

HTML



Markup Language
Content

CSS



Style sheet Language
Presentation

JS



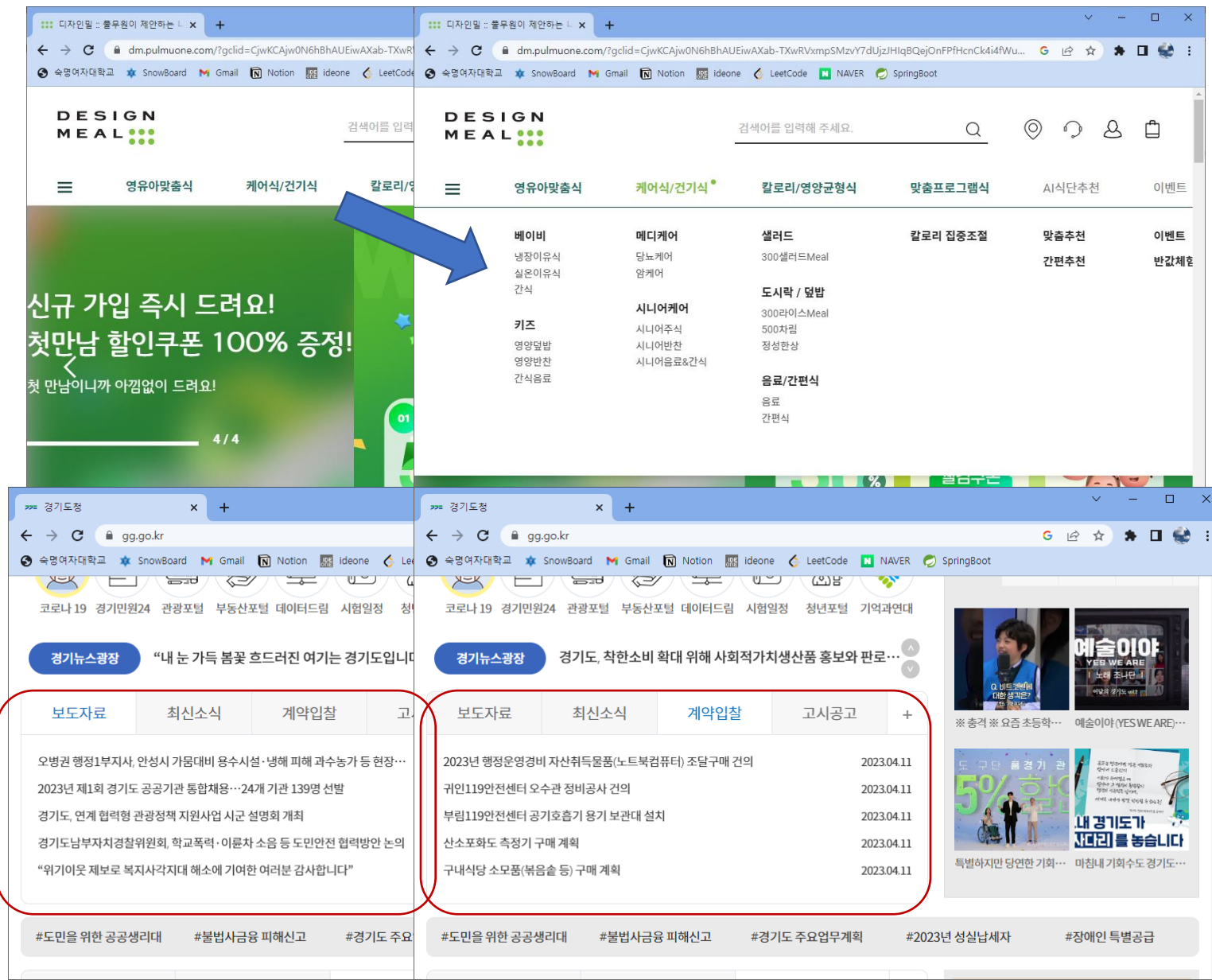
Programming Language
Behavior

자바스크립트 기본 문법 1

자바스크립트로 무엇을 할까

■ 웹 요소 제어

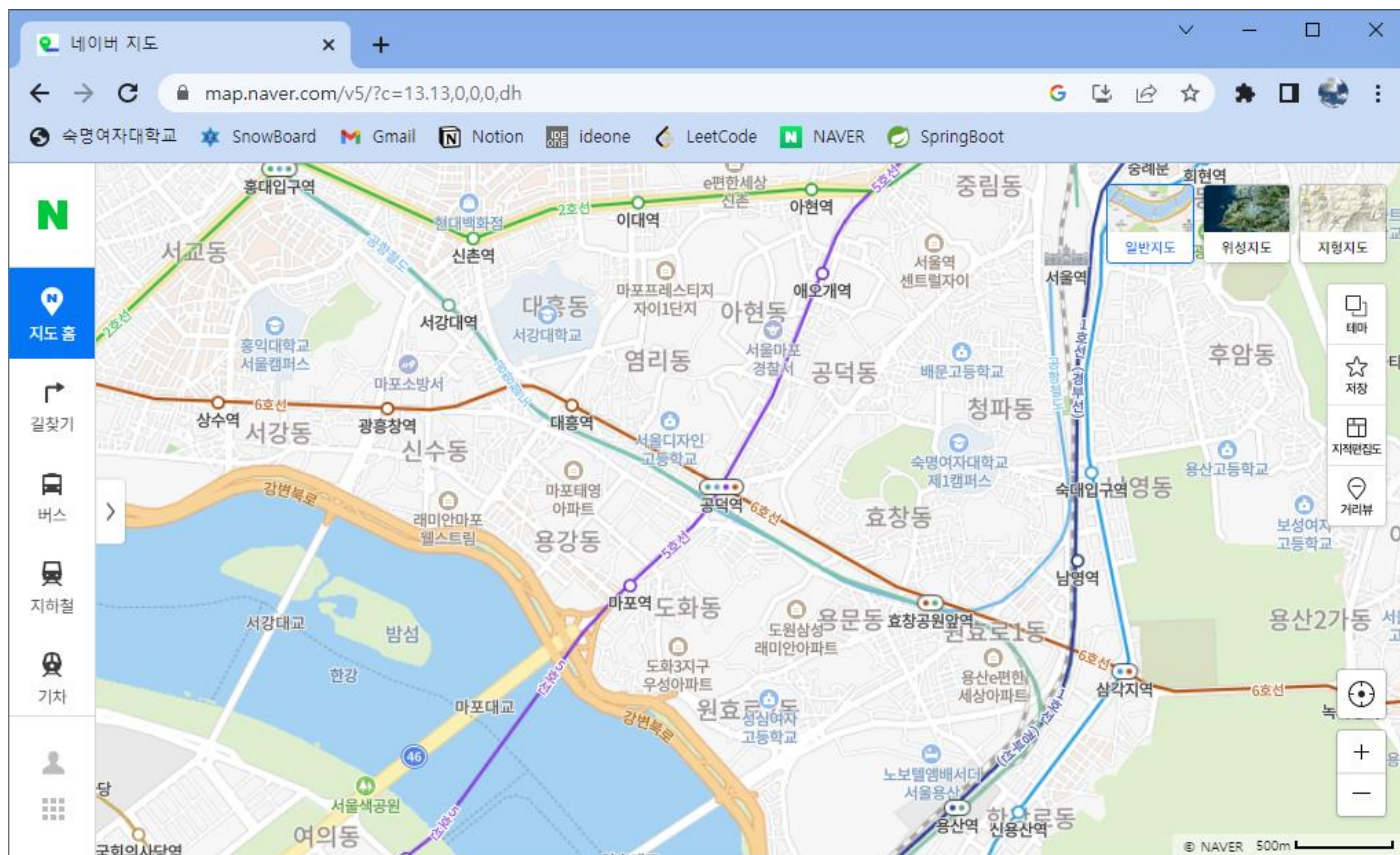
- 웹 요소를 가져와서 필요에 따라 스타일을 변경하거나 움직이게 할 수 있음
- 웹 사이트 UI 부분에 많이 활용
 - 예) 마우스 포인터를 올렸을 때 펼쳐지는 메뉴
 - 한 화면에서 탭을 눌러 내용만 바뀌도록 하는 콘텐츠



자바스크립트로 무엇을 할까

■ 웹 애플리케이션을 만듭니다.

