JavaScript 문법 + 응용

Session 12

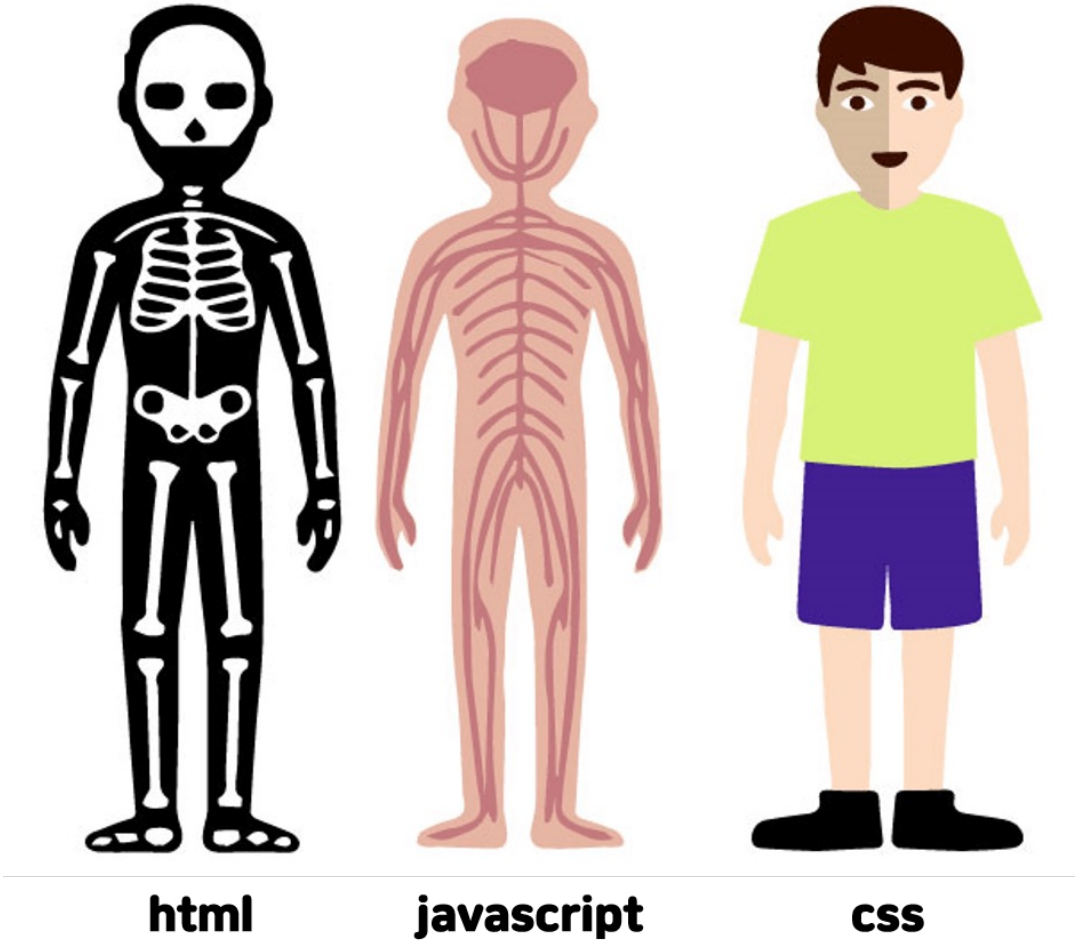
NEXT X LIKELION 박가영

Intro.

Ref: 2021 Stackoverflow 설문조사

목차

그래서 JS가 뭔데?



