03

Implicit Object

JavaScript

3장. Implicit Object

I. Javascript 내장 객체

II. 브라우저 내장 객체

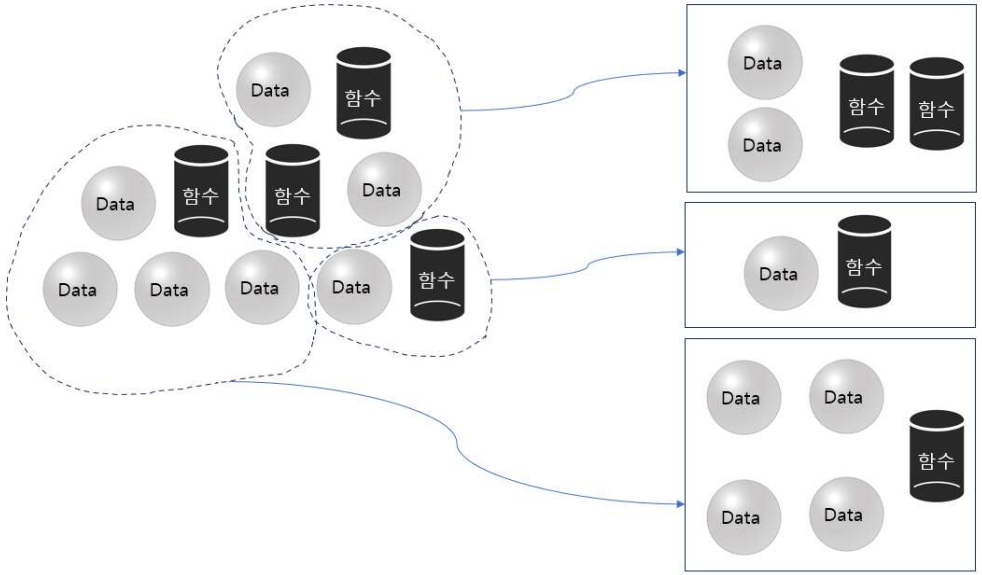
III.document

03-1. JavaScript 내장 객체

Implicit Object

객체란

- 객체란 함수와 데이터의 집합입니다.



OOP

- OOP 는 Object Oriented Programming 의 약어이며 '객체지향 프로그래밍'이라고 합니다.
- 객체를 선언하고 객체를 이용해 코드를 작성하는 기법을 이야기 하는 용어이며 자바스크립트 뿐만 아니라 대부분의 소프트웨어 언어에서 화자되는 용어입니다.
- 어떤 소프트웨어 언어에서 OOP 를 지원한다는 이야기는 그 언어에서 객체를 선언하고 이용하는 방법을 지원한다는 이야기이며, OOP 문법이라는 것은 객체를 선언하고 이용하기 위한 언어의 문법을 의미합니다.

객체란

- 자바스크립트에서는 객체를 만드는 방법이 크게 3가지가 있습니다.

- 객체 리터럴
- 함수 생성자
- 클래스

객체란

내장 객체란?

- 개발자가 직접 객체를 선언해서 필요한 객체를 만들 수도 있고 이미 존재하는 객체를 이용할 수도 있습니다.
- 애플리케이션을 개발하면서 이미 존재하는 객체를 내장 객체라고 합니다.
- 내장 객체는 애플리케이션을 개발하기 위한 소프트웨어 언어에서 제공되는 객체일 수도 있고 그 애플리케이션을 실행시켜 주는 플랫폼에서 제공되는 객체일 수도 있습니다.

- 자바스크립트 소프트웨어 언어에서 제공되는 객체일 수도 있으며 프런트 웹 애플리케이션이라면 브라우저에서 제공되는 객체일 수도 있습니다.
- 반대로 백앤드 애플리케이션이라면 Node.js 에서 제공되는 객체일 수도 있습니다.

Array

배열 이란

- 배열은 여러 개의 데이터를 하나의 변수로 활용하기 위한 프로그래밍 기법이며 자바스크립트에서는 Array 타입의 객체입니다.

- 배열을 이용한다는 것은 객체를 선언하고 그 객체에 여러 개의 데이터를 담아 이용하겠다는 의미입니다.