- 최근의 웹 사이트는 사용자와 실시간으로 정보를 주고 받으며 애플리케이션처럼 동작
- 예) 온라인 지도의 길찾기 서비스, 데이터 시각화 서비스, 공개된 API를 활용한 다양한 서비스



자바스크립트로 무엇을 할까

■ 다양한 라이브러리를 사용할 수 있습니다

- 웹을 중심으로 하는 서비스가 늘어나면서 브라우저에서 처리해야 할 일이 늘어남 → 라이브러리와 프레임워크가 계속 등장
- 예) 시각화를 위한 [d3.js](#), 머신러닝을 위한 tensorflow.js, DOM 조작을 위한 jQuery 등
- 예) 웹 애플리케이션 개발을 위한 React, Angular, Vue 등

■ 서버를 구성하고 서버용 프로그램을 만들 수 있습니다

- node.js : 프론트엔드 개발에 사용하던 자바스크립트를 백엔드 개발에서 사용할 수 있게 만든 프레임워크

웹 문서 안에 자바스크립트 작성

- <script> 태그와 </script> 태그 사이에 자바스크립트 소스 작성

```
<script>  
document.getElementById("demo").innerHTML = "My First JavaScript";  
</script>
```

- 웹 문서 안의 <head> 태그 또는 <body> 태그 안에 위치

- 주로 </body> 태그 앞에 작성 권장

- 외부 파일로 작성

- 확장자 .js
- 외부 스크립트 사용 <script src="myScript.js"></script>

자바스크립트 출력

- HTML 요소 내부에 내용 작성
 - innerHTML

```
<!DOCTYPE html>
<html>
<body>

<h2>My First Web Page</h2>
<p>My First Paragraph.</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

My First Web Page

My First Paragraph.

11

자바스크립트 출력 (계속)

■ HTML 출력

- document.write()

```
<!DOCTYPE html>
<html>
<body>

<h2>My First Web Page</h2>
<p>My first paragraph.</p>

<p>Never call document.write after the
document has finished loading.
It will overwrite the whole document.
</p>

<script>
document.write(5 + 6);
</script>

</body>
</html>
```

My First Web Page

My first paragraph.

Never call document.write after the document has finished loading. It will overwrite the whole document.

11

```
<!DOCTYPE html>
<html>
<body>

<h2>My First Web Page</h2>
<p>My first paragraph.</p>

<button type="button" onclick="document.write(5 + 6)">
Try it</button>

</body>
</html>
```

My First Web Page

My first paragraph.

Try it

11

자바스크립트 출력 (계속)

■ alert 창으로 출력

- window.alert()
 - window 생략 가능

```
<!DOCTYPE html>
<html>
<body>

<h2>My First Web Page</h2>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

www.w3schools.com 내용:

11

확인

■ 브라우저 콘솔에 출력

- console.log()

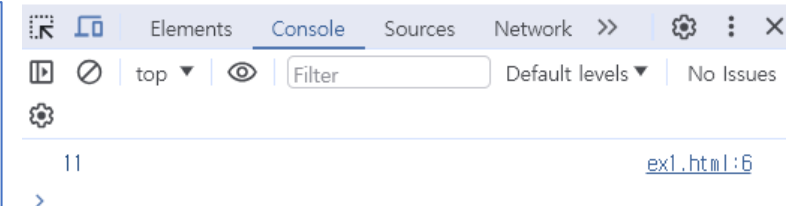
```
<!DOCTYPE html>
<html>
<body>

<h2>Activate Debugging</h2>

<p>F12 on your keyboard will activate debugging.</p>
<p>Then select "Console" in the debugger menu.</p>
<p>Then click Run again.</p>

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

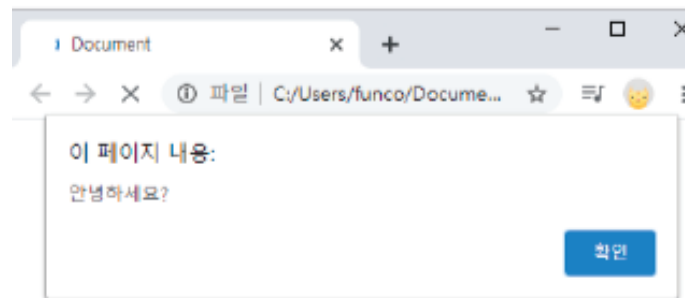


기본 입출력 방법_창

■ 알림 창 출력

기본형 alert(메시지)

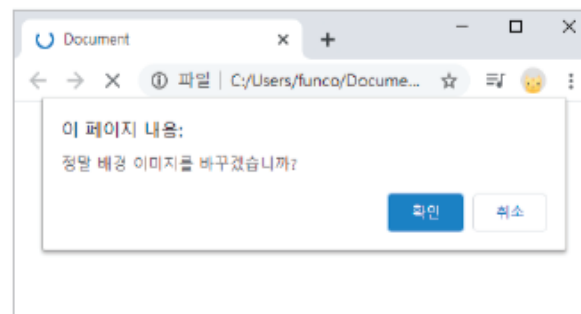
- '확인' 버튼이 있는 메시지 창 표시



■ 확인 창 출력

기본형 confirm(메시지)

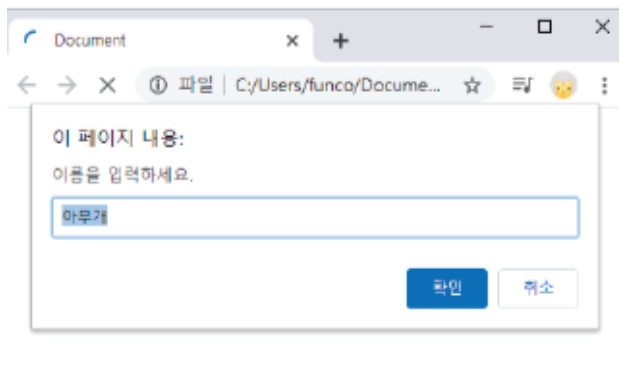
- '확인' 과 '취소' 버튼이 있는 창 표시
- 클릭하는 버튼에 따라 프로그램 동작



■ 프롬프트 창에서 입력받기

- 텍스트 필드가 있는 창 표시
- 사용자 입력 값을 가져와 프로그램에서 사용

기본형 prompt(메시지) 또는 prompt(메시지, 기본값)



자바스크립트 출력 (계속)

■ 인쇄