- 웹페이지에 “생동감을 불어넣기 위해” 만들어진 프로그래밍 언어
- HTML/CSS와 완전한 통합이 가능한 스크립트 언어
- Java와 1도 관계없다

JS 기초 문법_자료형

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…

JS 기초 문법_변수 선언

var vs let, const

	재 선언	재 할당
var	O	O
let	X	O
const	X	X

JS 기초 문법_변수 선언

var vs let, const

```
var name = '멋사';
console.log(name);          // 멋사

var name = '멋사 재할당';
console.log(name);          // 멋사 재할당
```

변수 재선언 — 변수 재할당

var

변수 재선언 및 재할당 가능

필요할 때 마다 변수를 사용할 수 있다는 장점이 있지만,

같은 변수명을 남용하는 문제를 야기할 수 있음

JS 기초 문법_변수 선언

var vs let, const

```
var name = '멋사';
console.log(name);          // 멋사

var name = '멋사 재할당';
console.log(name);          // 멋사 재할
```

변수 재선언

변수 재할당

var

변수 재선언 및 재할당 가능

필요할 때 마다 변수를 사용할 수 있다는 장점이 있지만,

같은 변수명을 남용하는 문제를 야기할 수 있음

Console.log("message")

- Python의 "print"와 같음
- Message를 찍어보거나, 변수를 넘겨주면 결과값을 보여줌

JS 기초 문법_변수 선언

var vs let, const

```
var name = '멋사';
console.log(name);          // 멋사

var name = '멋사 재할당';
console.log(name);          // 멋사 재할당
```

변수 재선언 — 변수 재할당

var

변수 재선언 및 재할당 가능

필요할 때 마다 변수를 사용할 수 있다는 장점이 있지만,

같은 변수명을 남용하는 문제를 야기할 수 있음

JS 기초 문법_변수 선언

var vs let, const

```
let name = '멋사';
console.log(name);          // 멋사

name = '멋사 재할당';
console.log(name);          // 멋사 재할당
```

let

변수 재할당 O

변수 재선언 X

일반적인 변수 개념으로 사용

X

```
let  name = '멋사 다시 선언';
console.log(name);
```

```
Uncaught SyntaxError: Identifier 'name' has already been declared  VM85428:2
```

JS 기초 문법_변수 선언

var vs let, const

```
const name = '멋사';
console.log(name);          // 멋사
```

X
```
name = '멋사 재할당';
console.log(name);
```

X
```
const  name = '멋사 다시 선언';
console.log(name);
```

const

변수 재할당 X

변수 재선언 X

일반적인 상수 개념으로 사용

JS 기초 문법_Object

- Python의 dict와 같음
- 참조에 의해(by reference) 저장되고 복사된다는 특징.

```
// 객체를 선언할 때

let user = {
    name: "Lion",
    age: 22
};


// 참조할 때 (1)점 표기법, (2)대괄호 표기법

user.name

user["name"]


// 동적으로 추가 및 수정 가능

user.name = "Tiger"

user["name"] = "Tiger"
```

※ 객체 내부가 아직 정의되지 않은 상태에서는 점 표기법 사용 시, 에러 발생

JS 기초 문법_Array

- Python의 list와 같지만, method 사용 방식이 서로 다름. (아래 링크 참고)
- 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

•[그 외에도]

•arr.slice([start], [end])

•arr.splice() 등등…

```
// push 메서드 활용 예시

let meotjaengi = [];

meotjaengi.push("!"); // Array "meotjaengi"에 !가 추가됨
meotjaengi.append("?"); // Error (Python과 다릅니다!)


// slice 메서드 활용 예시

let meotjaengi2 = ["L", "I", "O", "N"];

console.log(meotjaengi2.slice(1,3)); // ['I', 'O']

// (인덱스가 1인 요소(start)부터 3인 요소 이전까지(end - 1)의
"SubArray(하위 배열)" 형태로 복사)
```

JS 기초 문법_Function

- Python의 def와 같음
- JS에서 함수를 사용하는 방법은 두 가지 : (1) function 으로 정의하기, (2) Arrow function 으로 정의하기

```
// function으로 함수 정의하기

function add(a, b) {
    return a + b;
}


// Arrow function으로 함수 정의하기

let add = (a, b) => {
    return a + b;
};
```

JS 기초 문법_비교 연산자

[기본 수학 연산자]

사용 상황	연산자
크다/작다	< , >
이상/이하	>= , <=
같음(동등)	== (등호 2개)
다름(부등)	!=

[일치 연산자(===)]

