7강

코어객체(Date)

JavaScript

➢ 코어 객체

- 코어 객체는 기본 객체로서 표준이다.
- JavaScript 언어가 실행되는 어디서나 사용 가능한 기본 객체이며, 표준 객체이다.
- 언어 자체가 제공하는 기본 내장 객체(Built-in Object)를 의미한다.
- **Array, Date, String, Math** 타입 등이며, 웹 페이지 JavaScript 코드에서, 혹은 서버에서 사용 가능하다.
- 코어 객체는 **new 키워드**로 생성한다.

```
var today = new Date();          // 시간 정보를 다루는 Date 타입의 객체 생성
var msg = new String("Hello"); // "Hello" 문자열을 담은 String 타입의 객체 생성
```

- 객체가 생성되면 객체 내부에 프로퍼티와 메소드들이 존재한다.
- 객체와 멤버 사이에 **점(.)** 연산자 이용하여 작성한다.

```
obj.프로퍼티 = 값;              // 객체 obj의 프로퍼티 값 변경
변수 = obj.프로퍼티;            // 객체 obj의 프로퍼티 값 알아내기
obj.메소드(매개변수 값들);      // 객체 obj의 메소드 호출
```

JavaScript

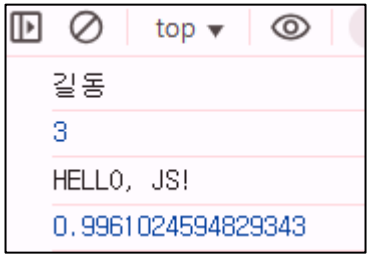
➤ 코어 객체 간단한 예시

```
// Object: 객체 생성
    let obj = new Object();
    obj.name = '길동';
    console.log(obj.name); // "길동"

    // Array: 배열 사용
    let arr = [1, 2, 3];
    console.log(arr.length); // 3

    // String: 문자열 처리
    let str = 'Hello, JS!';
    console.log(str.toUpperCase()); // "HELLO, JS!"

    // Math: 수학 함수
    console.log(Math.random()); // 0~1 사이 랜덤 숫자
```



JavaScript

➢ innerHTML과 innerText와 textContent 차이

- innerHTML : innerHTML 속성은 말 그대로 HTML을 그대로 반환한다.
- innerText : innerText 속성은 그 요소와 그 자손의 랜더링 된 텍스트 콘텐츠를 나타낸다. 즉, 눈에 보이는 텍스트만 반환한다는 것이다.
- textContent : **textContent**는 **<script>**와 **<style>**을 비롯한 모든 요소의 텍스트 콘텐츠를 가져온다

```
<div id="box">
  <style>#source {color: blue}</style>
  <p>Hi there!</p>
  <span style="display: none">hidden message</span>
</div>
 <script>
   let box = document.querySelector('#box');
   console.log('textContent : '+box.textContent);
   console.log('innerText : '+box.innerText);
```

```
textContent :
      #source {color: blue}
      Hi there!
      hidden message

innerText : Hi there!
```

HTML 사용빈도

JavaScript

➢ Date 객체

- **Date 객체는** 날짜와 시간(년, 월, 일, 시, 분, 초, 밀리초(천분의 1초(millisecond, ms)))을 위한 메소드를 제공하는 빌트인 객체이면서 생성자 함수이다.
- Date 생성자 함수로 생성한 Date 객체는 내부적으로 숫자 값을 갖는다. 이 값은 1970년 1월 1일 00:00(UTC)을 기점으로 현재 시간까지의 밀리초를 나타낸다.
- UTC(협정 세계시: Coordinated Universal Time)는 GMT(그리니치 평균시: Greenwich Mean Time)로 불리기도 하는데 UTC와 GMT는 초의 소숫점 단위에서만 차이가 나기 때문에 일상에서는 혼용되어 사용된다. 기술적인 표기에서는 UTC가 사용된다.
- KST(Korea Standard Time)는 UTC/GMT에 9시간을 더한 시간이다. 즉, KST는 UTC/GMT보다 9시간이 빠르다. 예를 들어, UTC 00:00 AM은 KST 09:00 AM이다.
- 현재의 날짜와 시간은 자바스크립트 코드가 동작한 시스템의 시계에 의해 결정된다. 시스템 시계의 설정(timezone, 시간)에 따라 서로 다른 값을 가질 수 있다.