Array

배열 객체 선언 – [ ] 표기법

- 배열 객체를 선언하고 싶다면 대괄호([ ])을 이용하면 됩니다.
- [ ] 으로 배열 객체를 선언하면서 콤마(,)를 구분자로 여러 데이터를 [ ] 안에 추가합니다.

[ ] 으로 배열 객체 선언

```
1   let products1 = ['나이키 운동화', '아식스 가방']
2   let products2 = []
3
4   console.log(products1 instanceof Array)//true
5   console.log(products2 instanceof Array)//true
```

Array

배열 객체 선언 – Array 생성자 이용

- 배열 객체를 선언하는 또다른 방법은 Array 생성자를 이용하는 방법입니다.

Array 생성자로 배열 객체 선언

```
1   let products1 = new Array('나이키 운동화', '아식스 가방')
2   let products2 = new Array()
3   console.log(products1 instanceof Array)//true
4   console.log(products2 instanceof Array)//true
```

Array

배열 데이터 개수 파악

- 배열에는 여러건의 데이터가 담기는데 경우에 따라 몇 개의 데이터가 저장되어 있는 것인지 알아야 하는 경우가 있습니다.
- 이를 위해 length 프로퍼티를 제공합니다.

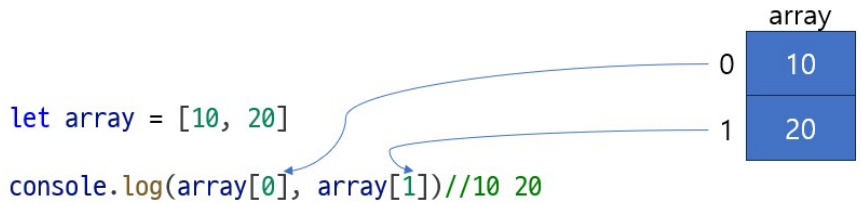
length 프로퍼티

```
1   let array = [10, 20]
2   console.log(array.length)//2
3
4   for(let i = 0; i<array.length; i++){
5     console.log(array[i])
6   }
7   //10
8   //20
```

Array

배열 데이터 획득

- 배열의 데이터를 획득하는 가장 기본적인 방법은 [ ]에 데이터의 위치인 인덱스를 명시해서 획득하는 것입니다.



Array

배열 데이터 획득 – forEach()

- 배열에 데이터를 저장된 순서대로 하나하나 획득해서 로직을 실행시키려면 forEach() 함수를 이용할 수도 있습니다.
- forEach() 의 매개변수에는 함수를 지정해야 하며, 배열의 개수만큼 함수가 반복적으로 호출됩니다.
- 함수가 호출될대 첫번째 매개변수에는 배열의 데이터, 두번째 매개변수에는 데이터의 인덱스 값이 전달되게 됩니다.

```
array.forEach((data, index) => {
  console.log(`array[${index}] = ${data}`)
})
```

Array

배열 데이터 수정

- 배열에 저장된 데이터를 수정하는 가장 기본적인 방법은 [ ] 로 데이터의 위치를 지정하고 그 위치에 새로운 값을 대입해서 수정하는 방법입니다.

```
array[0] = 100
array[1] = 200
```

배열 데이터 추가

- push() 함수를 이용해 배열에 데이터를 추가하면 배열의 마지막 위치에 데이터가 추가되게 됩니다.

```
array.push(30)
array.push(40)
```

Array

배열의 함수들

함수	설명
concat()	두 배열을 합쳐 하나의 배열을 만드는 함수
join()	배열 데이터를 구분자로 연결해 문자열로 만드는 함수
push()	배열에 데이터를 추가하는 함수
unshift()	배열에 맨 앞에 데이터를 추가하는 함수
pop()	배열에서 데이터를 제거하는 함수
shift()	맨 앞의 데이터를 제거하는 함수
splice()	지정된 위치의 데이터 삭제, 변경, 추가
slice()	특정 위치의 데이터 획득
forEach()	배열의 데이터 개수 만큼 함수 반복 실행
filter()	조건에 만족하는 배열 데이터만 추출
every()	배열의 데이터가 특정 조건에 모두 만족하는지 판단
map()	배열의 데이터로 함수를 실행, 반환 값을 모아서 배열로 만드는 함수

