

자바스크립트(JavaScript) 기초

# Session 14

NEXT X LIKELION 김나영

# 01\_자바스크립트란? 🤔

---

# Intro

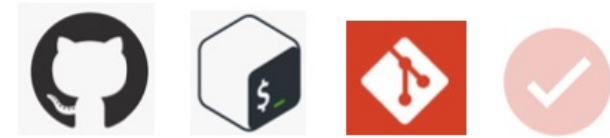
## Programming Language



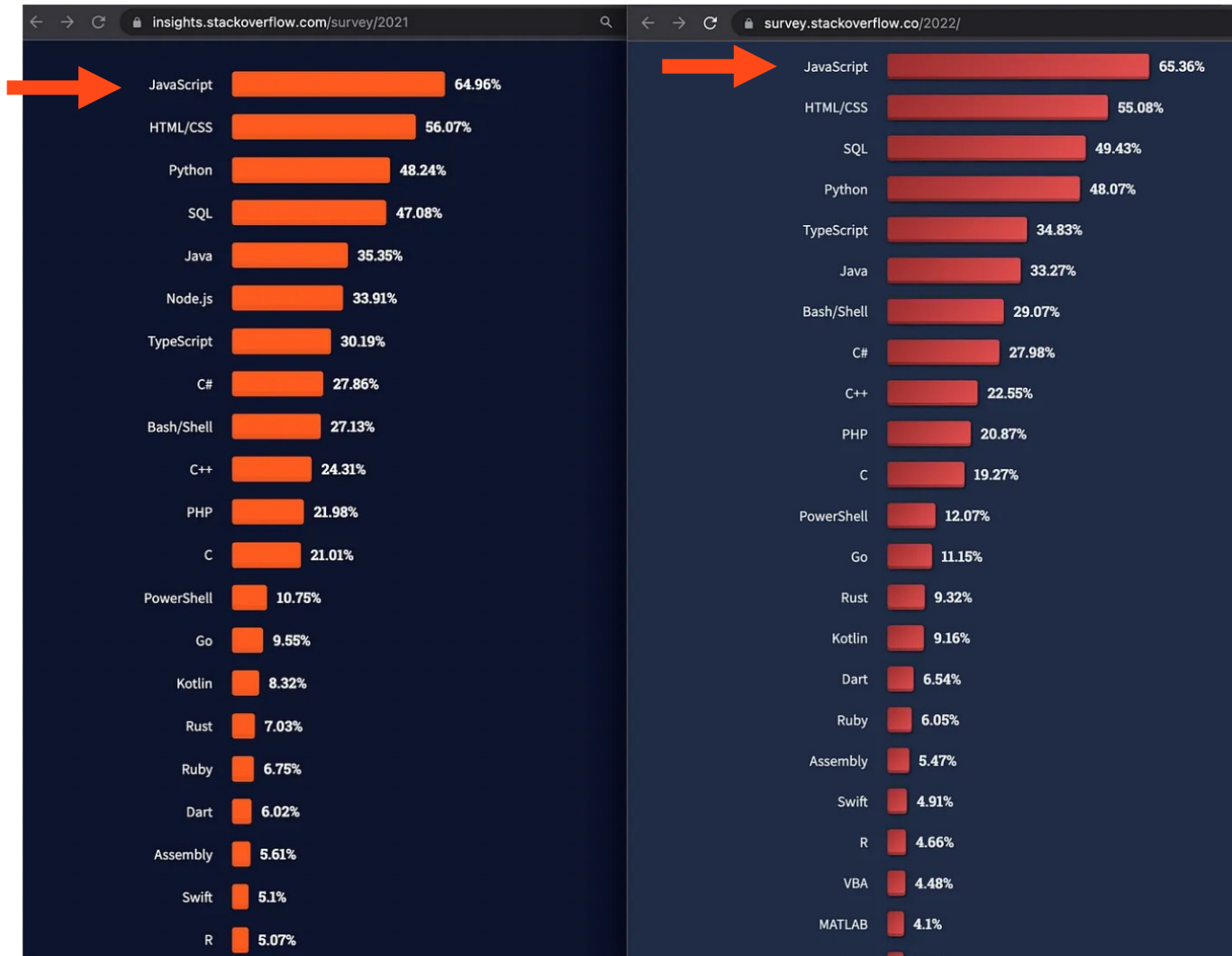
## Framework



## Tools



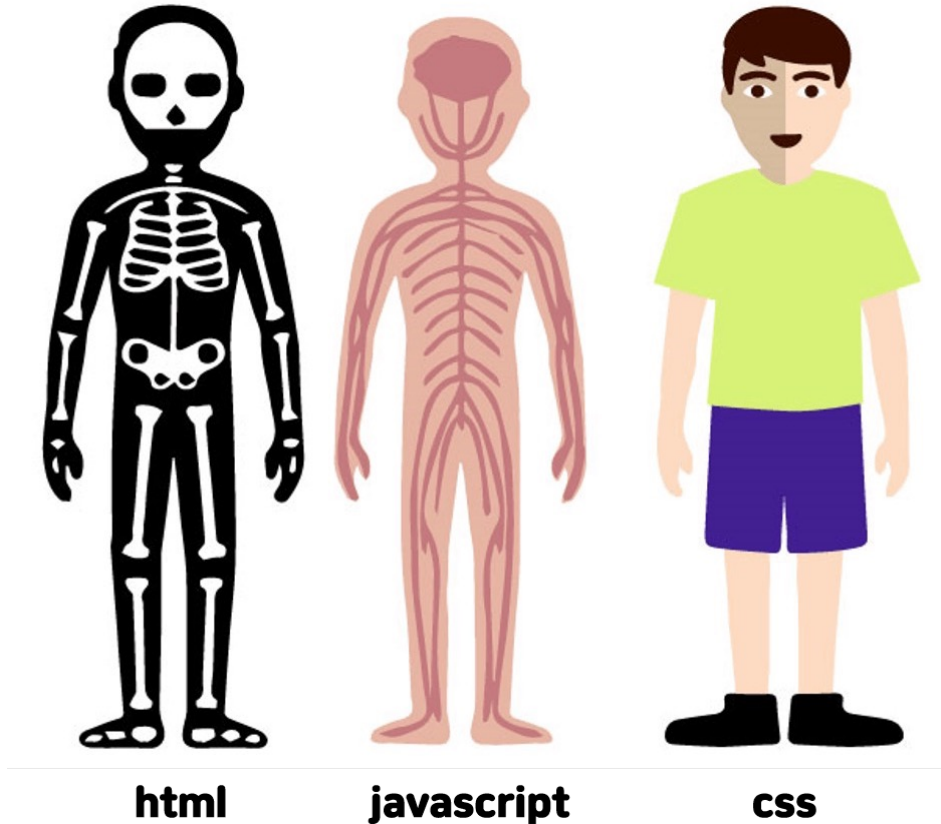
# Intro



2021 vs 2022 사용중인 언어 순위

# 자바스크립트?

JavaScript?



- 자바스크립트(javascript) : “프로그래밍 언어”

1. 웹 페이지에 “생동감을 불어넣기 위해” 만들어짐

2. HTML/CSS와 완전한 통합이 가능한 스크립트 언어

: 스크립트 == 자바스크립트로 작성한 프로그램.