JavaScript

➢ Date 객체

Date 객체는 생성자 함수이다. **Date 생성자 함수는 날짜와 시간을 가지는 인스턴스를 생성한다**. 생성된 인스턴스는 기본적으로 현재 날짜와 시간을 나타내는 값을 가진다. 현재 날짜와 시간이 아닌 다른 날짜와 시간을 다루고 싶은 경우, Date 생성자 함수에 명시적으로 해당 날짜와 시간 정보를 인수로 지정한다. Date 생성자 함수로 객체를 생성하는 방법은 4가지가 있다.

1. New Date

인수를 전달하지 않으면 현재 날짜와 시간을 가지는 인스턴스를 반환한다.

```
const today = new Date();
console.log(`오늘날짜 ${today}`);
```

```
오늘날짜 Fri Sep 08 2023 14:35:51 GMT+0900 (한국 표준시)   date_test.html:13
```

2. New Date(milliseconds)

인수로 숫자 타입의 밀리초를 전달하면 **1970년 1월 1일 00:00(UTC)을 기점**으로 **인수로 전달된 밀리초만큼 경과한 날짜와 시간을 가지는 인스턴스를 반환**한다.

JavaScript

```
// 86400000ms는 1day를 의미한다.
// 1s = 1,000ms
// 1m = 60s * 1,000ms = 60,000ms
// 1h = 60m * 60,000ms = 3,600,000ms
// 1d = 24h * 3,600,000ms = 86,400,000ms
const date = new Date(86400000);
console.log(date);
```

```
Fri Jan 02 1970 09:00:00 GMT+0900 (한국 표준시)
```

3. New Date(dateString)

인수로 날짜와 시간을 나타내는 문자열을 전달하면 지정된 날짜와 시간을 가지는 인스턴스를 반환한다. 이때 인수로 전달한 문자열은 Date.parse 메소드에 의해 해석 가능한 형식이어야 한다.

```
let date = new Date('September 08,2023 14:50:00');
console.log(date);

date = new Date('2023/09/08/14:50:00');
console.log(date);
```

```
Fri Sep 08 2023 14:50:00 GMT+0900 (한국 표준시)
Fri Sep 08 2023 14:50:00 GMT+0900 (한국 표준시)
```

JavaScript

4. New Date(year, month[, day, hour, minute, second, millisecond])

인수로 **년, 월, 일, 시, 분, 초, 밀리초를 의미하는 숫자를 전달하면 지정된 날짜와 시간을 가지는 인스턴스를 반환한다**. 이때 **년, 월은 반드시 지정하여야 한다**. 지정하지 않은 옵션 정보는 0 또는 1으로 초기화된다.

인수	내용
year	1900년 이후의 년
month	월을 나타내는 0 ~ 11까지의 정수 (주의: 0부터 시작, 0 = 1월)
day	일을 나타내는 1 ~ 31까지의 정수
hour	시를 나타내는 0 ~ 23까지의 정수
minute	분을 나타내는 0 ~ 59까지의 정수
second	초를 나타내는 0 ~ 59까지의 정수
millisecond	밀리초를 나타내는 0 ~ 999까지의 정수

※ 년, 월을 지정하지 않은 경우 1970년 1월 1일 00:00(UTC)을 가지는 인스턴스를 반환한다.

JavaScript

4. New Date(year, month[, day, hour, minute, second, millisecond])

```
// 월을 나타내는 8는 9월을 의미한다.
// 2023/9/1/00:00:00
let date = new Date(2023, 8);
console.log(date);

// 월을 나타내는 8는 9월을 의미한다.
// 2023/9/8/15:04:00:00
date = new Date(2023, 8, 8, 15, 4, 0, 0);
console.log(date);

// 가독성이 훨씬 좋다.
date = new Date('2023/9/08/15:04:00:00');
console.log(date);
```

```
Fri Sep 01 2023 00:00:00 GMT+0900 (한국 표준시)
Fri Sep 08 2023 15:04:00 GMT+0900 (한국 표준시)
Fri Sep 08 2023 15:04:00 GMT+0900 (한국 표준시)
```

