

웹프로그래밍

- Web Programming -

4. Javascript

정보보호학과 이병천 교수



4. Javascript

- 13. 자바스크립트와의 첫 만남
- 14. 자바스크립트 기본 문법
- 15. 함수와 이벤트
- 16. 자바스크립트와 객체
- 자바스크립트 예제

13. 자바스크립트와 첫 만남

13-1 자바스크립트로 무엇을 할까

13-2 웹 브라우저가 자바스크립트를 만났을 때

13-3 자바스크립트 용어와 기본 입출력 방법

13-4 자바스크립트 스타일 가이드



자바스크립트로 무엇을 할까

웹 요소를 제어합니다

- 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

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

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

서버를 구성하고 서버용 프로그램을 만들 수 있습니다

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

웹 브라우저가 자바스크립트를 만났을 때

웹 문서 안에 자바스크립트 작성하기

- `<script>` 태그와 `</script>` 태그 사이에 자바스크립트 소스 작성
- 웹 문서 안의 어디든 위치할 수 있지만, 주로 **`</body>` 태그 앞에 작성**
- 자바스크립트 소스가 있는 위치에서 실행됨.

 **Do it!** 내부 자바스크립트 사용하기

예제 파일 13\script-1.html

(... 생략 ...)

`<body>`

`<h1 id="heading">자바스크립트</h1>`

`<p id="text">위 텍스트를 클릭해 보세요</p>`

`<script>`

`var heading = document.querySelector('#heading');`

`heading.onclick = function() {`

`heading.style.color = "red";`

`}`

`</script>`

`</body>`

(... 생략 ...)



이곳에 자바스크립트 소스를 작성합니다.

외부 스크립트 파일 연결해서 작성하기

자바스크립트 소스를 별도의 파일(*.js)로 저장한 후 웹 문서에 연결

기본형 `<script src="외부 스크립트 파일 경로"></script>`

 **Do it!** 외부 스크립트 사용하기(JS 파일)

예제 파일 13\js\chage-color.js

`var heading = document.querySelector('#heading');`

`heading.onclick = function() {`

`heading.style.color = "red";`

`}`

 **Do it!** 외부 스크립트 사용하기(HTML 파일)

예제 파일 13\script-2.html

(... 생략 ...)

`<body>`

`<h1 id="heading">자바스크립트</h1>`

`<p id="text">위 텍스트를 클릭해 보세요</p>`

`<script src="js/change-color.js"></script>`

`</body>`

(... 생략 ...)



이곳에서 외부 스크립트 파일을 연결합니다.

자바스크립트

위 텍스트를 클릭해 보세요

자바스크립트

위 텍스트를 클릭해 보세요

웹 브라우저가 자바스크립트를 만났을 때

웹 브라우저에서 스크립트를 해석하는 과정

 **Do it!** 웹 브라우저가 스크립트를 해석하는 과정 살펴보기 예제 파일 13\script-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>글자색 바꾸기</title>
7   <style>
8     body { text-align: center; }
9     (... 생략 ...)
10  </style>
11 </head>
12 <body>
13   <h1 id="heading">자바스크립트</h1>
14   <p id="text">위 텍스트를 클릭해 보세요</p>
15
16   <script>
17     var heading = document.querySelector('#heading');
18     heading.onclick = function() {
19       heading.style.color = "red";
20     }
21   </script>
22 </body>
23 </html>
```

HTML 분석기

CSS 분석기

자바스크립트 해석기

- ① 1행에 있는 <!DOCTYPE html>를 보고 현재 문서가 웹 문서라는 것을 인식
→ 이제부터 HTML 표준에 맞춰 소스를 읽기 시작
- ② 웹 문서에서 HTML 태그의 순서와 포함 관계 확인
- ③ 태그 분석이 끝나면 7~14행의 스타일 정보 분석.
- ④ 20행의 <script> 태그를 만나면 자바스크립트 해석기에게 스크립트 소스 넘김
자바스크립트 해석기에서 <script>와 </script> 사이의 소스를 분석하고 해석
- ⑤ ②에서 분석한 HTML과 ③에서 분석한 CSS 정보에 따라 웹 브라우저 화면에 표시
- ⑥ 웹 브라우저에서 사용자가 동작하면 자바스크립트를 실행해서 결과를 화면에 표시

웹 브라우저가 자바스크립트를 만났을 때

웹 브라우저에서 스크립트를 해석하는 과정

 **Do it!** 웹 브라우저가 스크립트를 해석하는 과정 살펴보기 예제 파일 13\script-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>글자색 바꾸기</title>
7   <style>
8     body { text-align: center; }
9     (... 생략 ...)
10  </style>
11 </head>
12 <body>
13   <h1 id="heading">자바스크립트</h1>
14   <p id="text">위 텍스트를 클릭해 보세요</p>
15
16   <script>
17     var heading = document.querySelector('#heading');
18     heading.onclick = function() {
19       heading.style.color = "red";
20     }
21   </script>
22 </body>
23 </html>
```

HTML 분석기

CSS 분석기

자바스크립트 해석기

- ① 1행에 있는 <!DOCTYPE html>를 보고 현재 문서가 웹 문서라는 것을 인식
→ 이제부터 HTML 표준에 맞춰 소스를 읽기 시작
- ② 웹 문서에서 HTML 태그의 순서와 포함 관계 확인
- ③ 태그 분석이 끝나면 7~14행의 스타일 정보 분석.
- ④ 20행의 <script> 태그를 만나면 자바스크립트 해석기에게 스크립트 소스 넘김
자바스크립트 해석기에서 <script>와 </script> 사이의 소스를 분석하고 해석
- ⑤ ②에서 분석한 HTML과 ③에서 분석한 CSS 정보에 따라 웹 브라우저 화면에 표시
- ⑥ 웹 브라우저에서 사용자가 동작하면 자바스크립트를 실행해서 결과를 화면에 표시

자바스크립트 용어와 기본 입출력 방법

식과 문

식(expression)

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



자바스크립트의 다양한 식 나타내기

```
inch * 2.54 // 연산식은 식입니다.  
"안녕하세요?"; // 문자열도 식입니다.  
5 // 숫자도 식입니다.
```

문(statement)

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

자바스크립트 용어와 기본 입출력 방법

간단한 입출력 방법

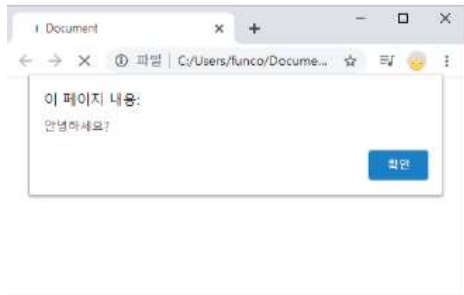
알림 창 출력

'확인' 버튼이 있는 메시지 창 표시

기본형 `alert(메시지)`

 **Do it!** 알림 창 만들기

```
alert("안녕하세요?")
```



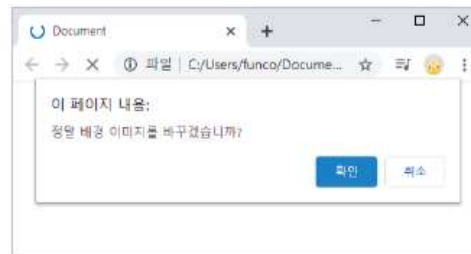
확인 창 출력

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

기본형 `confirm(메시지)`

 **Do it!** 확인 창 만들기


```
var reply = confirm("정말 배경 이미지를 바꾸겠습니까?");
```



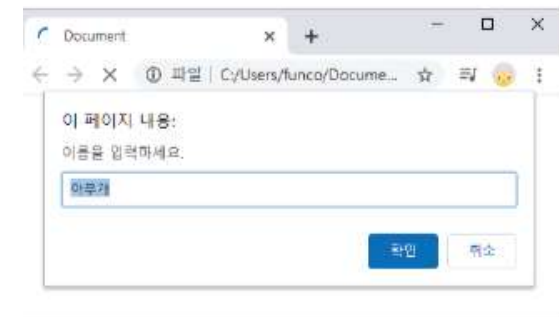
프롬프트 창에서 입력받기

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

기본형 `prompt(메시지)` 또는 `prompt(메시지, 기본값)`

 **Do it!** 프롬프트 창의 기본값 지정하기

```
var name = prompt("이름을 입력하세요.", "아무개");
```




자바스크립트 용어와 기본 입출력 방법

간단한 입출력 방법

document.write()

- 아직 document 객체를 배우지 않았으므로 웹 문서(document)에서 괄호 안의 내용을 표시(write)하는 명령문 이라는 정도로만 알아둘 것
- 괄호 안에서 큰따옴표(" ")나 작은 따옴표(' ') 사이에 입력한 내용은 웹 브라우저 화면에 그대로 표시됨
- 따옴표 안에는 HTML 태그도 함께 사용할 수 있음

 **Do it!** document.write() 문으로 제목 표시하기

```
<script>
  document.write("<h1>어서오세요</h1>");
</script>
```



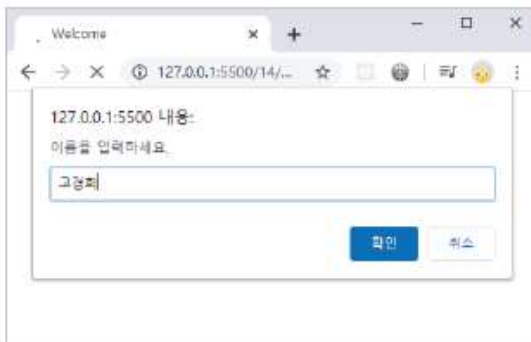
 **Do it!** 이름 받아서 화면에 표시하기 예제 파일 13\greeting.html

```
<script>
  var name = prompt("이름을 입력하세요.");
  document.write("<b><big>" + name + "</big></b>님, 환영합니다.");
</script>
```

① 프롬프트 창에 입력받은 내용을 name 변수에 저장합니다.

② name의 변수값과 "님, 환영합니다."를 연결해 웹 브라우저 창에 표시합니다.

이름:김윤




자바스크립트 용어와 기본 입출력 방법

간단한 입출력 방법

console.log()

웹 브라우저 개발자 도구 창에 포함되어 있는 창

- 괄호 안의 내용을 **콘솔 창**에 표시
- 괄호 안에 변수가 들어갈 수도 있고, 따옴표 안에 텍스트를 넣을 수도 있음
- 따옴표 안에 태그는 사용할 수 없음

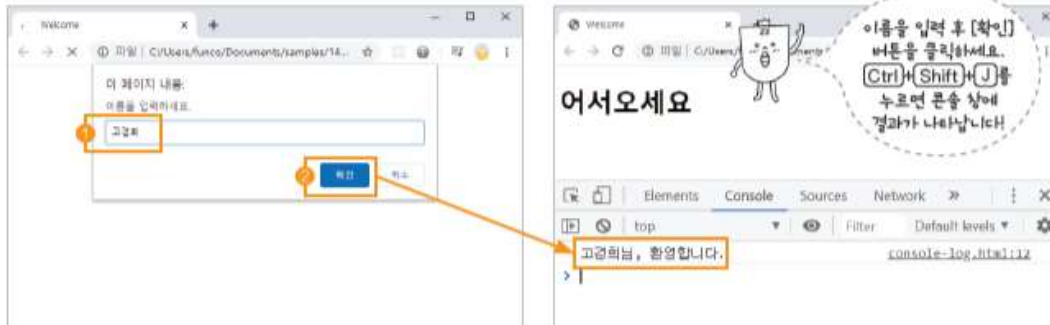
 **Do it!** 이름 받아서 콘솔 창에 표시하기

예제 파일 13\console-log.html

```
<body>
  <h1>어서오세요</h1>

  <script>
    var name = prompt("이름을 입력하세요.");
    console.log(name + "님, 환영합니다.");
  </script>
</body>
```

웹 브라우저에서 콘솔 창을 열려면
웹 브라우저 화면에서
[Ctrl] + [Shift] + [J] 를 누르세요



자바스크립트 스타일 가이드

스타일 가이드란

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



자바스크립트 스타일 가이드

자바스크립트 소스를 작성할 때 지켜야 할 기본 규칙

1. 코드를 보기 좋게 들여쓰다

- 'Tab' 키나 '스페이스바'를 눌러 2칸이나 4칸 들여쓰
- 최근에는 공백 2칸 들여쓰기를 많이 사용함

2. 세미콜론으로 문장을 구분한다

- 세미콜론을 붙일 것을 권장함
- 소스는 한 줄에 한 문장만 작성하는 것이 좋다

세미콜론 규칙

```
var n = 10 // 권장하지 않음
var n = 10; // 권장함
var n = 10; var sum = 0; // 권장하지 않음
```

3. 공백을 넣어 읽기 쉽게 작성한다

식별자나 연산자, 값 사이에 공백을 넣어 읽기 쉽게 작성한다

공백 규칙

```
var sum=num+10; // 권장하지 않음
var sum = num + 10; // 권장함
```

4. 코드를 설명하는 주석을 작성한다

- 한 줄 주석 : 슬래시 2개(//) 바로 뒤에 작성
- 여러 줄 주석 : 여는 기호(/*)를 맨 앞에, 닫는 기호(*/)를 맨 뒤에 넣고 그 사이에 주석 내용을 작성
- 주석 사이에 또다른 주석을 넣을 수 없음