HTML 안에 작성할 수 있으며,

페이지를 불러올 때, 스크립트가 자동으로 실행됨.

3. JAVA와 전혀 관계 없다 

: JAVA 인기가 높을 때 만들어져서, 홍보 차 지은 이름일 뿐...

더 알아보기: <https://ko.javascript.info/intro>

# 02\_JS 기초 문법

---

변수 / 자료형 / 객체 / 배열 / 조건문 / 반복문 / 함수

# 스크립트 사용하기

JS 기초 문법 0

## 1) 웹 페이지에 스크립트 삽입해보기: `<script>` 태그 이용

helloworld.html

```
<!DOCTYPE HTML>
<html>

<body>

  <script>
    alert( 'Hello, world!' );
  </script>

</body>

</html>
```

💡 HTML 안에 직접 스크립트를 작성하는 방식  
: 스크립트가 아주 간단할 때만 사용!

# 스크립트 사용하기

JS 기초 문법 0

## 2) 웹 페이지에 스크립트 삽입해보기: 외부 JS 파일 이용

helloworld.html

```
<!DOCTYPE HTML>
<html>
<body>
<scriptsrc="/helloworld.js"></script>
</body>
</html>
```

helloworld.js

```
alert( 'Hello, world!' );
```

💡 HTML과 별도의 파일로 스크립트를 작성하는 방식

: 성능상 이점(트래픽 절약 / 속도 빨라짐)

왜? 브라우저가 스크립트를 다운 -> 캐시(cache)에 저장해서.

즉, 여러 페이지에서 동일한 스크립트를 사용하는 경우,

브라우저는 스크립트를 1번만 다운받고,

페이지가 바뀔 때마다 스크립트를 새로 받지 X.

캐시로부터 가져와서 사용 O.



# 스크립트 사용하기

JS 기초 문법 0

⚠ **src 속성이 있으면 태그 내부의 코드는 무시된다!**

helloworld.html

```
<!DOCTYPE HTML>
<html>
<body>
  <script src="/helloworld.js">
    alert( 'Hello, world! (내부 코드)' );
  </script>
</body>
</html>
```

helloworld.js

```
alert( 'Hello, world! (src)' );
```

→ <script> 태그를 쓸 때 src 속성과 내부 코드, 둘 중 하나만 사용한다!

# Alert, Prompt, Confirm

JS 기초 문법 0

→ alert, prompt, confirm으로 상호작용하기

## Alert

사용자가 OK 누를 때까지 메시지 모달 창 띄우기

Javascript 파일

```
alert("Hello!");
```

## Prompt

사용자가 값을 입력할 수 있는 input field가 있으면서  
텍스트+확인+취소 버튼 있는 대화상자 모달 창 띄우기

Javascript 파일

```
result = prompt(title, [default]); // 형식  
let age = prompt('나이를 입력해주세요.', 20); // 예시
```

## Confirm

질문+확인+취소 버튼이 있어서,  
확인: true, 취소: false를 반환하는 모달 창 띄우기

Javascript 파일

```
result = confirm(question); // 형식  
let isAge = confirm("당신은 20세인가요?"); // 예시
```

# 세미콜론

JS 기초 문법 0

## ⚠ 세미콜론(;)은 문장 끝에 항상 달기

줄 바꿈이 있다면, 대부분의 경우, 세미콜론이 없어도 있는 것처럼 해석해주긴 하지만...  
‘항상’이 아니기 때문에 무조건 세미콜론을 써주자.

반대로 세미콜론이 없어야 할 곳에 세미콜론이 있다면,  
자바스크립트는 필요없는 세미콜론을 자동으로 없애고 해석한다.

무조건 쓰지 않을 이유가 없다!

# 주석

JS 기초 문법 0

## 1) 짧은 주석:

```
//주석주석주석
```

## 2) 긴 주석:

```
/*주석주석  
주석주석주석  
주석주석주석  
*/
```

 중첩 주석은 지원하지 않습니다.

`/*...*/` 안에 또 다른 `/*...*/` 이 있을 수 없습니다.

주석을 중첩해 쓰면 에러가 발생합니다.

```
1  /*  
2      /* 중첩 주석 ?!?! */  
3  */  
4  alert( 'World' );
```

# 변수 선언

JS 기초 문법 1

**var** : 옛날 방식. 쓰지마.

Var: 블록 스코프 없고, 중복 선언도 되고, 선언 전에 사용도 되고... ☹

**let** : 변수 선언    변수 선언 시 앞에 꼭 붙이기!!!

**const** : 상수 선언    파이썬에는 없고... C언어의 #define을 떠올려보면 됨. #define PI = 3.14;

# 변수 선언

JS 기초 문법 1

	재 선언	재 할당
var	O	O
let	X	O
const	X	X

# 변수 선언

JS 기초 문법 2

## let : 변수 선언

```
let next = 3;    // 변수 선언  
console.log(next);
```

```
next = 4;        // 재할당  
console.log(next);
```

```
let next = 3;    // 변수 선언  
console.log(next);
```

```
let next = 4;    // 재선언X  
console.log(next);
```

## const : 상수 선언

```
const next = 3;  // 상수 선언  
console.log(next);
```

```
next = 4;        // 재할당X  
console.log(next);
```

```
const next = 3;  // 상수 선언  
console.log(next);
```

```
const next = 4;  // 재선언X  
console.log(next);
```

# 자료형

JS 기초 문법 2



Q. 자바스크립트는 자료형을 어떻게 지정하나요?

A. 자바스크립트의 변수는 자료형에 관계없이 모든 데이터일 수 있습니다.

```
1 // no error
2 let message = "hello";
3 message = 123456;
```

따라서 변수는 어떤 순간에 문자열일 수 있고  
다른 순간엔 숫자가 될 수도 있습니다.



# 자료형

JS 기초 문법 2

자료형	상세 설명
숫자형(Number)	정수, 소수점 숫자 등을 포함한 숫자를 나타낼 때 사용
문자형(String)	빈 문자열 or 문자들로 이뤄진 문자열을 나타낼 때 사용 * 단일 문자를 나타내는 별도의 자료형은 없음.
불린형(Boolean)	true, false 를 나타낼 때 사용
null	알 수 없는 값을 나타냄
undefined	할당되지 않은 값을 나타냄
객체형(Object)	복잡한 데이터 구조를 표현할 때 사용할 때 사용

+ bigint, 심볼형, Function, Array 등등...