5. Date 생성자를 new 연산자 없이 호출

Date 생성자 함수를 new 연산자없이 호출하면 인스턴스를 반환하지 않고 **결과값을 문자열로 반환한다**.

```
let today = Date();
console.log(typeof today, today);
```

```
string Fri Sep 08 2023 15:11:54 GMT+0900 (한국 표준시)
```

JavaScript

➢ Date 메서드

1. Date.now()

1970년 1월 1일 00:00:00(UTC)을 기점으로 현재 시간까지 경과한 밀리초를 숫자로 반환한다.

```
let now = Date.now();
console.log((now/86400000)/365);
//  now => 1,694,153,925,531
// 19608.264520671295 일 // 55년 => 1970+55 : 2025년
```

2. Date.pasre

1970년 1월 1일 00:00:00(UTC)을 기점으로 **인수로 전달된 지정 시간(new Date(dateString)의 인수와 동일한 형식)까지의 밀리초를 숫자로 반환한다.**

```
let d = Date.parse('Jan 2, 1970 00:00:00 UTC'); // UTC
console.log(d); // 86400000

d = Date.parse('Jan 2, 1970 09:00:00'); // KST
console.log(d); // 86400000

d = Date.parse('1970/01/02/09:00:00'); // KST
console.log(d); // 86400000
```

```
86400000
86400000
86400000
```

JavaScript

➢ Date 메서드

3. Date.UTC

1970년 1월 1일 00:00:00(UTC)을 기점으로 인수로 전달된 지정 시간까지의 밀리초를 숫자로 반환한다.

Date.UTC 메소드는 new Date(year, month[, day, hour, minute, second, millisecond])와 같은 형식의 인수를 사용해야 한다. Date.UTC 메소드의 인수는 local time(KST)가 아닌 UTC로 인식된다..

```
let d = Date.UTC(1970, 0, 2);
console.log(d); // 86400000

d = Date.UTC('1970/1/2');
console.log(d); // NaN
```

4. Date.prototype.getFullYear

년도를 나타내는 4자리 숫자를 반환한다.

```
let today = new Date();
let year = today1.getFullYear();

console.log(today);
console.log(year);
```

```
Fri Sep 08 2023 15:42:24 GMT+0900 (한국 표준시)
2023
```

JavaScript

➢ Date 메서드

5. Date.prototype.setFullYear

년도를 나타내는 4자리 숫자를 설정한다. 년도 이외 월, 일도 설정할 수 있다.

dateObj.setFullYear(year[, month[, day]])

```
const today = new Date();
// 년도 지정
today.setFullYear(2000);

let year = today.getFullYear();
console.log(today); // Fri Sep 08 2000 15:46:24 GMT+0900 (한국 표준시)
console.log(year);  // 2000

// 년도 지정
today.setFullYear(1900, 0, 1);

year = today.getFullYear();
console.log(today); // Mon Jan 01 1900 17:42:40 GMT+0827 (한국 표준시)
console.log(year);  // 1900
```

```
Fri Sep 08 2000 15:47:30 GMT+0900 (한국 표준시)
2000
Mon Jan 01 1900 15:47:30 GMT+0827 (한국 표준시)
1900
```

JavaScript

6. Date.prototype.getMonth

월을 나타내는 0 ~ 11의 정수를 반환한다. 1월은 0, 12월은 11이다.

```
const today = new Date();
const month = today.getMonth();

console.log(today); //Fri Sep 08 2023 15:52:43 GMT+0900 (한국 표준시)
console.log(month); // 8
```

7. Date.prototype.setMonth

월을 나타내는 0 ~ 11의 정수를 설정한다. 1월은 0, 12월은 11이다. 월 이외 일도 설정할 수 있다.

dateObj.setMonth(month[, day])

```
const today = new Date();

//월을 지정
today.setMonth(0);

let month = today.getMonth();
console.log(today);
console.log(month);

//월, 일 지정
today.setMonth(11,1);
month = today.getMonth();
console.log(today);
console.log(month);
```

```
Sun Jan 08 2023 15:58:58 GMT+0900 (한국 표준시)
0
Fri Dec 01 2023 15:58:58 GMT+0900 (한국 표준시)
11
```

JavaScript

8. Date.prototype.getDate

날짜(1 ~ 31)를 나타내는 정수를 반환한다.

```
const today = new Date();
const date = today.getDate();

console.log(today); // Fri Sep 08 2023 16:04:50 GMT+0900 (한국 표준시)
console.log(date);  // 8
```

9. Date.prototype.setDate

날짜(1 ~ 31)를 나타내는 정수를 설정한다.

```
const today = new Date();

// 날짜 지정
today.setDate(1);

const date = today.getDate();
console.log(today); // Fri Sep 01 2023 16:06:10 GMT+0900 (한국 표준시)
console.log(date);  // 1
```

JavaScript

10. Date.prototype.getDay

요일(0 ~ 6)를 나타내는 정수를 반환한다.