JS는 약타입 언어,

```
console.log( 0 == false ); // true
```

피연산자의 묵시적 형변환이 일어나기 때문에 나타나는 현상

=== 를 통해 type의 일치 여부까지 검사한다.

```
console.log( 0 === false ); // false
```

JS 기초 문법_삼항 연산자

- Python의 if else 를 간편하게 사용가능
- 사용형태 → (조건문) ? (조건문이 참인 경우에 반환) : (조건문이 거짓인 경우에 반환)

```
// if, else를 사용한 조건문

let 적정age;
let age = prompt('나이를 입력해 주세요.', '');

if (age > 18) {
    적정age = true;
  } else {
    적정age = false;
  }
```

```
// 삼항 연산자 사용 형태

let result = condition ? value1 : value2;


// 삼항연산자 사용한 조건문

let 적정age = (age > 18) ? true : false;
```

JS 기초 문법_반복문

- for
- for in
- for of
- while
- do … while
- forEach, map, reduce 등등

```
// while 반복문 사용 예시

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


// for in 반복문 사용 예시

let obj = { name: 'Likelion', job: 'engineer' }

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

※ 주로 객체 자료형과 함께 사용됨

JS 기초 문법_조건문

- If 문
- switch case
- 삼항 연산자 () ? () : ();
- 짧은 조건문

  a || b → a 또는 b 둘 중 하나라도 참일 경우 실행

  a && b → a 와 b 모두 참이어야 실행

JS 기초 문법_간단 실습 1

```
1. 프로퍼티 합 구하는 코드 작성하기
-> 아래 객체 속 모든 팀원의 월급을 합한 값을 구하고, 그 값을 변수 sum에 저장해주는 코드를 작성해보세요!
[Hint]
※ salaries가 비어있다면 sum에 0이 저장되어야 합니다.
※ 반복문을 사용해보세요.
※ 객체의 값을 참조하는 방법을 떠올려보세요! */

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

/*

여기에 1번 문제의 정답 코드를 작성해보세요.

*/
```

JS 기초 문법_간단 실습 2

```
2. min(a, b) 함수 만들기
-> a 와 b 중에서 더 작은 값을 반환해주는 함수 min(a, b)를 만들어 보세요! */

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

// (2) ? 를 활용해 만들기
function min(a, b) {
  //(코드를 완성해보세요!)
}
```

JS 기초 문법_간단 실습 1_정답

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

let sum = 0;
for (let key in salaries) {
  sum += salaries[key];
}
```

[대괄호 표기법]을 사용하면 모든 표현식의 평가 결과를 프로퍼티 키로 사용할 수 있습니다. (변수로 접근 가능)
※ 점 표기법은 이런 방식이 불가능

JS 기초 문법_간단 실습 2_정답

```
// (1) if 문을 활용해 만들기
function min(a, b) {
  if (a < b) {
    return a;
  } else {
    return b;
  }
}

// (2) ? 를 활용해 만들기
function min(a, b) {
  return a < b ? a : b;
}
```