한 줄 주석 작성 방법

```
var today = new Date(); // 현재 날짜 가져오기
var h = today.getHours(); // 시간 추출하기
```



여러 줄 주석 작성 방법

```
/* 현재 날짜를 가져와
시와 분, 초로 추출하고
화면에 표시하는 스크립트
*/
function startTime() { ..... }
```

5. 식별자는 정해진 규칙을 지켜 작성한다

- 첫 글자는 반드시 영문자나 언더스코어(_), 달러 기호(\$)로 시작해야 한다
- 두 단어 이상이 하나의 식별자를 만들 때 단어 사이에 공백을 둘 수 없다
- 예약어는 식별자로 사용할 수 없다

14. 자바스크립트 기본 문법

14-1 변수 알아보기

14-2 자료형 이해하기

14-3 연산자 알아보기

14-4 조건문 알아보기

14-5 반복문 알아보기



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
 - 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
- 여러 단어를 연결할 때는 하이픈이나 언더스코어를 사용할 수 있고 중간에 대문자를 섞어 쓸 수도 있다
- 예) `total-area`, `total_area`, `totalArea` 등
- 변수 이름은 의미있게 작성한다



변수 알아보기

변수 선언하기

- var 뒤에 변수 이름 작성
- var를 한번만 쓰고 뒤에 여러 개의 변수를 한꺼번에 선언할 수도 있음

기본형 var 변수명

변수 선언하기

```
var currentYear; // 올해 연도 변수 선언
var birthYear;   // 태어난 연도 변수 선언
var age;         // 계산한 나이 변수 선언
```

변수 한꺼번에 선언하기

```
var currentYear, birthYear, age; // 올해 연도, 태어난 연도, 계산한 나이 변수 선언
```

변수에 값 할당

'=' 기호 다음에 값을 저장



변수 선언과 값 할당 따로 하기

```
var birthYear; // 태어난 연도 변수 선언
birthYear = 1995; // 변수에 값 할당
```



변수 선언과 값 할당 같이 하기

```
var currentYear = 2021; // 올해 연도 변수 선언하고 값 할당하기
```


자료형 이해하기

자료형이란

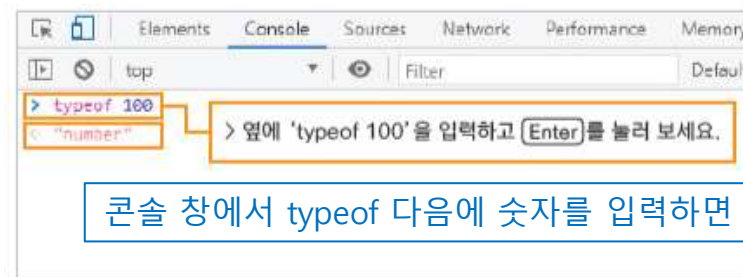
컴퓨터가 처리할 수 있는 자료의 형태

종류		설명	예시
기본 유형	숫자형	따옴표 없이 숫자로만 표기합니다.	<code>var birthYear = 2000;</code>
	문자열	작은따옴표(' ')나 큰따옴표(" ")로 묶어서 나타냅니다. 숫자를 따옴표로 묶으면 문자로 인식합니다.	<code>var greeting = "Hello!";</code> <code>var birthYear = "2000";</code>
	논리형	참(true)과 거짓(false)이라는 2가지 값만 있는 유형입니다. 이때 true와 false는 소문자로만 표시합니다.	<code>var isEmpty = true;</code>
복합 유형	배열	하나의 변수에 여러 개의 값을 저장합니다.	<code>var seasons = ['봄', '여름', '가을', '겨울'];</code>
	객체	함수와 속성을 함께 포함합니다.	<code>var date = new Date();</code>
특수 유형	undefined	자료형이 지정되지 않았을 때의 상태입니다. 예를 들어 변수 선언만 하고 값을 할당하지 않은 변수는 undefined 상태입니다.	
	null	값이 유효하지 않을 때의 상태입니다.	

숫자형(number)

숫자

- 정수 : 소수점 없는 숫자
 - 실수 : 소수점이 있는 숫자
- ※ 자바스크립트는 실수를 정밀하게 계산하지 못함



자료형 이해하기

문자열(string)

작은따옴표(' ')나 큰따옴표(" ")로 묶은 데이터

```
top Filter Default levels
> typeof "안녕하세요?"
< "string"
> typeof "12345"
< "string"
> |
```

콘솔 창에서 typeof 다음에 따옴표로 묶은 내용을 입력하면 string이라고 표시됨

논리형(boolean)

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

```
top Filter Default levels
> 100 < 10
< false
> 100 > 10
< true
> |
```

조건에 따라 true나 false 값을 표시하는 논리형

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

자료형 이해하기

배열(array)

하나의 변수에 여러 값을 저장할 수 있는 복합 유형

기본형 배열명["값1 ", "값2",]

배열명[]

빈 배열 선언

예) 계절 이름을 프로그램에 사용할 경우

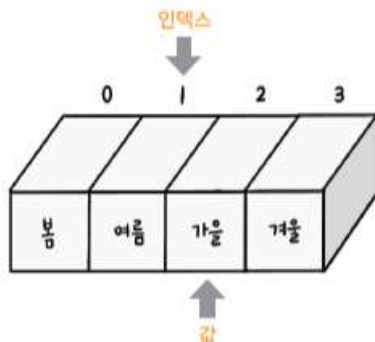
배열을 사용하지 않는다면

```
var spring = "봄";  
var summer = "여름";  
var fall = "가을";  
var winter = "겨울";
```

배열을 사용한다면

```
var season = ["봄", "여름", "가을", "겨울"];
```

변수 이름 → Season



자바스크립트의 데이터 유형 자동 변환

자바스크립트의 편리한 점이면서도 약점인 부분이 데이터 유형이 유연하다는 것입니다. 다시 말해 변수의 데이터 유형이 중간에 바뀔 수 있다는 것이죠.

책에 있는 '나이 계산 프로그램'에서는 프롬프트 창을 통해 사용자의 태어난 해를 입력받는데, 이때 프롬프트 창에서 입력받은 값은 문자열이지만 사칙연산에 사용된 문자열은 자동으로 숫자형으로 변환되어 계산됩니다

연산자 알아보기


산술 연산자

수학 계산을 할 때 사용하는 연산자

종류	설명	예시
+	두 피연산자의 값을 더합니다.	<code>c = a + b</code>
-	첫 번째 피연산자 값에서 두 번째 피연산자 값을 뺍니다.	<code>c = a - b</code>
*	두 피연산자의 값을 곱합니다.	<code>c = a * b</code>
/	첫 번째 피연산자 값을 두 번째 피연산자 값으로 나눕니다.	<code>c = a / b</code>
%	첫 번째 피연산자 값을 두 번째 피연산자 값으로 나눈 나머지를 구합니다.	<code>c = a % b</code>
++	피연산자를 1 증가시킵니다.	<code>a++</code>
--	피연산자를 1 감소시킵니다.	<code>b--</code>

나누기 연산자(/) : 나눈 값 자체

나머지 연산자(%) : 나눈 후에 남은 나머지 값

 나누기 연산자와 나머지 연산자 비교하기

```
var numberOne = 15 / 2; // numberOne은 7입니다
var numberTwo = 15 % 2; // numberTwo는 1입니다
```

할당 연산자(대입 연산자)


연산자 오른쪽의 실행 결과를 왼쪽 변수에 할당하는 연산자

종류	설명	예시
=	연산자 오른쪽의 값을 왼쪽 변수에 할당합니다.	<code>y = x + 3</code>
+=	<code>y = y + x</code> 를 의미합니다.	<code>y += x</code>
-=	<code>y = y - x</code> 를 의미합니다.	<code>y -= x</code>
*=	<code>y = y * x</code> 를 의미합니다.	<code>y *= x</code>
/=	<code>y = y / x</code> 를 의미합니다.	<code>y /= x</code>
%=	<code>y = y % x</code> 를 의미합니다.	<code>y %= x</code>

연결 연산자

둘 이상의 문자열을 합쳐서 하나의 문자열로 만드는 연산자

'+' 기호 사용

 연결 연산자 사용하여 출력하기

```
document.write (birthYear + "년에 태어난 사람의 나이는 " + age + "세입니다.");
```

연산자 알아보기

비교 연산자

피연산자 2개의 값을 비교해서 true나 false로 결과값 반환

종류	설명	예시	
		조건식	결과값
==	피연산자가 서로 같으면 true입니다.	3 == "3"	true
===	피연산자도 같고 자료형도 같으면 true입니다.	a === "3"	false
!=	피연산자가 서로 같지 않으면 true입니다.	3 != "3"	false
!==	피연산자가 같지 않거나 자료형이 같지 않으면 true입니다.	3 !== "3"	true
<	왼쪽 피연산자가 오른쪽 피연산자보다 작으면 true입니다.	3 < 4	true
<=	왼쪽 피연산자가 오른쪽 피연산자보다 작거나 같으면 true입니다.	3 <= 4	true
>	왼쪽 피연산자가 오른쪽 피연산자보다 크면 true입니다.	3 > 4	false
>=	왼쪽 피연산자가 오른쪽 피연산자보다 크거나 같으면 true입니다.	3 >= 4	false

== 연산자 와 != 연산자

피연산자의 자료형을 자동으로 변환해서 비교

```
3 == "3" // true
3 != "3" // false
```

=== 연산자 와 !== 연산자

피연산자의 자료형을 변환하지 않음

```
3 === "3" // false
3 !== "3" // true
```

← 프로그램에서 값을 비교할 때 더 많이 사용

논리 연산자

true와 false가 피연산자인 연산자
조건을 처리할 때 사용

종류	기호	설명
OR 연산자		피연산자 중 하나만 true여도 true가 됩니다.
AND 연산자	&&	피연산자가 모두 true일 경우에만 true가 됩니다.
NOT 연산자	!	피연산자의 반댓값을 지정합니다.


조건문 알아보기

if 문과 if ~ else 문

피연산자 2개의 값을 비교해서 true나 false로 결과값 반환
하나의 if ~ else 문 안에 다른 if ~ else 문을 넣을 수 있다

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

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

 **Do it!** 3의 배수 확인하기 2 예제 파일 14\if-2.html

```
(... 생략 ...)  
<script>  
    var userNumber = prompt("숫자를 입력하세요.");  
  
    if(userNumber !== null) { // 입력값이 null이 아니면 if~else 문을 실행  
        if(userNumber % 3 === 0)  
            alert("3의 배수입니다.");  
        else  
            alert("3의 배수가 아닙니다.");  
    }  
    else  
        alert("입력이 취소됐습니다."); // 입력값이 null이면 알림 창을 보여 줌  
</script>  
(... 생략 ...)
```

if~else 문 안에 중첩된 if~else 문을 사용합니다.

127.0.0.1:5500 내용:
숫자를 입력하세요.

확인 취소

127.0.0.1:5500 내용:
3의 배수가 아닙니다.

확인

조건문 알아보기

조건 연산자로 조건 체크하기

조건이 하나이고 true일 때와 false일 때 실행할 명령이 각각 하나뿐일 때 간단하게 사용할 수 있음

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

Do it! 3의 배수 확인하기 2

예제 파일 14\if-2.html

(... 생략 ...)

<script>

var userNumber = prompt("숫자를 입력하세요.");

```
if(userNumber !== null) { // 입력값이 null이 아니면 if~else 문을 실행
    if(userNumber % 3 === 0)
        alert("3의 배수입니다.");
    else
        alert("3의 배수가 아닙니다.");
}
```

if~else 문 안에 중첩된 if~else 문을 사용합니다.

```
else
    alert("입력이 취소됐습니다."); // 입력값이 null이면 알림 창을 보여 줌
```

</script>

(... 생략 ...)



Do it! 조건 연산자를 사용해 3의 배수 확인하기

예제 파일 14\if-3.html

(... 생략 ...)

<script>

var userNumber = prompt("숫자를 입력하세요.");

if(userNumber !== null)

조건 연산자를 사용했습니다.

(userNumber % 3 === 0) ? alert("3의 배수입니다.") : alert("3의 배수가 아닙니다.");

else

alert("입력이 취소됐습니다.");

</script>

(... 생략 ...)

127.0.0.1:5500 내용:

숫자를 입력하세요.

14

확인

취소

127.0.0.1:5500 내용:

3의 배수가 아닙니다.

확인

조건문 알아보기

논리 연산자로 조건 체크하기

- 조건을 2개 이상 체크할 경우에는 조건 연산자를 사용해 조건을 만들
- 두 조건이 true일 경우, 조건 1개만 true일 경우처럼 여러 경우를 따질 때 논리 연산자 사용

AND 연산자 (&&)

피연산자 2개 중에서 false가 하나라도 있으면 결과값은 false

op 1	op 2	op 1 && op 2
false	false	false
false	true	false
true	false	false
true	true	true

OR 연산자 (||)

피연산자 2개 중에서 true가 하나라도 있으면 결과값은 true

op 1	op 2	op 1 op 2
false	false	false
false	true	true
true	false	true
true	true	true

NOT 연산자 (!)