요일	반환값
일요일	0
월요일	1
화요일	2
수요일	3
목요일	4
금요일	5
토요일	6

```
const today = new Date();
const day = today.getDay();

console.log(today); // Fri Sep 08 2023 16:09:38 GMT+0900 (한국 표준시)
console.log(day);   // 5
```

JavaScript

11. Date.prototype.getHours

시간(0 ~ 23)를 나타내는 정수를 반환한다.

```
const today = new Date();
const hours = today.getHours();

console.log(today); // Fri Sep 08 2023 16:11:45 GMT+0900 (한국 표준시)
console.log(hours); // 16
```

12. Date.prototype.setHours

시간(0 ~ 23)를 나타내는 정수를 설정한다. 시간 이외 분, 초, 밀리초도 설정할 수 있다.
dateObj.setHours(hour[, minute[, second[, ms]]])

```
const today = new Date();

// 시간 지정
today.setHours(7);

let hours = today.getHours();
console.log(today); // Fri Sep 08 2023 07:13:52 GMT+0900 (한국 표준시)
console.log(hours); // 7

// 시간/분/초/밀리초 지정
today.setHours(0, 0, 0, 0); // 00:00:00:00

hours = today.getHours();
console.log(today); // Fri Sep 08 2023 00:00:00 GMT+0900 (한국 표준시)
console.log(hours); // 0
```

JavaScript

13. Date.prototype.getMinutes

분(0 ~ 59)를 나타내는 정수를 반환한다.

```
const today = new Date();
const minutes = today.getMinutes();

console.log(today);   // Fri Sep 08 2023 16:17:34 GMT+0900 (한국 표준시)
console.log(minutes); // 17
```

14. Date.prototype.setMinutes

분(0 ~ 59)를 나타내는 정수를 설정한다. 분 이외 초, 밀리초도 설정할 수 있다.
dateObj.setMinutes(minute[, second[, ms]])

```
const today = new Date();

// 분 지정
today.setMinutes(50);

let minutes = today.getMinutes();
console.log(today);   // Fri Sep 08 2023 16:50:18 GMT+0900 (한국 표준시)
console.log(minutes); // 50

// 분/초/밀리초 지정
today.setMinutes(5, 10, 999); // HH:05:10:999

minutes = today.getMinutes();
console.log(today);   // Fri Sep 08 2023 16:05:10 GMT+0900 (한국 표준시)
console.log(minutes); // 5
```

JavaScript

15. Date.prototype.getSeconds

초(0 ~ 59)를 나타내는 정수를 반환한다.

```
const today = new Date();
const seconds = today.getSeconds();

console.log(today);   // Fri Sep 08 2023 16:22:02 GMT+0900 (한국 표준시)
console.log(seconds); // 2
```

16. Date.prototype.setSeconds

초(0 ~ 59)를 나타내는 정수를 설정한다. 초 이외 밀리초도 설정할 수 있다.
dateObj.setSeconds(second[, ms])

```
const today = new Date();

// 초 지정
today.setSeconds(30);

let seconds = today.getSeconds();
console.log(today); // Fri Sep 08 2023 16:23:30 GMT+0900 (한국 표준시)
console.log(seconds); // 30

// 초/밀리초 지정
today.setSeconds(10, 0); // HH:MM:10:000

seconds = today.getSeconds();
console.log(today); // Fri Sep 08 2023 16:23:10 GMT+0900 (한국 표준시)
console.log(seconds); // 10
```

JavaScript

17. Date.prototype.getMilliseconds

주어진 날짜의 현지 시간 기준 밀리초를 나타내는 0에서 999 사이의 정수.

```
const today = new Date();
const ms = today.getMilliseconds();

console.log(today); // Fri Sep 08 2023 16:26:18 GMT+0900 (한국 표준시)
console.log(ms); // 110
```

