17장: 생성자 함수에 의한 객체 생성

생성자 함수

- 생성자 함수는 객체를 생성하는 함수이다.
- 함수를 new 연산자와 함께 호출하여 사용한다.
- 동일한 프로퍼티를 가지는 객체를 여러개 생성해야 할 때 용이하다.
- 일반 함수와 비슷하면서도, 다른 점들이 존재한다.

Object와 원시타입 생성자 함수

- new 연산자와 Object 생성자 함수를 호출하면 빈 객체를 생성하여 반환한다.
- String, Number, Boolean, Function, Array, Date, RegExp, Promise 등의 빌트인 생성자 함수를 제공한다,

```
const obj = new Object() // const obj = {} 와 동일

// 다음의 코드는 "원시 객체"를 반환한다!
// 웬만하면 사용하지 않는 것이 좋다
const strObj = new String('lee');
const numObj = new Number('lee');
const boolObj = new Boolean('lee');
const regExpObj = new RegExp('lee');
```

생성자 함수의 인스턴스 생성 과정

```
function Circle(radius) {
  // 1. 암묵적으로 빈 객체가 생성되고 this에 바인딩된다

  // 2. this에 바인딩되어 있는 인스턴스를 초기화 한다.
  this.radius = radius;

  this.getDiameter = function () {
    return 2 * this.radius;
  };


  // 3. this에 바인딩된 인스턴스를 암묵적으로 반환한다.
  // 명시적으로 "객체"를 반환하면 암묵적인 this 반환이 무시된다.
  // 명시적으로 원시 값을 반환하면 원시 값 반환은 무시되고 this가 반환된다.
}

const circle = new Circle(1);
console.log(circle) // Circle {radius: 1, getDiameter: f}
```

내부 메서드 [[Call]]과 [[Construct]]

- 함수는 일반 객체와 동일하게 동작할 수 있다.
  - 함수는 [[Environment]], [[FormalParameters]] 등의 내부 슬롯과 [[Call]], [[Construct]] 같은 내부 메서드를 을 가지는 객체이다.
  - 일반 함수처럼 호출시 [[Call]]이 호출된다 => 이를 callable이라 한다.
  - new 연산자와 함께 생성자 함수로서 호출되면 내부 메서드 [[Construct]]가 호출된다 => 이를 constructor(생성자) 함수라고 한다
  - 모든 함수는 Callable이지만, constructor와 non-constructor로 나뉜다.

Constructor와 Non-constructor

- Constructor: 함수 선언문, 함수 표현식, 클래스(클래스도 함수다)
- Non-Constructor: 메서드(ES6 메서드 축약 표현), 화살표 함수

```
const obj = {
  // 일반 함수로 정의된 메서드는 ES6에서 메서드로 인정받지 못한다
  method1: function () {},
  // 메서드 축약표현만 메서드로 인정한다
  method2 () {}
}

new obj.method1() // {}
new obj.method2() // TypeError: obj.method2 is not a constructor
```

일반 함수와 생성자 함수

```
function Circle (radius) {
  this.radius = radius;

  this.getDiameter = function () {
    return 2*this.radius;
  }
}

const circle = Circle(10);
```

- 일반 함수와 생성자 함수의 형식적 차이는 없다
- New 연산으로 일반 함수를 호출해도 생성자 함수처럼 호출된다.
- New 연산없이 생성자 함수를 호출해도 일반 함수처럼 호출된다.
  - 다만, 이때 this는 생성할 객체가 아니라, 전역 객체(window)를 가르킨다
  - => 생성자 함수는 **파스칼 케이스**로 명명하여 구별할 수 있도록 한다

ES6 new.target

```
function Circle (radius) {
  if (!new.target) return new Circle(radius)

  this.radius = radius;

  this.getDiameter = function () {
    return 2*this.radius;
  }
}
```

- 생성자 함수가 new 연산자 없이 호출되는 것을 방지하기 위해 new.target이 새로 생겼다.
- new.target은 this와 유사하게 consturctor인 모든 함수 내부에서 암묵적인 지역 변수와 같이 사용되며 메타 프로퍼티라고 부른다.
- new 연산자와 함께 생성자 함수로서 호출되면 함수 내부의 new.target은 함수 자신을 가르킨다.
- New 연산자 없이 호출된 함수는 undefined를 가르킨다.

스코프 세이프 생성자 패턴 (new.target 지원하지 않는 경우)

```
function Circle (radius) {
  if (!this instanceof Circle) return new Circle(radius)

  this.radius = radius;

  this.getDiameter = function () {
    return 2*this.radius;
  }
}
```