# Object(객체)

JS 기초 문법 3

## 💡 무엇이냐

- Python의 dict와 같음
- 키(key)과 값(value)으로 구성된 프로퍼티(Property)들의 집합
- (+) 프로퍼티(property, 데이터)와 메소드(method, 데이터를 참조하고 조작할 수 있는 동작)로 구성된 집합

## 💡 중요성

- 자바스크립트는 객체(object) 기반의 스크립트 언어이다.
  - 자바스크립트를 이루는 거의 모든 것은 객체이다
- (원시 타입(Primitives)을 제외한 함수, 배열, 정규표현식 등)

## 💡 특징: 참조에 의해 저장되고 복사됨.(by reference)

# Object(객체)

JS 기초 문법 3

// 객체 선언

```
let user = {  
  name: "김나영",  
  age: "23",  
};
```

// 객체 참조

`user.name` //(1) 점 표기법(객체 내부가 정의되지 않은 상태에서는 사용X)

`user["name"]` //(2) 대괄호 표기법

// 동적으로 추가 및 수정 가능

`user.name = "홍길동"` //(1) 점 표기법(객체 내부가 정의되지 않은 상태에서는 사용X)

`user["name"] = "홍길동"` //(2) 대괄호 표기법

# Array(배열)

JS 기초 문법 4

## 💡 무엇이나

- Python의 list와 같음 (단, method 사용 방식이 서로 다름)

[https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global\\_Objects/Array](https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array)

## 💡 대표적인 메서드들

- **arr.push** : array 맨 끝에 요소 추가
- **arr.pop** : array 맨 끝 요소 제거
- **arr.shift** : array 맨 앞 요소 제거
- **arr.unshift** : array 맨 앞에 요소 추가
- **arr.slice([start], [end])** : start ~ end 슬라이싱
- **arr.splice()** [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/splice](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice)

```
// push 메서드 활용 예시
let nextArray = [];
//javascript
nextArray.push("!"); // Array "nextArray"에 !가 추가됨
//python
nextArray.append("?"); // Error (Python과 다릅니다!)

// slice 메서드 활용 예시
let nextArray2 = ["N", "E", "X", "T"];
console.log(nextArray2.slice(1,3)); // ['E', 'X']
// (인덱스가 1인 요소(start = 1)부터
// 3인 요소 이전까지(end - 1 = 3-1 = 2)의
// "SubArray(하위 배열)" 형태로 복사)
```

# 조건문

JS 기초 문법 5

## 1) if 문

```
let year = prompt('올해는 몇년도?', '');

if (year < 2023) {
    alert( '숫자를 좀 더 올려보세요.' );
} else if (year > 2023) {
    alert( '숫자를 좀 더 내려보세요.' );
} else {
    alert( '정답입니다!' );
}
```

## 2) switch case

```
let a = 2023;

switch (a) {
    case 2022:
        alert( '비교하려는 값보다 작습니다.' );
        break;
    case 2023:
        alert( '비교하려는 값과 일치합니다.' );
        break;
    case 2024:
        alert( '비교하려는 값보다 큼니다.' );
        break;
    default:
        alert( "어떤 값인지 파악이 되지 않습니다." );
}
```

# 조건문

JS 기초 문법 5

## 3) 삼항 연산자 `() ? () : ()`;

```
// 형식: true면 value1, false면 value2 반환
let result = condition ? value1 : value2;
// 예시
let pass = (age > 18) ? true : false;

//if문이었다면?
let pass;
let age = prompt('나이를 입력해 주세요.', '');
if (age > 18) {
    pass = true;
} else {
    pass = false;
}
```

## 4) 짧은 조건문

등등

```
a || b
// a 또는 b 중 하나라도 참이면 실행

a && b
// a와 b 모두 참이어야 실행
```

# 반복문

JS 기초 문법 6

## 1) for 문

## 2) for in 문 : 주로 객체 자료형과 함께 사용함

```
//forin 반복문 사용 예시
let obj = { name: '김나영', job: 'student' }

for (let key in obj){
    console.log(`${key} : ${obj[key]}`);
} // name : 김나영, job : student
```

## 3) for of 문

```
let array = ["1","2","3"]

for(let arr of array){
    console.log(arr);
}
```

# 반복문

JS 기초 문법 6

## 4) while 문

```
let i = 0;
while(i<3){//0,1,2가 출력됩니다.
    console.log(i);
    i++;
}
```

## 5) do ... while문 : 주로 객체 자료형과 함께 사용함

+) “배열”을 반복 시: forEach, map, reduce 등등

```
const numbers = [1,2,3];
numbers.forEach(number => console.log(number));
```



# 함수(Function)

JS 기초 문법 7

💡 무엇이냐: Python의 def와 같음

💡 정의하는 방식 3가지

- 1) function으로 정의, 함수 선언문
- 2) function으로 정의, 함수 표현식
- 2) **Arrow function**으로 정의하기

★ Arrow function을 왜 쓰나요?

- 1) 코드 길이가 짧아진다
- 2) 'context'가 없는 짧은 코드를 담을 용도인데... 추후 심화적으로 공부할 때 보강!
- 3) 심화 맛보기: Arrow function에는 'this'가 없다!

```
//function으로 함수 정의하기, 함수선언문
function add(a, b) {
    return a + b;
}
```

```
//function으로 함수 정의하기, 함수표현식
let function sayHi(){
    alert("안녕하세요!");
}
```

```
// Arrow function으로 함수 정의하기
let add = (a, b) => a + b;

let sayHi = () => alert("안녕하세요!");
```

# 03\_연습(JS 기초 문법)

---

# JS 기초 문법 연습 (1)

/\*

1. 프로퍼티 합 구하는 코드 작성하기

-> 아래 객체 속 모든 팀원의 월급을 합한 값을 구하고, 그 값을 변수 `sum`에 저장해주는 코드를 작성해보세요!

[Hint] 객체의 값을 참조하는 방법을 떠올려보세요! \*/

```
let salaries = {  
  John: 100,  
  Ann: 160,  
  Pete: 130,  
};
```

// 이 아래에 정답 코드를 작성하세요.

//HINT1. `sum` 변수 선언하기

//HINT2. 반복문 사용하기

## JS 기초 문법 연습 (2)

```
/*
  2. min(a, b) 함수 만들기
  -> a 와 b 중에서 더 작은 값을 반환해주는 함수 min(a, b)를 만들어 보세요!
  -> a == b인 경우엔 a나 b 중 어떤 것을 반환해도 상관없습니다. */

// (1) if 문을 활용해 만들기
function min(a, b) {
}

// (2) ? 를 활용해 만들기
function min(a, b) {
}

// console.log(min(5,4));
```

쉬는 시간

# 04\_DOM 🌲

---

# DOM

Document Object Model

- **DOM이란?** Document Object Model

- **의미:** HTML 문서의 프로그래밍 interface, 쉽게 말해 웹 브라우저가 HTML 페이지를 인식하는 방식
- **좁은 의미:** document 객체와 관련된 객체의 집합
- **자료구조:** tree 형식 - nodes와 objects로 문서를 표현한다.
- 즉, DOM은 문서의 '구조화'된 표현을 제공하며,  
프로그래밍 언어가 DOM 구조에 접근할 수 있는 방법을 제공하여  
문서 구조, 스타일, 내용 등을 변경할 수 있게 한다.