18. Date.prototype.setMilliseconds

밀리초(0 ~ 999)를 나타내는 정수를 설정한다.

```
const today = new Date();

// 밀리초 지정
today.setMilliseconds(123);

const ms = today.getMilliseconds();
console.log(today); // Fri Sep 08 2023 16:27:43 GMT+0900 (한국 표준시)
console.log(ms); // 123
```

JavaScript

19. Date.prototype.getTime

1970년 1월 1일 00:00:00(UTC)를 기점으로 현재 시간까지 경과된 밀리초를 반환한다.

```
const today = new Date();
const time = today.getTime();

console.log(today); // Fri Sep 08 2023 16:30:12 GMT+0900 (한국 표준시)
console.log(time); // 1694158232509
```

20. Date.prototype.setTime

1970년 1월 1일 00:00:00(UTC)를 기점으로 현재 시간까지 경과된 밀리초를 설정한다.
dateObj.setTime(time)

```
const today = new Date();

// 1970년 1월 1일 00:00:00(UTC)를 기점으로 현재 시간까지 경과된 밀리초 지정
today.setTime(86400000); // 86400000 === 1day

const time = today.getTime();
console.log(today); // Fri Jan 02 1970 09:00:00 GMT+0900 (한국 표준시)
console.log(time); // 86400000
```

JavaScript

21. Date.prototype.getTimezoneOffset

UTC와 지정 로케일(Locale) 시간과의 차이를 분 단위로 반환한다.

```
const today = new Date();
const x = today.getTimezoneOffset() / 60; // -9

console.log(today); // Fri Sep 08 2023 16:34:20 GMT+0900 (한국 표준시)
console.log(x); // -9
```

※ KST(Korea Standard Time)는 UTC에 9시간을 더한 시간이다. 즉, UTC = KST - 9h이다.

22. Date.prototype.toDateString

사람이 읽을 수 있는 형식의 문자열로 날짜를 반환한다.

```
const d = new Date('2023/9/08/18:30');

console.log(d.toString()); // Fri Sep 08 2023 18:30:00 GMT+0900 (한국 표준시)
console.log(d.toDateString()); // Fri Sep 08 2023
```

23. Date.prototype.toTimeString

사람이 읽을 수 있는 형식의 문자열로 시간을 반환한다.

```
const d = new Date('2023/9/08/18:30');

console.log(d.toString()); // Fri Sep 08 2023 18:30:00 GMT+0900 (한국 표준시)
console.log(d.toTimeString()); // 18:30:00 GMT+0900 (한국 표준시)
```

JavaScript

24. Date.prototype.toLocalDateString

생성된 Date 객체에서 날짜 부분을 현재 지역 표기법으로 변환하여 가져옵니다.
toLocaleDateString() 메서드는 locales(로케일 - 언어와 국가)과 options(옵션)을 파라미터(Parameter) 전달받아 현재 지역 표기법으로 변환합니다.

```
const today = new Date();
console.log(today.toDateString());//Fri Sep 08 2023
```

25. Date.prototype.toLocaleTimeString

생성된 Date 객체에서 시간 부분을 현재 지역 표기법으로 변환하여 가져옵니다.

```
const today = new Date();
console.log(today.toLocaleTimeString()); //오후 5:00:31
```

26. setTimeOut()

- 어떤 코드를 바로 실행하지 않고 일정 시간 기다린 후 실행해야하는 경우가 있다. 이럴 때는 **setTimeout()** 함수를 사용할 수 있다.
- **setTimeout()** 함수는 첫번째 인자로 실행할 코드를 담고 있는 함수를 받고, 두번째 인자로 지연 시간을 **밀리초(ms) 단위**로 받는다.

```
//1 밀리초 = 0.001 초,  10 밀리초 = 0.01 초,  2500 밀리초 = 2.5 초
setTimeout(() => console.log('2초 후에 실행됨'), 2000);
```

