은 아니지만, 개발 시 편의를 고려한 관례이다.

=> 프레임워크에 따라서 이는 필수 사항이 되기도 한다!

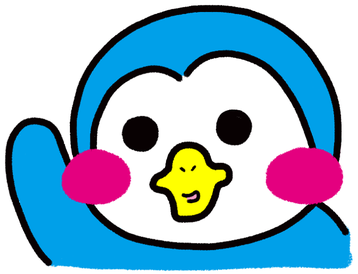
this



웹문서 조작을 위해 작성된 자바스크립트에서, 전역 키워드 this는 window 객체를 의미한다. 이벤트 발생 시에 this는 이벤트가 발생한 DOM을 의미한다.
그러나 메소드 내에서 this는 클래스 오브젝트를 의미한다.

```
1  console.log(this)
2
3  document.getElementById("p").onclick = function(){
4    console.log(this)
5  }
6
7  class Member{
8    getName(){
9      console.log(this)
10     return "이름";
11   }
12 }
13 let memberObj = new Member();
14 console.log(memberObj.getName());
```

constructor

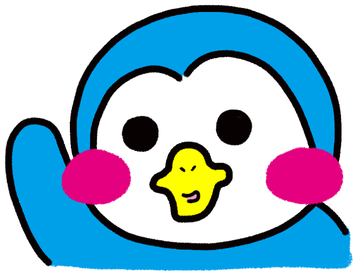


클래스에는 '생성자' 함수가 존재한다. 이는 new 연산자를 통한 클래스 오브젝트 생성 시 최초로 동작하는 메소드이며, constructor라는 이름을 사용한다.

```
1  class Pengsoo{
2    constructor(){
3      console.log("생성자 호출!")
4      this.name = "펭수"
5    }
6    sayMyName(){
7      window.alert(`펭-하! 저는 ${this.name}입니다!`)
8      return this.name
9    }
10 }
11
12 const peng = new Pengsoo()
13 const name = peng.sayMyName()
14 console.log(`${name}에 반환값이 저장되었습니다.`)
```

=> 주로 생성한 객체의 멤버 변수를 초기화하는 역할을 한다!

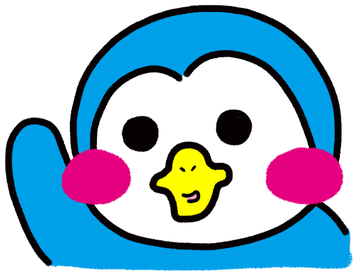
constructor 매개변수



생성자 함수 또한 함수이므로 매개변수 사용이 가능하며, new 연산 시 인자를 받는다.

```
class Pengsoo{
  constructor(name){
    console.log("생성자 호출!")
    this.name = name
  }
  sayMyName(){
    return this.name
  }
}
const peng1 = new Pengsoo("펭수")
const name1 = peng1.sayMyName()
const peng2 = new Pengsoo()
const name2 = peng2.sayMyName()
console.log(`1: ${name1}, 2: ${name2}`)
```

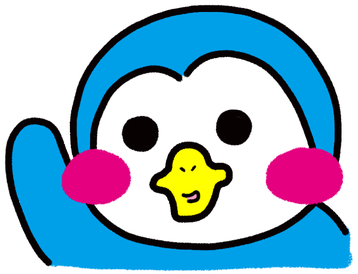
static



static 선언을 해주면 객체 생성 전부터 메소드를 사용할 수 있다.

```
1  class Person{
2    constructor(name){
3      this.name = name
4    }
5
6    // non-static 메소드
7    getName(){
8      return this.name
9    }
10
11   // static 메소드
12   static getSmile(){
13     return "스마일"
14   }
15 }
16
17 console.log(Person.getSmile()) // 가능
18 // console.log(Person.getName()) // 불가능
```

클래스의 상속



상속은 선언을 마친 클래스로부터 메소드와 속성을 물려받는 것을 의미한다.
상속 시에는 기존 클래스로부터 extends 키워드를 통해 상속을 받는다.