피연산자를 반대로 뒤집음

op	!op
false	true
true	false

조건문 알아보기

switch문

- 처리할 명령이 많을 경우 switch 문이 편리

```
기본형 switch (조건)
{
    case 값1: 명령1
        break
    case 값2: 명령2
        break
    .....
    default: 명령n
}
```

- 조건은 case 문의 값과 일대일로 일치해야 함
- case 문의 명령 실행 후 switch 문 빠져나옴
- 조건과 일치하는 case 문이 없다면 default 문 실행
- default 문에는 break 문이 없음



switch 문으로 조건 체크하기

예제 파일 14\switch.html

(... 생략 ...)

<script>

```
var session = prompt("관심 세션을 선택해 주세요. 1-마케팅, 2-개발, 3-디자인");
```

```
switch(session) {
```

```
    case "1": document.write("<p>마케팅 세션은 <strong>201호</strong>에서 ..... </p>")
```

```
        break;
```

```
    case "2": document.write("<p>개발 세션은 <strong>203호</strong>에서 ..... </p>")
```

```
        break;
```

```
    case "3": document.write("<p>디자인 세션은 <strong>205호</strong>에서 ..... </p>")
```

```
        break;
```

```
    default: alert("잘못 입력했습니다."); // 1, 2, 3이 아닌 값을 입력받으면 출력
```

```
}
```

</script>

(... 생략 ...)

127.0.0.1:5500 내용:

관심 세션을 선택해 주세요. 1-마케팅, 2-개발, 3-디자인

2

확인

취소

개발 세션은 **203호**에서 진행됩니다.

반복문 알아보기

for 문

기본형

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

- ❶ 초깃값: 카운터 변수를 초기화합니다. 초깃값은 0이나 1부터 시작합니다.
- ❷ 조건: 명령을 반복하기 위해 조건을 체크합니다. 이 조건을 만족해야 그다음에 오는 명령을 실행할 수 있습니다.
- ❸ 증가식: 명령을 반복한 후 실행합니다. 보통 카운터 변수를 1 증가시키는 용도로 사용합니다.



for 문을 사용해 1부터 5까지 숫자 더하기

예제 파일 14\repeat-2.html

(... 생략 ...)

<script>

var i;

var sum = 0;

```
for(i = 1; i < 6; i++) {  
    sum += i;  
}
```



for 문을 사용해
코드를 간단하게 작성했죠!

document.write("1부터 5까지 더하면 " + sum);


</script>

(... 생략 ...)

- ❶ 카운터로 사용할 변수 i에 초깃값 1 지정
- ❷ i = 1 → i < 6 체크 → (조건 만족함) → sum += i 실행 → i++ 실행
- ❸ i = 2 → i < 6 체크 → (조건 만족함) → sum += i 실행 → i++ 실행
- ❹ i = 3 → i < 6 체크 → (조건 만족함) → sum += i 실행 → i++ 실행
- ❺ i = 4 → i < 6 체크 → (조건 만족함) → sum += i 실행 → i++ 실행
- ❻ i = 5 → i < 6 체크 → (조건 만족함) → sum += i 실행 → i++ 실행
- ❼ i = 6 → i < 6 체크 → (조건 만족하지 않음) → for 문을 빠져나옴

반복문 알아보기

중첩된 for 문

 Do it! for 문 2개로 구구단 만들기

예제 파일 14\gugudan-1.html

(... 생략 ...)

```
<h1>구구단</h1>
```

```
<script>
```

```
var i, j;
```

```
for (i = 1; i <= 9; i++) {
```

```
    document.write("<h3>" + i + "단</h3>");
```

```
    for (j = 1; j <= 9; j++) {
```

```
        document.write(i + " X " + j + " = " + i*j + "<br>");
```

```
    }
```

```
}
```

```
</script>
```

(... 생략 ...)

구구단

1단

```
1 X 1 = 1
1 X 2 = 2
1 X 3 = 3
1 X 4 = 4
1 X 5 = 5
1 X 6 = 6
1 X 7 = 7
1 X 8 = 8
1 X 9 = 9
```

2단

```
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
```

3단

```
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
```

...

while 문과 do ~ while 문

while 문

조건을 체크하고 true라면 { }안의 명령 실행

→ 조건이 false라면 명령은 한 번도 실행하지 않을 수 있음

```
기본형 while (조건) {  
    실행할 명령  
}
```

do ~ while 문

일단 명령을 한번 실행한 후 조건 체크.

true라면 { } 안의 명령 실행, false라면 { }을 빠져나옴

→ 조건이 false라도 명령은 최소한 한 번은 실행

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



Do it! while 문으로 팩토리얼 만들기

예제 파일 14\factorial.html

```
(... 생략 ...)  
<h1>while 문을 사용한 팩토리얼 계산</h1>  
<script>  
    var n = prompt("숫자를 입력하세요.");  
    var msg = "";  
  
    if(n !== null) { // 취소 버튼을 누르지 않았는지 체크  
        var nFact = 1; // 곱값  
        var i = 1; // 카운터  
  
        while(i <= n) {  
            nFact *= i;  
            i++;  
        }  
        msg = n + "! = " + nFact; // 곱값을 표시할 문자열  
    }  
    else  
        msg = "값을 입력하지 않았습니다.";  
  
    document.write(msg); // 결과 표시  
</script>  
(... 생략 ...)
```

break 문과 continue 문

break 문

종료 조건이 되기 전에 반복문을 빠져 나와야 할 때 사용

기본형 break

Do it! break 문으로 구구단을 3단까지만 표시하기 예제 파일 14\gugudan-3.html

```
(... 생략 ...)  
<script>  
  var i, j;  
  
  for(i = 1; i <= 9; i++) {  
    document.write("<div>");  
    document.write("<h3>" + i + "단</h3>");  
    for(j = 1; j <= 9; j++) {  
      document.write(i + " X " + j + " = " + i*j + "<br>");  
    }  
    document.write("</div>");  
  
    if(i === 3) break; // i값이 3이면 break 문을 실행합니다.  
  }  
</script>
```

continue 문

조건에 해당되는 값을 만나면 반복문의 맨 앞으로 이동
→ 결과적으로 반복 과정을 한 차례 건너 뛴

기본형 continue

Do it! 1부터 10까지 짝수만 더하기 예제 파일 14\even.html

```
(... 생략 ...)  
<h1>짝수끼리 더하기</h1>  
<script>  
  var i;  
  var n = 10;  
  var sum = 0;  
  
  for(i = 1; i <= n; i++) {  
    if (i % 2 === 1) // 1이 홀수라면 반복문을 건너뛴  
      continue  
    sum += i;  
  
    document.write(i + " ----- " + sum + "<br>");  
  }  
</script>  
(... 생략 ...)
```

15. 함수와 이벤트

15-1 함수 알아보기

15-2 var를 사용한 변수의 특징

15-3 let와 const의 등장

15-4 재사용할 수 있는 함수 만들기

15-5 함수 표현식

15-6 이벤트와 이벤트 처리기

15-7 DOM을 이용한 이벤트 처리기



함수 알아보기

함수란

- 동작해야 할 목적대로 명령을 묶어 놓은 것
- 각 명령의 시작과 끝을 명확하게 구별할 수 있음
- 묶은 기능에 이름을 붙여서 어디서든 같은 이름으로 명령을 실행할 수 있음
- 자바스크립트에는 이미 여러 함수가 만들어져 있어서 가져다 사용할 수 있음

예) alert()

함수의 선언 및 호출

함수 선언 : 어떤 명령을 처리할지 미리 알려주는 것

```
기본형 function 함수명() {  
    명령  
}
```

함수 호출 : 선언한 함수를 사용하는 것

```
기본형 함수명( ) 또는 함수명(변수)
```

Do it! 함수를 사용해 두 수 더하기 예제 파일 15\using-function.html

(... 생략 ...)

```
<script>  
function addNumber() {  
    var num1 = 2;  
    var num2 = 3;  
    var sum = num1 + num2;  
    alert("결과값: " + sum);  
}  
  
addNumber();  
addNumber();  
</script>  
(... 생략 ...)
```

함수를 선언

함수를 2번 호출

127.0.0.1:5500 내용:
결과값: 5

확인

127.0.0.1:5500 내용:
결과값: 5

확인

알림 상에서
결과값이 두 번 나타납니다.


var를 사용한 변수의 특징

스코프 : 변수가 적용되는 범위

스코프에 따라 지역 변수(로컬 변수)와 전역 변수(글로벌 변수)로 나뉨

지역 변수

- 함수 안에서 선언하고 함수 안에서만 사용함
- var과 함께 변수 이름 지정

 **Do it!** var 예약어로 지역 변수 선언하기 예제 파일 15\var-1.html

```
(... 생략 ...)  
<script>  
  fu  addNumber() {  
    var sum = 10 + 20; // 지역 변수 선언  
  }  
  addNumber();  
  
  console.log(sum); // 지역 변수를 사용  
</script>  
(... 생략 ...)
```


변수 sum 적용 범위

오류 발생

uogcunj:를윤

전역 변수

- 스크립트 소스 전체에서 사용함
- 함수 밖에서 선언하거나 함수 안에서 var 없이 선언

 **Do it!** var 예약어를 사용한 지역 변수와 전역 변수 예제 파일 15\var-2.html

```
(... 생략 ...)  
<script>  
  function addNumber() {  
    v  sum = 10 + 20; // 지역 변수 선언  
    multi = 10 * 20; // 전역 변수 선언  
  }  
  addNumber();  
  console.log(multi); // 전역 변수를 사용  
</script>  
(... 생략 ...)
```

200

JBA:를윤

var를 사용한 변수의 특징

var 변수와 호이스팅

```
Do it! 변수와 호이스팅 예제 파일 15\var-3.html

(... 생략 ...)
<script>
  var x = 10;

  function displayNumber() {
    console.log("x is " + x);
    console.log("y is " + y);
    var y = 20;
  }
  displayNumber();
</script>
(... 생략 ...)
```

호이스팅

- 변수를 뒤에서 선언하지만, 마치 앞에서 미리 앞에서 선언한 것처럼 인식함
- 함수 실행문을 앞에 두고 선언 부분을 뒤에 두더라도 앞으로 끌어올려 인식함

재선언과 재할당이 가능하다

- 재선언 : 이미 선언한 변수를 다시 선언할 수 있음
 - 재할당 : 같은 변수에 다른 값을 할당할 수 있음
- 재선언과 재할당이 가능하면 실수로 변수를 잘못 조작할 확률이 높아짐

```
Do it! var 예약어를 사용한 변수의 재할당과 재선언 예제 파일 15\var-4.html

(... 생략 ...)
<script>
  f addNumber(num1, num2) {
    return num1 + num2; // 2개의 수 더하기
  }
  var sum = addNumber(10, 20); // sum 변수 선언, 함수 호출
  console.log(sum);
  sum = 50; // sum 변수 재할당
  console.log(sum);
  var sum = 100; // sum 변수 재선언
  console.log(sum);
</script>
(... 생략 ...)
```

	Elements	Console	Sources	>>	⋮	×
		top			Filter	Default lev
30						
50						
100						
>						

let과 const의 등장

let을 사용한 변수의 특징

- 블록 변수 – 블록({ }) 안에서만 사용할 수 있다
→ 전역 변수는 변수 이름과 초깃값만 할당하면 됨

전역 변수

```
(... 생략 ...)  
function calcSum(n) {  
  sum = 0;  
  for(let i = 1; i < n + 1; i++) {  
    sum += i;  
  }  
}  
calcSum(10);  
console.log(sum);  
(... 생략 ...)
```

블록 변수

- 재할당은 가능하지만 재선언은 할 수 없다
- 호이스팅이 없다

const를 사용한 변수의 특징

- 상수 – 변하지 않는 값을 선언할 때 사용
- 재선언, 재할당할 수 없음

자바스크립트 변수, 이렇게 사용하자

- 전역 변수는 최소한으로 사용
- var 변수는 함수의 시작 부분에서 선언
- for 문에서 카운터 변수는 var보다 let 변수로 선언
- ES6를 사용한다면 var보다 let를 사용하는 것이 좋다

재사용할 수 있는 함수 만들기

매개 변수와 인수, return 문

- 매개 변수 : 하나의 함수를 여러 번 실행할 수 있도록 실행할 때마다 바뀌는 값을 변수로 처리한 것
- 인수 : 함수를 실행할 때 매개 변수 자리에 넘겨주는 값



Do it! 매개변수를 사용한 함수 선언하고 호출하기

(... 생략 ...)

```
function addNumber(num1, num2) { ❶
```

```
    var sum = num1 + num2; ❷
```

```
    return sum; ❸
```

```
}
```

```
var result = addNumber(2, 3); // 함수 호출 ❹
```

```
document.write("두 수를 더한 값: " + result); ❺
```

(... 생략 ...)

- ❶ 자바스크립트 해석기가 `function`이라는 예약어를 만나면 함수를 선언하는 부분이라는 걸 인식하고 함수 블록(`{ }`)을 해석합니다. 아직 실행하지 않습니다.
- ❷ `addNumber(2, 3)`을 만나면 해석해 두었던 `addNumber()` 함수를 실행합니다.
- ❸ `addNumber()` 함수에서 2는 `num1`로, 3은 `num2`로 넘기고 더한 값을 `sum` 변수에 저장합니다.
- ❹ 함수 실행이 모두 끝나면 결과값 `sum`을 함수 호출 위치, 즉 `var result`로 넘깁니다.
- ❺ 넘겨받은 결과값을 `result`라는 변수에 저장합니다.
- ❻ `result` 변수에 있는 값을 화면에 표시합니다.