```
<!DOCTYPE html>
<html>
<body>

<h2>The window.print() Method</h2>

<p>Click the button to print the current
page.</p>

<button onclick="window.print()">Print
this page</button>

</body>
</html>
```

The window.print()

Click the button to print the c

Print this page

24. 4. 12. 오후 2:45W3Schools Try! Editor

The window.print() Method

Click the button to print the current page.

Print this page

인쇄

용지 1장

대상

Hancom PDF

페이지

전체

레이아웃

세로 방향

컬러

컬러

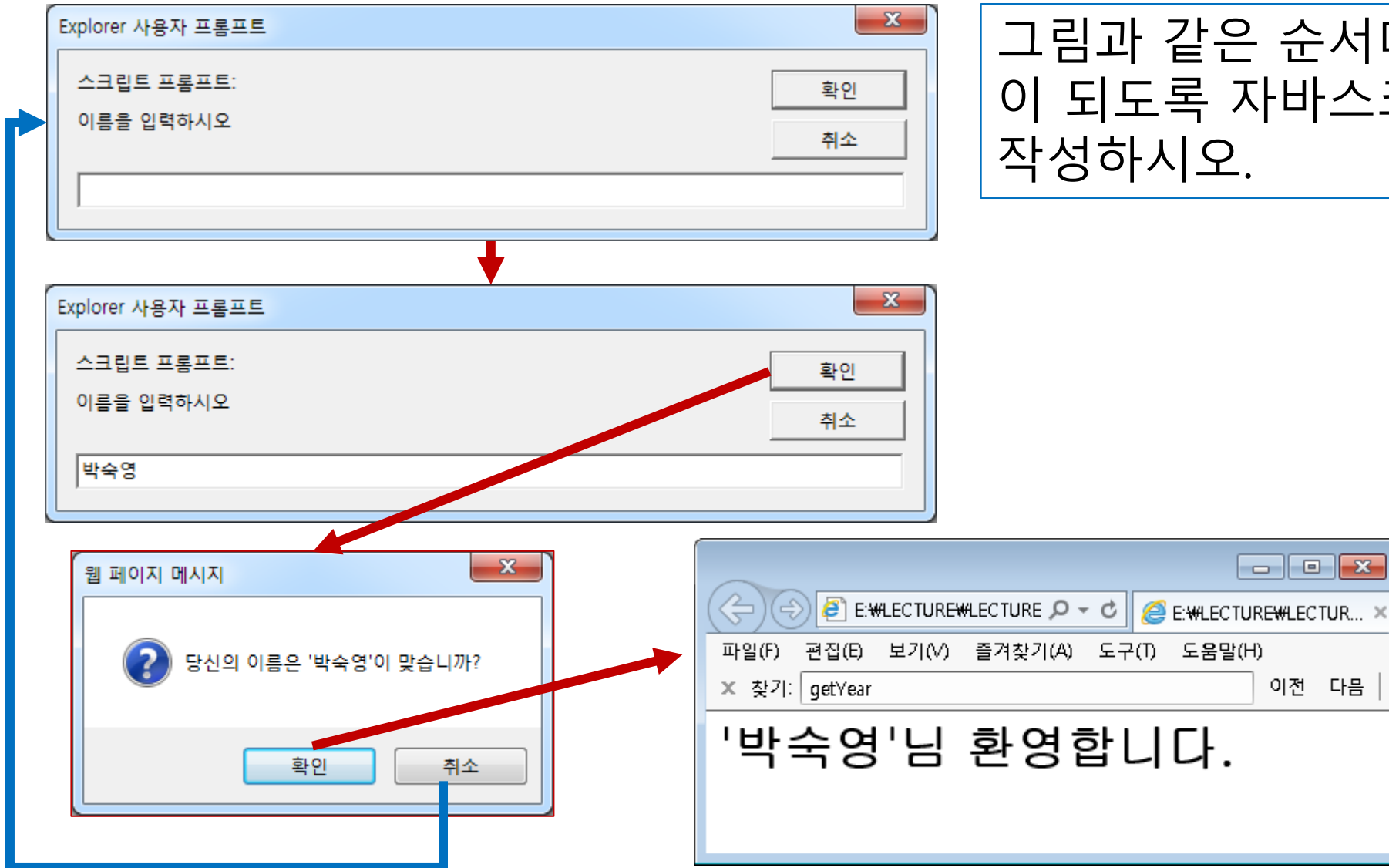
설정 더보기

인쇄

취소

11

실습



그림과 같은 순서대로 진행
이 되도록 자바스크립트를
작성하십시오.

■ 코딩 규칙을 '스타일 가이드', '코딩 컨벤션', '코딩 스타일', '표준 스타일' 등으로 부름

■ 코딩 규칙이 왜 필요할까?

- 자바스크립트는 다른 프로그래밍 언어에 비해 데이터 유형이 유연해서 오류 발생이 잦다
- 오픈 소스에 기여하거나 누군가와 공유할 소스라면 더욱 깔끔한 소스가 중요하다
- 팀 프로젝트를 진행한다면 통일된 코딩 규칙이 필요하다
- 코딩 규칙에 따라 작성된 웹 사이트는 유지 보수도 수월하고 그만큼 비용도 줄어든다

■ 자바스크립트 스타일 가이드

- 구글 자바스크립트 스타일 가이드
(google.github.io/styleguide/jsguide.html) 또는
- 에어비앤비 자바스크립트 스타일 가이드(github.com/airbnb/javascript) 참고
- 회사 프로젝트의 경우 팀 내에서 상의해서 결정

Google JavaScript Style Guide

Table of Contents

1 Introduction	5.8 Control structures
1.1 Terminology notes	5.9 this
1.2 Guide notes	5.10 Equality Checks
	5.11 Disallowed features
2 Source file basics	6 Naming
2.1 File name	6.1 Rules common to all identifiers
2.2 File encoding: UTF-8	6.2 Rules by identifier type
2.3 Special characters	6.3 Camel case: defined
3 Source file structure	7 JSDoc
3.1 License or copyright information, if present	7.1 General form
3.2 @fileoverview, JSDoc, if present	7.2 Markdown
3.3 goog.module statement	7.3 JSDoc tags
3.3.3 goog.module Exports	7.4 Line wrapping
3.4 ES modules	7.5 Topfile-level comments
3.5 goog.setTestOnly	7.6 Class comments
3.6 goog.require and goog.requireType statements	7.7 Enum and typedef comments
3.7 The file's implementation	7.8 Method and function comments
4 Formatting	7.9 Property comments
4.1 Braces	7.10 Type annotations
4.2 Block indentation: +2 spaces	7.11 Visibility annotations
4.3 Statements	
4.4 Column limit: 80	8 Policies
4.5 Line wrapping	8.1 Issues unspecified by Google Style: Be Consistent!

Airbnb JavaScript Style Guide() {

A mostly reasonable approach to JavaScript

Note: this guide assumes you are using Babel, and requires that you use babel-preset-airbnb or the equivalent. It also assumes you are installing shims/polyfills in your app, with [airbnb-browser-shims](#) or the equivalent.

downloads: [6.2M/month](#) downloads: [12M/month](#) [github](#) [join chat](#)

This guide is available in other languages too. See [Translation](#)

Other Style Guides

- [ES5 \(Deprecated\)](#)
- [React](#)
- [CSS-in-JavaScript](#)
- [CSS & Sass](#)
- [Ruby](#)

Table of Contents

1. Types
2. References
3. Objects
4. Arrays
5. Destructuring
6. Strings
7. Functions

자바스크립트 용어

■ 식(expression)

- 값을 만들어 낼 수 있다면 모두 식이 될 수 있다
- 식은 변수에 저장된다

■ 문(statement)

- 문의 끝에는 세미콜론(;)을 붙여서 구분하는게 좋다
- 넓은 의미에서 식이나 값을 포함할 수 있다

■ 주석(Comments)

- 한 줄 주석
 - //
- 여러줄 주석
 - /* */

자바스크립트 키워드

Keyword	Description
var	Declares a variable
let	Declares a block variable
const	Declares a block constant
if	Marks a block of statements to be executed on a condition
switch	Marks a block of statements to be executed in different cases
for	Marks a block of statements to be executed in a loop
function	Declares a function
return	Exits a function
try	Implements error handling to a block of statements

변수 알아보기

■ 변수란

- 변수(variable) : 값이 여러 번 달라질 수 있는 데이터
- 상수(constant) : 값을 한번 지정하면 바뀌지 않는 데이터

■ 변수 선언의 규칙

- 변수 이름
 - 영어 문자, 언더스코어(_), 숫자를 사용한다
 - 첫 글자는 영문자, _기호, \$기호를 사용한다. (숫자로 시작 불가)
 - 띄어쓰기나 기호는 허용하지 않는다
예) now, _now, now25 (사용할 수 있음)
예) 25now, now 25, *now (사용할 수 없음)
- 영어 대소문자를 구별하며 예약어는 변수 이름으로 사용할 수 없다
- L 변수 이름은 의미있게 작성한다

변수 선언 키워드

■ var

- 1995~2015
- 이전 브라우저를 지원하려면 사용

■ let

- 변수 선언
- 2015~

■ const

- 상수 선언
- 2015~

■ 사용 예제

- `var sum=100;`
- `var grade; // undefined`
- `let total; // undefined`
- `let name="sook";`
- `const pi = 3.14;`
- `const test = 5;`

let과 var 비교

■ let

- block scope
- 사용하기 전에 선언되어야 한다.
- 동일 영역 내에 재선언 불가

■ var

- function scope
- 동일 영역 내에 재선언 가능
 - 문제 발생 가능성 있음

	Scope	Redeclare	Reassign	Hoisted	Binds this
var	No	Yes	Yes	Yes	Yes
let	Yes	No	Yes	No	No
const	Yes	No	No	No	No

const 사용 예제

- 선언할 때 바로 값이 할당되어야 함
- 배열, 객체, 함수 등을 선언할 때 const 사용
 - 해당 배열, 객체 자체는 변경 못하지만
 - 상수 배열의 요소 변경 가능
 - 상수 객체의 속성 변경 가능

