Javascript 객체

키워드 : 객체 (https://www.w3schools.com/js/js_objects.asp), 배열 객체 (https://www.w3schools.com/js/js_arrays.asp), Math 객체 (https://www.w3schools.com/js/js_math.asp)

우리가 사는 현실 세계는 수 많은 객체(Objects)로 이루어져 있습니다. 예를 들어, 강의에서 살펴봤듯이 고양이는 하나의 객체입니다. 이러한 객체는 색, 크기 등과 같은 속성(Properties)을 갖습니다. 또한, 객체는 '냐옹'이나 '냥펀치' 등과 같은 고유한 행동(Methods)을 할 수 있습니다.

이번 장에선 현실 세계의 객체를 자바스크립트 코드로 작성하는 방법에 대해서 살펴보도록 하겠습니다. 우선 계속해서 고양이를 통해 개념에 대해서 살펴보겠습니다.

모든 고양이는 색과 크기 등의 속성을 갖습니다. 하지만, 색과 크기에 해당하는 **값**(Values)은 고양이마다 서로 다릅니다. 마찬가지로, 모든 고양이는 '냐옹'과 '냥펀치' 등의 행동을 할 수 있습니다. 하지만, 해당 행동은 서로 다른 시간에 필요한 경우에 하게 됩니다.

객체 선언

아래 코드 예시는 자바스크립트 코드로 고양이 객체를 선언하고, 사용하는 방법을 나타냅니다.

```
var cat = {name:"껌껌이", color:"black", age:3, eyeColor:"blue"};
console.log(typeof cat); //object
```

객체는 변수를 선언할 때 사용했던 var 키워드를 이용해서 선언하고, 객체를 감싸는 코드 블록({})으로 구성합니다. 또한, 객채의 속성은 `property:value` 형태로 작성하고, 속성이 여러개일 경우 쉼표(,)로 구분 짓습니다.

이렇게 사용자가 직접 정의한 객체를 일반적으로 '사용자 정의 객체'라고 부릅니다. 객체를 선언만 하고 사용하지 않는다면 의미가 없겠죠? 우리는 객체를 선언하고 속성 값을 얻어오거나(get), 값을 변경(set) 할 수 있습니다. 다음은, get과 set을 하는 방법에 대해서 살펴보도록 하겠습니다.

아래 코드는 위에서 선언한 cat 객체의 속성을 get, set 하는 방법에 대한 예를 보여줍니다.

```
//기본 형태
객체.속성
or
객체[속성]

//1번
console.log(cat.name, cat.color); //껌껌이, black
cat.age = 5;
cat.eyeColor = "yellow";
console.log(cat.age, cat.eyeColor); //5, yellow

//2번
console.log(cat["name"], cat["color"]); //껌껌이, black
cat["age"] = 5;
cat["eyeColor"] = "yellow";
console.log(cat["age"], cat["eyeColor"]); //5, yellow
```

객체를 선언하고 값을 가져오는 작업(get)을 하기 위해서 닷(.) 연산자를 사용해 1번 코드처럼 객체.속성 형태로 접근하거나, 2번 코드처럼 배열의 값을 접근할 때처럼 객체["속성"] 형태로 접근할 수 있습니다. 반대로 기존의 값을 변경하는 작업(set)을 하기 위해선 변수에 값을 할당 할 때처럼 객체.속성= 값 혹은, 객체["속성"] = 값 형태로 작성합니다.

위에서 객체는 속성(Properties)과 행동(Methods)으로 구성된다고 언급했습니다. 하지만, 메서드에 대한 내용은 자바스크립트 함수에 대한 선행 지식이 필요하므로, 이번 장에서는 사용자 정의 객체에 메서드를 추가하는 방법에 대해선 다루지 않습니다. 사용자 정의 객체에 메서드를 추가하는 방법은 해당 내용에 필요한 선행 지식을 더 공부하고 알아보도록 하겠습니다. :)

new 연산자

앞서 우리가 사용자 정의 객체를 선언하기 위해선 코드 블록({})에 property:value를 쉼표(,)로 구분 짓고 나열한다고 언급했습니다. 이런 식으로 중괄호로 감싼 코드 블록({}) 형태로 객체를 선언하는 방법을 리터럴(Literal) 형식으로 선언한다고 표현합니다.

자바스크립트에선 객체를 리터럴 형식으로 선언하는 방식과 더불어 new 키워드를 이용한 객체 선언 방식도 지원합니다. 우리가 변수를 이야기할 때 살펴본 자료형도 사실 하나의 객체로 선언할 수 있습니다. 선언 방법은 var 객체명 = new Object() 형태로 선언합니다.