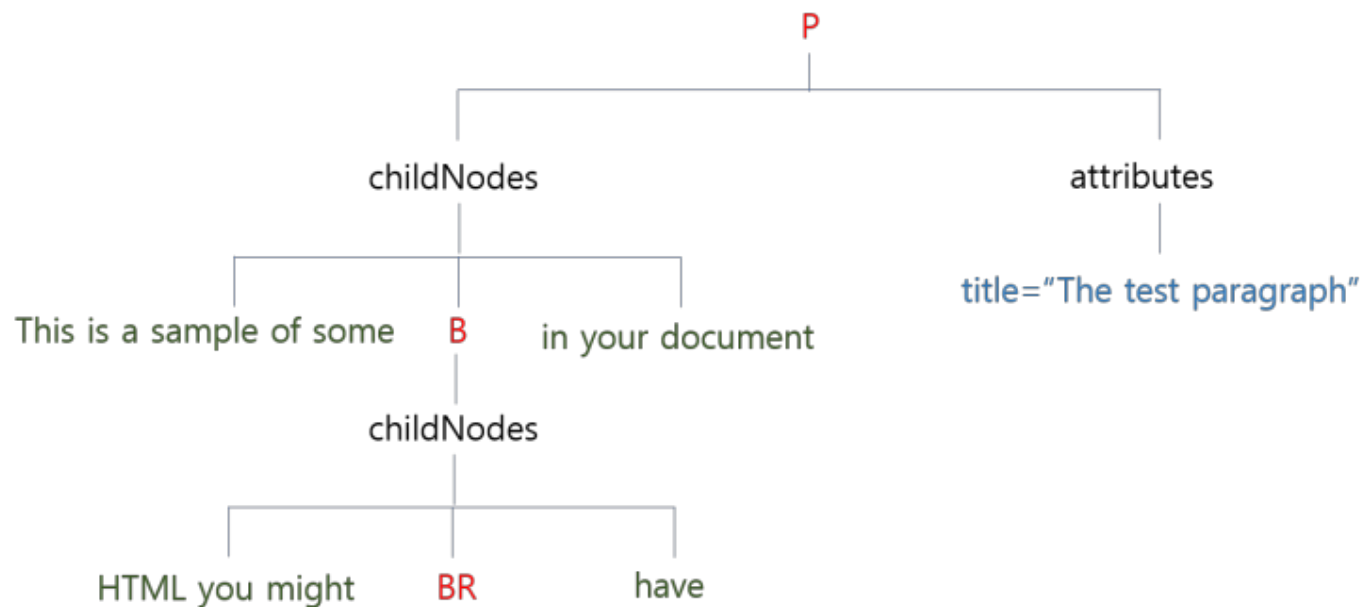
# DOM

Document Object Model

- Dom 트리

**DOM** = Document Object Model

```
<p title="The test paragraph">  
  This is a sample of some <b>HTML you might <br> have</b>in your document  
</p>
```