함수 표현식

익명 함수

- 함수 이름이 없는 함수
- 함수 자체가 식이므로 함수를 변수에 할당할 수도 있고 다른 함수의 매개변수로 사용할 수도 있음

Do it! 익명 함수 실행하기

```
(... 생략 ...)  
var sum = f (a, b) {  
    return a + b;  
}  
document.write("함수 실행 결과: " + sum(10, 20) );  
(... 생략 ...)
```

즉시 실행 함수

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

기본형 (function() {
 명령
})();

또는

기본형 (function(매개변수) {
 명령
})(인수));

Do it! 매개변수가 있는 즉시 실행 함수 만들기

```
(... 생략 ...)  
<script>  
    (function(a, b) {  
        sum = a + b;  
    })(100, 200);  
    document.write("함수 실행 결과: " + sum);  
</script>  
(... 생략 ...)
```

매개변수
인수

함수 표현식

화살표 함수

- ES6 이후 사용하는 => 표기법
- 익명 함수에서만 사용할 수 있음

기본형 (매개변수) => { 함수 내용 }

```
const hi = function() {  
  return alert("안녕하세요?");  
}
```



```
const hi = () => { return alert("안녕하세요");};
```

return 생략해서



```
const hi = () => alert("안녕하세요");
```

```
let hi = function(user) {  
  document.write (user + "님, 안녕하세요?");  
}
```



```
let hi = user => { document.write (user + "님, 안녕하세요?"); }
```

```
let sum = function(a, b) {  
  return a + b;  
}
```



```
let sum = (a, b) => a + b;
```

이벤트와 이벤트 처리기

이벤트

- 웹 브라우저나 사용자가 행하는 동작
- 웹 문서 영역안에서 이루어지는 동작만 가리킴
- 주로 마우스나 키보드를 사용할 때, 웹 문서를 불러올 때, 폼에 내용을 입력할 때 발생

표 15-1 마우스 이벤트

종류	설명
click	사용자가 HTML 요소를 클릭할 때 이벤트가 발생합니다.
dblclick	사용자가 HTML 요소를 더블클릭할 때 이벤트가 발생합니다.
mousedown	사용자가 요소 위에서 마우스 버튼을 눌렀을 때 이벤트가 발생합니다.
mousemove	사용자가 요소 위에서 마우스 포인터를 움직일 때 이벤트가 발생합니다.
mouseover	마우스 포인터가 요소 위로 옮겨질 때 이벤트가 발생합니다.
mouseout	마우스 포인터가 요소를 벗어날 때 이벤트가 발생합니다.
mouseup	사용자가 요소 위에 놓인 마우스 버튼에서 손을 뗄 때 이벤트가 발생합니다.

표 15-2 키보드 이벤트

종류	설명
keydown	사용자가 키를 누르는 동안 이벤트가 발생합니다.
keypress	사용자가 키를 눌렀을 때 이벤트가 발생합니다.
keyup	사용자가 키에서 손을 뗄 때 이벤트가 발생합니다.

표 15-3 문서 로딩 이벤트

종류	설명
abort	문서가 완전히 로딩되기 전에 불러오기를 멈췄을 때 이벤트가 발생합니다.
error	문서 가 정확히 로딩되지 않았을 때 이벤트가 발생합니다.
load	문서 로딩이 끝나면 이벤트가 발생합니다.
resize	문서 화면 크기가 바뀌었을 때 이벤트가 발생합니다.
scroll	문서 화면이 스크롤되었을 때 이벤트가 발생합니다.
unload	문서에서 벗어날 때 이벤트가 발생합니다.

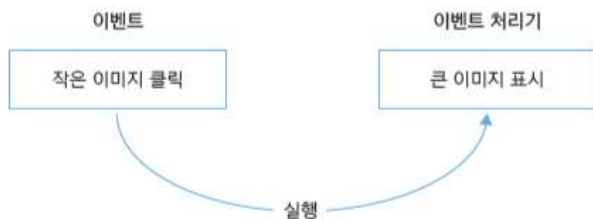
표 15-4 폼 이벤트

종류	설명
blur	폼 요소에 포커스를 잃었을 때 이벤트가 발생합니다.
change	목록이나 체크 상태 등이 변경되면 이벤트가 발생합니다. <input>, <select>, <textarea> 태그에서 사용합니다.
focus	폼 요소에 포커스가 놓였을 때 이벤트가 발생합니다. <label>, <select>, <textarea>, <button> 태그에서 사용합니다.
reset	폼이 리셋되었을 때 이벤트가 발생합니다.
submit	submit 버튼을 클릭했을 때 이벤트가 발생합니다.

이벤트와 이벤트 처리기

이벤트 처리기

- 이벤트가 발생했을 때 처리하는 함수
- 이벤트 핸들러(event handler)라고도 함



- 이벤트가 발생한 HTML 태그에 이벤트 처리기를 직접 연결

기본형 <태그 on이벤트명 = "함수명">

Do it! 버튼을 클릭하면 알림 창 표시하기 예제 파일 15\event-1.html

```
(... 생략 ...)  
<body>  
  <ul>  
    <li><a href="#" onclick="alert('버튼을 클릭했습니다.')">Green</a></li>  
    <li><a href="#" onclick="alert('버튼을 클릭했습니다.')">Orange</a></li>  
    <li><a href="#" onclick="alert('버튼을 클릭했습니다.')">Purple</a></li>  
  </ul>  
(... 생략 ...)
```

The screenshot shows the web application in two states. On the left, there are three buttons labeled 'Green', 'Orange', and 'Purple'. The 'Green' button is highlighted with an orange border. On the right, an alert dialog box is displayed with the text '127.0.0.1:5500 내용: 버튼을 클릭했습니다.' (Content: Button clicked). A '확인' (OK) button is visible in the bottom right corner of the dialog.

DOM을 이용한 이벤트 처리기

DOM을 사용하면 자바스크립트가 주인이 되어 HTML의 요소를 가져와서 이벤트 처리기를 연결

기본형 웹 요소.onclick = 함수;

 **Do it!** 버튼 클릭해서 글자색 바꾸기 예제 파일 15\event-3.html

(... 생략 ...)

```
<body>
  <button id="change">글자색 바꾸기</button>
  <p>Reprehenderit ..... laborum quis.</p>
```

방법 1

방법 3

방법 2

// 방법 1: 웹 요소를 변수로 지정 & 미리 만든 함수 사용

```
var changeBtn = document.querySelector("#change");
changeBtn.onclick = changeColor;
```

함수 이름 뒤에 괄호가 없다는 것을 기억하세요!

```
function changeColor() {
  document.querySelector("p").style.color = "#f00";
}
```

// 방법 3: 함수를 직접 선언

```
document.querySelector("#change").onclick = function() {
  document.querySelector("p").style.color = "#f00";
};
```

// 방법 2: 웹 요소를 따로 변수로 만들지 않고 사용

```
document.querySelector("#change").onclick = changeColor;
```

```
function changeColor() {
  document.querySelector("p").style.color = "#f00";
}
```


16. 자바스크립트와 객체

16-1 객체 알아보기

16-2 자바스크립트와 내장 객체

16-3 브라우저와 관련된 객체



객체 알아보기

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

객체의 인스턴스 만들기

- 객체는 객체 자체가 아니라 인스턴스 형태로 만들어서 사용
- 인스턴스 : 객체를 틀처럼 사용해서 같은 모양으로 찍어낸 것.

기본형 new 객체명

예) Date 객체의 인스턴스를 만들어서 날짜와 시간 표시하기

```
var now = new Date(); // Date 인스턴스 만들고 변수에 저장하기
document.write("현재 시각은 " + now);
```

현재 시각은 Wed Nov 11 2020 15:58:24
GMT+0900 (대한민국 표준시)

객체 알아보기

프로퍼티(property)와 메서드(method)

- 프로퍼티 : 객체의 특징이나 속성
- 메서드 : 객체에서 할 수 있는 동작



프로퍼티	메서드
제조사	시동 걸기
모델명	움직이기
색상	멈추기
배기량	주차하기

그림 16-2 자동차의 프로퍼티와 메서드

- 2) now 변수에 저장 ← 1) Date 객체의 인스턴스를 만들어서

```
var now = new Date();  
document.write("현재 시각은 " + now.toLocaleString());
```

- 3) Date 객체의 메서드 사용

내장 객체 – Array 객체

배열 만들기

1) 초깃값이 없는 경우

```
var numbers = new Array();    // 배열의 크기를 지정하지 않음  
var numbers = new Array(4);   // 배열의 크기를 지정함
```

2) 초깃값이 있는 경우

```
var numbers = ["one", "two", "three", "four"];    // 배열 선언  
var numbers = Array("one", "two", "three", "four"); // Array 객체를 사용한 배열 선언
```

배열의 크기 – Array 객체의 length 프로퍼티

```
var numbers = ["one", "two", "three", "four"]; // 배열 선언  
  
for(i = 0; i < numbers.length; i++) {          // 배열 선언  
    document.write("<p>" + numbers[i] + "</p>");  
}
```

배열명 옆에 마침표(.)와
length 프로퍼티를
작성하면 됩니다!

배열의 각 요소

one
two
three
four

내장 객체 – Array 객체

배열 만들기 – Array 객체의 인스턴스 만들기

1) 초깃값이 없는 경우

```
var numbers = new Array();    // 배열의 크기를 지정하지 않음  
var numbers = new Array(4);   // 배열의 크기를 지정함
```

2) 초깃값이 있는 경우

```
var numbers = ["one", "two", "three", "four"];    // 배열 선언  
var numbers = Array("one", "two", "three", "four"); // Array 객체를 사용한 배열 선언
```

배열의 크기 – Array 객체의 length 프로퍼티

```
var numbers = ["one", "two", "three", "four"]; // 배열 선언  
  
for(i = 0; i < numbers.length; i++) {          // 배열 선언  
    document.write("<p>" + numbers[i] + "</p>");  
}
```

배열명 옆에 마침표(.)와
length 프로퍼티를
작성하면 됩니다!

배열의 각 요소

one
two
three
four

내장 객체 – Array 객체

Array 객체의 메서드

종류	설명
concat	기존 배열에 요소를 추가해 새로운 배열을 만듭니다.
every	배열의 모든 요소가 주어진 함수에 대해 참이면 true를 반환하고 그렇지 않으면 false를 반환합니다.
filter	배열 요소 중에서 주어진 필터링 함수에 대해 true인 요소만 골라 새로운 배열을 만듭니다.
forEach	배열의 모든 요소에 대해 주어진 함수를 실행합니다.
indexOf	주어진 값과 일치하는 값이 있는 배열 요소의 첫 인덱스를 찾습니다.
join	배열 요소를 문자열로 합칩니다. 이때 구분자를 지정할 수 있습니다.
push	배열의 맨 끝에 새로운 요소를 추가한 후 새로운 length를 반환합니다.
unshift	배열의 시작 부분에 새로운 요소를 추가합니다.
pop	배열의 마지막 요소를 꺼내 그 값을 결과로 반환합니다.
shift	배열에서 첫 번째 요소를 꺼내 그 값을 결과로 반환합니다.
splice	배열에 요소를 추가하거나 삭제합니다.
slice	배열에서 특정한 부분만 잘라 냅니다.
reverse	배열의 배치 순서를 역순으로 바꿉니다.
sort	배열 요소를 지정한 조건에 따라 정렬합니다.
toString	배열에서 지정한 부분을 문자열로 반환합니다. 이때 각 요소는 쉼표(,)로 구분합니다.