```
// You can create a constant array:  
const cars = ["Saab", "Volvo", "BMW"];
```

```
// You can change an element:  
cars[0] = "Toyota";
```

```
// You can add an element:  
cars.push("Audi");
```

```
// You can create a const object:  
const car = {type:"Fiat", model:"500", color:"white"};
```

```
// You can change a property:  
car.color = "red";
```

```
// You can add a property:  
car.owner = "Johnson";
```

산술 연산자

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

비교 연산자/논리 연산자/타입 연산자

■ 비교 연산자

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

■ 논리 연산자

Operator	Description
&&	logical and
	logical or
!	logical not

■ 타입 연산자

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

비트 연산자/연결 연산자

■ 비트 연산자

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
	OR	5 1	0101 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	left shift	5 << 1	0101 << 1	1010	10
>>	right shift	5 >> 1	0101 >> 1	0010	2
>>>	unsigned right shift	5 >>> 1	0101 >>> 1	0010	2

■ 연결 연산자

- 둘 이상의 문자열을 합쳐서 하나의 문자열로 만드는 연산자
 - '+' 기호 사용

자료형

■ 숫자

- 정수 : 소수점 없는 숫자
- 실수 : 소수점이 있는 숫자

■ 문자열(string)

- 작은따옴표(' ')나 큰따옴표(" ")로 묶은 데이터

■ 논리형(boolean)

- 참true이나 거짓false의 값을 표현하는 자료형. 불린 유형이라고도 함.
- 조건을 확인해서 조건이 맞으면 true, 맞지 않으면 false라는 결괏값 출력

■ undefined 유형

- 자료형이 정의되지 않았을 때의 데이터 상태
- 변수 선언만 하고 값이 할당되지 않은 자료형

■ null 유형

- 데이터 값이 유효하지 않은 상태
- 변수에 할당된 값이 유효하지 않다는 의미

■ 배열(array)

- 하나의 변수에 여러개의 값을 저장

■ 객체(object)

- 함수와 속성을 함께 포함

조건문(if)

■ if /if-else

```
if(조건){  
    조건 결과값이 true일 때 실행할 명령  
}
```

```
if(조건){  
    조건 결과값이 true일 때 실행할 명령  
}  
else{  
    조건 결과값이 false일 때 실행할 명령  
}
```

■ 조건 연산자

```
(조건) ? true일 때 실행할 명령 : false일 때 실행할 명령
```

조건문(switch)

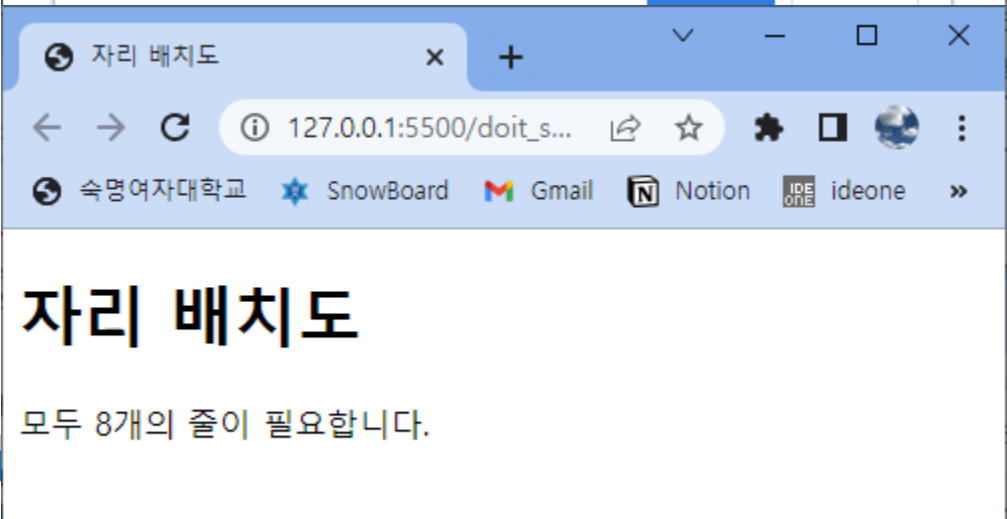
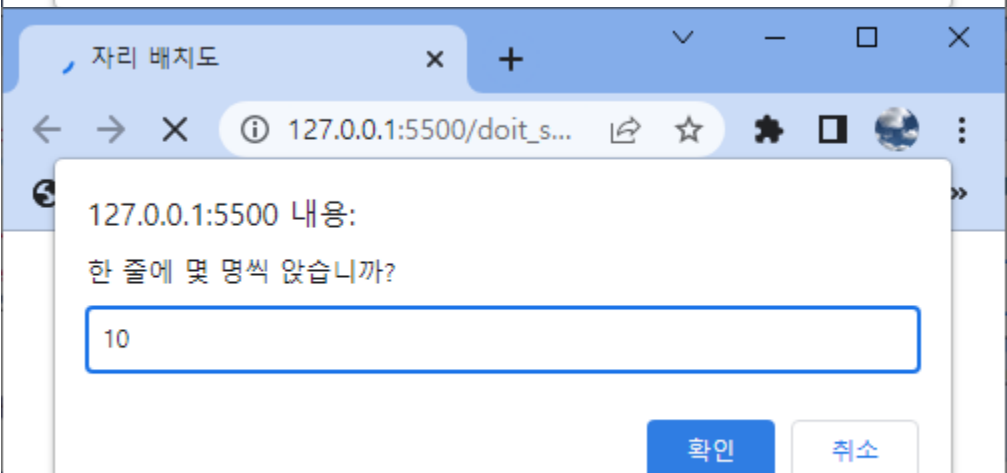
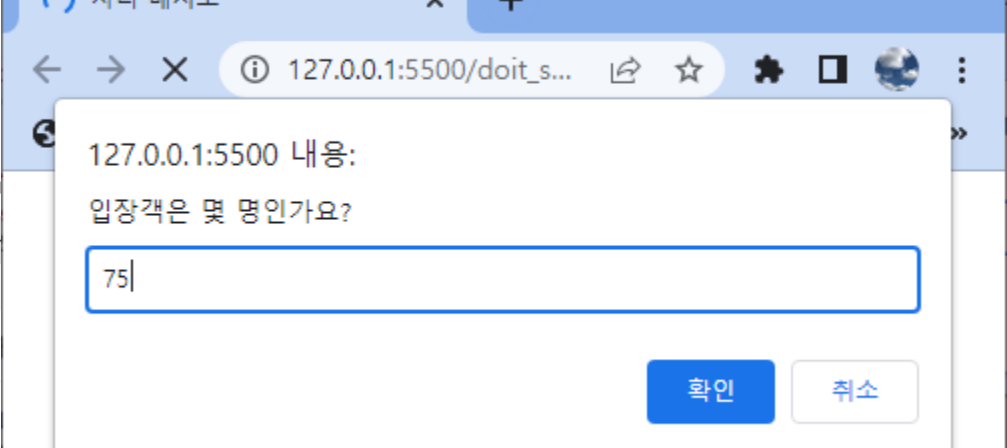
```
switch(조건)
{
    case 값1:
        조건이 값1일 때 실행할 명령
        break;
    case 값2:
        조건이 값2일 때 실행할 명령
        break;
    default:
        조건이 위의 case에 해당하지 않을 때 실행할 명령
        break;
}
```


자리 배치도 만들기

