30. Date

TABLE OF CONTENTS

표준 빌트인 객체(standard built-in object)인 Date는 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 숫자값을 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.

#1. Date 생성자 함수

Date는 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.

1.1. new Date()

Date 생성자 함수에 인수를 전달하지 않으면 현재 날짜와 시간을 가지는 인스턴스를 반환한다.

JAVASCRIPT

```
const todoy = new Date();
// → Thu Mar 26 2020 14:03:18 GMT+0900 (대한민국 표준시)
```

1.2. new Date(milliseconds)

Date 생성자 함수에 숫자 타입의 밀리초를 인수로 전달하면 1970년 1월 1일 00:00(UTC)을 기점으로 인수로 전달된 밀리초만큼 경과한 날짜와 시간을 가지는 인스턴스를 반환한다.

JAVASCRIPT

```
// KST(Korea Standard Time)는 GMT(그리니치 평균시: Greenwich Mean Time)에 9시간을 더한 시간이다.
new Date(0); // → Thu Jan 01 1970 09:00:00 GMT+0900 (대한민국 표준시)

// 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
new Date(86400000); // → Fri Jan 02 1970 09:00:00 GMT+0900 (대한민국 표준시)
```

1.3. new Date(dateString)

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

JAVASCRIPT

```
new Date('May 26, 2020 10:00:00');
// → Tue May 26 2020 10:00:00 GMT+0900 (대한민국 표준시)

new Date('2020/03/26/10:00:00');
// → Thu Mar 26 2020 10:00:00 GMT+0900 (대한민국 표준시)
```

1.4. 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.

인수는 다음과 같다.

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

```
// 월을 나타내는 2는 3월을 의미한다.
// 2020/3/1/00:00:00:00
new Date(2020, 2);
// → Sun Mar 01 2020 00:00:00 GMT+0900 (대한민국 표준시)

// 월을 나타내는 2는 3월을 의미한다.
// 2020/3/26/10:00:00:00
new Date(2020, 2, 26, 10, 00, 00, 0);
// → Thu Mar 26 2020 10:00:00 GMT+0900 (대한민국 표준시)

// 가독성이 훨씬 좋다.
new Date('2020/3/26/10:00:00:00');
// → Thu Mar 26 2020 10:00:00 GMT+0900 (대한민국 표준시)
```

april이어야하지 않나

1.5. Date 생성자 함수를 new 연산자없이 호출

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

1.1 new연산자로 호출한 것과 결과값 동일한 것 같은데

JAVASCRIPT

```
Date(); // → Thu Mar 26 2020 14:10:47 GMT+0900 (대한민국 표준시)
```

2. Date 메소드

2.1. Date.now

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

JAVASCRIPT

```
Date.now(); // → 1585199541377
```

2.2. Date.parse

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

JAVASCRIPT

```
// UTC
Date.parse('Jan 2, 1970 00:00:00 UTC'); // → 86400000

// KST
date = Date.parse('Jan 2, 1970 09:00:00'); // → 86400000

// KST
date = Date.parse('1970/01/02/09:00:00');  // → 86400000
```

2.3. 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. month는 월을 의미하는 0~11까지의 정수이다. 0부터 시작하므로 주의가 필요하다.

JAVASCRIPT

```
Date.UTC(1970, 0, 2); // → 86400000
Date.UTC('1970/1/2'); // → NaN
```

2.4. Date.prototype.getFullYear

년도를 나타내는 4자리 숫자를 반환한다.

JAVASCRIPT

```
const today = new Date();
// → Thu Mar 26 2020 14:13:33 GMT+0900 (대한민국 표준시)

today.getFullYear(); // → 2020
```

2.5. Date.prototype.setFullYear

Date 객체에 년도를 나타내는 4자리 숫자를 설정한다. 년도 이외에 옵션으로 월, 일도 설정할 수 있다.

JAVASCRIPT

```
const today = new Date();

// 년도 지정
today.setFullYear(2000);
today.getFullYear(); // → 2000

// 년도/월/일 지정
today.setFullYear(1900, 0, 1);
today.getFullYear(); // → 2000
```

2.6. Date.prototype.getMonth

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

JAVASCRIPT

```
const today = new Date();
// → Thu Mar 26 2020 14:15:25 GMT+0900 (대한민국 표준시)

today.getMonth(); // → 2
```

2.7. Date.prototype.setMonth

Date 객체에 월을 나타내는 0 ~ 11의 정수를 설정한다. 1월은 0, 12월은 11이다. 월 이외에 옵션으로 일도 설정할 수 있다.