내장 객체 – Array 객체

배열끼리 합치는 concat() 메서드

- 서로 다른 배열 2개를 합쳐서 새로운 배열을 만들
- 기존 배열에 영향을 주지 않음

```
// 배열 2개 합치기
var nums = [1, 2, 3];
var chars = ["a", "b", "c", "d"];

var numsChars = nums.concat(chars);
var charsNums = chars.concat(nums);
document.write("nums에 chars 합치면: ", numsChars, "<br> chars에 nums 합치면: ",
charsNums);
```

nums에 chars 합치면: 1,2,3,a,b,c,d
chars에 nums 합치면: a,b,c,d,1,2,3

배열 요소끼리 합치는 join() 메서드

- 배열 요소를 연결해서 하나의 문자열로 만들
- 요소 사이에 원하는 구분자를 넣을 수 있음.
- 구분자를 지정하지 않으면 쉼표(,)로 구분

```
// 배열 안의 요소 합치기
var nums = [1, 2, 3];
var chars = ["a", "b", "c", "d"];

.....
var string1 = nums.join();
document.write("구분자 없이: ", string1);
document.write("<br>");
var string2 = chars.join('/');
document.write("'/' 구분자 지정: ", string2);
```

구분자 없이: 1,2,3
'/' 구분자 지정: a/b/c/d

내장 객체 – Array 객체

새로운 요소를 추가하는 push(), unshift() 메서드

- push() 메서드 : 배열 맨 끝에 요소 추가
- unshift() 메서드 : 배열 맨 앞에 요소 추가
- 배열의 길이가 반환, 기존 배열이 바뀜

```
// 요소 추가하기 – 새로운 length값 반환
var nums = [1, 2, 3];

.....
var ret1 = nums.push(4, 5); // 배열 맨 끝에 새로운 요소를 추가
document.write("length: ", ret1, " | 배열: ", nums);
document.write("<br>");
var ret2 = nums.unshift(0); // 배열 맨 앞에 새로운 요소를 추가
document.write("length: ", ret2, " | 배열: ", nums);
```

length: 5 | 배열: 1,2,3,4,5
length: 6 | 배열: 0,1,2,3,4,5

배열에서 요소를 꺼내는 pop(), shift() 메서드

- pop() 메서드 : 배열 뒤쪽에서 요소를 꺼냄
- shift() 메서드 : 배열 앞쪽에서 요소를 꺼냄
- 꺼낸 요소를 반환, 기존 배열을 꺼낸 요소가 빠진 상태로 변경됨

```
// 요소 꺼내기 – 꺼낸 요소값 반환
var chars = ["a", "b", "c", "d"];

.....
var popped1 = chars.pop(); // 마지막 요소 꺼냄
document.write("꺼낸 요소: ", popped1, " | 배열: ", chars);
document.write("<br>");
var popped2 = chars.shift(); // 1번째 요소 꺼냄
document.write("꺼낸 요소: ", popped2, " | 배열: ", chars);
```

꺼낸 요소: d | 배열: a,b,c
꺼낸 요소: a | 배열: b,c

내장 객체 – Array 객체

중간에 요소를 추가하거나 삭제하는 splice() 메서드

- 배열 중간에 2개 이상의 요소를 추가하거나 삭제할 수 있음
- 새로운 배열이 결과값으로 반환됨

1) 괄호 안에 인수가 1개일 경우

인수가 지정한 인덱스의 요소부터 배열의 맨 끝 요소까지 삭제

```
// 인수가 1개인 경우
var numbers = [1, 2, 3, 4, 5];
var newNumbers = numbers.splice(2);
document.write("반환된 배열: " + newNumbers + "<br>");
document.write("변경된 배열: " + numbers);
```

반환된 배열: 3,4,5
변경된 배열: 1,2

2) 괄호 안에 인수가 2개일 경우

- 첫 번째 인수는 인덱스값이고 두 번째 인수는 삭제할 요소의 개수
- 메서드를 실행한 후에는 삭제한 요소를 반환하고, 기존 배열은 나머지 요소만 남음

```
// 인수가 2개일 경우
var study = ["html", "css", "web", "jquery"];
var newStudy = study.splice(2, 1);
document.write("반환된 배열 : " + newStudy + "<br>");
document.write("변경된 배열 : " + study);
```

반환된 배열: web
변경된 배열: html,css,jquery

3) 괄호 안에 인수가 3개 이상일 경우

첫 번째 인수는 배열에서 삭제할 시작 위치, 두 번째 인수는 삭제할 개수, 세 번째 인수부터는 삭제한 위치에 새로 추가할 요소를 지정

```
// 인수가 3개 이상인 경우
var newStudy2 = study.splice(2, 1, "js");
document.write("반환된 배열: " + newStudy2 + "<br>");
document.write("변경된 배열: " + study);
```

반환된 배열: jquery
변경된 배열: html,css,js

내장 객체 – Array 객체

기존 배열을 바꾸지 않으면서 삭제하는 slice() 메서드

- 요소를 여러 개 꺼낼 수 있음
- 요소를 삭제한 후에도 기존 배열이 바뀌지 않음

1) 괄호 안에 인수가 1개일 경우

인수가 지정한 인덱스의 요소부터 마지막 요소까지 꺼내서 반환

```
var colors = ["red", "green", "blue", "white", "black"];  
var colors2 = colors.slice(2); //인덱스 값이 2인 요소부터 마지막 요소까지 꺼내기  
document.write(colors2);
```

blue,white,black

2) 괄호 안에 인수가 2개일 경우

첫 번째 인수는 시작 인덱스, 두 번째 인수는 끝 인덱스의 직전 인덱스

```
var colors = ["red", "green", "blue", "white", "black"];  
.....  
var colors3 = colors.slice(2, 4); // 인덱스값이 2부터 3인 요소까지 꺼내기  
document.write(colors3);
```

blue,white

slice() 메서드는 기존 배열에 영향을 주지 않지만, splice() 메서드는 요소를 추가·삭제하면 기존 배열 자체가 수정됨
→ 기존 배열에서 꺼낸 요소로 새로운 배열을 만들어 사용하려면 slice() 메서드를 사용하고, 기존 배열의 일부 요소만 삭제하려면 splice() 메서드를 선택하는 것이 좋다

내장 객체 – Date 객체

Date 객체 인스턴스 만들기

현재 날짜로 설정할 경우



Data 객체로 현재 날짜 나타내기

```
new Date();
```

특정 날짜로 설정할 경우 – 괄호 안에 날짜 또는 날짜와 시간 입력



Data 객체로 특정 날짜 나타내기

```
new Date("2020-02-25")
```



Data 객체로 특정 날짜와 시간 나타내기

```
new Date("2020-02-25T18:00:00")
```

자바스크립트의 날짜와 시간 입력 방식

1) YYYY-MM-DD 형식

```
new Date("2020")  
new Date("2020-02")  
new Date("2020-02-25")
```

2) YYYY-MM-DDTHH 형식

```
new Date("2020-02-25T18:00:00")  
new Date("2020-02-25T18:00:00Z")
```

3) MM/DD/YYYY 형식

```
new Date("02/25/2020")
```

4) 이름 형식

```
new Date("Mon Jan 20 2020 15:00:41 GMT+0900 (대한민국 표준시)")
```

내장 객체 – Date 객체

Date 객체의 메서드

날짜/시간 정보를 가져오는 메서드,
날짜/시간 정보를 설정하는 메서드,
날짜/시간 형식을 바꿔주는 메서드로 구분됨

	구분	설명
날짜·시간 정보 가져오기	getFullYear()	연도를 4자리 숫자로 표시합니다.
	getMonth()	0~11 사이의 숫자로 월을 표시합니다. 0부터 1월이 시작되고 11은 12월입니다.
	getDate()	1~31 사이의 숫자로 일을 표시합니다.
	getDay()	0~6 사이의 숫자로 요일을 표시합니다. 0부터 일요일이 시작되고 6은 토요일입니다.
	getTime()	1970년 1월 1일 자정 이후의 시간을 밀리 초(1/1000초)로 표시합니다.
	getHours()	0~23 사이의 숫자로 시를 표시합니다.
	getMinutes()	0~59 사이의 숫자로 분을 표시합니다.
	getSeconds()	0~59 사이의 숫자로 초를 표시합니다.
	getMilliseconds()	0~999 사이의 숫자로 밀리초를 표시합니다.
날짜·시간 설정하기	setFullYear()	연도를 4자리 숫자로 설정합니다.
	setMonth()	0~11 사이의 숫자로 월을 설정합니다. 0부터 1월이 시작되고 11은 12월입니다.
	setDate()	1~31 사이의 숫자로 일을 설정합니다.
	setTime()	1970년 1월 1일 자정 이후의 시간을 밀리초로 설정합니다.
	setHours()	0~23 사이의 숫자로 시를 설정합니다.
	setMinutes()	0~59 사이의 숫자로 분을 설정합니다.
	setSeconds()	0~59 사이의 숫자로 초를 설정합니다.
	setMilliseconds()	0~999 사이의 숫자로 밀리초를 설정합니다.
날짜·시간 형식 바꾸기	toLocaleString()	현재 날짜와 시간을 현지 시간(local time)으로 표시합니다.
	toString()	Data 객체 타입을 문자열로 표시합니다.

내장 객체 – 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

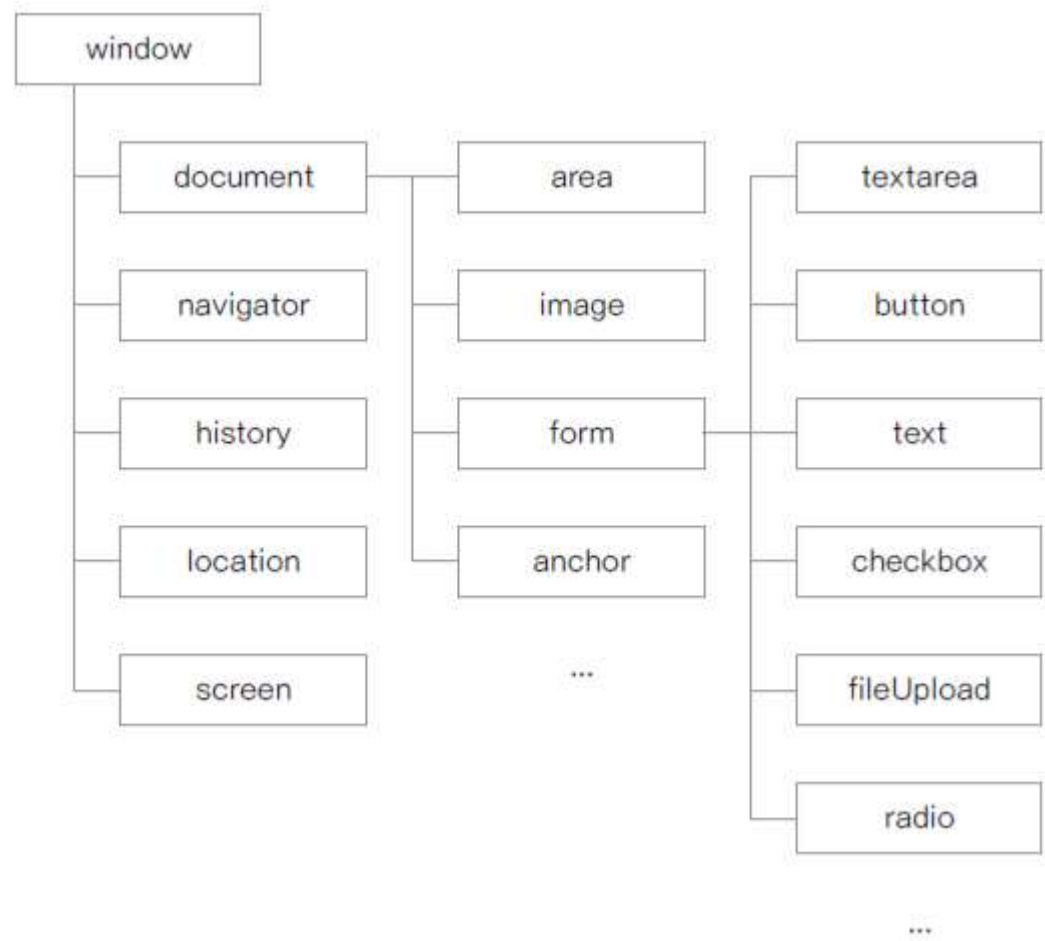
종류	설명
E	오일러 상수
PI	원주율(π) (약 3.141592653589793의 값)
SQRT2	$\sqrt{2}$ (약 1.4142135623730951의 값)
SQRT1_2	$1/\sqrt{2}$ (약 0.7071067811865476의 값)
LN2	\log_2 (약 0.6931471805599453의 값)
LN10	\log_{10} (약 2.302585092994046의 값)
LOG2E	$\log_2 e$ (약 1.4426950408889634의 값)
LOG10E	$\log_{10} e$ (약 0.4342944819032518의 값)

Math 객체의 메서드

종류	설명
abs()	절댓값을 반환합니다.
acos()	아크 코사인(arc cosine)값을 반환합니다.
asin()	아크 사인(arc sine)값을 반환합니다.
atan()	아크 탄젠트(arc tangent)값을 반환합니다.
atan2()	아크 탄젠트(arc tangent)값을 반환합니다.
ceil()	매개변수의 소수점 이하 부분을 올립니다.
cos()	코사인(cosine)값을 반환합니다.
exp()	지수 함수를 나타냅니다.
floor()	매개변수의 소수점 이하 부분을 버립니다.
log()	매개변수에 대한 로그(log)값을 반환합니다.
max()	매개변수 중 최댓값을 반환합니다.
min()	매개변수 중 최솟값을 반환합니다.
pow()	매개변수의 지수값을 반환합니다.
random()	0과 1 사이의 무작위 수를 반환합니다.
round()	매개변수의 소수점 이하 부분을 반올림합니다.
sin()	사인(sine)값을 반환합니다.
sqrt()	매개변수에 대한 제곱근을 반환합니다.
tan()	탄젠트(tangent)값을 반환합니다.

브라우저 관련 객체

브라우저 관련 객체의 계층 구조



종류	설명
window	브라우저 창이 열릴 때마다 하나씩 만들어집니다. 브라우저 창 안의 요소 중에서 최상위에 있습니다.
document	웹 문서마다 하나씩 있으며 <body> 태그를 만나면 만들어집니다. HTML 문서의 정보가 담겨 있습니다.
navigator	현재 사용하는 브라우저의 정보가 들어 있습니다.
history	현재 창에서 사용자의 방문 기록을 저장합니다.
location	현재 페이지의 URL 정보가 담겨 있습니다.
screen	현재 사용하는 화면 정보를 다룹니다.

