17장
생성자 함수에 의한
객체 생성

발표자: 최별🌟

17.1 OBJECT 생성자 함수

```
// 빈 객체의 생성
const person = new Object();
    인스턴스          생성자 함수
// 프로퍼티 추가
person.name = 'Lee';
person.sayHello = function () {
  console.log('Hi! My name is ' + this.name);
};

console.log(person); // {name: "Lee", sayHello: f}
person.sayHello(); // Hi! My name is Lee
```

STRING, NUMBER, BOOLEAN, FUNCTION, ARRAY, DATE, REGEXP(정규표현식), PROMISE 등의 빌트인 생성자 함수도 존재

17.2 생성자 함수

17.2.1 객체 리터럴에 의한 객체 생성 방식의 문제점

```
const circle1 = {
  radius: 5,
  getDiameter() {
    return 2 * this.radius;
  }
};
console.log(circle1.getDiameter()); // 10

const circle2 = {
  radius: 10,
  getDiameter() {
    return 2 * this.radius;
  }
};
console.log(circle2.getDiameter()); // 20
```

비효율적

17.2 생성자 함수

17.2.2 생성자 함수에 의한 객체 생성 방식의 장점

```
// 생성자 함수
function Circle(radius) {
  // 생성자 함수 내부의 this는 생성자 함수가 생성할 인스턴스를 가리킨다.
  this.radius = radius;
  this.getDiameter = function () {
    return 2 * this.radius;
  };
}

// 인스턴스 생성
const circle1 = new Circle(5); // 반지름이 5인 Circle 객체 생성
const circle2 = new Circle(10); // 반지름이 10인 Circle 객체 생성

console.log(circle1.getDiameter()); // 10
console.log(circle2.getDiameter()); // 20
```

NEW와 함께 사용

17.2 생성자 함수

17.2.3 생성자 함수의 인스턴스 생성 과정

생성자 함수의 역할? 인스턴스 생성(필수), 인스턴스 초기화(옵션)

1. 인스턴스 생성과 THIS 바인딩
2. 인스턴스 초기화
3. 인스턴스 반환

17.2 생성자 함수

17.2.3 생성자 함수의 인스턴스 생성 과정

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

  // 2. this에 바인딩되어 있는 인스턴스를 초기화
  this.radius = radius;
  this.getDiameter = fucntion () {
    return 2 * this.radius;
  };

  // 3. 완성된 인스턴스가 바인딩된 this가 암묵적으로 반환
}

// 인스턴스 생성. Circle 생성자 함수는 암묵적으로 this를 반환
const circle = new Circle(1);
console.log(circle); // Circle {radius: 1, getDiameter: f}
```

RETURN 문 반드시 생략!

17.2 생성자 함수

17.2.4 내부 메서드 [[CALL]]과 [[CONSTRUCT]]

```
function foo() {}

// 일반적인 함수로서 호출: [[call]]이 호출
foo();

// 생성자 함수로서 호출: [[construct]]가 호출
new foo();
```

일반 객체는 호출할 수 없지만 함수는 호출 가능

일반 함수로서 호출: 함수 객체의 내부 메서드 [[CALL]이 호출
NEW 연산자와 함께 생성자 함수로서 호출: 내부 메서드 [[CONSTRUCT]]가 호출
[[CALL]]을 갖는 객체 - CALLABLE
[[CONSTRUCT]]를 갖는 객체 - CONSTRUCTOR

17.2 생성자 함수

17.2.5 CONSTRUCTOR와 NON-CONSTRUCTOR의 구분

자바스크립트 엔진은 함수 정의 방식에 따라 구분

- CONSTRUCTOR: 함수 선언문, 함수 표현식, 클래스
- NON-CONSTRUCTOR: 메서드(ES6 메서드 축약 표현), 화살표 함수

NON-CONSTRUCTOR인 함수 객체를 생성자 함수로서 호출하면 에러가 발생

17.2 생성자 함수

17.2.6 NEW 연산자

NEW 연산자와 함께 함수를 호출하면 해당 함수는 생성자 함수로 동작.

단, CONSTRUCTOR이어야 한다

일반 함수 – [[CALL]] 호출

생성자 함수 – [[CONSTRUCTOR]] 호출

17.2 생성자 함수

17.2.7 NEW.TARGET

```
// 생성자 함수
function Circle(radius) {
  // 이 함수가 new 연산자와 함께 호출되지 않았다면 new.target은 undefined
  if (!new.target) {
    // new 연산자와 함께 생성자 함수를 재귀 호출하여 생성된 인스턴스를 반환
    return new Circle(radius);
  }
  this.radius = radius;
  this.getDiameter = function () {
    return 2 * this.radius;
  };
}

// new 연산자 없이 생성자 함수를 호출하여도 new.target을 통해 생성자 함수로서 호출
const circle = Circle(5);
console.log(circle.getDiameter());
```

17.2 생성자 함수

17.2.7 NEW.TARGET

대부분의 빌트인 함수(OBJECT, STRING, NUMBER, BOOLEAN, FUNCTION, ARRAY, DATE, REGEXP, PROMISE 등)는 NEW 연산자와 함께 호출되었는지를 확인한 후 적절한 값을 반환

STRING, NUMBER, BOOLEAN 생성자 함수는 NEW 연산자 없이 호출하면 문자열, 숫자, 불리언 값을 반환
→ 이를 통해 데이터 타입을 변환하기도 한다.

감사합니다

발표자: 최별🌟