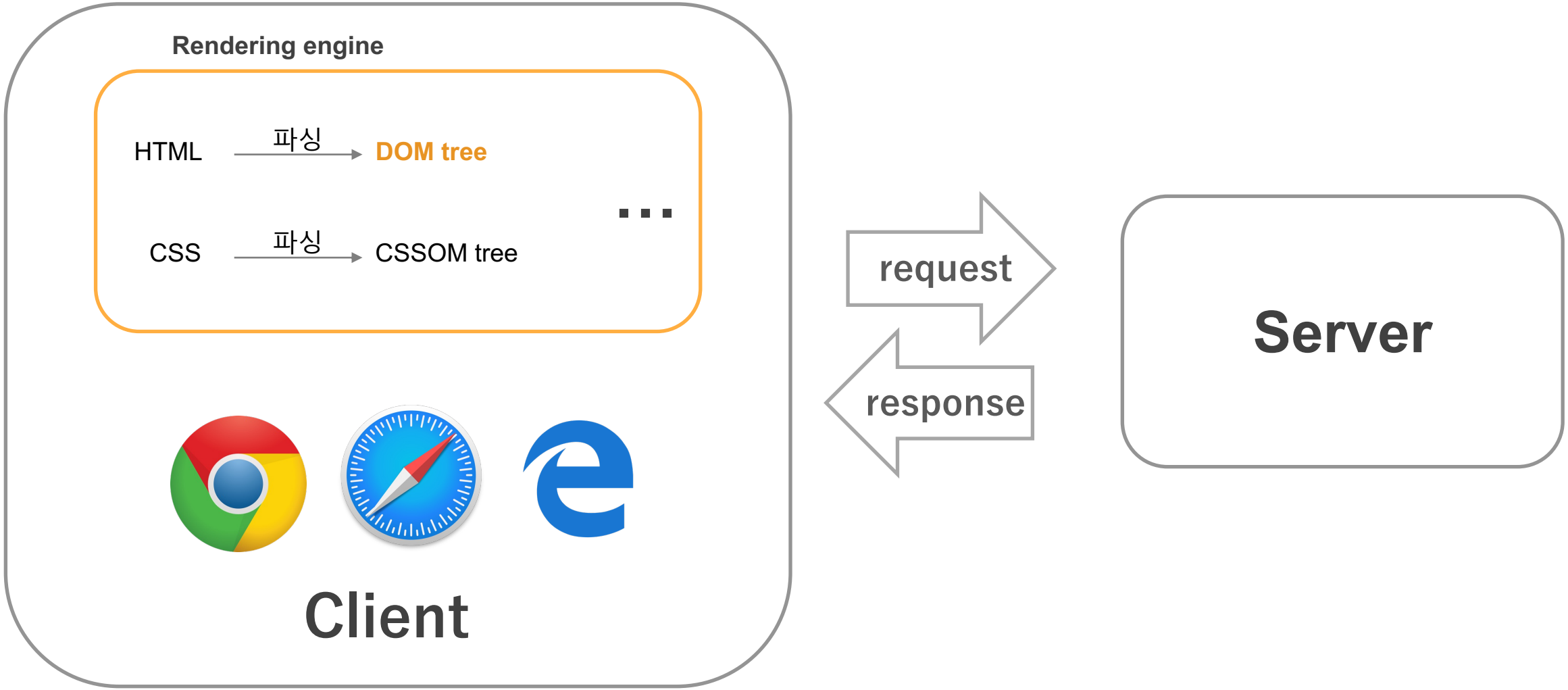
DOM

Document Object Model

문서 객체 모델(The Document Object Model, 이하 DOM) 은 HTML, XML 문서의 프로그래밍 interface 이다. DOM은 문서의 구조화된 표현(structured representation)을 제공하며 프로그래밍 언어가 DOM 구조에 접근할 수 있는 방법을 제공하여 그들이 문서 구조, 스타일, 내용 등을 변경할 수 있게 돕는다. DOM 은 nodes와 objects로 문서를 표현한다. 이들은 웹 페이지를 스크립트 또는 프로그래밍 언어들에서 사용될 수 있게 연결시켜주는 역할을 담당한다.

DOM

<브라우저의 렌더링 과정>



DOM

바이트(2진수)