JAVASCRIPT

```
const today = new Date();

// 월 지정
today.setMonth(0); // 1월
today.getMonth(); // → 0

// 월/일 지정
today.setMonth(11, 1); // 12월 1일
today.getMonth(); // → 11
```

2.8. Date.prototype.getDate

날짜(1 ~ 31)를 나타내는 정수를 반환한다.

JAVASCRIPT

```
const today = new Date();
// → Thu Mar 26 2020 14:16:43 GMT+0900 (대한민국 표준시)

today.getDate(); // → 26
```

2.9. Date.prototype.setDate

Date 객체에 날짜(1 ~ 31)를 나타내는 정수를 설정한다.

JAVASCRIPT

```
const today = new Date();

// 날짜 지정
today.setDate(1);
today.getDate(); // → 1
```

2.10. Date.prototype.getDay

요일(0 ~ 6)를 나타내는 정수를 반환한다. 반환값은 아래와 같다.

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

JAVASCRIPT

```
const today = new Date();
// → Thu Mar 26 2020 14:17:23 GMT+0900 (대한민국 표준시)

today.getDay(); // → 4
```

2.11. Date.prototype.getHours

시간(0 ~ 23)를 나타내는 정수를 반환한다.

JAVASCRIPT

```
const today = new Date();
// → Thu Mar 26 2020 14:17:39 GMT+0900 (대한민국 표준시)

today.getHours(); // → 14
```

2.12. Date.prototype.setHours

Date 객체에 시간(0 ~ 23)를 나타내는 정수를 설정한다. 시간 이외에 옵션으로 분, 초, 밀리초도 설정할 수 있다.

JAVASCRIPT

```
const today = new Date();

// 시간 지정
today.setHours(7);
today.getHours(); // → 7

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

2.13. Date.prototype.getMinutes

분(0 ~ 59)를 나타내는 정수를 반환한다.

JAVASCRIPT

```
const today = new Date();
// → Thu Mar 26 2020 14:18:34 GMT+0900 (대한민국 표준시)

today.getMinutes(); // → 18
```

2.14 Date.prototype.setMinutes

Date 객체에 분(0 ~ 59)를 나타내는 정수를 설정한다. 분 이외에 옵션으로 초, 밀리초도 설정할 수 있다.

JAVASCRIPT

```
const today = new Date();

// 분 지정
today.setMinutes(50);
today.getMinutes(); // → 50

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

2.15. Date.prototype.getSeconds

초(0 ~ 59)를 나타내는 정수를 반환한다.

JAVASCRIPT

```
const today = new Date();
// → Thu Mar 26 2020 14:19:41 GMT+0900 (대한민국 표준시)

today.getSeconds(); // → 41
```

2.16 Date.prototype.setSeconds

Date 객체에 초(0 ~ 59)를 나타내는 정수를 설정한다. 초 이외에 옵션으로 밀리초도 설정할 수 있다.

JAVASCRIPT

```
const today = new Date();

// 초 지정
today.setSeconds(30);
today.getSeconds(); // → 30

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

2.17. Date.prototype.getMilliseconds

밀리초(0 ~ 999)를 나타내는 정수를 반환한다.

JAVASCRIPT

```
const today = new Date();
// → Thu Mar 26 2020 14:20:27 GMT+0900 (대한민국 표준시)

today.getMilliseconds(); // → 424
```

2.18. Date.prototype.setMilliseconds

Date 객체에 밀리초(0 ~ 999)를 나타내는 정수를 설정한다.

JAVASCRIPT

```
const today = new Date();

// 밀리초 지정
today.setMilliseconds(123);
today.getMilliseconds(); // → 123
```

2.19. Date.prototype.getTime

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

JAVASCRIPT

```
const today = new Date();
// → Thu Mar 26 2020 14:21:08 GMT+0900 (대한민국 표준시)

today.getTime(); // → 1585200068364
```

2.20. Date.prototype.setTime

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

JAVASCRIPT

```
const today = new Date();