Array

배열의 함수 – concat()

- concat() 함수는 두 배열을 결합시켜 새로운 배열을 만들어 주는 함수입니다.

```
let array3 = array1.concat(array2)
```

배열의 함수 – join()

- 배열의 데이터를 하나로 묶어서 문자열로 만드는 함수입니다.
- 각 데이터를 구분할 문자열을 join() 함수의 매개변수로 구분해 줍니다.

```
let result = array.join('-')
```

Array

배열의 함수 – push(), unshift()

- 배열에 데이터를 추가하는 함수는 push() 와 unshift() 가 있습니다.
- 두 함수의 차이는 추가되는 위치입니다. push() 함수는 마지막 위치에 데이터가 추가되며 unshift() 는 맨 처음 위치에 데이터가 추가됩니다.
- push(), unshift() 함수의 매개변수에 지정한 데이터가 배열에 추가되는데 한꺼번에 여러 개의 데이터를 추가할 수도 있습니다.

```
array.push(300, 400)
array.unshift(500, 600)
```

Array

배열의 함수 – pop(), shift()

- 배열에 있는 데이터를 제거하기 위해 pop(), shift() 함수가 제공됩니다.
- 두 함수의 차이는 제거되는 데이터의 위치로 pop() 함수는 배열의 마지막 데이터가 제거되며 shift() 는 맨 앞의 데이터가 제거 됩니다.

pop(), shift() 함수

```
1   let array = [10, 20, 30, 40]
2   array.pop()
3   console.log(array)//[10, 20, 30]
4   array.shift()
5   console.log(array)//[20, 30]
```

Array

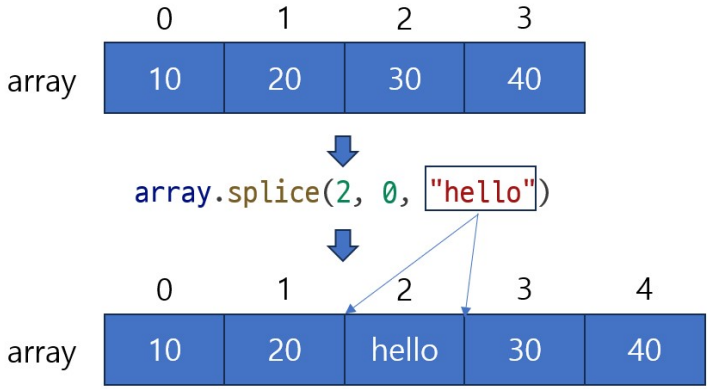
배열의 함수 – splice()

- 중간 특정 위치에 데이터를 추가하거나, 제거 혹은 교체해야 하는 경우가 있습니다. 이때 사용되는 함수가 splice() 입니다.
- splice() 함수는 매개변수로 배열의 특정 위치를 지정하고 그 위치에 데이터를 추가, 제거, 교체할 수 있는 함수입니다.

Array

배열의 함수 – splice() – 데이터 추가

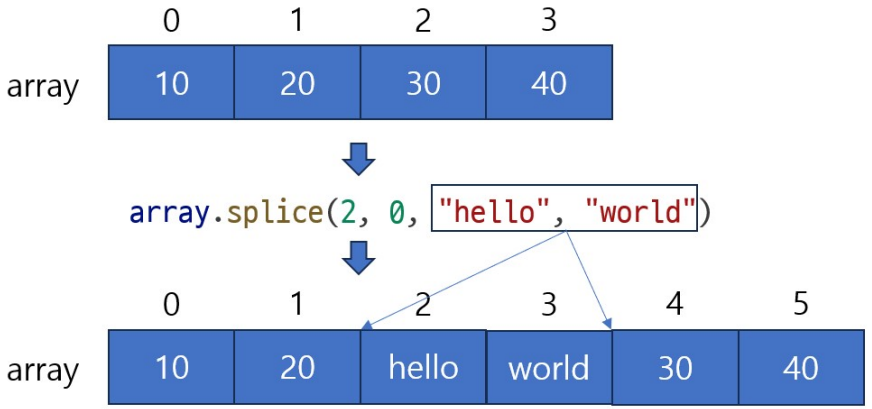
- splice() 함수의 첫번째 매개변수에 데이터가 추가되어야 하는 위치를 인덱스로 지정하고 두번째 매개변수에 0 을 지정하니다.
- splice() 함수의 두번째 매개변수는 교체하고자 하는 데이터 개수인데 이 값을 0으로 지정하면 교체하는 데이터 개수가 0 임으로 기존의 데이터는 변경되지 않고 데이터 추가만 되게 됩니다.



