17장. 생성자 함수에 의한 객체 생성

17.1 Object 생성자 함수

new연산자와 함께 Object 생성자 함수를 호출하면 빈 객체를 생성하여 반환

빈 객체를 생성한 이후 프로퍼티 또는 메서드를 추가하여 객체 완성

- **생성자 함수란?**

  new 연산자와 함께 호출하여 객체(인스턴스)를 생성하는 함수

- **인스턴스란?**

  생성자 함수에 의해 생성된 객체

반드시 Object 생성자 함수를 사용해 빈 객체를 사용하는 것이 아님 → 특별한 이유 없으면 그닥 유용X

17.2 생성자 함수

객체 리터럴에 의한 객체 생성 방식은 직관적이고 간편

But, 객체 리터럴에 의한 객체 생성 방식은 단 하나의 객체만 생성

> 동일한 프로퍼티를 갖는 객체를 여러 개 생성해야 하는 경우 매번 같은 프로퍼티를 기술해야해서 비효율적

생성자 함수에 의한 객체 생성 방식

마치 객체(인스턴스)를 생성하기 위한 템플릿(클래스)처럼 생성자 함수를 사용하여 프로퍼티 구조가 동일한 객체를 여러 개 간편하게 생성 가능

- **this란?**

  객체 자신의 프로퍼티나 메서드를 참조하기 위한 자기 참조 변수다. this가 가리키는 값, 즉 this 바인딩은 함수 호출 방식에 따라 동적으로 결정된다.

일반 함수와 동일한 방법으로 생성자 함수를 정의하고 new 연산자와 함께 호출하면 해당 함수는 생성자 함수로 동작

→ 만약 new 연산자와 함께 생성자 함수를 호출하지 않으면 생성자 함수가 아닌 일반 함수로 동작

생성자 함수의 역할

== 프로퍼티 구조가 동일한 인스턴스를 생성하기 위한 템플릿(클래스)로서 동작하여 인스턴스를 생성하는 것과 생성된 인스턴스를 초기화(인스턴스 프로퍼티 추가 및 초기값 할당)하는 것

1) 생성자 함수가 인스턴스 생성하는 것 == 필수

2) 생성자 함수가 생성된 인스턴스를 초기화하는 것 == 옵션

생성자 함수의 인스턴스 생성과정

new 연산자와 함께 생성자 함수를 호출하면 자바스크립트 엔진은 과정을 거쳐 암묵적으로 인스턴스를 생성하고 인스턴스를 초기화한 후 암묵적으로 인스턴스를 반환

1. 인스턴스 생성과 this 바인딩

   암묵적으로 빈 객체가 생성되고. 이 객체가 생성자 함수가 생성한 인스턴스

   그리고 암묵적으로 생성된 빈 객체(인스턴스)는 this에 바인딩 된다.

   - 바인딩이란?

     식별자와 값을 연결하는 과정

2. 인스턴스 초기화

생성자 함수에 기술되어 있는 코드가 한 줄씩 실행되어 this에 바인딩되어 있는 인스턴스를 초기화

3. 인스턴스 반환

   생성자 함수 내부에서 모든 처리가 끝나면 완성된 인스턴스가 바인딩된 this를 암묵적으로 반환

   명시적으로 원시 값을 반환하면 원시 값 반환은 무시되고 암묵적으로 this가 반환된다.

   따라서 생성자 함수 내부에서 return문을 반드시 생략해야한다

일반 객체는 호출할 수 없지만 함수는 호출할 수 있다.

```
function foo() {}
    //일반적인 함수로 호출: [[Call]]이 호출
    foo();

    //생성자 함수로 호출: [[Construct]]가 호출
    new foo();
```

함수 객체는 constructor일 수도 있고 non-constructor일 수도 있다.

→ 모든 함수 객체는 호출할 수 있지만 모든 함수 객체를 생성자 함수로서 호출할 수 있는 것은 아님

- **constructor란?**

  함수 선언문, 함수 표현식, 클래스

- **non-constructor란?**

  메서드, 화살표 함수

non-constructor인 함수 객체를 생성자 함수로서 호출하면 에러 발생

!주의: 생성자 함수로서 호출될 것을 기대하고 정의하지 않은 일반 함수(callable이면서 constructor)에 new 연산자를 붙여 호출하면 생성자 함수처럼 동작할 수 있다.

생성자 함수가 new 연산자 없이 호출되는 것을 방지하기 위해 new.target을 지원

- **new.target이란?**

  this와 유사하게 constructor인 모든 함수 내부에서 암묵적인 지역 변수와 같이 사용되며 메타 프로퍼티라고 부름

  new 연산자와 함께 생성자 함수로서 호출되면 함수 내부의 new.target은 함수 자신을 가리킨다.

  new 연산자 없이 일반 함수로서 호출된 함수 내부의 new.target은 undefined다.

→ 함수 내부에서 new.target을 사용하여 new 연산자와 생성자 함수로서 호출했는지 확인하여 그렇지 않은 경우 new 연산자와 함께 재귀 호출을 통해 생성자 함수로서 호출 가능