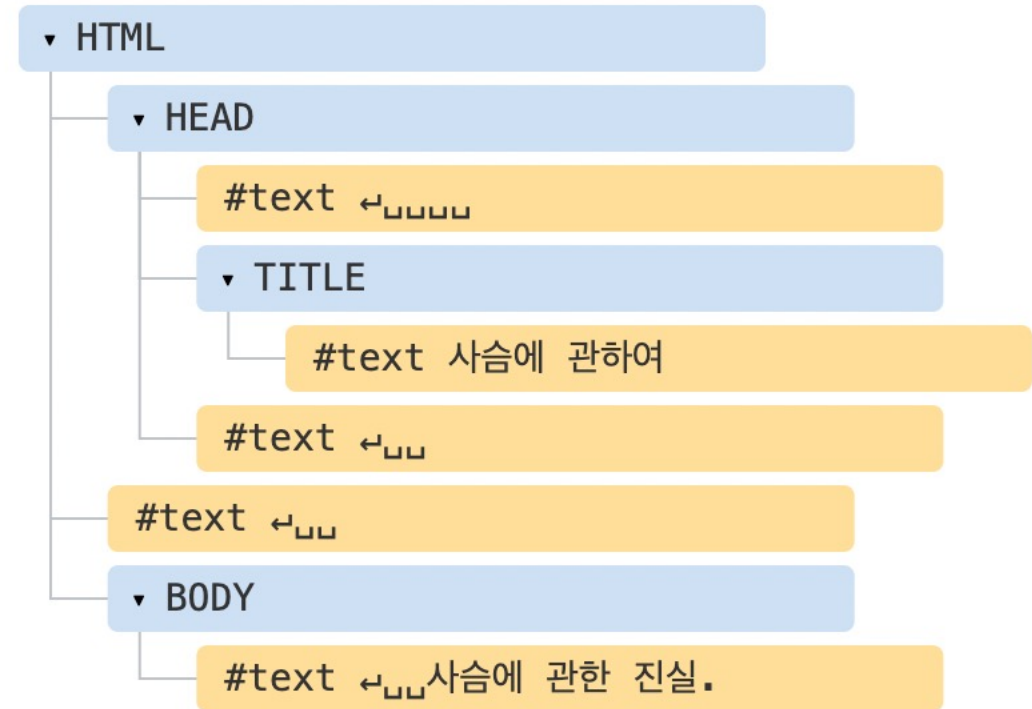
# DOM

Document Object Model

## • Dom 트리2

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <title>사슴에 관하여</title>
5 </head>
6 <body>
7   사슴에 관한 진실.
8 </body>
9 </html>
```

⚡ 요소 노드(Element node): 자식을 가질 수 있음

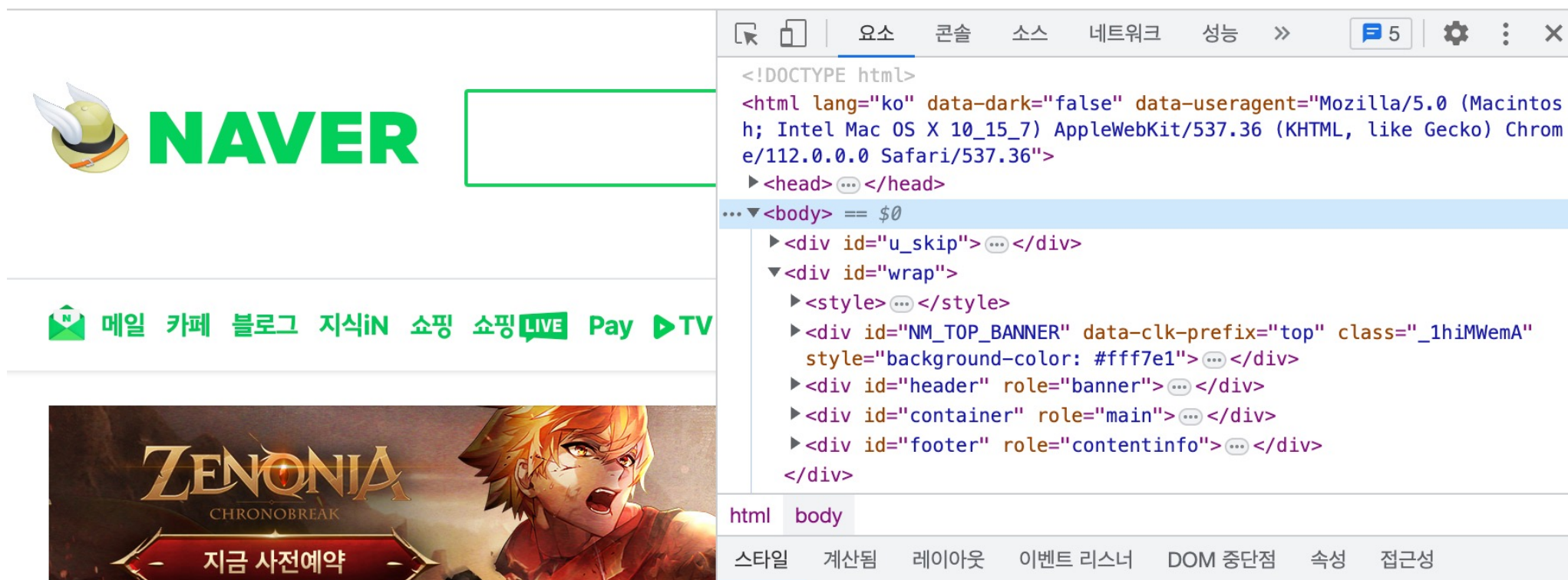


⚡ 텍스트 노드(text node): 자식을 가질 수 없음, 요소 내의 문자

# DOM

Document Object Model

- Dom 구조 직접 보기 개발자 도구 - Element(요소) 보기

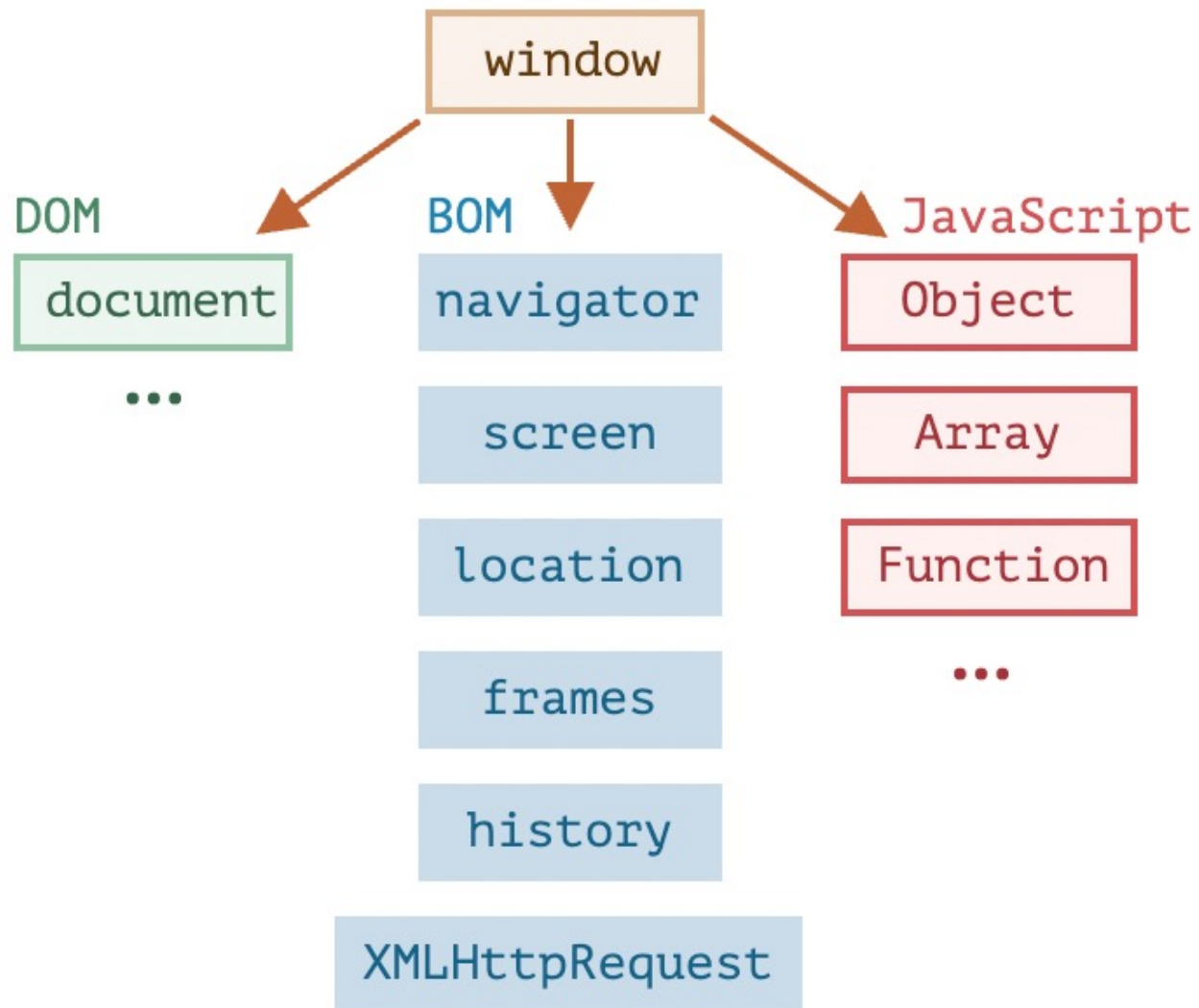


The image shows a screenshot of the Naver homepage. The Naver logo is at the top left. Below it is a navigation bar with links: 메일, 카페, 블로그, 지식iN, 쇼핑, 쇼핑 LIVE, Pay, and TV. At the bottom is a banner for 'ZENONIA CHRONOBREAK' with the text '지금 사전예약' (Now pre-order).

The Chrome DevTools 'Element' panel is open on the right side of the browser window. It shows the DOM tree structure of the page. The root element is `<!DOCTYPE html>`. The `<html>` element has attributes `lang="ko"`, `data-dark="false"`, and `data-useragent="Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36"`. The `<body>` element is selected, and its children are listed: `<div id="u_skip">`, `<div id="wrap">`, `<style>`, `<div id="NM_TOP_BANNER" data-clk-prefix="top" class="_1hiMwemA" style="background-color: #fff7e1">`, `<div id="header" role="banner">`, `<div id="container" role="main">`, and `<div id="footer" role="contentinfo">`.

# DOM

Document Object Model



# DOM에 접근하기

Document Object Model

- **document 객체** : 페이지의 기본 '진입점' 역할

참고) 반환값

```
document.getElementById('id값');
```

HTML에 있는 형식 그대로

id를 통한 요소 노드 취득

```
document.getElementsByTagName('태그 이름');
```

HTMLCollection[]

해당 태그 이름을 갖는 모든

요소 노드들을 탐색하여 반환