Array

배열의 함수 – splice() – 데이터 여러 개 추가

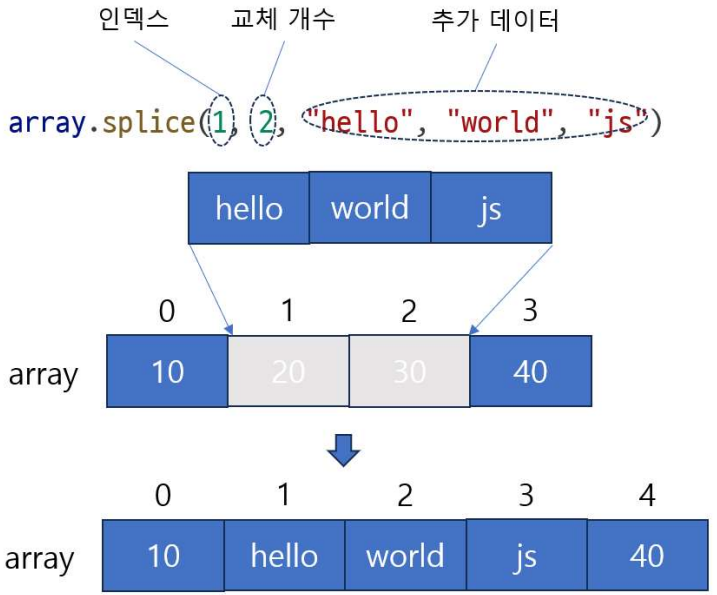
- splice() 함수를 이용해 특정 위치에 데이터 여러 개를 한꺼번에 추가할 수 있습니다.
- 이때는 추가하고자 하는 데이터를 세번째 매개변수부터 여러 개를 나열하면 됩니다.



Array

배열의 함수 – splice() – 데이터 교체

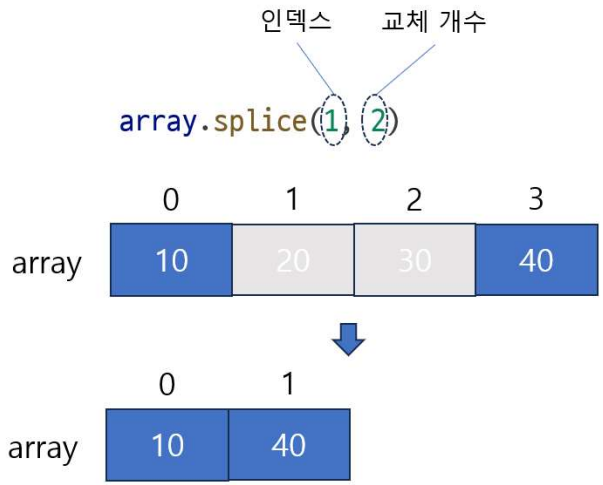
- splice() 의 두번째 매개변수가 교체되는 데이터 개수이며 1로 지정하면 1개를 교체하게 되며 2라고 지정하면 2개를 한꺼번에 교체하게 됩니다.