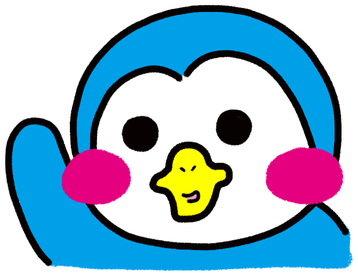
```
class Parent{
  constructor(name){
    this.name = name
  }
  sayName(){
    console.log(this.name)
  }
}
```

```

class Child extends Parent{
  set setName(name){
    this.name = name;
  }
}

let obj = new Child("원래 이름")
obj.setName = "새 이름"
obj.sayName()
```

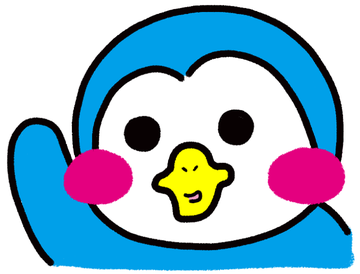
빌트인 오브젝트 상속



내장 객체를 상속받아 기능을 확장할 수 있다.

```
1  class ExtendArray extends Array{
2    constructor(){
3      super();
4    }
5    getTotal(){
6      let total = 0;
7      for(let i = 0; i < this.length; i++){
8        total += this[i]
9      }
10     return total
11   }
12 }
13 let obj = new ExtendArray();
14 obj.push(10,20)
15 console.log(obj.getTotal())
```

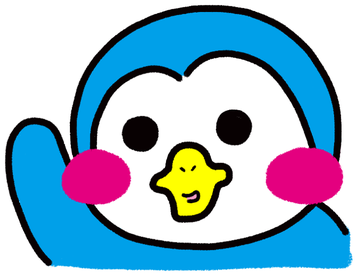
super?



super 는 상속을 해 준
부모클래스를
의미하는 키워드이다.
super 키워드를 호출하여
부모 클래스의 메소드를
호출할 수 있다.

```
1  class ExtendArray extends Array{
2    constructor(){
3      super();
4      super.push(1); super.push(2);
5      super.push(3); super.push(4);
6    }
7    getTotal(){
8      let total = 0;
9      for(let i = 0; i < this.length; i++){
10       total += this[i]
11     }
12     return total
13   }
14 }
15 let obj = new ExtendArray();
16 console.log(obj.getTotal())
```

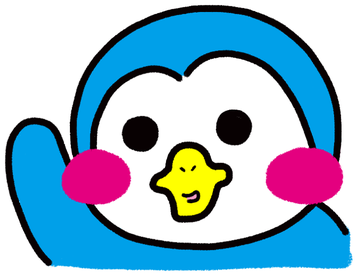
메소드 오버라이딩



물려받은 메소드를 자식 클래스가 재정의할 수 있다.

```
14  class StrangeExtendArray extends ExtendArray{
15    constructor(){
16      super();
17    }
18    getTotal(){
19      let total = 0;
20      for(let i = 0; i < this.length; i++){
21        total += (this[i] * 5)
22      }
23      return total
24    }
25  }
26
27  let obj = new StrangeExtendArray();
28  obj.push(10)
29  obj.push(20)
30  console.log(obj.getTotal())
```

내용 정리



- 클래스는 객체를 만드는 설계도이다.
- 클래스 내부에서 this는 생성된 객체를 의미하는 키워드이다.
- constructor 함수는 생성자 함수로써, 생성 시에 호출되는 함수이다.
- static 키워드는 메소드를 객체 생성 전부터 사용할 수 있도록 만든다.
- 상속은 기존 클래스로부터 신규 클래스가 자원을 물려받는 것이다.
- 상속받은 메소드를 재정의할 수 있다. 이를 오버라이딩이라 한다.