```
document.getElementsByClassName('class 값');
```

HTMLCollection[]

해당 클래스 값을 갖는 모든

요소 노드들을 탐색하여 반환

# DOM에 접근하기

Document Object Model

- document 객체 : 페이지의 기본 '진입점' 역할

document.querySelector('css선택자')

첫번째 요소만

CSS 선택자로 요소 선택 가능

: 해당하는 첫번째 요소만 선택

document.querySelectorAll('css선택자');

NodeList[]

CSS 선택자로 요소 선택 가능

: 해당하는 모든 요소를 선택

## 💡 CSS 선택자가 뭐였냐고?

\*

전체 선택자 (Universal Selector)

tag

태그 선택자 (Type Selector)

#id

ID 선택자 (ID Selector)

.class

클래스 선택자 (Class Selector)

일치 선택자 **a.b**

: a 와 b의 조건을 동시에 만족하는 요소 선택

자식 선택자 **a > b**

: a의 자식 요소인 b를 선택

후손 선택자 **a b**

: a의 하위 요소인 b를 선택

인접 형제 선택자 **a + b**

: a의 바로 다음 형제 요소인 b 하나만 선택

일반 형제 선택자 **a ~ b**

: a의 다음 형제 요소 b 모두 선택

# DOM 조작하기

Document Object Model

- **innerHTML**: Element node 내부의 HTML 코드를 문자열로 리턴, 내부 HTML 자체를 수정할 때 자주 활용
- **innerText**: Element node 내부의 HTML 중에서 사용자에게 보여지는 텍스트 값을 리턴
- **classList**: Element node의 className을 읽는 프로퍼티로, DOM 객체의 class를 조작할 때 자주 활용

innerHTML과 innerText의 차이점을 더 자세히 알고 싶다면: <https://hianna.tistory.com/480>

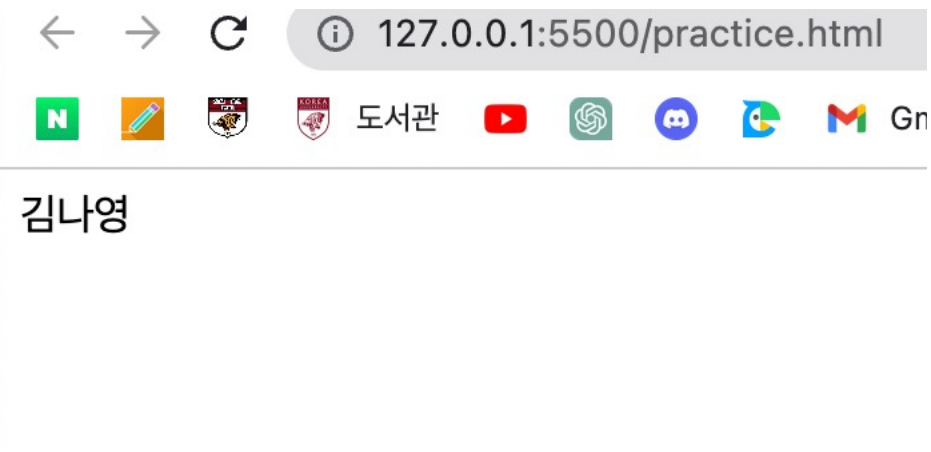
# DOM 조작하기: 예시1

Document Object Model

dom1.html

```
<!-- 쉬운 예시 -->

<p id="name">홍길동</p>
<script>
    document.getElementById( 'name' ).innerHTML="김나영";
    document.getElementById( 'name' ).classList="nameclass";
</script>
```



# DOM 조작하기 : 예시2

Document Object Model

<!-- 조~금 더 복잡한 예시 1 -->

dom2.js

```
const dc = document; //document 너무 기니까 dc로 줄여보기
const body = dc.body; //매번 dc.body 치기 귀찮으니까 body로 줄여보기

const container = dc.createElement("div"); //createElement로 div 요소 생성, container로 명명
container.innerText = "메인 컨테이너"; //새로 생성한 div 즉 container의 내부 Text를 "메인 컨테이너"로 쓰기
container.classList.add("container");//새로 생성한 div 즉 container의 class에 "container" 추가하기

const title = dc.createElement('h1');//createElement로 h1 요소 생성, title로 명명
title.innerText = "타이틀"; //새로 생성한 h1 즉 title의 내부 Text를 "타이틀"로 쓰기
title.classList.add('title');//새로 생성한 h1 즉 title의 class에 "title" 추가하기

body.append(container); //document.body 밑에 container을 추가
container.append(title); //container 아래에 title을 추가
```



# DOM 조작하기 : 예시2

Document Object Model

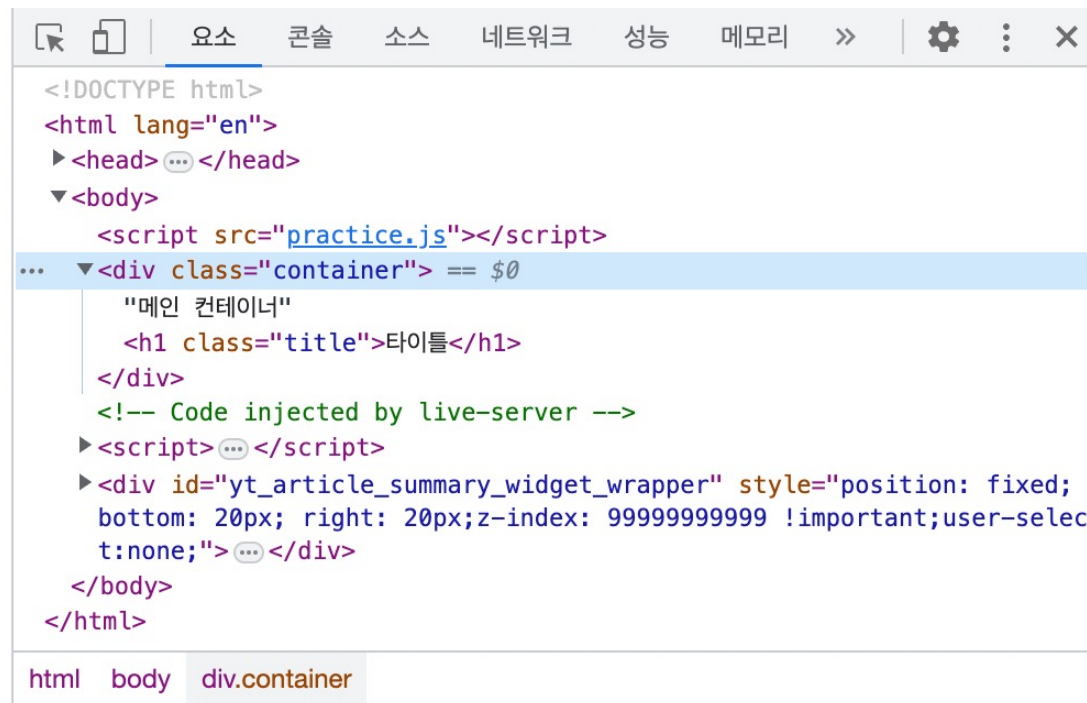
dom2.html

```
<!-- 조~금 더 복잡한 예시 1 -->
```

```
<body>  
  <script src="dom2.js"></script>  
</body>
```

메인 컨테이너

## 타이틀



```
<!DOCTYPE html>  
<html lang="en">  
  <head>...</head>  
  <body>  
    <script src="practice.js"></script>  
    <div class="container"> == $0  
      "메인 컨테이너"  
      <h1 class="title">타이틀</h1>  
    </div>  
    <!-- Code injected by live-server -->  
    <script>...</script>  
    <div id="yt_article_summary_widget_wrapper" style="position: fixed;  
      bottom: 20px; right: 20px; z-index: 9999999999 !important; user-select: none;">...</div>  
  </body>  
</html>
```

```
<div class="container">  
  "메인 컨테이너"  
  <h1 class="title">타이틀</h1>  
</div>  
</body>
```

이것과 같은 결과

# DOM 조작하기 : 예시3

Document Object Model

dom3.html

```
<!-- 조~금 더 복잡한 예시 2 -->

<body>
  <div class="container">
    <h1 class="heading">DOM 예제</h1>

    <ul class="myList">
      <li>기본 항목</li>
    </ul>
  </div>
  <script src="dom3.js"></script>
</body>
```

# DOM 조작하기 : 예시3

Document Object Model

dom3.js