문자 <meta charsest = “UTF-8”> …

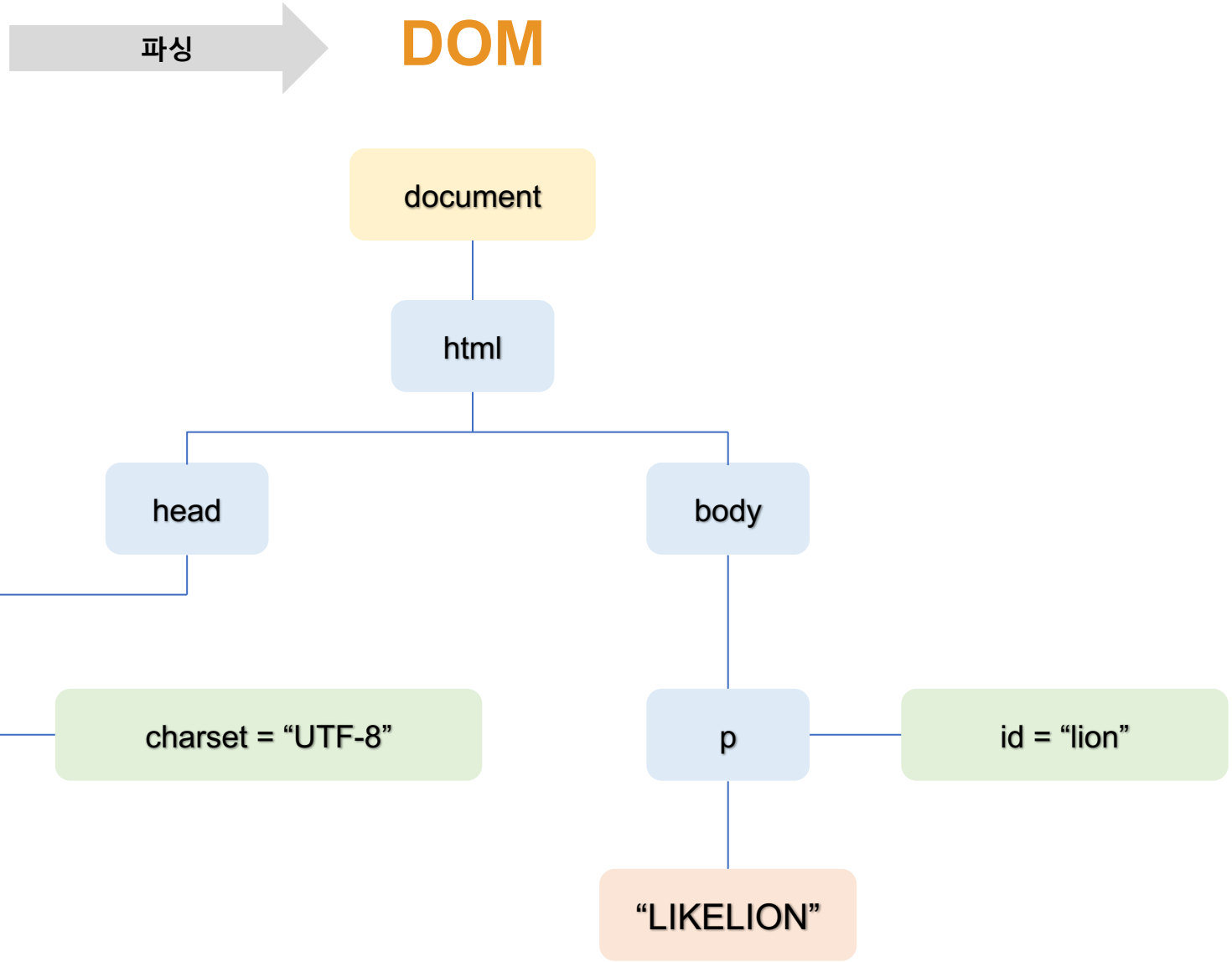
토큰

노드 html head body …

DOM HTML 파싱의 결과

DOM

```
<html>
  <head>
    <meta charset = “UTF-8”>
  </head>
  <body>
    <p id = “lion”>LIKELION</p>
  </body>
</html>
```



DOM

document.getElementById('id값');

id를 이용한 요소 노드 취득

document.getElementsByTagName('태그 이름');

인수에 들어있는 태그 이름을 갖는 **모든 요소 노드들**을 탐색하여 반환

(HTMLCollection 객체 반환)

document.getElementsByClassName('class 값');

인수로 전달한 클래스 값을 갖는 **모든 요소 노드들**을 탐색하여 반환

(HTMLCollection 객체 반환)

DOM

document.querySelector(‘css선택자’);

document.querySelectorAll(‘css선택자’);