Array

배열의 함수 – splice() – 데이터 삭제

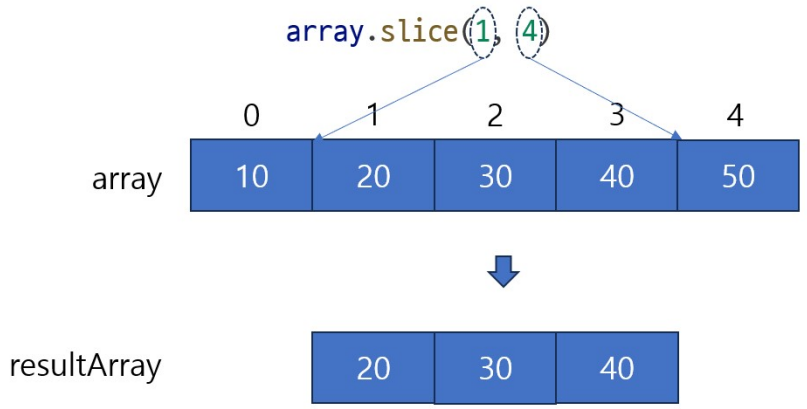
- splice() 로 데이터를 삭제 하기 위해서는 인덱스 위치와 삭제하고자 하는 데이터 개수만 지정하고 추가 데이터를 지정하지 않으면 됩니다. 신규로 추가되는 데이터가 없음으로 기존 데이터만 삭제되게 됩니다.



Array

배열의 함수 – slice()

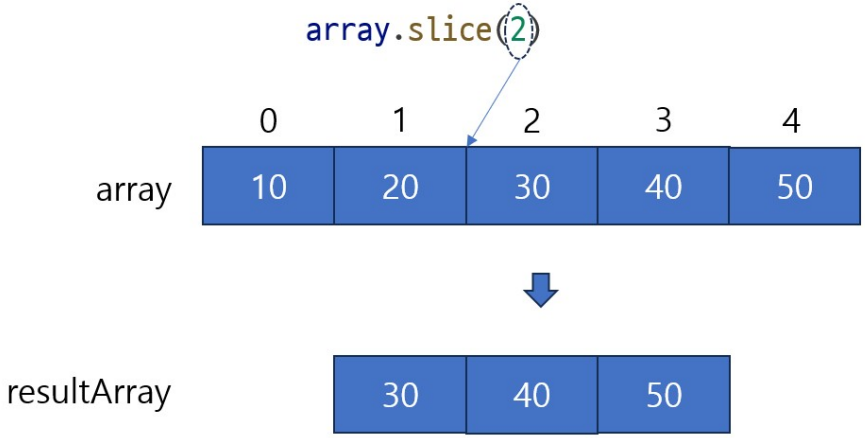
- slice() 함수는 배열의 데이터를 획득할 때 사용됩니다.
- slice() 를 사용하면 두개의 인덱스를 지정하고 그 인덱스 사이의 데이터 여러 개여 한꺼번에 획득할 수 있습니다.
- slice() 에 의해 획득되는 데이터가 여러 개임으로 slice() 는 획득한 데이터를 배열로 반환합니다.



Array

배열의 함수 – slice()

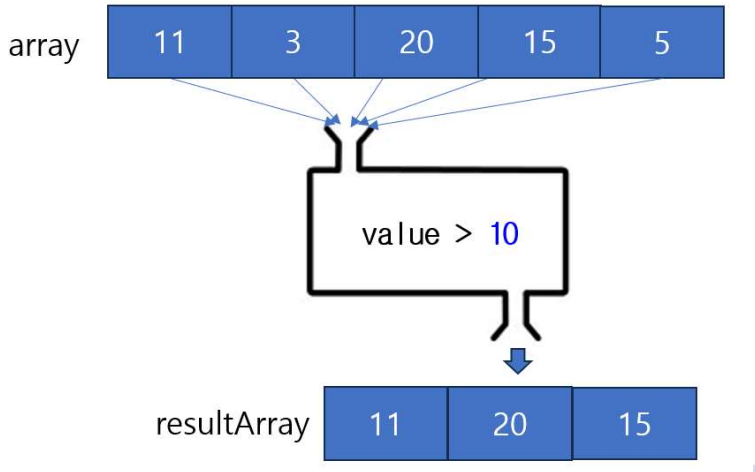
- 만약 slice() 함수를 이용하면서 매개변수를 1개만 지정하게 되면 그 인덱스 위치의 데이터부터 오른쪽의 모든 데이터가 획득되게 됩니다.