브라우저 관련 객체 – window 객체

window 객체의 프로퍼티

주로 웹 브라우저 창의 정보를 가져오거나 값을 바꿀 때 사용

종류	설명
document	브라우저 창에 표시된 웹 문서에 접근할 수 있습니다.
frameElement	현재 창이 다른 요소 안에 포함되어 있을 경우 그 요소를 반환하고, 반대로 포함되어 있지 않으면 null을 반환합니다.
innerHeight	내용 영역의 높이를 나타냅니다.
innerWidth	내용 영역의 너비를 나타냅니다.
localStorage	웹 브라우저에서 데이터를 저장하는 로컬 스토리지를 반환합니다.
location	window 객체의 위치 또는 현재 URL을 나타냅니다.
name	브라우저 창의 이름을 가져오거나 수정합니다.
outerHeight	브라우저 창의 바깥 높이를 나타냅니다.
outerWidth	브라우저 창의 바깥 너비를 나타냅니다.
pageXOffset	스크롤했을 때 수평으로 이동하는 픽셀 수로 scrollX와 같습니다.
pageYOffset	스크롤했을 때 수직으로 이동하는 픽셀 수로 scrollY와 같습니다.
parent	현재 창이나 서브 프레임의 부모입니다.
screenX	브라우저 창의 왼쪽 테두리가 모니터 왼쪽 테두리에서 떨어져 있는 거리를 나타냅니다.
screenY	브라우저 창의 위쪽 테두리가 모니터 위쪽 테두리에서 떨어져 있는 거리를 나타냅니다.
scrollX	스크롤했을 때 수평으로 이동하는 픽셀 수를 나타냅니다.
scrollY	스크롤했을 때 수직으로 이동하는 픽셀 수를 나타냅니다.
sessionStorage	웹 브라우저에서 데이터를 저장하는 세션 스토리지를 반환합니다.

window 객체의 메서드

window 객체는 기본 객체이므로 'window.'를 생략하고 메서드 이름만 사용해도 됨

종류	설명
alert()	알림 창을 표시합니다.
blur()	현재 창에서 포커스를 제거합니다.
close()	현재 창을 닫습니다.
confirm()	[확인], [취소] 버튼이 있는 확인 창을 표시합니다.
focus()	현재 창에 포커스를 부여합니다.
moveBy()	현재 창을 지정한 크기만큼 이동합니다.
moveTo()	현재 창을 지정한 좌표로 이동합니다.
open()	새로운 창을 엽니다.
postMessage()	메시지를 다른 창으로 전달합니다.
print()	현재 문서를 인쇄합니다.
prompt()	프롬프트 창에 입력한 텍스트를 반환합니다.
resizeBy()	지정한 크기만큼 현재 창의 크기를 조절합니다.
resizeTo()	동적으로 브라우저 창의 크기를 조절합니다.
scroll()	문서에서 특정 위치로 스크롤합니다.
scrollBy()	지정한 크기만큼씩 스크롤합니다.
scrollTo()	지정한 위치까지 스크롤합니다.
sizeToContent()	내용에 맞게 창의 크기를 맞춥니다.
stop()	로딩을 중지합니다.

브라우저 관련 객체 – navigator 객체, history 객체

navigator 객체

- 사용하는 브라우저가 많아지고, 웹 애플리케이션이 등장하면서 navigator 객체에 여러 프로퍼티가 등장하고 있음.
- 일부 브라우저에서만 지원하는 프로퍼티도 있음

주요 프로퍼티

종류	설명
battery	배터리 충전 상태를 알려 줍니다.
cookieEnabled	쿠키 정보를 무시하면 false, 허용하면 true를 반환합니다.
geolocation	모바일 기기를 이용한 위치 정보를 나타냅니다.
language	브라우저 UI의 언어 정보를 나타냅니다.
oscpu	현재 운영체제 정보를 나타냅니다.
userAgent	현재 브라우저 정보를 담고 있는 사용자 에이전트 문자열입니다.

history 객체

방문한 사이트 주소가 배열 형태로 저장됨

프로퍼티와 메서드

구분		설명
프로퍼티	length	현재 브라우저 창의 history 목록에 있는 항목의 개수, 즉 방문한 사이트 개수가 저장됩니다.
메서드	back()	history 목록에서 이전 페이지를 현재 화면으로 불러옵니다
	forward()	history 목록에서 다음 페이지를 현재 화면으로 불러옵니다
	go()	history 목록에서 현재 페이지를 기준으로 상대적인 위치에 있는 페이지를 현재 화면으로 불러옵니다. 예를 들어 history.go(1)은 다음 페이지를 가져오고, history.go(-1)은 이전 페이지를 불러옵니다.

브라우저 관련 객체 – location 객체, screen 객체

location 객체

- 현재 문서의 URL 주소 정보가 담겨 있음
- 이 정보를 편집해서 브라우저 창에 열 사이트/문서 지정

구분		설명
프로퍼티	hash	URL 중에서 #로 시작하는 해시 부분의 정보를 담고 있습니다.
	host	URL의 호스트 이름과 포트 번호를 담고 있습니다.
	hostname	URL의 호스트 이름이 저장됩니다.
	href	전체 URL입니다. 이 값을 변경하면 해당 주소로 이동할 수 있습니다.
	pathname	URL 경로가 저장됩니다.
	port	URL의 포트 번호를 담고 있습니다.
	protocol	URL의 프로토콜을 저장합니다.
	password	도메인 이름 앞에 username과 password를 함께 입력해서 접속하는 사이트의 URL 일 경우에 password 정보를 저장합니다.
	search	URL 중에서 ?로 시작하는 검색 내용을 저장합니다.
	username	도메인 이름 앞에 username을 함께 입력해서 접속하는 사이트의 URL일 경우에 username 정보를 저장합니다.
메서드	assign()	현재 문서에 새 문서 주소를 할당해서 새 문서를 가져옵니다.
	reload()	현재 문서를 다시 불러옵니다.
	replace()	현재 문서의 URL을 지우고 다른 URL의 문서로 교체합니다.
	toString()	현재 문서의 URL을 문자열로 반환합니다.

screen 객체

사용자의 화면 크기, 정보

구분		설명
프로퍼티	availHeight	UI 영역(예를 들어 윈도우의 작업 표시줄이나 Mac의 독)을 제외한 영역의 높이를 나타냅니다.
	availWidth	UI 영역을 제외한 내용 표시 영역의 너비를 나타냅니다.
	colorDepth	화면에서 픽셀을 렌더링할 때 사용하는 색상 수를 나타냅니다.
	height	UI 영역을 포함한 화면의 높이를 나타냅니다.
	orientation	화면의 현재 방향을 나타냅니다.
	pixelDepth	화면에서 픽셀을 렌더링할 때 사용하는 비트 수를 나타냅니다.
	width	UI 영역을 포함한 화면의 너비를 나타냅니다.
메서드	lockOrientation()	화면 방향을 잠금니다.
	unlockOrientation()	화면 방향 잠금을 해제합니다.

17. 문서 객체 모델(DOM)

17-1 문서 객체 모델 알아보기

17-2 DOM 요소에 접근하고 속성 가져오기

17-3 DOM에서 이벤트 처리하기

17-4 DOM에서 노드 추가, 삭제하기



문서 객체 모델 알아보기

문서 객체 모델이란

자바스크립트를 이용하여 웹 문서에 접근하고 제어할 수 있도록 객체를 사용해 웹 문서를 체계적으로 정리하는 방법

웹 문서와 그 안의 모든 요소를 '객체'로 인식하고 처리함

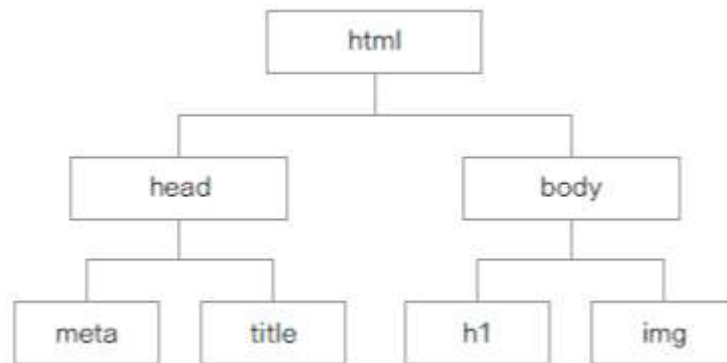
예) 웹 문서 전체는 document 객체, 삽입한 이미지는 image 객체



Do it! HTML의 요소 관계 알아보기

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>DOM Tree 알아보기</title>
</head>
<body>
  <h1>Do it!</h1>
  
</body>
</html>
```

요소의 계층 관계



문서 객체 모델 알아보기

DOM 트리

- 웹 문서에 있는 요소들 간의 부모, 자식 관계를 계층 구조로 표시한 것
- 나무 형태가 되기 때문에 "DOM 트리"라고 함.
- 노드(node) : DOM 트리에서 가지가 갈라져 나간 항목
- 루트 노트(root node) : DOM 트리의 시작 부분(html)

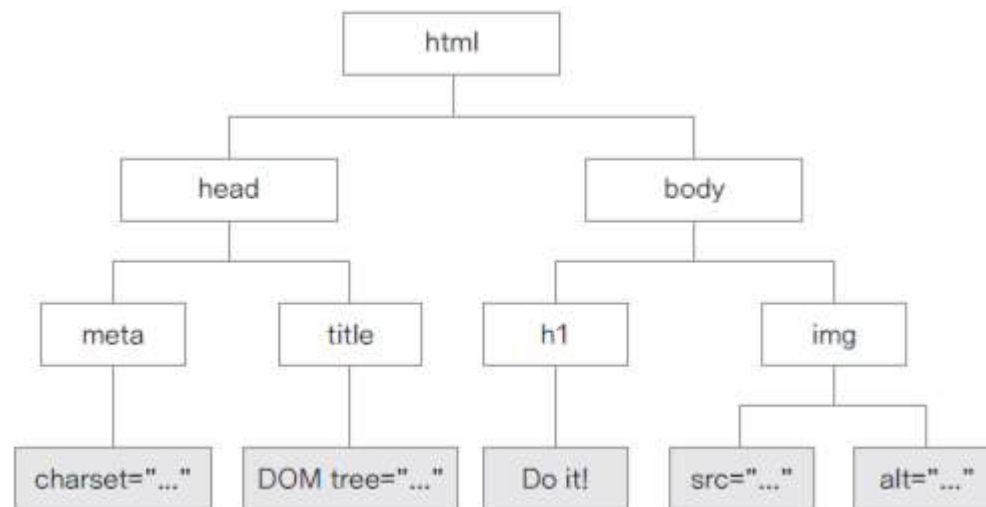


Do it! HTML의 요소 관계 알아보기

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>DOM Tree 알아보기</title>
</head>
<body>
  <h1>Do it!</h1>
  
</body>
</html>
```

DOM 을 구성하는 원칙

- 모든 HTML 태그는 요소(element) 노드
- 웹 문서의 텍스트 내용은 요소 노드의 자식 노드인 텍스트(text) 노드
- 태그의 속성은 요소 노드의 자식 노드인 속성(attribute) 노드
- 주석은 주석(comment) 노드

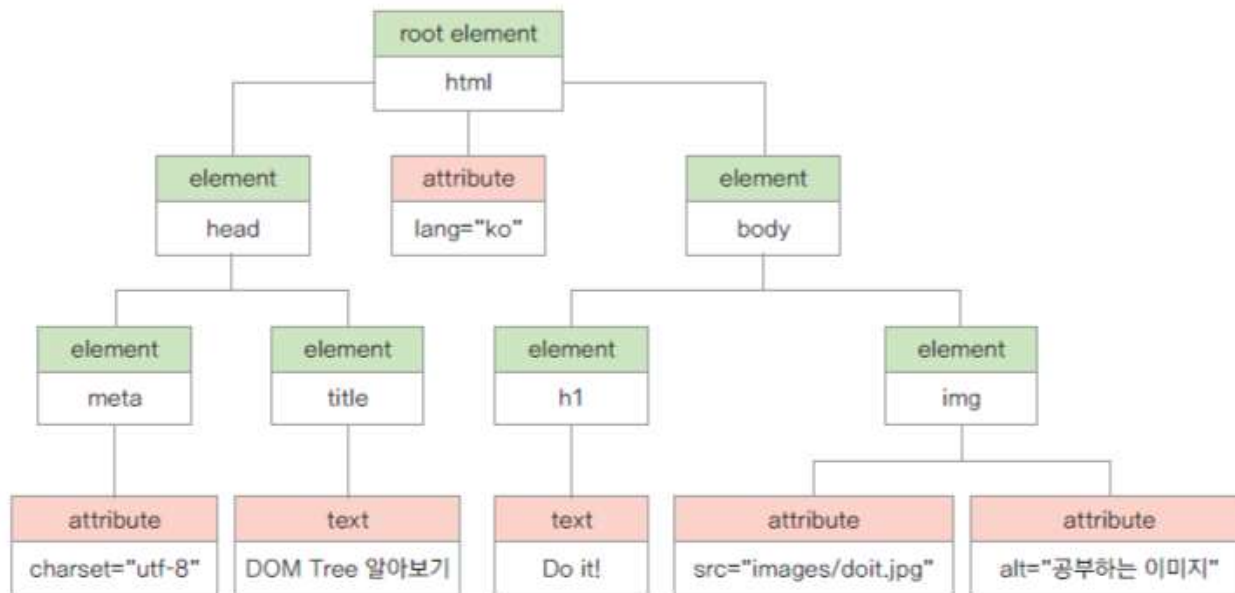


문서 객체 모델 알아보기



Do it! HTML의 요소 관계 알아보기

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>DOM Tree 알아보기</title>
</head>
<body>
  <h1>Do it!</h1>
  