DOM

DOM 조작

innerHTML

classList

Event

Event

어떤 “사건”을 발생 시키는것

Event type

이벤트의 종류를 나타내는 문자열

이벤트 타깃

EventTarget.addEventListener(

‘eventType’, function , useCapture);

이벤트 타입 이벤트 핸들러 캡쳐 사용 여부

Event

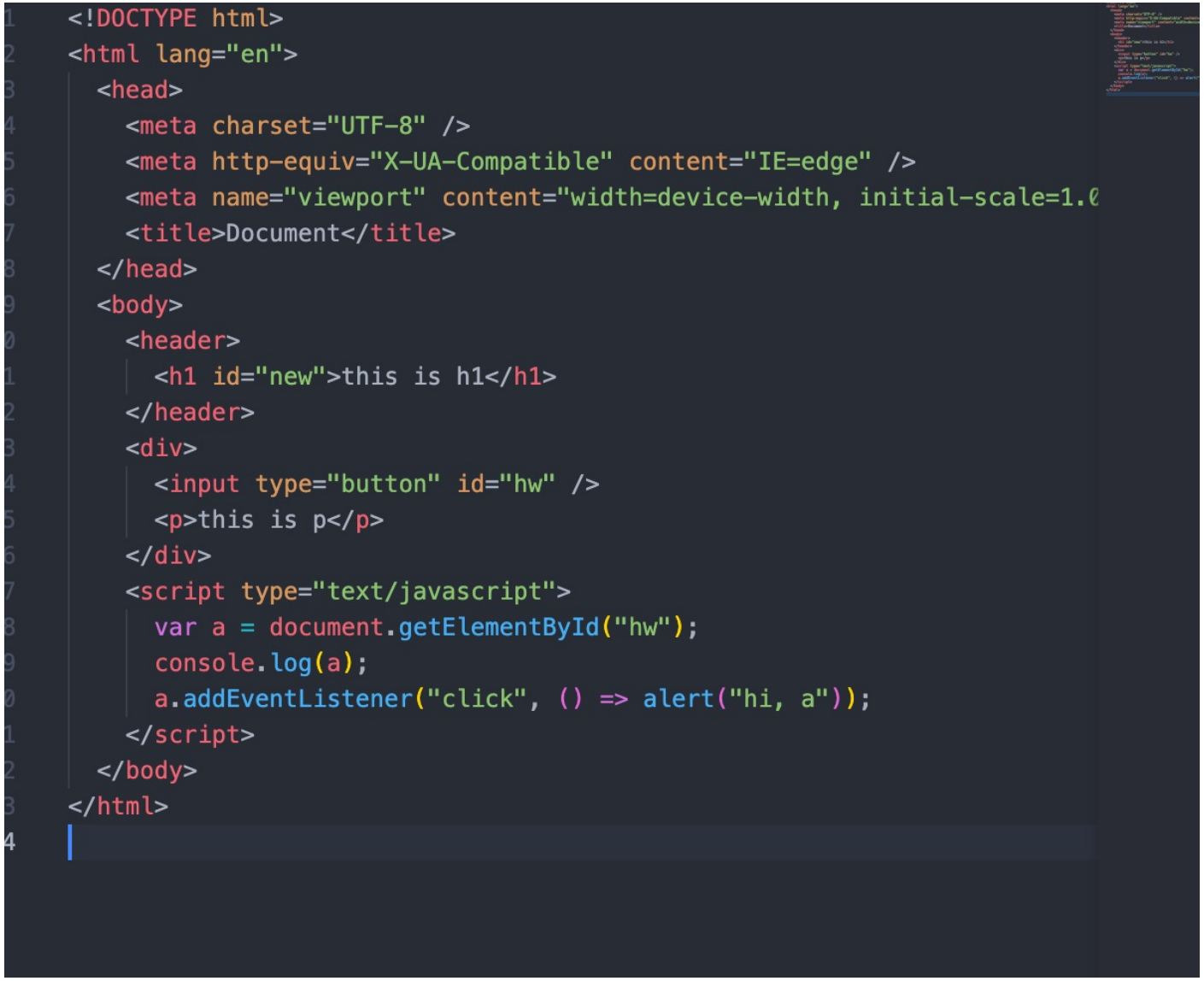
이벤트 타깃 이벤트 타입 이벤트 핸들러

```
document.getElementById("minus").addEventListener("click", function () {
    num--;
    document.getElementById("number").innerHTML = num;
});
```