Array

배열의 함수 – filter()

- filter() 함수는 배열의 데이터 중 조건에 맞는 데이터만 추출하고자 할 때 사용하는 함수입니다.
- filter() 의 매개변수에 함수가 지정되고 이렇게 되면 배열의 개수 만큼 순차적으로 매개변수에 지정된 함수가 호출되게 됩니다.
- 매개변수의 함수에서 true 를 반환하는 데이터만 추출해서 결과를 반환하게 됩니다.



Array

배열의 함수 – every()

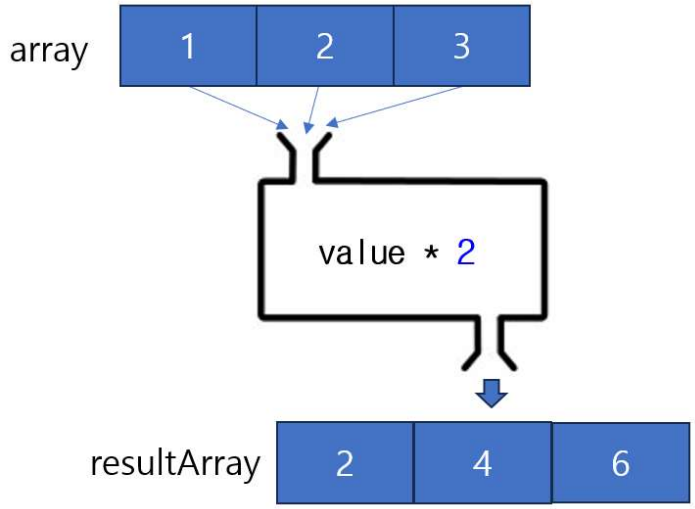
- every() 함수는 배열의 데이터가 특정 조건에 모두 만족하는 데이터인지를 판단하기 위해서 사용이 됩니다.
- 매개변수에 함수를 지정하고 그 함수에 배열의 데이터가 순차적으로 전달되며 결과는 true/false 가 반환됩니다.
- 함수의 반환 값이 모두 true 이면 최종 every() 함수의 최종 결과 값이 true 이며 하나라도 false 를 반환하면 every() 함수의 최종 결과 값은 false 가 됩니다.

```
let result = array.every((value) => {
  return value > 4
})
```

Array

배열의 함수 – map()

- map() 함수는 배열의 데이터로 특정 로직을 실행하고 그 결과를 다시 모아서 반환할 때 사용됩니다.
- 배열의 데이터가 3개라면 최종 반환되는 배열의 데이터도 3개입니다.



Date

- 시간/날짜 데이터는 애플리케이션에서 이용할 때 다양한 형태로 이용되며 다양한 연산이 필요한데 이를 도와주기 위한 자바스크립트 내장 객체가 Date 이며 이 Date 의 다양한 함수를 이용해 시간/날짜를 이용하게 됩니다.

Date() 객체 이용하기

- Date 객체는 new 를 이용하여 생성합니다. 생성된 Date 객체의 데이터는 현재 시간입니다.

```
let date1 = new Date()
console.log(date1.toString())
//Tue Apr 23 2024 11:32:02 GMT+0900 (한국 표준시)
```