```
<!-- 조~금 더 복잡한 예시 1 -->
```

```
const dc = document; //document 너무 기니까 dc로 줄여보기
const myList = dc.querySelector('.myList'); //myList class에 해당하는 첫번째 요소 선택

for (let i=1; i<=3; i++) { //반복문
    let item = dc.createElement('li'); //li 요소를 생성, item으로 명명
    item.innerText = i + " 번째 추가 항목"; //item의 내부 Text를 "i번째 추가 항목"으로 쓰기
    myList.append(item); //myList에 item을 추가
}
```

# DOM 조작하기 : 예시3

Document Object Model

## DOM 예제

- 첫번째 항목
- 1 번째 추가 항목
- 2 번째 추가 항목
- 3 번째 추가 항목

```
<body>
  <div class="container">
    <h1 class="heading">DOM 예제</h1>
    ... <ul class="myList"> == $0
      <li>
        ::marker
        "첫번째 항목"
      </li>
      <li>
        ::marker
        "1 번째 추가 항목"
      </li>
      <li> ... </li>
      <li> ... </li>
    </ul>
  </div>
```

# 05\_Event ⚡

---

# Event

javascript Event

- Event 란?

- 의미: “**사건**” - 사용자가 클릭하는 사건 / 스크롤을 하는 사건 / 무언가를 입력한 사건 …….

- **이벤트 타입** 예시:

- 포커스 이벤트(focus, blur …)
- 폼 이벤트(reset, submit …)
- 뷰 이벤트(scroll, resize …)
- 마우스 이벤트(mouseenter, mouseover, click, dbclick …)
- 드래그앤 드롭 이벤트(dragstart, drag, dragleave, drop …)
- 키보드 이벤트(keydown, keyup …)

# EventListener

javascript Event

- **EventListener** : DOM 객체에서 이벤트가 발생할 경우, 특정 함수(이벤트 처리 핸들러)를 호출

DOM객체.**add**EventListener(이벤트 타입, 실행할 함수명, 옵션);

DOM객체.**remove**EventListener(이벤트 타입, 실행할 함수명, 옵션);

# Event: 예시(외부파일 방식)

event.html

```
<!-- 예시 -->

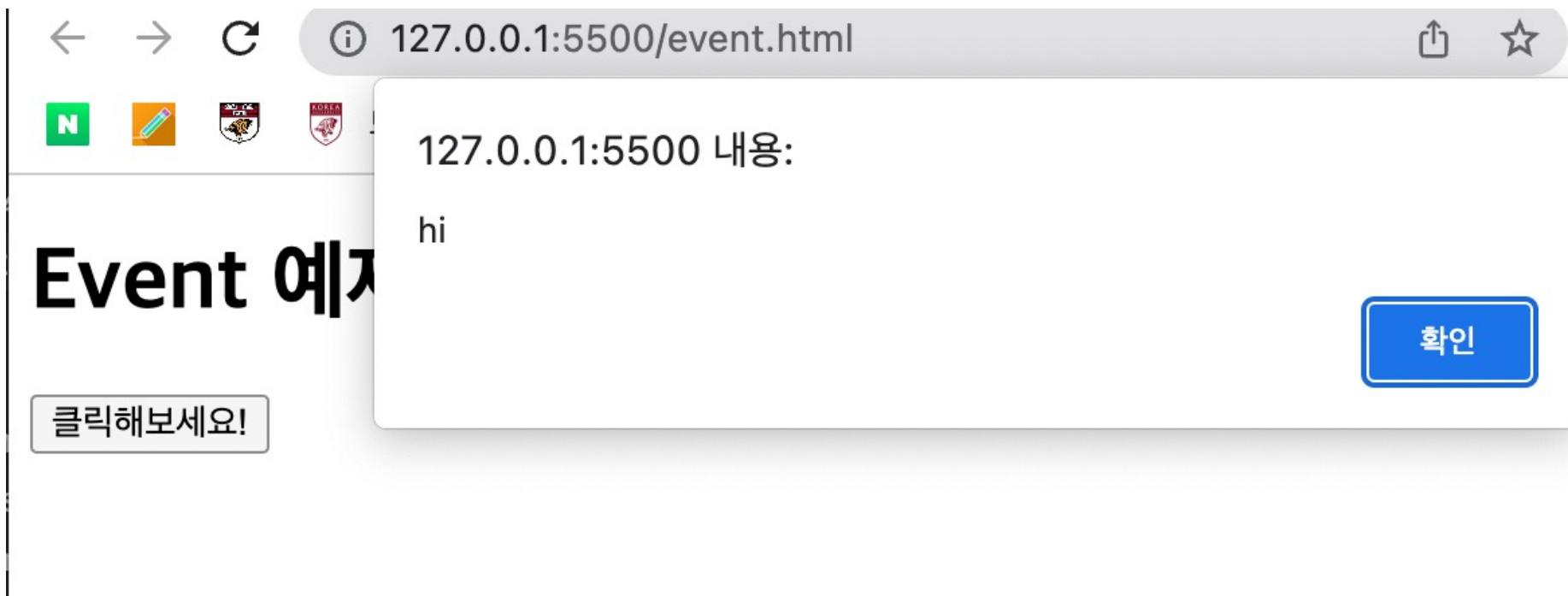
<body>
  <div class="container">
    <h1>Event 예제</h1>
    <button id="eventTarget">클릭해보세요!</button>
  </div>
</body>
<script src="event.js"></script>
```

event.js

```
let eventTarget = document.getElementById('eventTarget');
eventTarget.addEventListener('click', () => alert("hi"));
```



## Event: 예시



# Event: 예시(script 내부 태그 방식)

event.html

```
<!-- 예시/ -->

<body>
  <div class="container">
    <h1>Event 예제</h1>
    <button id="eventTarget">클릭해보세요!</button>
  </div>
</body>