```
seat-1.html - WebWorkspace - Visual Studio Code

seat-1.html x
doit_sources > 14 > seat-1.html > ...

1 <!DOCTYPE html>
2 <html lang="ko">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>자리 배치도</title>
7 </head>
8 <body>
9   <h1>자리 배치도</h1>
10  <script>
11    var memNum = prompt("입장객은 몇 명인가요?"); // 전체 입장객
12    var colNum = prompt("한 줄에 몇 명씩 앉습니까?"); // 한 줄에 앉을 사람
13
14    if (memNum % colNum === 0)
15      rowNum = parseInt(memNum / colNum);
16    else
17      rowNum = parseInt(memNum / colNum) + 1;
18
19    document.write("모두 " + rowNum + "개의 줄이 필요합니다.");
20  </script>
21 </body>
22 </html>
```



반복문

■ for 문

```
for(초깃값; 조건; 증가식){  
    실행할 명령  
}
```

■ while 문

```
while(조건){  
    실행할 명령  
}
```

■ do ~ while 문

```
do{  
    실행할 명령  
}while(조건);
```

■ continue 문

- 반복문의 다음 스텝으로 건너뛴다

■ break 문

- 반복문 하나 빠져나가는 문장

for 문

■ for in Loop

```
for (key in object) {  
    // code block to be executed  
}
```

```
const numbers = [45, 4, 9, 16, 25];  
let txt = "";  
for (let x in numbers) {  
    txt += numbers[x];  
}
```

■ Array.forEach()

```
const numbers = [45, 4, 9, 16, 25];  
  
let txt = "";  
numbers.forEach(myFunction);  
  
function myFunction(value, index, array) {  
    txt += value;  
}
```

■ for of Loop

```
for (variable of iterable) {  
    // code block to be executed  
}
```

```
const cars = ["BMW", "Volvo", "Mini"];  
  
let text = "";  
for (let x of cars) {  
    text += x;  
}
```

자리 배치도 만들기 2

```
7 <style>
8   table, td {
9     border:1px solid #ccc;
10    border-collapse: collapse;
11  }
12  td {
13    padding:5px;
14    font-size:0.9em;
15  }
16 </style>
```

```
20 <script>
21   var i, j;
22   var memNum = prompt("입장객은 몇 명인가요?"); // 전체 입장객
23   var colNum = prompt("한 줄에 몇 명씩 앉습니까?"); // 한 줄에 앉을 사람
24
25   if (memNum % colNum == 0)
26     rowNum = parseInt(memNum / colNum);
27   else
28     rowNum = parseInt(memNum / colNum) + 1;
29
30   // document.write("모두 " + rowNum + "개의 줄이 필요합니다.");
31
32   document.write("<table>");
33   for (i = 0; i < rowNum; i++) {
34     document.write("<tr>");
35     for (j = 1; j <= colNum; j++) {
36       seatNo = i * colNum + j; // 좌석 번호
37       if (seatNo > memNum) break;
38       document.write("<td> 좌석 " + seatNo + " </td>");
39     }
40     document.write("</tr>");
41   }
42   document.write("</table>");
43 </script>
```

자리 배치도

좌석 1	좌석 2	좌석 3	좌석 4	좌석 5	좌석 6	좌석 7	좌석 8	좌석 9	좌석 10
좌석 11	좌석 12	좌석 13	좌석 14	좌석 15	좌석 16	좌석 17	좌석 18	좌석 19	좌석 20
좌석 21	좌석 22	좌석 23	좌석 24	좌석 25	좌석 26	좌석 27	좌석 28	좌석 29	좌석 30
좌석 31	좌석 32	좌석 33	좌석 34	좌석 35	좌석 36	좌석 37	좌석 38	좌석 39	좌석 40
좌석 41	좌석 42	좌석 43	좌석 44	좌석 45	좌석 46	좌석 47	좌석 48	좌석 49	좌석 50
좌석 51	좌석 52	좌석 53	좌석 54	좌석 55	좌석 56	좌석 57	좌석 58	좌석 59	좌석 60
좌석 61	좌석 62	좌석 63	좌석 64	좌석 65	좌석 66	좌석 67	좌석 68	좌석 69	좌석 70
좌석 71	좌석 72	좌석 73	좌석 74	좌석 75					

함수

- 특정 작업을 수행하기 위해 설계된 코드 블록
- 자바스크립트에는 이미 여러 함수가 만들어져 있어서 사용할 수 있음
 - 예) alert()

- 함수 선언 (함수 정의)

- 함수가 어떤 명령을 처리해야 할지 미리 알려주는 것
- function 예약어를 사용하고, { } 안에 실행할 명령을 작성

```
function name(parameter1, parameter2, parameter3) {  
    // code to be executed  
}
```

```
function myFunction(a, b) {  
    return a * b;  
}
```

- 함수 호출 (함수 실행)

- 함수 이름을 사용해 함수 실행

```
함수명(전달할 변수들);
```

```
let rs = myFunction(4, 6); // 24
```

Scope

■ 전역 변수

- 자바스크립트 코드 내의 어느곳에서나 접근할 수 있는 변수
- 방법
 - 선언하지 않고 사용
 - var로 선언 시 함수 밖에서 선언된 변수
 - let으로 선언 시 블록 밖에서 선언된 변수

■ 지역 변수

- 블록 내에서만 사용할 수 있는 변수
- 방법
 - var: 함수내에서 선언시 선언된 함수 내에서만 유효
 - let: 블록내에서 선언시 선언된 블록 내에서만 유효

호이스팅(hoisting)

- 변수를 뒤에서 선언하지만, 마치 앞에서 미리 선언한 것처럼 인식
- 함수 실행문을 앞에 두고 함수 정의 부분을 뒤에 두더라도 앞으로 끌어올려 인식
- var로 선언된 변수는 호이스팅 가능, let으로 선언된 변수는 호이스팅 불가
- 그러나 var는 재선언이 가능한 변수로 실수로 변수를 잘못 조작할 확률이 높아지므로 권장하지 않음