아래 코드는 위에서 선언한 여러 가지 자료형을 new 연산자를 이용해 선언하는 방법에 대한 예를 보여줍니다.

```
var obj = new Object(); //객체
var str = new String(); //문자열 객체
var num = new Number(); //숫자 객체
var bool = new Boolean(); //논리 객체
var arr = new Array(); //배열 객체
```

일반적으로 new 연산자를 사용해 객체를 선언하는 방법보단 우리가 기존에 알고있는 리터럴 자료형 선언 방식을 권장합니다. 위 코드는 아래와 같이 리터럴 방식으로 선언할 수 있습니다.

```
var obj = {}; //빈 객체 선언
var str = ""; //빈 문자열 선언
var num = 0; //숫자타입 선언
var bool = false; //논리타입 선언
var arr = []; //빈 배열 선언
```

단, 예외적으로 내장 객체 부분에서 살펴볼 날짜 정보를 표현하는 Date 객체와 같은 경우 new 연산자를 이용해 선언해서 사용해야 하는 경우도 있습니다. 이러한 부분은 한 번에 모두 이해하기 다소 어려울 수 있습니다. 코드를 직접 작성하면서 하나하나 익혀 나가도록 해보세요! :)

배열 객체

사용자 정의 객체가 존재하듯 자바스크립트가 기본적으로 제공하는 객체가 있습니다. 우리가 따로 정의하지 않았지만 기존의 자바스크립트가 제공하는 객체를 '내장 객체'라고 부릅니다. 내장 객체와 관련된 내용은 이번 장과 다음 장에서 계속해서 살펴보도록 하겠습니다.

우선, 반복문에서 잠깐 살펴보았던 배열도 하나의 객체로 처리됩니다. 이번 절에선 배열과 관련된 내용에 대해서 알아보겠습니다.

배열은 다음과 같은 큰 특징을 갖습니다.

- 여러 개의 자료(값)를 한꺼번에 저장할 수 있는 자료형 객체
- 배열의 크기는 자동 관리
- 배열 요소 탐색은 인덱스 사용, 인덱스는 0부터 시작

배열도 하나의 객체로 처리 되기 때문에, 배열의 길이를 나타내는 `length` 속성을 갖습니다. 또한, 배열과 관련된 다양한 행동을 할 수 있는 메서드들이 존재합니다. 배열과 관련된 메서드는 굉장히 다양합니다. 아래 코드를 통해 사용법을 살펴보도록 하겠습니다.

```
//배열 선언
// 같은 표현 -> var array = new Array("html", "css", "Javascript", "Python");
var array = ["html", "css", "javascript", "python"];
array.push("java"); //배열의 끝 부분에 새로운 값 추가
array.unshift("php"); //배열의 시작 부분에 새로운 값 추가
console.log(array.slice(0, 3)); //배열의 특정 부분을 인덱싱(시작, 끝) -> ["php",
"html", "css"]
array.pop(); //배열의 끝 부분의 요소 삭제
array.shift(); //배열의 첫 부분의 요소 삭제
console.log(array.splice(2, 1, "jsp")); //배열의 특정 부분에 값 추가 및 해당 인덱스 다음에
나오는 요소 삭제(추가할 인덱스, 삭제할 인덱스 갯수, 추가할 값) -> "javascript" 삭제
console.log(array); //["html", "css", "jsp", "python"]
```

적절한 배열의 사용은 코드 작성 시 많은 부분에서 이점으로 작용합니다. 배열을 선언하고 다양한 메서드들을 사용하는 시간을 가져보세요 :)

Math 객체

강의 영상에서 수학과 관련된 정보를 제공하는 Math라는 내장 객체에 대해서 간단히 살펴보았습니다. 이번 절에선 몇 가지 Math 객체의 메서드에 대해 알아보고, 강의에서 보았던 Random 메서드에 대해서 살펴보겠습니다.

아래 코드를 통해 자주 사용되는 Math 객체의 메서드에 대해 살펴보도록 하겠습니다.

```
var x = -1, y = 12.34;
console.log(Math.abs(x)); //숫자의 절댓값을 반환 -> 1
console.log(Math.ceil(y)); //소수점 이하를 올림 -> 13
console.log(Math.floor(y)); //소수점 이하를 버림 -> 12
console.log(Math.round(y)); //소수점 이하를 반올림 -> 12
```