<script>
  let eventTarget = document.getElementById('eventTarget');
  eventTarget.addEventListener('click', () => alert("hi"));
</script>
```

# Event: 예시(inline 방식)

event.html

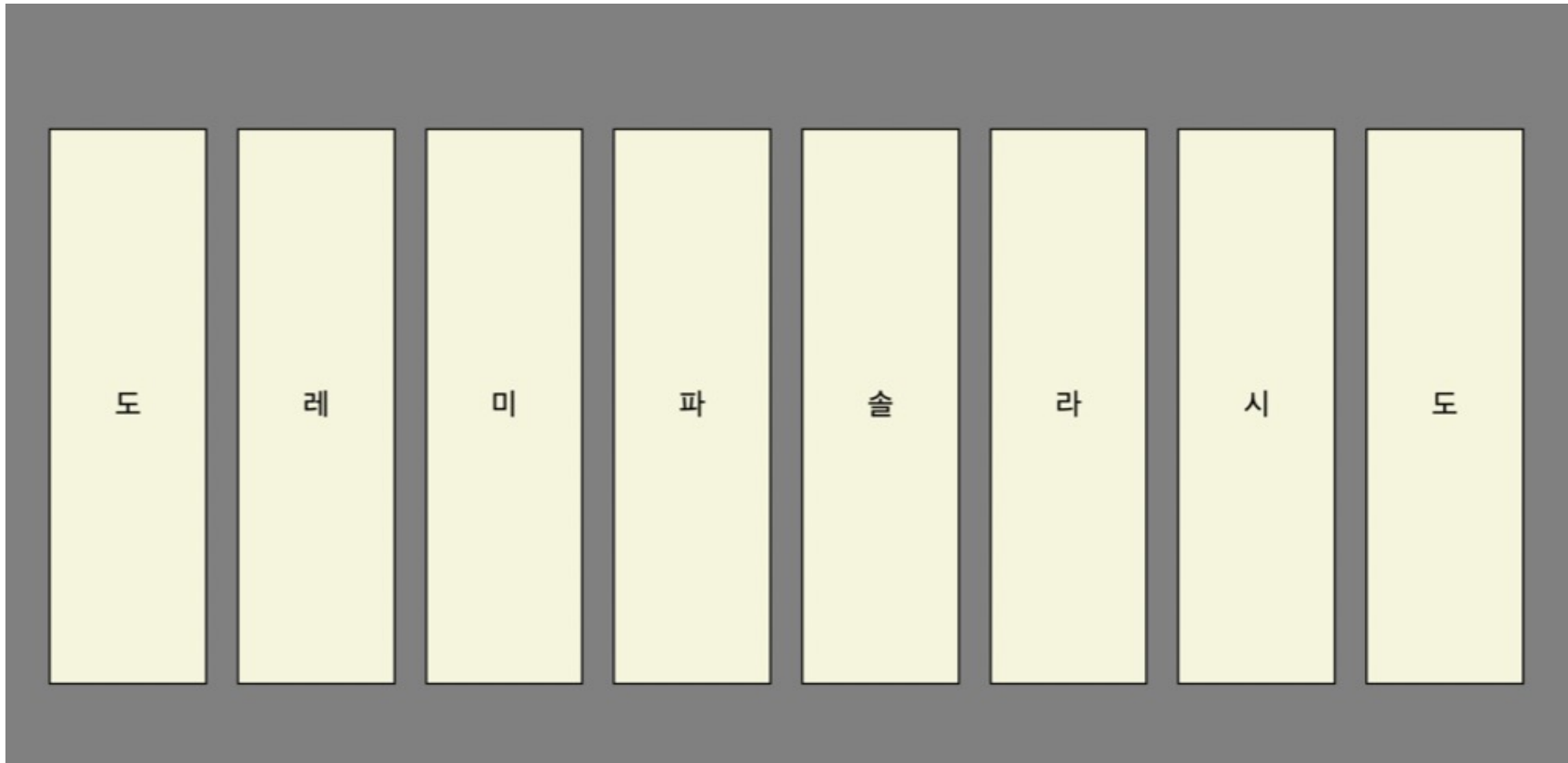
```
<!-- 예시/ -->

<body>
  <div class="container">
    <h1>Event 예제</h1>
    <button id="eventTarget" onclick="alert('hi')">클릭해보세요!</button>
  </div>
</body>
```

# 06\_실습(DOM & Event)

---

# | JS 응용 간단 실습: 키보드 피아노 만들기



<https://javascript30.com/>

# | JS 응용 간단 실습: 키보드 피아노 만들기

## 문제 설명

1) 키보드자판의 **ASDFGHJK**를 누르면 **도레미파솔라시도**가 재생되기

: 아스키 코드 활용

도	레	미	파	솔	라	시	도
A 65	S 83	D 68	F 70	G 71	H 72	J 74	K 75
a 97	b 115	c 110	d 102	e 103	f 104	g 106	h 107

2) 키보드를 눌렀을 때, 해당하는 음의 버튼이 **scale up**되면서 **노랗게 하이라이트** 되기

# | JS 응용 간단 실습: 키보드 피아노 만들기

## 문제 푸는 법 설명

1) index.html에서: main.js파일을 연결한다

2) main.js에서:

- play 함수를 작성한다.
- pause 함수를 작성한다.
- “keypress” 시 .play 클래스를 추가하고, “keydown” 시 .play클래스를 remove한다.

3) style.css에서:

- css 파일에서 .play 클래스에 해당하는 css 내용을 추가한다.

# | JS 응용 간단 실습: 키보드 피아노 만들기

## 우리가 해야 할 일

### 1. index.html과 main.js를 연결하기❤️

- 외부 파일 방식을 기억하자!

### 2. Main.js 코드 짜기❤️

- 키보드를 누르면 play 함수가 실행
- play함수: 누른 키에 해당하는 음을 재생
- 키보드를 뗄때 pause함수가 실행
- pause함수: 뗄 키에 해당하는 음재생을 멈춤



# JS 응용 간단 실습: 키보드 피아노 만들기

## 참고해야할 문법(html)

### 1.1.1. data-

'data-' 속성은 모든 태그에 적용할 수 있는 글로벌 속성이다. 기본적으로 제공되는 속성이 아닌, 나만의 새로운 속성을 추가할 때 사용된다. 기본적으로 속성만 입력했을 때는 ""처럼 빈 문자열이 기본값이다.

피아노에서는 *keypress* 이벤트에 쓸 함수에 사용할 *data-press* 속성과 *keyup* 이벤트에 쓸 함수에 사용할 *data-key* 속성을 이용해 새로운 속성값을 부여할 수 있었다.

## 참고해야할 문법(css)

```
###transform: scale();  
scale(가로, 세로),, 하나만 쓰면 가로세로 공통 적용  
확대하는 효과!
```

```
###border-color + box-shadow  
border-color 와 box-shadow 에 동일한 색상을 부여하여 자연스럽게 강조되는 효과를 주도록 하자.
```

# JS 응용 간단 실습: 키보드 피아노 만들기

## 참고해야할 문법(javascript)

### 1.3.1. addEventListener

`target.addEventListener(type, listener..);` type 반응할 **이벤트 유형**을 나타냄, listener 지정된 타입의 이벤트 발생시 알림을 받는 객체

### 1.3.2 keypress/ keyup

keypress : 키가 눌린 상태일 때 (연속적으로 실행됨)

keyup : 키 누름이 해제될 때

keydown : 키가 눌렸을 때 (불연속)

### 1.3.3 querySelector 사용시 class 가져오기

아래 예제처럼 정말 강력한 선택자도 사용할 수 있습니다. 예제의 결과는 클래스가 "user-panel main"인 `<div>`(`<div class="user-panel main">`) 안의, 이름이 "login"인 `<input>` 중 첫 번째 요소입니다.

```
var el = document.querySelector("div.user-panel.main input[name=login]");
```

내 코드에서 쓰인 예시로는

```
const audio = document.querySelector(`audio[data-press="${e.keyCode}"]`); 이다.
```

`document.querySelector(Element선택하기)` 에서...

- `Element[attribute = "value"]`: Element의 attribute = value인 element를 선택
- `${백틱 표현식}`: 문자열 내에서 변수나 함수를 표현할 수 있는 표현식, python의 f-string과 비슷하다

## 과제 내용

피아노 실습 구현 완료하기

## 제출 방법

- 본인 레포지토리에 git push한 링크
- 구현한 피아노 기능 화면 녹화

## 제출 기한

다음 세션까지