Date 객체 간단한 예제

JavaScript

```
<h3>Date 객체로 현재 시간 알아내기</h3>
<hr />
<script>
  var now = new Date(); // 현재 시간 값을 가진 Date 객체 생성
  var weekString = ['일', '월', '화', '수', '목', '금', '토'];
  document.write(now.getFullYear() + '년도<br>');
  document.write(now.getMonth() + 1 + '월<br>');
  document.write(now.getDate() + '일<br>');
  var week = now.getDay();
  for (let i = 0; i < weekString.length; i++) {
    if (week === i) {
      document.write(weekString[i] + '요일<br>');
    }
  }
  document.write(now.getHours() + '시<br>');
  document.write(now.getMinutes() + '분<br>');
  document.write(now.getSeconds() + '초<br>');
  document.write(now.getMilliseconds() + '밀리초<br><hr>');
  document.write(`toLocalDateString : ${now.toLocaleDateString()} <br>`);
  document.write(`toLocalTimeString : ${now.toLocaleTimeString()}`);
```

Date 객체로 현재 시간 알아내기

2023년도
9월
25일
월요일
20시
48분
57초
270밀리초

toLocalDateString : 2023. 9. 25.
toLocalTimeString : 오후 8:48:57

시작일자 : Fri Sep 01 2023 00:00:00 GMT+0900 (한국 표준시)
시작일의 요일 : 5
마지막일자 : 2023. 9. 30.
마지막일 : 30
이틀전 일자 : 2023. 8. 30.

기초
연습문제

Date 기초 연습문제

조건] 아래 주어진 1번 ~ 10번 까지 아래와 같이 button태그와 p태그를 이용하여 작성하시오.

```
<button onclick="showYear()">1.현재 연도 출력</button>
<p id ="output"></p>
```

1. 현재 연도 출력
문제: 현재 날짜를 Date 객체로 가져와서 콘솔에 "올해는 2025년입니다." 형식으로 출력하세요.

2. 오늘 요일 출력
문제: 현재 날짜를 이용하여 오늘 요일을 한글로 출력하세요.
예: "오늘은 화요일입니다."

3. 현재 시간 출력
문제: 현재 시간을 시:분:초 형식으로 출력하세요.
예: "현재 시간: 14:35:20"

4. 특정 날짜까지 남은 일수 계산
문제: 내 생일까지 남은 일수를 계산하여 출력하세요.
예: "생일까지 120일 남았습니다."

5. 하루 후, 일주일 후 날짜 출력
문제: 오늘 날짜를 기준으로 **하루 후**와 **일주일 후** 날짜를 출력하세요.

Date 기초 연습문제

6. 날짜 비교
문제: 두 날짜(2025-09-23과 2025-12-25) 중 어떤 날짜가 더 미래인지 비교하고, 미래 날짜를 출력하세요.

7. 월의 마지막 날 구하기
문제: 현재 달의 마지막 날짜(일)를 구하여 출력하세요.
예: "9월의 마지막 날은 30일입니다."

8. 특정 요일 찾기
문제: 이번 달 1일부터 시작해서 **첫 번째 금요일**이 몇 일인지 찾아 출력하세요.

9. 시간 차 계산
문제: 현재 시간과 오후 6시(18:00:00) 사이의 남은 시간을 시, 분, 초 단위로 계산하여 출력하세요.

10. 요일에 따라 메시지 출력
문제: 오늘 요일이 주말(토, 일)이면 "주말이에요!", 평일이면 "열심히 일하는 날!"을 출력하세요.

연습문제

Date 연습문제 3

예제3] 아래 주어진 조건에 만족하도록 html 문서를 작성하시오.

조건

① 해당 년월의 마지막 일자 산출하기
② 해당 년월의 시작인 1일의 요일 산출하기
③ 토요일을 만나면 줄 바꿈 하기
④ 이전, 다음버튼 클릭시 년월일 증가, 감소 시키기
⑤ 위의 ①,②,③,④번의 조건을 이용하여 달력을 작성하시오.
⑥ Function display( ) 형식의 함수 이용하여 작성하시오.
⑦ innerHTML을 이용하여 작성하시오.
⑧ 2차원 배열을 이용하여 작성하시오.
⑨ <table>태그를 이용하여 작성하시오.
⑩ 다음페이지의 **<출력 결과물>**을 보고 디자인은 알아서 작성하시오.
⑪ 저장할 파일명은 Ex_calendar.html 로 작성 하시오.

Date 연습문제

<출력 결과물>