```
<script>
    function displayNumber(){
        var x = 10;
        let a = 10;
        console.log(x);
        console.log(y);
        console.log(a);
        console.log(b);
        var y = 20;
        let b = 20;
    }
    displayNumber();
</script>
```

10

undefined

10

✖ ▶ Uncaught ReferenceError: Cannot access 'b' before initialization
at displayNumber (ex_hoist.html:16:25)
at ex_hoist.html:20:9

매개변수와 반환 값

```
function functionName(parameter1, parameter2, parameter3) {  
    // code to be executed  
    return value;  
}
```

■ 매개변수(parameter)

- 함수 정의부에 열거된 변수명. 함수의 입력으로 들어오는 변수들

■ 인수(arguments)

- 함수를 실행할 때 매개 변수 자리에 넘겨주는 값. 호출 시 실제 전달되는 값

■ 반환 값(return)

- 함수를 실행한 결과값을 함수를 호출한 쪽으로 넘겨줌

디폴트 매개변수

- 함수 정의부에 선언된 매개변수보다 적은 수의 인수가 전달되면, 전달값이 없는 매개변수에 undefined 가 설정

```
function myFunction(x, y) {  
  if (y === undefined) {  
    y = 2;  
  }  
}
```

- 디폴트 매개변수 값 설정

```
function multiple(a, b = 5, c = 10){ // b = 5, c = 10 으로 기본값 지정  
  return a * b + c;  
}  
var result1 = multiple(5, 10, 20); // a = 5, b = 10, c = 20  
var result2 = multiple(10, 20);    // a = 10, b = 20, c = 10  
var result3 = multiple(30);        // a = 30, b = 5, c = 10
```

가변길이 매개변수

```
function sum(...args) {  
  let sum = 0;  
  for (let arg of args) sum += arg;  
  return sum;  
}  
  
let x = sum(4, 9, 16, 25, 29, 100, 66, 77);
```

익명 함수

- 함수 이름이 없는 함수
- 함수 자체가 식이므로 함수를 변수에 할당할 수도 있고 다른 함수의 매개변수로 사용할 수도 있음
- 변수에 저장된 익명 함수는 함수 이름 대신 변수를 이용해 함수를 실행함

```
const x = function (a, b) {return a * b};  
let z = x(4, 3);
```

즉시 실행 함수(Self-Invoking Functions)

- 함수를 정의함과 동시에 실행하는 함수(한 번만 실행하는 함수)
 - 함수를 실행하는 순간 자바스크립트 해석기에서 함수를 해석함
- 식 형태로 선언하기 때문에 함수 선언 끝에 세미콜론(;) 붙임

```
(function () {  
    let x = "Hello!!"; // I will invoke myself  
})();
```

```
(function(a, b){  
    sum = a + b;  
})(100, 200);
```

화살표 함수(람다함수)

- ES6 이후 사용 => 표기를 이용해 함수 식을 작성하는 간단한 문법
- function 키워드 생략
- 문장이 한개밖에 없을 경우 return, 중괄호 생략 가능

```
// ES5  
var x = function(x, y) {  
    return x * y;  
}
```



```
const x = (x, y) => { return x * y };
```



```
const x = (x, y) => x * y;
```

콜백함수

■ 다른 함수에 인수로 전달되는 함수

```
function myDisplayer(some) {  
  document.getElementById("demo").innerHTML = some;  
}
```

```
function myCalculator(num1, num2, myCallback) {  
  let sum = num1 + num2;  
  myCallback(sum);  
}
```

```
myCalculator(5, 5, myDisplayer);
```

- 함수가 인수로 전달될 때 소괄호 사용안함

콜백함수 예

```
// Create an Array
const myNumbers = [4, 1, -20, -7, 5, 9, -6];

// Call removeNeg with a callback
const posNumbers = removeNeg(myNumbers, (x) => x >= 0);

// Display Result
document.getElementById("demo").innerHTML = posNumbers;

// Keep only positive numbers
function removeNeg(numbers, callback) {
  const myArray = [];
  for (const x of numbers) {
    if (callback(x)) {
      myArray.push(x);
    }
  }
  return myArray;
}
```

비동기 함수

- 다른 함수와 병렬로 실행되는 함수

- setTimeout()

- 시간 초과 시 실행될 콜백 함수 지정

```
setTimeout(myFunction, 3000);  
function myFunction() {  
    document.getElementById("demo").innerHTML = "I love You !!";  
}
```

- setInterval()

- 간격마다 실행될 콜백 함수 지정

```
setInterval(myFunction, 1000);  
function myFunction() {  
    let d = new Date();  
    document.getElementById("demo").innerHTML =  
        d.getHours() + ":" +  
        d.getMinutes() + ":" +  
        d.getSeconds();  
}
```

최근엔 비동기 프로그래밍을
위해 콜백대신 promise 사용


```
let myPromise = new Promise(function(myResolve, myReject) {  
  // "Producing Code" (May take some time)
```

```
    myResolve(); // when successful  
    myReject();  // when error  
});  
// "Consuming Code" (Must wait for a fulfilled Promise)  
myPromise.then(  
  function(value) { /* code if successful */ },  
  function(error) { /* code if some error */ }  
);
```

Promise Object

■ Syntax

```
function myDisplayer(some) {  
  document.getElementById("demo").innerHTML = some;  
}
```

```
let myPromise = new Promise(function(myResolve, myReject) {  
  let x = 0;  
  // The producing code (this may take some time)  
  if (x == 0) {  
    myResolve("OK");  
  } else {  
    myReject("Error");  
  }  
});
```

```
myPromise.then(  
  function(value) {myDisplayer(value);},  
  function(error) {myDisplayer(error);}  
);
```

Callback vs Promise

■ Example Using Callback

```
setTimeout(function() { myFunction("I love You !!!"); }, 3000);

function myFunction(value) {
  document.getElementById("demo").innerHTML = value;
}
```

■ Example Using Promise

```
let myPromise = new Promise(function(myResolve, myReject) {
  setTimeout(function() { myResolve("I love You !!"); }, 3000);
});

myPromise.then(function(value) {
  document.getElementById("demo").innerHTML = value;
});
```

async, await

```
function myFunction() {  
  return Promise.resolve("Hello");  
}
```

■ async

- Promise 객체를 반환하는 함수 작성하는 키워드

```
async function myFunction() {  
  return "Hello";  
}  
myFunction().then(  
  function(value) {myDisplayer(value);}  
);
```

■ await

- async 함수 내에서 사용

```
async function myDisplay() {  
  let myPromise = new Promise(function(resolve) {  
    resolve("I love You !!");  
  });  
  document.getElementById("demo").innerHTML = await myPromise;  
}  
myDisplay();
```