JS & HTML _ inline

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0
    <title>Document</title>
  </head>
  <body>
    <header>
      <h1 id="newId" onclick="alert('hihi')">this is h1</h1>
    </header>
    <div>
      <p>this is p</p>
    </div>
  </body>
</html>
```

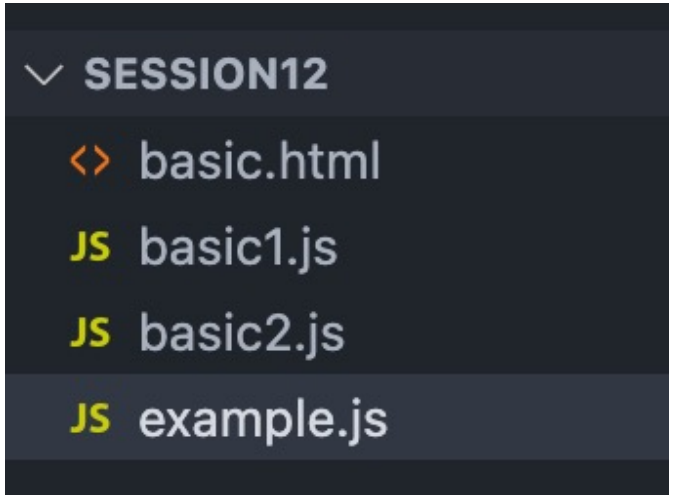
JS & HTML _ script



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <header>
      <h1 id="new">this is h1</h1>
    </header>
    <div>
      <input type="button" id="hw" />
      <p>this is p</p>
    </div>
    <script type="text/javascript">
      var a = document.getElementById("hw");
      console.log(a);
      a.addEventListener("click", () => alert("hi, a"));
    </script>
  </body>
</html>
```

JS & HTML _ 외부파일

```html
    <header>
      <h1 id="new">this is h1</h1>
    </header>
    <div>
      <input type="button" id="hw" />
      <p>this is p</p>
    </div>
    <script type="text/javascript" src="example.js"></script>
  </body>
</html>
```

