객체

객체와 프로퍼티

- 객체, 프로퍼티 정의

```
var obj = {
   name : '아이유',
   phone : '010-1234-5678'
};
```

- 프로퍼티 접근

```
obj.name = 'IU';
obj['phone'] = '010-5678-1234';
var propName = 'friend';
obj[propName] = 'YuInna'
```

- 없는 프로퍼티 접근?

```
student.age
```

객체와 메소드

- 객체 내 메소드 정의

```
var obj = {
    name : '아이유',
    phone : '010-1234-5678',
    sing : function() {
        console.log('좋은날~');
    }
};
obj.sing();
```

객체와 메소드

- this : 객체 내 참조
- 메소드 내 프로퍼티 접근

```
var obj = {
   name : '아이유',
   introduce : function() {
      console.log('내 이름은 ' + this.name);
   }
};
```

객체와 메소드

- 객체 내 메소드 추가

```
obj.dance = function() {
    console.log('분홍신');
}
```

실습

- 06.Class/object

함수를 이용한 객체 정의

- 함수 생성자를 이용한 클래스 정의
  - 프로퍼티, 메소드

```
function Greet() {
    message = 'Hello';
    this.hi = function() {
        console.log(message);
    }
}
```

- 객체 생성(new)과 사용

```
var obj = new Greet();

obj.hi();
```

함수를 이용한 객체 정의

- 파라미터가 있는 생성자 함수

```
function Actor(name, movie) {
    this.name = name;
    this.movie = movie;
    this.act = function() {
        console.log(this.name + ' perform in ' + this.movie);
    }
};
```

- 객체 생성과 사용하기 : new

```
var johansson = new Actor('Johansson', 'Avengers');
johansson.act();
```

함수를 이용한 객체 정의

- 함수 생성자 - 생성자 접근

```
var johansson = new Actor('Johansson', 'Avengers');

johansson.constructor
```

- 타입 비교

```
var johansson = new Actor('Johansson', ‘Avengers');
var alba = new Actor('Jessica Albl', 'Into the blue’);
var iu = new Singer('IU', '좋은날');
johansson.constructor === alba.constructor
iu.constructor === alba.constructor
```

객체 정의하는 방법의 차이

- 다수의 객체 생성하기
- Object Literal

```
var johansson = { name : ‘Johansson', movie : ‘Avengers’
                  act : function() { console.log('Act'); }}
var alba = { name : ‘Alba', movie : ‘Into the blue’
             act : function() { console.log('Act'); }}
```

- Object from Constructor

```
var johansson = new Actor('Johansson', 'Avengers');

var alba = new Actor('Jessica Albl', 'Into the blue');
```

실습

- 06.Class/constructor

프로퍼티

- 프로퍼티 : public, private

```
function Person(name, age) {
    this.name = name;
    var age = age;
}

var obj = new Person('설현', 21);
console.log(obj.name);
console.log(obj.age);
```

프로퍼티

- 프로퍼티 접근

```
var obj = new MyClass();
obj.prop1
obj["prop2"]
var name = "prop3";
obj[name]
```

- 프로퍼티 추가/삭제

```
obj.newProperty = 4;
delete obj.newProperty;
```

- 프로퍼티 확인

```
obj.hasOwnProperty(‘newProperty')
```

실습

- 06.Class/property

객체 생성 없이 사용하는 메소드

- 객체 생성 없이 사용하는 메소드 예

```
Math.random()
```

- 생성자 함수 정의

```
function Area() {}
Area.rectangle = function(width, height) {
  return width * height;
};
```

- 사용하기

```
Area.rectangle(10, 20)
```

프로토타입

- 모든 클래스는 프로토타입 프로퍼티가 있다.
- 객체는 프로토타입에 작성한 프로퍼티와 메소드를 상속

```
Rectangle.prototype.[메소드 이름] = function(){}
```

프로토타입을 사용한 정의

- Constructor

```
function Rectangle(width, height) {
     this.width = width;
     this.height = height;
};
```

```
Rectangle.prototype.size = function() {
    return this.width * this.height;
};
```

- 사용

```
var rect = new Rectangle(20, 30);
console.log(rect.size());
```

프로토타입

- 프로토타입과 프로퍼티

```
function Person() {}

Person.prototype.name = 'No name';

var who = new Person();
console.log(who.name);
```

실습

- 06.Class/prototype

this

- this
- 동작하는 context 에 따라서
- 클로저의 경우!

this

- 객체 접근 : this

```
function Person(name) {
    this.name = name;
    this.getName = function() {
        console.log(this.constructor);
        return this.name;
    }
}
```

this

- 객체 접근 : this

```
function Person(name) {
    this.name = name;
    this.getName = function() {}
    this.letMeIntroduce = function() {
        return function() {
            console.log('My Name is ' + this.name);
        }
    }
}
```

this

- 객체 접근 : this

```
function Person(name) {
    this.name = name;
    this.getName = function() {}

    this.letMeIntroduce = function() {
        var self = this;
        return function() {
            console.log('My Name is ' + this.name);
            console.log('My Name is ' + self.name);
        }
    }
}
```

this

- 객체 접근 : this
- Arrow Function의 경우 Object가 되지 않는다.

```
function Person(name) {
   this.name = name;

   this.withArrow = function() {
      return () => {
         console.log('this instanceof Person :' ,this instanceof Person);
         console.log('My Name is ' + this.name);
      }
   }
}
```

- 예제 코드 : 06.Class/this.js

클래스

- 클래스. ES6

```
class Person {
    constructor(name) {
        this.name = name;
    }

    sayHello() {
        console.log('Hello. I am ' + this.name);
    }
}

var iu = new Person('IU');
iu.sayHello();
```

- 예제 코드 : 06.Class/class.js