</body>
</html>
```



DOM 요소에 접근하고 속성 가져오기

DOM 요소에 접근하기

웹 문서에서 원하는 요소를 찾아가는 것을 “접근한다(access)”고 함

getElementById() 메서드

기본형 요소명.getElementById("id명")

getElementsByClassName() 메서드

기본형 요소명.getElementsByClassName("class명")

getElementsByTagName() 메서드

기본형 요소명.getElementsByTagName("태그명")

- 반환 값이 2개 이상일 수 있음
- HTMLCollection 객체에 저장됨

querySelector() 메서드, querySelectorAll() 메서드

기본형 노드.querySelector(선택자)
노드.querySelectorAll(선택자 또는 태그)

- querySelector() 메서드는 한 개의 값만 반환
- querySelectorAll() 메서드는 반환 값이 여러 개일 때 모두 반환 → 노드 리스트로 저장됨
- id 이름 앞에는 해시 기호(#), class 이름 앞에는 마침표(.), 태그는 기호 없이 태그명 사용

DOM 요소에 접근하고 속성 가져오기

innerText, innerHTML 프로퍼티

웹 요소의 내용을 수정하는 프로퍼티

- innerText : 텍스트 내용 지정
- innerHTML : HTML 태그까지 포함해서 텍스트 내용 지정

기본형 요소명.innerText = 내용
요소명.innerHTML = 내용

```
<button onclick="inntext()">innerText로 표시하기</button>
<button onclick="innhtml()">innerHTML로 표시하기</button>
<h1>현재 시각: </h1>
<div id="current"></div>

<script>
  var now = new Date();

  function inntext(){
    document.getElementById("current").innerText = now;
  }
  f      innhtml() {
    document.getElementById("current").i      = "<em>" + now + "</em>";
  }
</script>
```



DOM 요소에 접근하고 속성 가져오기

getAttribute() 메서드, setAttribute() 메서드

- getAttribute() 메서드 : 속성 노드의 값을 가져옴
- setAttribute() 메서드 : 속성 노드의 값을 바꿈

기본형 `getAttribute("속성명")`

기본형 `setAttribute("속성명", "값")`

```
<div id="prod-pic">
  
  *****
</div>

<script>
function displaySrc() {
  var cup = document.querySelector("#cup");           // id="cup"인 요소에 접근하기
  alert("이미지 소스: " + cup.getAttribute("src"));    // cup 속성을 알림 창에 표시
}
</script>
```

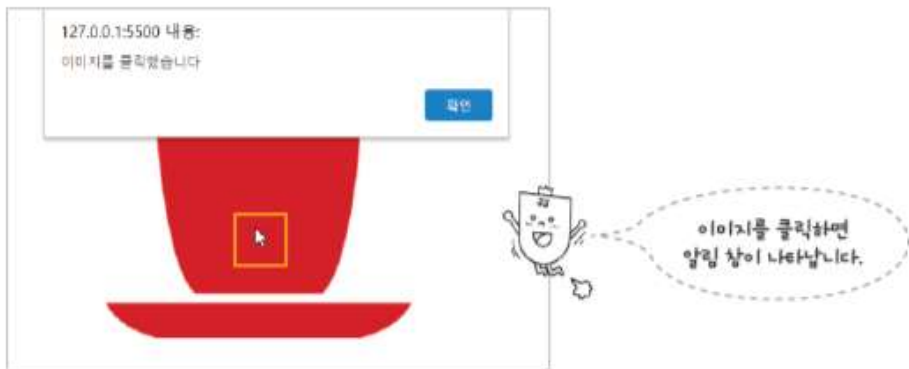


DOM에서 이벤트 처리하기

DOM 요소에 함수 직접 연결하기

DOM 요소에 이벤트 처리기 함수를 직접 연결

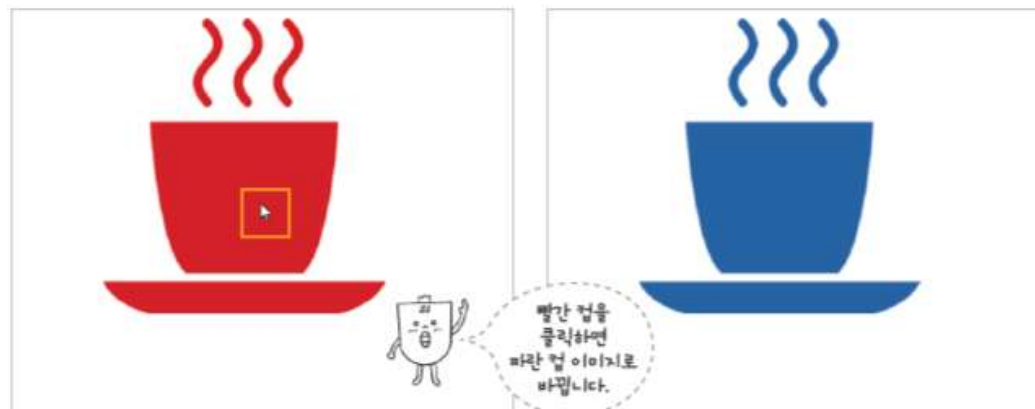
```
(... 생략 ...)  
  
  
.....  
<script>  
  var cup = document.querySelector("#cup");  
  cup.onclick = function(){  
    alert("이미지를 클릭했습니다");  
  }  
</script>
```



함수 이름을 사용해 연결하기

- 함수를 따로 정의해 놓았다면 DOM 요소에 함수 이름을 사용해 연결
- 이 때 함수 이름 다음에 괄호를 추가하지 않음

```
  
  
.....  
<script>  
  var cup = document.querySelector("#c"); // id=cup 요소를 가져옴  
  cup. = changePic; // cup을 클릭하면 ChangPic 함수를 실행  
  
  function changePic() {  
    cup.src = "images/cup-2.png"; // cup 요소의 경로를 다른 이미지 경로로 바꿈  
  }  
</script>
```



DOM에서 이벤트 처리하기

DOM의 event 객체

웹 문서에서 이벤트 발생한 요소가 무엇인지,
어떤 이벤트가 발생했는지 등의 정보가 담긴 객체



Do it! 이미지에서 클릭한 위치 알아내기

예제 파일 17\event-3.html

(... 생략 ...)

```

```

.....

```
<script>
```

```
var cup = document.querySelector("#cup");
```

```
cup.onclick = function(event) {
```

```
    alert("이벤트 유형: " + event.type + ", 이벤트 발생 위치: " + event.pageX + ", " +  
event.pageY);
```

```
}
```

```
</script>
```

127.0.0.1:5500 내용:

이벤트 유형: click, 이벤트 발생 위치: 351,211

확인



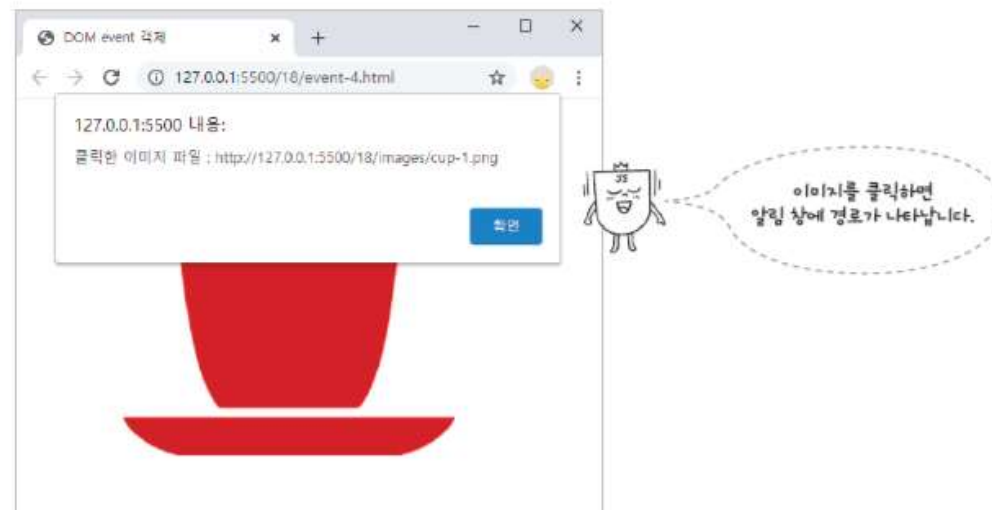
이번에는 이미지를 클릭하면
알림 상에 클릭한 위치가
나타납니다.

구분		설명
프로퍼티	altKey	이벤트가 발생할 때 (Alt) 를 눌렀는지 여부를 boolean값으로 반환합니다.
	button	마우스에서 누른 버튼의 키값을 반환합니다.
	charCode	keypress 이벤트가 발생할 때 어떤 키를 눌렀는지 유니코드값으로 반환합니다.
	clientX	이벤트가 발생한 가로 위치를 반환합니다.
	clientY	이벤트가 발생한 세로 위치를 반환합니다.
	ctrlKey	이벤트가 발생했을 때 (Ctrl) 를 눌렀는지 여부를 boolean값으로 반환합니다.
	pageX	현재 문서 기준으로 이벤트가 발생한 가로 위치를 반환합니다.
	pageY	현재 문서 기준으로 이벤트가 발생한 세로 위치를 반환합니다.
	screenX	현재 화면 기준으로 이벤트가 발생한 가로 위치를 반환합니다.
	screenY	현재 화면 기준으로 이벤트가 발생한 세로 위치를 반환합니다.
	shiftKey	이벤트가 발생할 때 (Shift) 를 눌렀는지 여부를 boolean값으로 반환합니다.
	target	이벤트가 최초로 발생한 대상을 반환합니다.
	timeStamp	이벤트가 발생한 시간을 반환합니다.
	type	발생한 이벤트 이름을 반환합니다.
	which	키보드와 관련된 이벤트가 발생할 때 키의 유니코드값을 반환합니다.
메서드	preventDefault()	이벤트를 취소할 수 있는 경우에 취소합니다.

DOM에서 이벤트 처리하기

this

이벤트가 발생한 대상에 접근할 때 사용하는 예약어



DOM에서 이벤트 처리하기

addEventListener() 메서드 사용하기

이벤트 객체를 사용해 이벤트 처리기 연결

기본형 요소.addEventListener(이벤트, 함수, 캡처 여부);

1

2

3

- 1 이벤트: 이벤트 유형을 지정합니다. 단, click과 keypress처럼 on을 붙이지 않고 씁니다.
- 2 함수: 이벤트가 발생하면 실행할 명령이나 함수를 지정합니다. 여기에서 함수를 정의할 때는 event 객체를 인수로 받습니다.
- 3 캡처 여부: 이벤트를 캡처하는지 여부를 지정하며 기본값은 false이고 true와 false 중에서 선택할 수 있습니다. true이면 캡처링, false이면 버블링한다는 의미입니다. 이벤트 캡처링은 DOM의 부모 노드에서 자식 노드로 전달되는 것이고, 이벤트 버블링은 DOM의 자식 노드에서 부모 노드로 전달되는 것입니다.

DOM에서 이벤트 처리하기

addEventListener() 메서드 사용하기



Do it! 마우스 포인터를 올리면 이미지 바꾸기

예제 파일 17\event-5.html

(... 생략 ...)

```

```

.....

```
<script>
```

```
var cover = document.getElementById("cover");
```

```
cover.addEventListener("mouseover", changePic); // 포인터를 올리면 changePic() 실행
```

```
cover.addEventListener("mouseout", originPic); // 포인터를 내리면 originPic() 실행
```

```
function changePic() {
```

```
    cover.src = "images/easys-2.jpg"; // 이미지 경로를 easys-2.jpg로 바꿈
```

```
}
```

```
function originPic() {
```

```
    cover.src = "images/easys-1.jpg"; // 이미지 경로를 easys-1.jpg로 바꿈
```

```
}
```

```
</script>
```

(... 생략 ...)

메서드 안에서 함수 표현식으로 사용 가능

```

```

```
<script>
```

```
var cover = document.getElementById("cover");
```

```
cover.addEventListener("mouseover", changePic);
```

```
cover.addEventListener("mouseout", originPic);
```

```
function changePic() {
```

```
    cover.src = "images/easys-2.jpg";
```

```
}
```

```
function originPic() {
```

```
    cover.src = "images/easys-1.jpg";
```

```
}
```

```
</script>
```

DOM에서 이벤트 처리하기

addEventListener() 메서드 사용하기



Do it! 마우스 포인터를 올리면 이미지 바꾸기

예제 파일 17\event-5.html

(... 생략 ...)

```

```

.....

```
<script>
```

```
var cover = document.getElementById("cover");
```

```
cover.addEventListener("mouseover", changePic); // 포인터를 올리면 changePic() 실행
```

```
cover.addEventListener("mouseout", originPic); // 포인터를 내리면 originPic() 실행
```

```
function changePic() {
```

```
    cover.src = "images/easys-2.jpg"; // 이미지 경로를 easys-2.jpg로 바꿈
```

```
}
```

```
function originPic() {
```

```
    cover.src = "images/easys-1.jpg"; // 이미지 경로를 easys-1.jpg로 바꿈
```

```
}
```

```
</script>
```

(... 생략 ...)



Do it! 메서드 안에서 함수 선언하기

예제 파일 17\event-6.html

(... 생략 ...)