UTC/GMT

- 소프트웨어에서 시간/날짜 데이터를 활용하기 위해서는 UTC, GMT 라는 용어가 먼저 정리되어야 합니다.
- UTC, GMT 는 자바스크립트에서만 이용되는 용어는 아니며 대부분의 소프트웨어에서 사용하고 있습니다.
- 먼저 UTC 를 보면 Coordinated universal Time 의 약어이며 한국어로는 '협정세계시' 정로로 번역되어 불리웁니다.
- 세계 각국의 시간은 다릅니다. 즉 서울이 1월 1일 자정일 때 뉴욕도 1월 1일 자정이 아닙니다.
- 그럼으로 한국에서만 시간을 이야기 한다면 상관없지만 서울에 있는 사람과 뉴욕에 있는 사람이 대화를 나누다가 "9시에 봅시다" 라고 이야기 하면 이 9시가 서울의 9시인지? 뉴욕의 9시인지가 불 분명해집니다.
- 그럼으로 전세계 각국에서 시간을 표현할 때 통일된 기준 시간이 있어야 하고 이를 위해 정의된 개념이 UTC입니다. UTC 는 영국의 그리니치 천문대를 기준으로 합니다. 영국의 시간이 UTC+0:00 이라면 대한민국은 9시간이 빠름으로 UTC+9:00 이며 뉴욕은 영국보다 5시간 느림으로 UTC-5:00 입니다.
- GMT(Greenwich Mean Time) 는 UTC 보다 더 오래된 개념이며 초의 소수점 단위에서 약간의 차이가 있지만 거의 동일한 개념이며 일반적으로 GMT, UTC 가 혼용되어 사용되고 있습니다.

타임스템프

- 어떤 특정 순간의 시간/날짜를 표현하는 데이터입니다.
- 일반적으로 사람들은 2000년 1월 1일 12시 00 분 00 초라고 이야기 하지만 소프트웨어 적으로 년/월/일/시/분/초를 따로 데이터적으로 유지하기 불편하며 어떤 날짜가 빠른지를 계산하기도 불편합니다.
- 그래서 일관된 하나의 데이터로 표현하고 이 데이터로 둘 이상의 시각을 비교하거나 기간을 계산하는데 이 때 사용되는 것이 타임스템프(Timestamp) 입니다.
- 타임스템프는 UTC 기준 1970년 1월 1일 자정부터 경과된 밀리초 값입니다.

Date

Date() 객체 이용하기

- 시간을 한국어를 적용해서 출력하고 싶다면 toLocaleString() 이라고 지정하면 됩니다.

```
console.log(date1.toLocaleString())
//2024. 4. 23. 오전 11:32:02
```

- new Date() 로 객체를 생성하면 현재 시간의 데이터를 가지는 객체가 생성되는데 만약 특정 시간을 설정하고 싶다면 매개변수에 시간을 지정해 주면 됩니다.

시간 지정해서 Date 객체 생성

```
1   let date2 = new Date("2024-10-09T10:10:10")
```

```
let date3 = new Date(2024, 10-1, 9, 10, 10, 10);
```

Date

원하는 데이터만 추출

- Date 객체가 가지는 시간 날짜 데이터중 원하는 데이터만 추출하기 위한 함수를 제공합니다.

- getFullYear() : 년도를 반환
- getMonth() : 월을 반환, 0이 1월
- getDate() : 일을 반환
- getDay() : 요일을 반환, 0이 일요일
- getHours() : 시간을 반환
- getMinutes() : 분을 반환
- getSeconds() : 초를 반환
- getTime(): 타임스템프값 반환

Date

두 시간 비교하기

- Date 객체를 비교해 어느 객체의 시간이 더 이전 시간인지를 판단하는 경우도 많습니다.
- 두 Date 객체의 데이터를 비교하는 여러가지 방법이 있지만 타임스템프 값을 비교해서 어느 시간이 더 이른 시간인지를 판단할 수 있습니다.

```
if(regDate.getTime() < eventStartDate.getTime()){
```

Math

- 수학과 관련된 다양한 작업이 필요하며 이를 도와주기 위한 Math 라는 내장객체를 제공합니다.

- Math.PI – 원주율 값을 가지는 프로퍼티
- Math.abs() – 절대값을 반환하는 함수
- Math.ceil() – 올림 값을 반환하는 함수
- Math.floor() – 내림 값을 반환하는 함수
- Math.round() – 반올림 값을 반환하는 함수
- Math.max() – 최대값을 구하는 함수
- Math.min() – 최소값을 구하는 함수
- Math.pow() – 제곱근을 구하는 함수
- Math.random() – 난수를 구하는 함수

Math

- Math 객체의 함수 중 가장 많이 사용되는 함수는 난수 발생 함수입니다.
- 난수 발생은 Math.random() 함수를 이용하며 발생하는 난수는 0 과 1 사이의 실수 값입니다.

```
console.log(Math.random())//0.6958603372854415
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare