Javascript 내장 객체

키워드 : Date 객체 (https://www.w3schools.com/js/js_dates.asp),
문자열 객체 (https://www.w3schools.com/js/js_strings.asp),
Number 객체 (https://www.w3schools.com/js/js_numbers.asp)

자바스크립트 코드 상에서 표현되는 다양한 정보의 의미 있는 것들을 묶어 하나의 개념으로 추상화 시킨 것을 객체라고 이야기 했습니다. 또한, 우리가 직접 객체를 선언하고, 사용할 수 있다고 객체 관련 강의에서 살펴보았습니다.

자바스크립트를 비롯한 여러 프로그래밍 언어는 개발자가 처음부터 모든 것을 개발하는 수고를 덜고자, 프로그래밍 언어 수준에서 기본적으로 제공하는 내장 객체 혹은, 내장 함수라는 것이 존재합니다. 이번 장에서는 자바스크립트에서 제공하는 내장 객체 세 가지에 대해서 알아보도록 하겠습니다.

Date 객체

우선, 강의에서도 살펴보았던 날짜 관련 객체입니다. 자바스크립트 코드를 작성하는 데 있어서 날짜를 계산하고 표현하는 방법은 굉장히 자주 등장하는 내용 중 하나입니다. 자바스크립트 코드에서 제공하는 날짜 객체를 선언하는 방법은 아래와 같이 크게 네 가지가 있습니다.

선언 방법	의미
new Date()	기본 선언
new Date(년, 월, 일, 시간, 분, 초, 밀리초)	날짜 정보를 각각 기입
new Date(밀리초)	밀리초 단위(1/1000초)로 기입
new Date(date string)	정해진 날짜 포멧으로 기입

UTC (국제 표준시)

> 1일: 24시간, 1시간: 60분, 1분: 60초, 1초: 1000밀리초

기본적으로 날짜 객체는 UTC(세계 협정 시간대) 체계 안에서 1970년 1월 1일을 기준으로 밀리초 단위로 값을 표현하게 됩니다. 우리는 보통 밀리초(1/1000초) 까지는 계산하지 않지만, 자바스크립트 코드로 날짜를 연산하는 과정에선 이 밀리초 단위를 기준으로 연산한다는 점을 기억해주세요!

아래 코드는 우리가 잘 알고 있는 변수에 날짜 객체 초기화하고 할당하는 네 가지 방법입니다.

```
var date1 = new Date();
var date2 = new Date(2018, 12, 31, 1, 1, 1); //각각의 인자(Parameter)는 생략 가능.
var date3 = new Date(0); //"January 01, 1970 00:00:00 UTC."
var date4 = new Date("December 31, 2018 1:1:1");
```

날짜 포멧(Format)

위에서 살펴본 방법들은 우리가 일반적으로 날짜를 표현할 때 사용하는 방법이 아니기 때문에 바로 사용하기에는 낯선 부분들이 다소 존재합니다. 이러한 부분을 해소하기 위해서 우리는 우리가 일반적으로 사용하는 날짜 형식(포멧)을 사용해서 날짜 객체를 초기화 할 수 있습니다.

아래 코드는 우리가 알고 있는 형태로 날짜를 나타내는 예시를 보여줍니다.

선언 방법	의미
"2018-12-31"	국제 표준 선언(ISO Date)

"12/31/2018" 축약 선언(Short Date)
"Dec 31 2018" 또는 "31 Dec 2018" 비 축약 선언(Long Date)

위 표에서 ISO Date라고 나와있는 형태("YYYY-MM-DD")로 작성하는 방법이 국제 표준화 기구(International Organization for Standardization)에서 권장하는 방법입니다. 일반적으로 ISO 8601라고 불리우는 이 방법은 년도(YYYY), 월(MM), 일(DD) 형태로 작성하게 됩니다.

권장하는 방법은 위 ISO Date 형태로 작성하는 것이지만, 경우에 따라서 다른 방법으로 표현하는 경우도 있으므로, 나머지 방법도 기억해주세요 :)

날짜 메서드

우리가 날짜 객체를 이용해서 날짜를 계산하는 연산을 하기 위해선, 문자 형태로 나타나 있는 날짜 정보를 연산할 수 있도록 숫자화 하는 과정이 필요합니다. 위에서 살펴본 1970년 1월 1일을 기준으로(0)하는 밀리초 단위로의 변환을 통해 날짜 계산을 가능케 할 수 있습니다.

```
var date = new Date();
console.log(date.getTime()); //현재 날짜를 밀리초 단위로 반환
```

위 코드와 같이 밀리초 단위로 현재 날짜를 반환하는 메서드인 getTime()을 비롯해 Date 객체의 여러 메서드를 이용해서 각종 날짜 정보 중 필요한 정보만을 추출할 수 있습니다.

더 많은 Date 객체 메서드와 관련된 내용은 여기 (https://www.w3schools.com/js/js_date_methods.asp)와 여기 (https://www.w3schools.com/js/js_date_methods_set.asp)를 참고해주세요.

문자열 객체

우리는 강의를 통해 문자열을 할당하고 조작하기 위해서 자바스크립트에서 string 타입의 변수를 선언하고 할당할 수 있다는 것에 대해서 알아보았습니다.

```
var str1 = ""; //빈 문자열 선언
var str2 = ''; //빈 문자열 선언
```

문자열은 위 코드와 같이 큰 따옴표("") 혹은, 작은 따옴표('')로 특정 문자열을 감싸는 형태로 선언할 수 있습니다. 만약, 문자열 안에서 큰 따옴표 혹은 작은 따옴표 자체를 문자열에 포함시켜 표현하고 싶다면 아래와 같이 서로 다른 형태로 감싸는('"'", '"'') 코드를 작성해야합니다.

```
var str1 = "My name is 'Seongjae Moon.'";
console.log(str1); //My name is 'Seongjae Moon.'
var str2 = 'My name is "Seongjae Moon."';
console.log(str2);//My name is "Seongjae Moon."
```

기본적으로 객체는 속성(Properties)과 메서드(Methods)를 가질 수 있습니다. 이는 문자열을 우리가 선언함과 동시에 자바스크립트에 이미 내장되어 있는 문자열 관련 속성과 메서드를 이용할 수 있다는 말이 됩니다.

아래 코드는 대표적인 문자열 객체의 속성인 length와 indexOf, substring 메서드의 사용의 예시입니다.

```
var str = "My name is Seongjae Moon.";
console.log(str.length); //25
console.log(str.index('name')); //3
consoel.log(str.substring(0, 2)); //My
```

먼저, length 속성 값은 문자열의 길이를 나타내며 1부터 시작합니다.

indexOf 메서드는 해당 문자열에 찾고자 하는 문자열의 처음 시작 위치(인덱스)를 처음 문자열을 0부터 계산하여 반환합니다.

substring 메서드는 특정 문자열을 (시작 인덱스, 끝 인덱스) 형태로 작성하여 추출할 수 있는 메서드입니다.

더 많은 문자열 객체 메서드와 관련된 내용은 여기 (https://www.w3schools.com/js/js_string_methods.asp)를 참고해주세요.

문자열 처리와 관련하여 자바스크립트의 중요한 내용 중 하나이지만, 우리 강의 과정에서 내용의 난이도로 인해 다루지 않은 정규 표현식 객체가 존재합니다. 정규 표현식 객체를 이용하면 다양한 문자열을 처리하는 과정을 수월하게 해낼 수 있습니다. 해당 내용은 내용이 굉장히 길어지므로 따로 정리해서 URL을 첨부해드리도록 하겠습니다. 만약, 지금 당장 궁금하시다면 여기 (https://www.w3schools.com/js/js_regexp.asp)를 참고해주세요!

Number 객체

자바스크립트 코드를 작성할 때 다양한 수학적 연산이 필요한 경우가 많습니다. 우리가 연산자를 공부할 때 살펴보았던 기본적인 사칙연산뿐만 아니라, 논리적으로 필요에 의해서 행하게 되는 다양한 연산들이 있습니다.

이번 절에서는 논리적인 수식 연산을 위해 Math 객체와 더불어 자주 사용되는 Number 객체의 속성과 메서드에 대해서 알아보겠습니다.

우선, Number 객체에서 제공하는 주요 속성은 다음과 같습니다.

선언 방법	의미
Number.MAX_VALUE	자바스크립트에서 표현할 수 있는 가장 큰 수 반환
Number.MIN_VALUE	자바스크립트에서 표현할 수 있는 가장 작은 수 반환
Number.POSITIVE_INFINITY	양의 무한대 반환
Number.NEGATIVE_INFINITY	음의 무한대 반환

위의 표에 나타난대로 자바스크립트에서는 표현할 수 있는 숫자의 범위가 정해져 있습니다. MAX_VALUE: 1.7976931348623157e+308,MIN_VALUE: 5e-324와 같은 식 입니다.

다음은 Number 객체의 전역 메서드(Global Methods)라고 불리는 메서드에 대해서 알아보겠습니다.

선언 방법	의미
Number(인자)	인자를 숫자형으로 변환하여 반환
parseFloat(인자)	인자를 실수로 변환하여 반환
parseInt(인자)	인자를 정수로 변환하여 반환

문자열 형태로 표현된 숫자 데이터를 수학적 연산을 위해 숫자형으로 변환해야 하는 경우가 생길 수 있습니다. 그런 경우 Number(문자열) 형태로 작성하면 문자열을 숫자 자료형(number)으로 변환할 수 있습니다.

또한, 문자열을 실수, 혹은 정수 자료형으로 특정 지어 변환할 수도 있습니다.

```
var str = "10";
var num = Number(str);
console.log(num, typeof num); //10, number
console.log(parseFloat(10)); //10
console.log(parseInt(10.1)); //10
```

더 많은 Number 객체 메서드와 관련된 내용은 여기 (https://www.w3schools.com/js/js_number_methods.asp)를 참고해주세요.

참고로, 문자열은 객체로 선언하지 않고 일반적으로 선언해도 선언과 동시에 문자열 객체 속성과 메서드를 사용할 수 있습니다. 하지만, 숫자 자료형은 기본 자료형 선언과 Number 객체 선언과의 차이가 있기 때문에 Number 객체의 속성과 메서드를 사용하기 위해선 Number 객체로 선언해야 합니다.

```
var str1 = "hi"; //속성, 메서드 사용 가능
var str2 = new String("hi"); //속성, 메서드 사용 가능

var num1 = 10; //속성, 메서드 사용 불가
var num2 = new Number(10); //속성, 메서드 사용 가능
```

자바스크립트에는 위에서 설명한 내장 객체뿐만 아니라, 잠시 언급한 정규 표현식 객체와 같은 다양한 내장 객체가 존재합니다. 아마 이런 생각이 드실지도 모르겠습니다.

> "이걸 언제 다 공부하지?, 다 알아야 하나?"

결론적으로, 처음부터 모든 내용을 다 암기하실 필요는 없습니다. 사실, 익숙해지는데 꽤나 오랜 시간이 걸리는 내용들입니다. 다만, 주요 포인트를 기억해두었다가 필요할 때 찾아볼 수 있는 정도의 이해만 있다면 충분합니다. 기본적인 이해를 갖고, 해당 내용을 필요에 따라 찾아보며 이해할 수 있는 능력만 있다면, 언제든 입맛에 맞게 사용할 수 있을 겁니다. :)