다음은 Math 객체의 Math.random()메서드와 관련된 내용입니다. Math.random()메서드는 0과 1 사이(0 <= X < 1)의 랜덤한 값을 반환합니다. 강의 영상에서 랜덤한 값을 반환한 결괏값이 소숫점 이하가 굉장히 긴 숫자를 나타내는 것을 확인 할 수 있었습니다. 만약, 1~10과 같이 특정 범위에 있는 정수값을 반환하고 싶을 경우에는 다음과 같이 위에서 살펴본 소수점 이하를 버리는 Math.floor() 메서드와 함께 수학적으로 범위를 지정해서 사용하면 됩니다.

```
    var rand = Math.floor(Math.random() * (max - min + 1) + min);
```

우선, 위 코드를 하나하나 살펴보겠습니다. Math.random()메서드가 0과 1 사이(0 <= X < 1)의 랜덤한 값을 반환하기 때문에 (상한값 - 하한값 + 1)을 곱해주면 0 x (max - min + 1) <= X < 1 x (max - min + 1) 형태가 됩니다. 여기에 하한값을 더해주면 0 x (max - min + 1) + min <= X < 1 x (max - min + 1) + min형태가 됩니다. 0에 어떤 값을 곱해도 0이기 때문에 마지막에 더한 min에 해당하는 값이 랜덤한 값의 하한값이 됩니다. 또한, 랜덤 범위의 상한값은 이하가 아닌 미만을 나타내기 때문에 1을 더해준 값이 포함되지 않습니다.

마지막으로, 반환 받은 랜덤한 값을 Math.floor()메서드로 소수점 이하를 버리는 연산을 진행하면 우리가 원하는 정수 형태로 반환 받게됩니다. 정수 범위의 랜덤한 값을 반환하는 최종 코드는 아래와 같습니다.

```
    var rand1to10 = Math.floor(Math.random() * (10 - 1 + 1) + 1);
    console.log(rand1to10); //1~10 범위의 랜덤한 값 반환

    var rand10to100 = Math.floor(Math.random() * (100 - 10 + 1) + 10);
    console.log(rand10to100); //10~100 범위의 랜덤한 값 반환
```

더 많은 Math 객체와 관련된 내용은 [여기 (https://www.w3schools.com/jsref/jsref_obj_math.asp)](https://www.w3schools.com/jsref/jsref_obj_math.asp)를 참고해주세요.

추가 내용

null 값

우리가 자바스크립트 자료형에서 다루지 않았던 내용 중에 null 값이라는 게 있었습니다. null 값은 기본적으로 아무것도 없음을 나타냅니다. 쉽게 말해 '존재 하지 않는 값'을 지칭합니다. 또한, 자바스크립트에서 null 값은 하나의 객체로 표현됩니다.

```
var user = {firstname: "seongjae",
            lastname: "moon",
            city: "Suwon",
            gender: "male"};

var user = null; //user 객체에 null 값 할당
console.log(typeof user); //object
```

말이 조금 어려울 수 있는데요, 예를 들어 위와 같은 코드가 있다고 가정해보겠습니다. 위의 코드에서 'user' 객체를 선언하고 user 객체에 null 값을 할당했습니다. null 값을 할당함으로써 user 객체가 가지고 있던 모든 속성은 사용할 수 없습니다. 하지만, user 객체는 여전히 객체를 나타내는 object 형을 갖습니다.

그렇다면, 우리가 알고 있는 타입이 아직 정해지지 않은 형태인 'undefined'와 차이점은 무엇일까요? 바로 타입과 관련 있습니다. null 값은 object라는 객체를 나타내는 타입이지만, undefined는 undefined 자체로 고유한 타입을 갖습니다.

```
console.log(typeof undefined); //undefiend
console.log(typeof null); //object

console.log(null == undefined); //true
console.log(null === undefined); //false
```

일반적으로 null 값은 객체를 초기화하기 위해서 사용됩니다. 쉽게 생각해서 아래 코드와 같이 의도적으로 어떤 객체를 사용하기 전에 객체를 담을 변수에 null 값을 할당하고 나중에 해당 객체의 특정 값을 할당하는 형태라고 할 수 있습니다.

```
var obj = null;
//코드...
```

자바스크립트는 객체 지향(Object-Oriented) 프로그래밍 언어 중 하나입니다. 프로그램 코드 상에 표현되는 다양한 정보의 의미 있는 것들을 묶어 하나의 개념으로 추상화시켜 객체로 표현하고 사용하게 됩니다. 객체와 관련된 내용은 전체 자바스크립트 챕터 중에서도 난이도 있는 파트 중에 하나입니다. 또한, 굉장히 중요한 파트이기도 합니다. 당장은 이해하기 어려운 부분들이 존재하겠지만, 직접 다양한 코드를 작성해보고 익숙해지도록 노력해보세요 :)