```
async function myDisplay() {  
  let myPromise = new Promise(function(resolve) {  
    setTimeout(function() {resolve("I love You !!");}, 3000);  
  });  
  document.getElementById("demo").innerHTML = await myPromise;  
}  
myDisplay();
```

객체

■ 객체(object)란

- 프로그램에서 인식할 수 있는 모든 대상
- 데이터를 저장하고 처리하는 기본 단위

■ 자바스크립트 객체


- 자바스크립트 안에 미리 객체로 정의해 놓은 것
- 문서 객체 모델(DOM) : 문서 뿐만 아니라 웹 문서 안에 포함된 이미지·링크·텍스트 필드 등을 모두 별도의 객체로 관리
- 브라우저 관련 객체 : 웹 브라우저 정보를 객체로 관리
- 내장 객체 : 웹 프로그래밍에서 자주 사용하는 요소를 객체로 정의해 놓음.

■ 사용자 정의 객체

- 필요할 때마다 사용자가 직접 만드는 객체

객체(Objects)

■ 실세계에서의 자동차 객체 사례

Object	Properties	Methods
	car.name = Fiat car.model = 500 car.weight = 850kg car.color = white	car.start() car.drive() car.brake() car.stop()

```
const person = {  
  firstName: "John",  
  lastName: "Doe",  
  age: 50,  
  eyeColor: "blue"  
};
```

```
const car = {type:"Fiat", model:"500", color:"white"};
```

```
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

■ 객체 속성 접근 방법

`objectName.propertyName`

or

`objectName["propertyName"]`

객체 메소드(Objects Methods)

- 함수 정의를 포함하는 객체의 속성

```
const person = {  
  firstName: "John",  
  lastName: "Doe",  
  id: 5566,  
  fullName: function() {  
    return this.firstName + " " + this.lastName;  
  }  
};
```

- 객체 메소드 사용 방법

```
objectName.methodName()
```

```
name = person.fullName();
```

객체 출력

■ 반복문에서 객체 출력

```
const person = {  
  name: "John",  
  age: 30,  
  city: "New York"  
};  
  
let txt = "";  
for (let x in person) {  
  txt += person[x] + " ";  
};
```

■ Object.values()

- 객체를 배열로 반환

```
const person = {  
  name: "John",  
  age: 30,  
  city: "New York"  
};  
const myArray = Object.values(person);
```

■ JSON.stringify()

- 객체를 JSON 형식의 문자열로 변환

```
const person = {  
  name: "John",  
  age: 30,  
  city: "New York"  
};  
let myString = JSON.stringify(person);
```

배열 (Array)

■ 배열 생성, 요소 접근

```
const cars = ["Saab", "Volvo", "BMW"];  
let car = cars[0];
```

■ length : 배열의 길이 반환

```
let size = fruits.length;
```

■ sort() : 정렬

■ reverse() 배열의 요소를 역순으로 반환

■ Array.forEach() 함수 사용

- 배열의 모든 요소에 주어진 함수 적용

```
const fruits =  
["Banana", "Orange", "Apple", "Mango"];  
let text = "<ul>";  
fruits.forEach(myFunction);  
text += "</ul>";  
function myFunction(value) {  
    text += "<li>" + value + "</li>";  
}
```

- map(): 배열 각 요소에 함수 적용 결과값을 새로운 배열로 반환
- filter(): 배열 각 요소에 함수 적용 결과값이 참인 요소들만 새로운 배열로 반환
- reduce(): 배열 각 요소에 함수 적용 → 하나의 결과값 반환

배열 객체 주요 메소드 (계속)

■ toString() : 문자열 반환

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
document.getElementById("demo").innerHTML = fruits.toString();
```

Banana,Orange,Apple,Mango

■ at()

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
let fruit = fruits[2];  
let fruit = fruits.at(2);
```

■ join() : 구분자와 함께 연결된 문자열 반환

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
document.getElementById("demo").innerHTML = fruits.join(" * ");
```

Banana * Orange * Apple * Mango

배열 객체 주요 메소드 (계속)

- `pop()` : 마지막 요소 제거/반환
- `push()`: 마지막 요소 추가

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
let fruit = fruits.pop();  
fruits.push("Kiwi");
```

- `shift()`: 첫번째 요소 제거/반환
- `unshift()`: 첫번째 요소 추가

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
let fruit = fruits.shift();
```

배열 객체 주요 메소드 (계속)

■ concat()

- 서로 다른 배열 2개를 합쳐서 새로운 배열을 만듦

```
const myGirls = ["Cecilie", "Lone"];  
const myBoys = ["Emil", "Tobias", "Linus"];  
const myChildren = myGirls.concat(myBoys);
```

■ splice(추가위치, 삭제되는 요소 수, 새롭게 추가될 요소들)

- 배열 중간에 새로운 요소를 추가하거나 삭제
- 삭제된 아이템들 반환

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.splice(2, 0, "Lemon", "Kiwi");
```

■ slice()

- 원본 배열은 그대로 두고, 주어진 인덱스 범위의 요소들로 구성된 새로운 배열을 반환

```
const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];  
const citrus = fruits.slice(1);  
const citrus2 = fruits.slice(1, 3);
```

■ 생성

```
const d = new Date();
```

```
const d = new Date("2022-03-25");
```

■ 날짜 출력 문자열 메소드

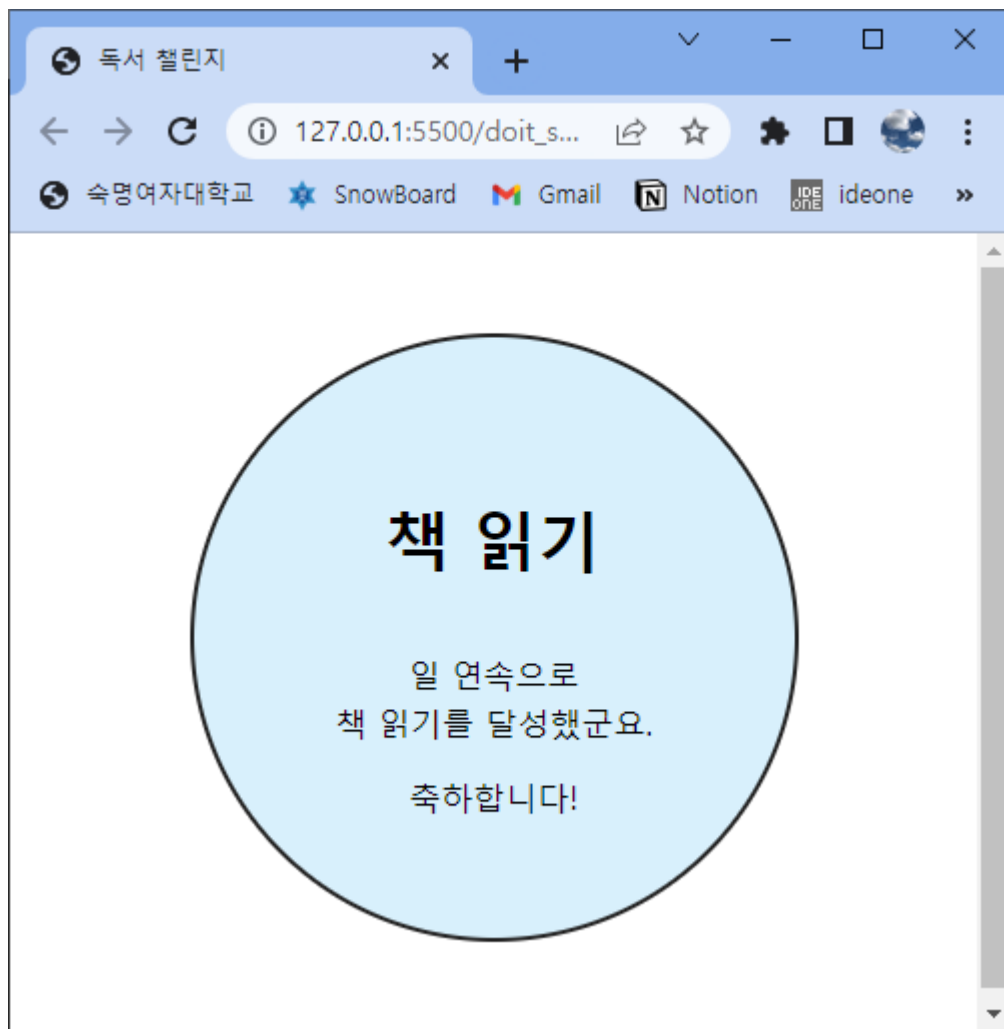
- toString() Sat Apr 20 2024 13:00:22 GMT+0900 (한국 표준시)
- toDateString() Sat Apr 20 2024
- toUTCString() Sat, 20 Apr 2024 04:00:47 GMT
- toISOString() 2024-04-20T04:01:30.984Z