```

```

.....

```
<script>
```

```
var cover = document.getElementById("cover");
```

```
cover.addEventListener("mouseover", function() {
```

```
    cover.src = "images/easys-2.jpg";
```

```
});
```

```
cover.addEventListener("mouseout", function() {
```

```
    cover.src = "images/easys-1.jpg";
```

```
});
```

```
</script>
```

(... 생략 ...)

원본: getElementById, addEventListener



왼쪽 이미지 위에
마우스 포인터를
올리면 오른쪽
이미지로
바뀝니다


DOM에서 이벤트 처리하기

CSS 속성에 접근하기

자바스크립트를 사용해 각 요소의 스타일을 자유롭게 수정할 수 있음

기본형 `document.요소명.style.속성명`

예) id가 desc인 요소의 글자를 파란색으로 변경하려면

 id가 desc인 요소의 글자색 바꾸기

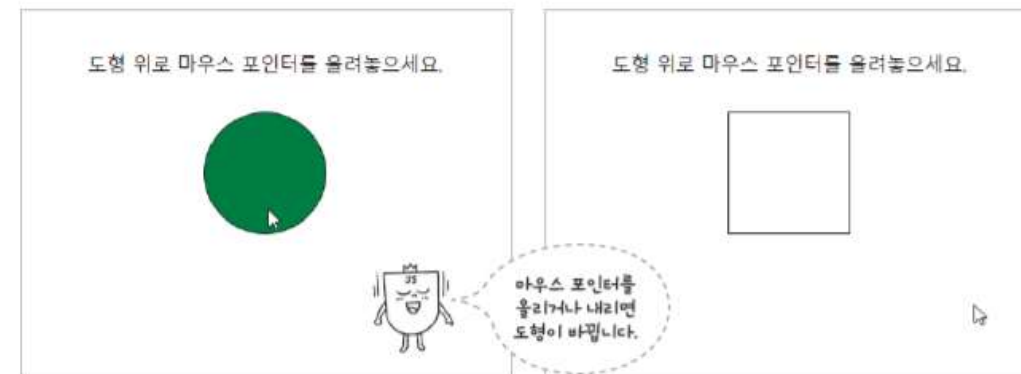
```
document.getElementById("desc").style.color = "blue";
```

- color처럼 한 단어인 속성명은 그대로 사용
- background-color, border-radius처럼 중간에 하이픈(-)이 있는 속성은 backgroundColor나 borderRadius처럼 두 단어를 합쳐서 사용
- 이때 두 번째 단어의 첫 글자는 Color와 Radius처럼 대문자로 표시

```
Do it! 도형의 테두리와 배경색 바꾸기 예제 파일 17\domCss.html

(... 생략 ...)
<div id="rect"></div>

.....
<script>
var myRect = document.querySelector("#rect");
myRect.addEventListener("mouseover", function() { // 마우스 포인터를 올리면
    myRect.style.backgroundColor = "green";        // 초록색으로 지정
    myRect.style.borderRadius = "50%";            // 테두리 둥글기를 50%로 지정
});
myRect.addEventListener("mouseout", function() { // 마우스 포인터를 내리면
    myRect.style.backgroundColor = "";            // 배경색 원래 값으로 지정
    myRect.style.borderRadius = "";              // 테두리 원래 값으로 지정
});
</script>
```



DOM에서 노드 추가, 삭제하기

노드 리스트란

querySelectorAll() 메서드를 사용해 가져온 여러 개의 노드를 저장한 것

```
(... 생략 ...)  
<h1>Web Programming</h1>  
<ul id="itemList">  
  <li>HTML</li>  
  <li>CSS</li>  
  <li>Javascript</li>  
</ul>
```



DOM에서 노드 추가, 삭제하기

텍스트 노드를 사용하는 새로운 요소 추가하기

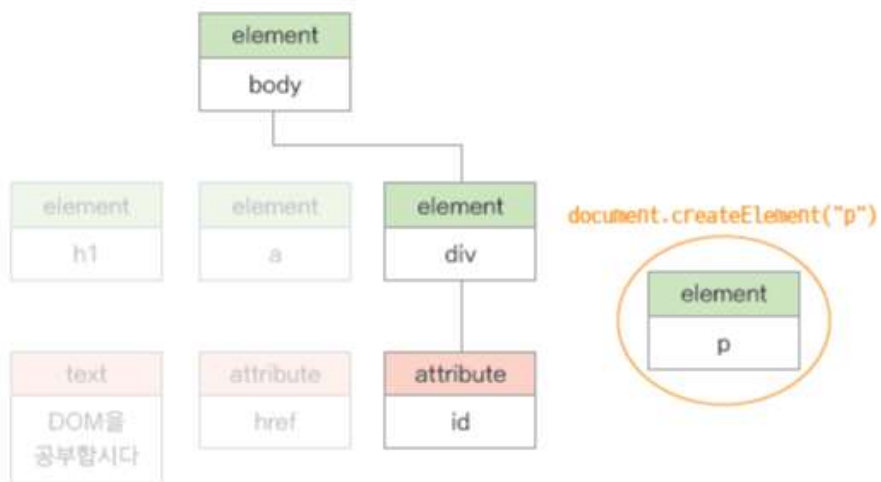
→ '더 보기' 링크를 클릭하면 텍스트 표시하기

1. 요소 노드 만들기 – createElement() 메서드

기본형 `document.createElement(노드명)`


 p 요소 노드 만들기

```
var newP = document.createElement("p");
```

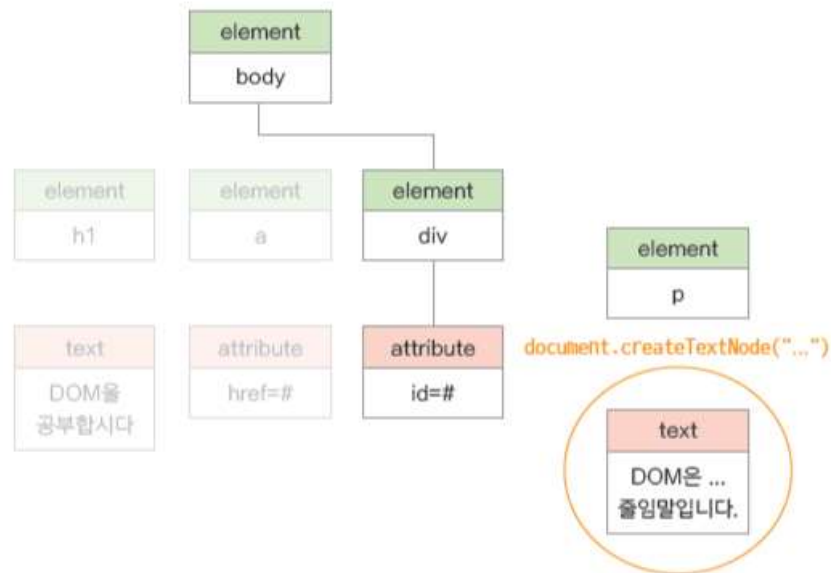


2. 텍스트 노드 만들기 – createTextNode() 메서드

기본형 `document.createTextNode(텍스트);`

 p 요소의 텍스트 노드 만들기

```
var txtNode = document.createTextNode("DOM은 document object model의 줄임말입니다.");
```



DOM에서 노드 추가, 삭제하기

텍스트 노드를 사용하는 새로운 요소 추가하기

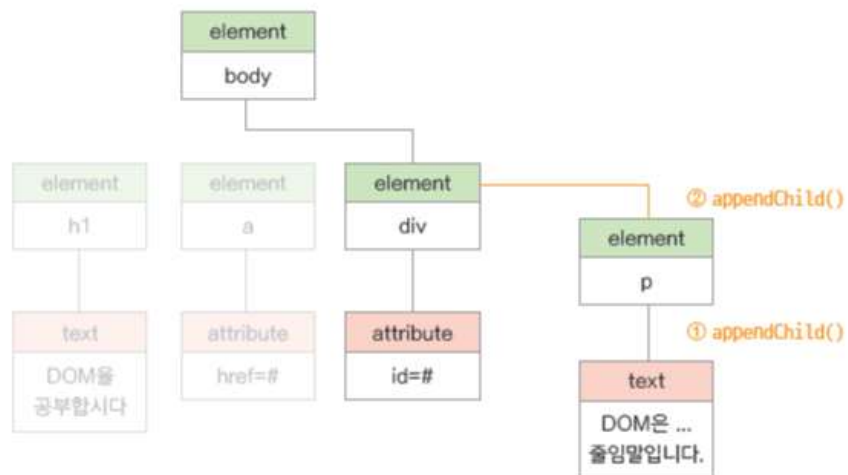
3. 자식 노드 연결하기 – appendChild() 메서드

기본형 부모노드.appendChild(자식노드)



텍스트 노드를 자식 노드로 연결하기

```
newP.appendChild(txtNode);  
document.getElementById("info").appendChild(newP);
```



전체 소스



Do it! 링크를 클릭하면 텍스트 표시하기

예제 파일 17\domNode-2.html

(... 생략 ...)

```
<div id="container">  
  <h1>DOM을 공부합니다</h1>  
  <a href="#" onclick="addP(); this.onclick='';">더 보기</a>  
  <div id="info"></div>  
</div>
```

링크를 클릭하면 addP() 함수가 한 번만 실행되도록 합니다.

<script>

```
function addP() {  
  var newP = document.createElement("p");  
  var txtNode = document.createTextNode("DOM은 document object model의 줄임말입니다.");  
  newP.appendChild(txtNode);  
  document.getElementById("info").appendChild(newP);  
}
```

</script>


(... 생략 ...)

DOM에서 노드 추가, 삭제하기

속성 값이 있는 새로운 요소 추가하기

→ '더 보기' 링크를 클릭하면 텍스트와 함께 이미지 표시하기


1. 요소 노드 만들기 – createElement() 메서드

 이미지 노드 추가하기

```
var newImg = document.createElement("img");
```

2. 속성 노드 만들기 – createAttribute() 메서드

[기본형] document.createAttribute(속성명)

 이미지의 src와 alt 속성 만들고 지정하기

```
var srcNode = document.createAttribute("src");  
var altNode = document.createAttribute("alt");  
srcNode.value = "images/dom.jpg";    // src 속성값 지정  
altNode.value = "돔 트리 예제 이미지"; // alt 속성값 지정
```

3. 속성 노드 연결하기 – setAttributeNode() 메서드

[기본형] 요소명.setAttributeNode(속성노드)

 이미지의 src 속성 노드 연결하기

```
newImg.setAttributeNode(srcNode);
```

4. 자식 노드 연결하기 – appendChild() 메서드

```
document.getElementById("info").appendChild(newImg);
```

DOM에서 노드 추가, 삭제하기

텍스트 노드를 사용하는 새로운 요소 추가하기

전체 소스



Do it! 링크를 클릭하면 텍스트와 이미지 표시하기

예제 파일 17\domNode-3.html

(... 생략 ...)

```
<div id="container">
  <h1>DOM을 공부합시다</h1>
  <a href="#" onclick="addContents(); this.onclick='';">더 보기</a>
  <div id="info"></div>
</div>
<script>
  function addContents() {
    var newP = document.createElement("p");
    var txtNode = document.createTextNode("DOM은 document object model의 줄임말입
니다.");
    newP.appendChild(txtNode);
```

```
    var newImg = document.createElement("img");
    var srcNode = document.createAttribute("src");
    var altNode = document.createAttribute("alt");
    srcNode.value = "images/dom.jpg";
    altNode.value = "돔 트리 예제 이미지";
    newImg.setAttributeNode(srcNode);
    newImg.setAttributeNode(altNode);
```

```
    document.getElementById("info").appendChild(newP);
    document.getElementById("info").appendChild(newImg);
  }
</script>
```

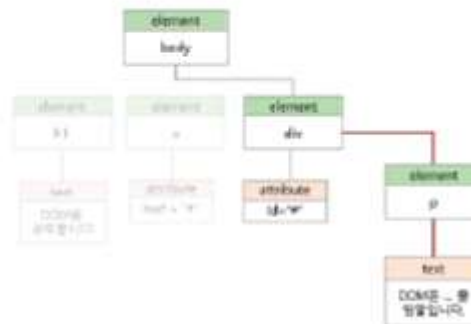
(... 생략 ...)

앞의 예제에서 추가된 부분입니다.

DOM을 공부합시다



DOM은 document object model의 줄임말입니다.



3.0.1.17.17\domNode-3.html

DOM에서 노드 추가, 삭제하기

노드 삭제하기

노드를 삭제할 때는 부모 노드에서 자식 노드를 삭제해야 한다

→ 노드를 삭제하려면 부모 노드부터 찾아야 함

parentNode 프로퍼티

현재 노드의 부모 노드에 접근해서 부모 노드의 요소 노드를 반환

기본형 `노드.parentNode`

removeChild() 메서드

자식 노드 삭제

기본형 `부모노드.removeChild(자식노드)`

자바스크립트 예제

<https://github.com/lbcsultan/javascript>

<https://jscripts.netlify.app/>

Javascript 예제 서비스

[자바스크립트 기본 문법 익히기](#)

[이미지 갤러리](#)

[서적 등록](#)

[사용자 관리](#)

[음식점 메뉴 관리](#)

[탁구 스코어 관리](#)

[QR Code 활용 예제](#)