// 1970년 1월 1일 00:00:00(UTC)를 기점으로 현재 시간까지 경과된 밀리초 설정
today.setTime(86400000); // → 86400000 === 1day
console.log(today); // → Fri Jan 02 1970 09:00:00 GMT+0900 (대한민국 표준시)
```

2.21. Date.prototype.getTimezoneOffset

UTC와 지정 로케일(Locale) 시간과의 차이를 분단위로 반환한다. KST(Korea Standard Time)는 UTC에 9시간을 더한 시간이다. 즉, UTC = KST - 9h이다.

JAVASCRIPT

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

2.22. Date.prototype.toDateString

사람이 읽을 수 있는 형식의 문자열로 날짜를 반환한다.

JAVASCRIPT

```
const today = new Date('2020/3/26/10:00');

today.toString();     // → Thu Mar 26 2020 10:00:00 GMT+0900 (대한민국 표준시)
today.toDateString(); // → Thu Mar 26 2020
```

2.23. Date.prototype.toISOString

ISO 8601 형식으로 날짜와 시간을 표현한 문자열을 반환한다.

JAVASCRIPT

```
const today = new Date('2020/3/26/10:00');

today.toString();    // → Thu Mar 26 2020 10:00:00 GMT+0900 (대한민국 표준시)
today.toISOString(); // → 2020-03-26T01:00:00.000Z

today.toISOString().slice(0, 10); // → 2020-03-26
today.toISOString().slice(0,10).replace(/-/g, ''); // → 20200326
```

2.24. Date.prototype.toLocaleString

인수로 전달한 로케일을 기준으로 날짜와 시간을 표현한 문자열을 반환한다. 인수를 생략한 경우, 브라우저가 동작 중인 시스템의 로케일을 적용한다.

JAVASCRIPT

```
const today = new Date('2020/3/26/10:00');
```

```
today.toString(); // → Thu Mar 26 2020 10:00:00 GMT+0900 (대한민국 표준시)
today.toLocaleString(); // -> 2020. 3. 26. 오전 10:00:00
today.toLocaleString('ko-KR'); // -> 2020. 3. 26. 오전 10:00:00
today.toLocaleString('en-US'); // → 3/26/2020, 10:00:00 AM
today.toLocaleString('ja-JP'); // → 2020/3/26 10:00:00
```

2.25. Date.prototype.toLocaleTimeString

인수로 전달한 로케일을 기준으로 시간을 표현한 문자열을 반환한다. 인수를 생략한 경우, 브라우저가 동작 중인 시스템의 로케일을 적용한다.

JAVASCRIPT

```
const today = new Date('2020/3/26/10:00');

today.toString(); // → Thu Mar 26 2020 10:00:00 GMT+0900 (대한민국 표준시)
today.toLocaleTimeString(); // -> 오전 10:00:00
today.toLocaleTimeString('ko-KR'); // -> 오전 10:00:00
today.toLocaleTimeString('en-US'); // → 10:00:00 AM
today.toLocaleTimeString('ja-JP'); // → 10:00:00
```

2.26. Date.prototype.toTimeString

사람이 읽을 수 있는 형식으로 시간을 표현한 문자열을 반환한다.

JAVASCRIPT

```
const today = new Date('2020/3/26/10:00');

today.toString();     // → Thu Mar 26 2020 10:00:00 GMT+0900 (대한민국 표준시)
today.toTimeString(); // → 10:00:00 GMT+0900 (대한민국 표준시)
```

#3. Date를 활용한 시계 예제

현재 날짜와 시간을 초단위로 반복 출력하는 예제이다.

JAVASCRIPT

```
(function printNow() {
  const today = new Date();

  const dayNames = ['(일요일)', '(월요일)', '(화요일)', '(수요일)', '(목요일)',
'(금요일)', '(토요일)'];
  // getDay: 해당 요일(0 ~ 6)를 나타내는 정수를 반환한다.
  const day = dayNames[today.getDay()];

  const year = today.getFullYear();
  const month = today.getMonth() + 1;
  const date = today.getDate();
  let hour = today.getHours();
  let minute = today.getMinutes();
  let second = today.getSeconds();
  const ampm = hour ⩾ 12 ? 'PM' : 'AM';

  // 12시간제로 변경
  hour %= 12;
  hour = hour || 12; // 0 ⇒ 12

  // 10미만인 분과 초를 2자리로 변경
  minute = minute < 10 ? '0' + minute : minute;
  second = second < 10 ? '0' + second : second;

  const now = `${year}년 ${month}월 ${date}일 ${day} ${hour}:${minute}:${seco
nd} ${ampm}`;

  console.log(now);
  setTimeout(printNow, 1000);
}());
```