Date 객체 주요 메소드

Method	Description
getFullYear()	Get year as a four digit number (yyyy)
getMonth()	Get month as a number (0-11)
getDate()	Get day as a number (1-31)
getDay()	Get weekday as a number (0-6)
getHours()	Get hour (0-23)
getMinutes()	Get minute (0-59)
getSeconds()	Get second (0-59)
getMilliseconds()	Get millisecond (0-999)
getTime()	Get time (milliseconds since January 1, 1970)

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

날짜 계산 프로그램



```
<style>
  #container{
    margin:50px auto;
    width:300px;
    height:300px;
    border-radius:50%;
    border:2px double ■ #222;
    background-color:□ #d8f0fc;
    text-align: center;
  }
  h1 {
    margin-top:80px;
  }
  .accent {
    font-size:1.8em;
    font-weight:bold;
    color:■ red;
  }
</style>
</head>
<body>
  <div id="container">
    <h1>책 읽기</h1>
    <p><span class="accent" id="result"></span>
      일 연속으로 <br> 책 읽기를 달성했군요.</p>
    <p>축하합니다!</p>
  </div>
```

```

<div id="container">
  <h1>책 읽기</h1>
  <p><span class="accent" id="result"></span>일 연속으로 <br> 책 읽기를 달성했군요.</p>
  <p>축하합니다!</p>
</div>

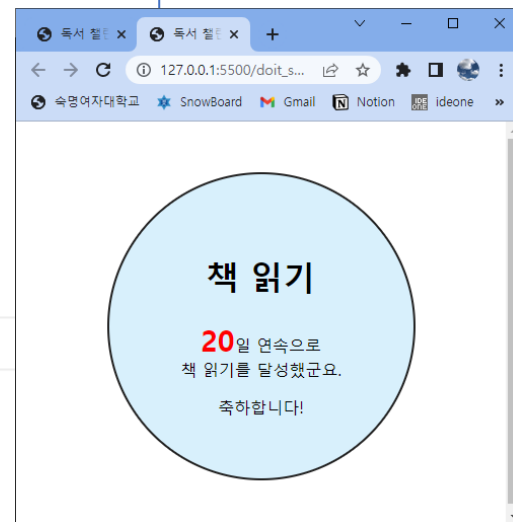
<script>
  var now = new Date();          // 오늘 날짜를 객체로 지정
  var firstDay = new Date("2023-04-01"); // 시작 날짜를 객체로 지정

  var toNow = now.getTime();      // 오늘까지 지난 시간(밀리 초)
  var toFirst = firstDay.getTime(); // 첫날까지 지난 시간(밀리 초)
  var passedTime = toNow - toFirst; // 첫날부터 오늘까지 지난 시간(밀리 초)

  passedTime = Math.round(passedTime/(1000*60*60*24)); // 밀리 초를 일 수로 계산하고 반올림

  document.querySelector('#result').innerText = passedTime;
</script>

```



Math 객체

■ Math 객체의 특징

- 수학 계산과 관련된 메서드가 많이 포함되어 있지만 수학식에서만 사용하는 것은 아님.
- 무작위 수가 필요하거나 반올림이 필요한 프로그램 등에서도 Math 객체의 메서드 사용함
- Math 객체는 인스턴스를 만들지 않고 프로퍼티와 메서드 사용

■ Math 객체의 프로퍼티

```
Math.E          // returns Euler's number
Math.PI         // returns PI
Math.SQRT2      // returns the square root of 2
Math.SQRT1_2    // returns the square root of 1/2
Math.LN2        // returns the natural logarithm of 2
Math.LN10       // returns the natural logarithm of 10
Math.LOG2E      // returns base 2 logarithm of E
Math.LOG10E     // returns base 10 logarithm of E
```


Math 객체 주요 메소드

- 사용 형식: *Math.method(number)*

Method	Description
<u>abs(x)</u>	Returns the absolute value of x
<u>acos(x)</u>	Returns the arccosine of x, in radians
<u>acosh(x)</u>	Returns the hyperbolic arccosine of x
<u>asin(x)</u>	Returns the arcsine of x, in radians
<u>asinh(x)</u>	Returns the hyperbolic arcsine of x
<u>atan(x)</u>	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
<u>atan2(y, x)</u>	Returns the arctangent of the quotient of its arguments
<u>atanh(x)</u>	Returns the hyperbolic arctangent of x
<u>cbrt(x)</u>	Returns the cubic root of x
<u>ceil(x)</u>	Returns x, rounded upwards to the nearest integer
<u>cos(x)</u>	Returns the cosine of x (x is in radians)
<u>cosh(x)</u>	Returns the hyperbolic cosine of x
<u>exp(x)</u>	Returns the value of E^x
<u>floor(x)</u>	Returns x, rounded downwards to the nearest integer

Math 객체 주요 메소드(계속)

<u>log(x)</u>	Returns the natural logarithm (base E) of x
<u>max(x, y, z, ..., n)</u>	Returns the number with the highest value
<u>min(x, y, z, ..., n)</u>	Returns the number with the lowest value
<u>pow(x, y)</u>	Returns the value of x to the power of y
<u>random()</u>	Returns a random number between 0 and 1
<u>round(x)</u>	Rounds x to the nearest integer
<u>sign(x)</u>	Returns if x is negative, null or positive (-1, 0, 1)
<u>sin(x)</u>	Returns the sine of x (x is in radians)
<u>sinh(x)</u>	Returns the hyperbolic sine of x
<u>sqrt(x)</u>	Returns the square root of x
<u>tan(x)</u>	Returns the tangent of an angle
<u>tanh(x)</u>	Returns the hyperbolic tangent of a number
<u>trunc(x)</u>	Returns the integer part of a number (x)

Math 객체 메소드 사용 예제

- random()

```
// Returns a random integer from 0 to 9:  
Math.floor(Math.random() * 10);
```

```
// Returns a random integer from 1 to 100:  
Math.floor(Math.random() * 100) + 1;
```

- min에서 max(포함) 사이의 난수 반환하는 함수 예

```
function getRndInteger(min, max) {  
    return Math.floor(Math.random() * (max - min + 1) ) + min;  
}
```

이벤트 당첨자 뽑기 프로그램

- `Math.random()` 활용
 - 0에서 1 사이(1포함 하지 않음)의 랜덤 숫자 출력
- 1~100 사이의 무작위 수 출력하기
 - `Math.random() * 100 + 1`
- 소수점 이하의 수는 버리고 출력하기
 - `Math.floor(Math.random() * 100 + 1)`

```

<body>
  <h1>당첨자 발표</h1>
  <script>
    var seed = prompt("전체 응모자 수 : ","");
    var picked = Math.floor((Math.random() * seed) + 1);

    document.write("전체 응모자 수 : " + seed + "명");
    document.write("<br>");
    document.write("당첨자 : " + picked + "번");
  </script>
</body>

```