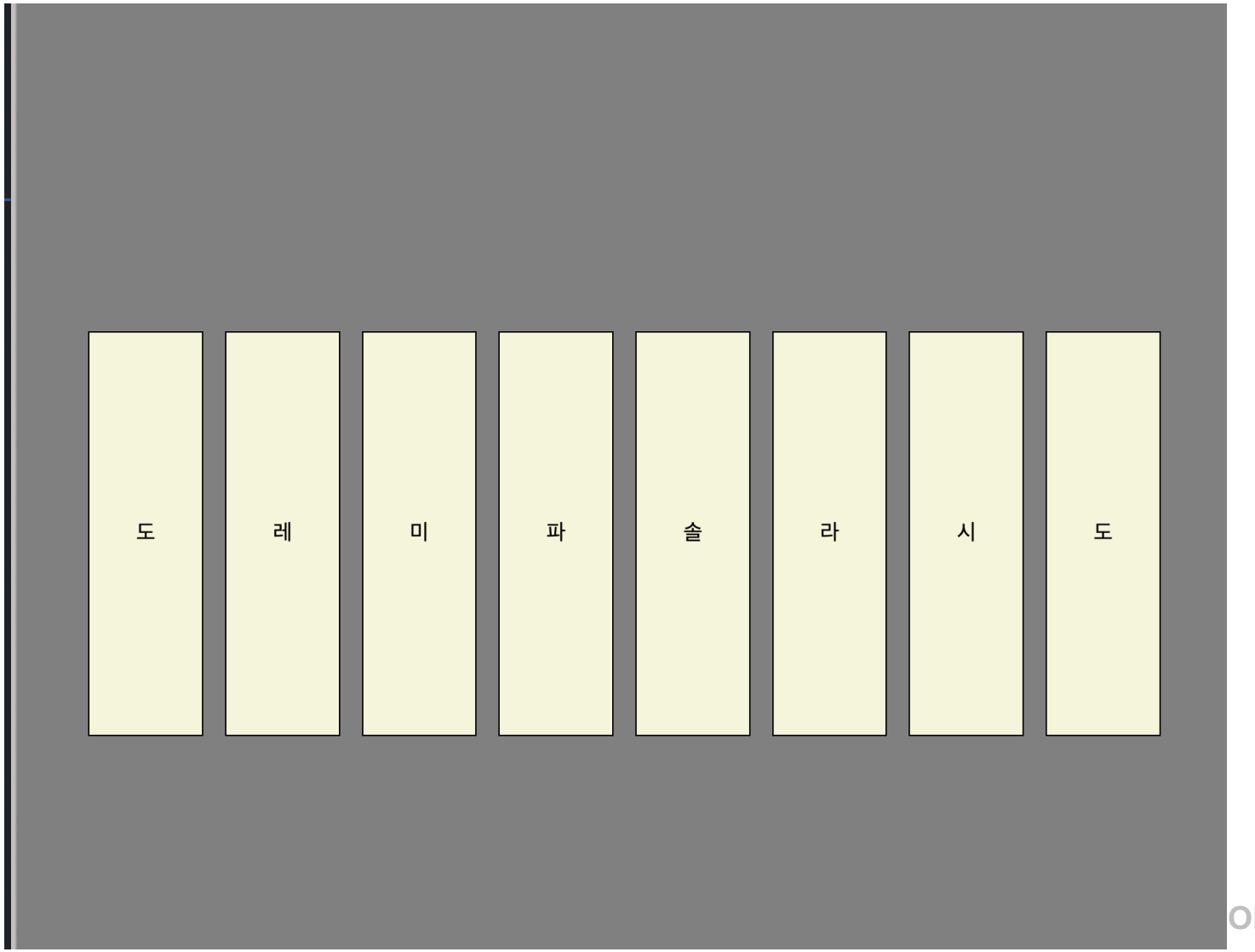


example.js > ...

```js
var hw = document.getElementById("hw");
hw.addEventListener("click", () => alert("hi"));
```

JS 응용 간단 실습
키보드 피아노 만들기

https://javascript30.com/



JS 기

1. JavaScript Drum Kit

1.1. 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.
피아노 에서는 keypress 이벤트에 쓸 함수에 사용할 data-press 속성과 keyup 이벤트에 쓸 함수에 사용할 data-key 속성을 이용해 새로운 속성값을 부여할 수 있었다.

1.2. CSS

1.2.1. transform: scale();

---

scale(가로, 세로),, 하나만 쓰면 가로세로 공통 적용 확대하는 효과!

1.2.2. border-color + box-shadow

이 피아노에서 border-color 와 box-shadow 에 동일한 색상을 부여하여 자연스럽게 강조되는 효과를 준다 . 기본 li 태그의 속성중 transition all .12s 가 있기에 더 자연스럽게 확장되는 느낌이다.

JS 기초

1.3. Javascript

1.3.1. addEventListener

`target.addEventListener(type, listener..);` type 반응할 이벤트 유형을 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)
강의에서는 keydown을 이용해 구현했지만 키가 눌리는 동안 계속해서 소리가 나기를 원하는 것이 내 목표였기 때문에 keypress와 keydown 으로 바꿔 구현함

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.

미션 : 시간 좀 많이 드릴예정

1. index.html과 main.js를 연결해주세요

2. Main.js에 들어가 키보드를 누르면 play 함수가 실행되게, play함수는 누른 키에 헤당하는 음을 재생하도록/ 키보드를 뗀다면 pause함수가 실행되고 pause함수는 뗀 키에 헤당하는 음재생을 멈추도록 코드를 짜주세요

정답

https://github.com/rkdud007/30javascript/tree/main/day1

미리 보지 말기