

AI 활용 빅데이터분석 풀스택웹서비스 SW 개발자 양성과정

JavaScript

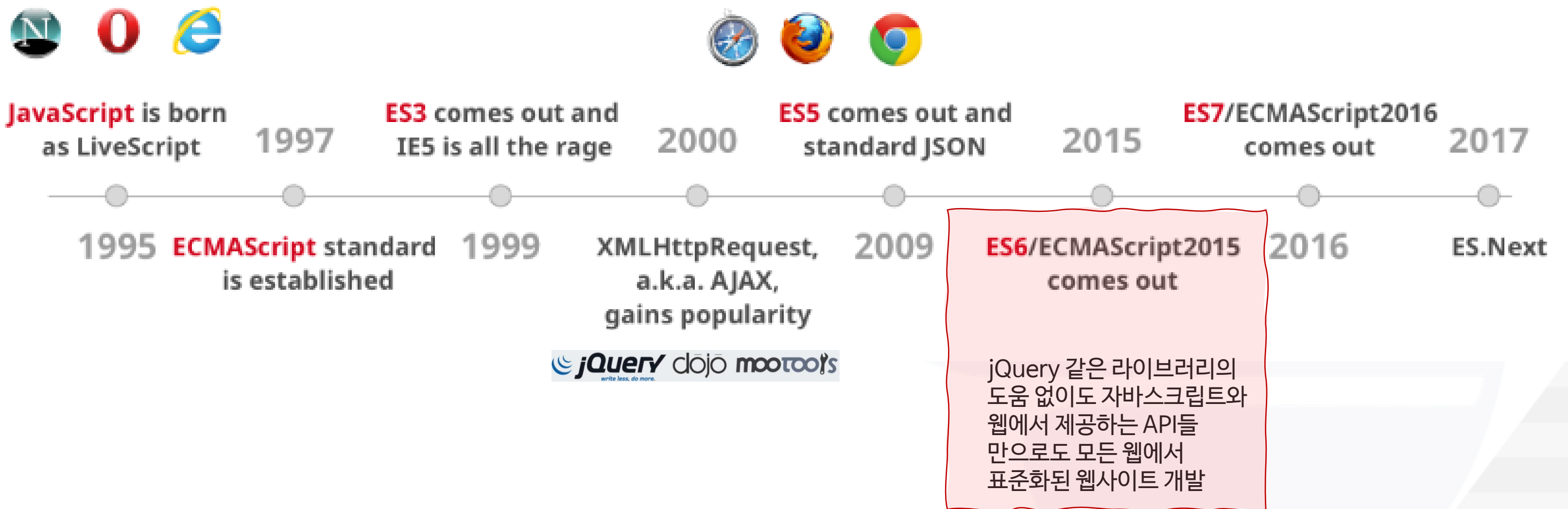


부산대학교 소프트웨어교육센터
PUSAN NATIONAL UNIVERSITY SOFTWARE EDUCATION CENTER



JavaScript

- 가벼운, 인터프리터 혹은 just-in-time 컴파일 프로그래밍 언어
 - 웹 브라우저에 인터프리터가 내장되어 주로 클라이언트 측 프로그래밍 작성
 - Node.js와 같은 프레임워크를 사용하면 서버 측 프로그래밍에서도 사용 가능



JavaScript

- 자바스크립트 엔진 (JavaScript engine)
 - 브라우저엔 '자바스크립트 가상 머신'이라 불리는 엔진이 내장
 - V8(Chrome , Opera), SpiderMonkey(Firefox)
- 자바스크립트로 할 수 있는 일
 - 페이지에 새로운 HTML을 추가하거나 기존 HTML, 혹은 스타일 수정하기
 - 마우스 클릭이나 포인터의 움직임, 키보드 키 눌림 등과 같은 사용자 행동에 반응하기
 - 네트워크를 통해 원격 서버에 요청을 보내거나, 파일 다운로드, 업로드하기 (AJAX나 COMET과 같은 기술 사용)
 - 쿠키를 가져오거나 설정하기. 사용자에게 질문을 건네거나 메시지 보여주기
 - 클라이언트 측에 데이터 저장하기 (로컬 스토리지)
- 자바스크립트 버전
 - https://www.w3schools.com/js/js_versions.asp

JavaScript 시작하기

- 내부 자바스크립트 코드로 작성

- <script> 요소로 삽입

- 외부 자바스크립트 파일 삽입하여 작성

- <script src=""> 요소의 src 속성을 이용하여 외부 자바스크립트 파일 추가

- 단, 작성된 외부 자바스크립트 파일의 확장자는 .js로 사용

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>자바스크립트</title>
  <script>
    alert("안녕하세요.") ;
  </script>
</head>
<body>
  <h1>JAVASCRIPT 실습</h1>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>자바스크립트</title>
</head>
<body>
  <h1>JAVASCRIPT 실습</h1>
  <script>
    alert("안녕하세요.") ;
  </script>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>자바스크립트</title>
  <script src="./scripts/js01.js"></script>
</head>
<body>
  <h1>JAVASCRIPT 실습</h1>
</body>
</html>
```

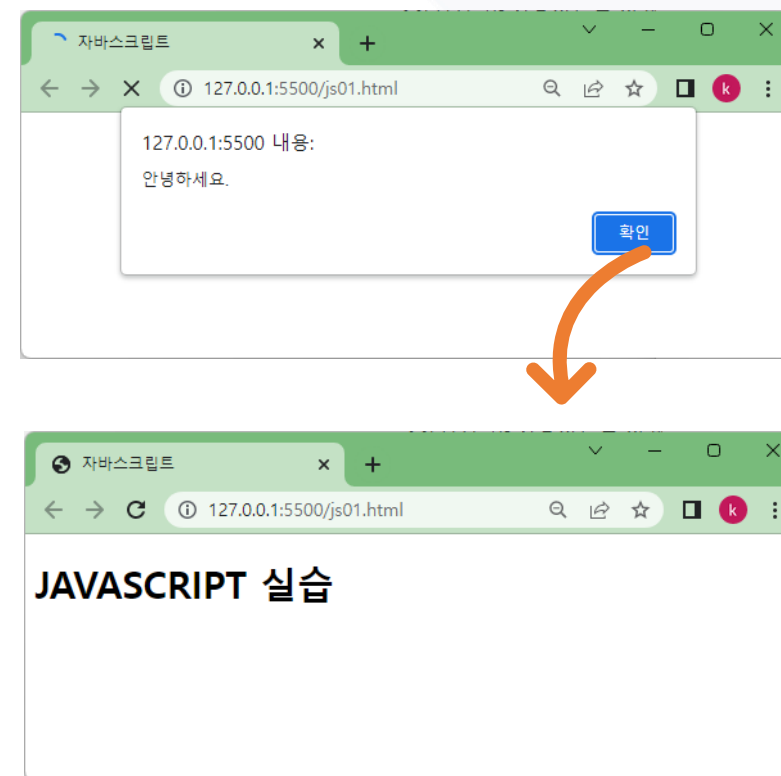
scripts > JS js01.js

```
1 alert("안녕하세요.") ;
2
```

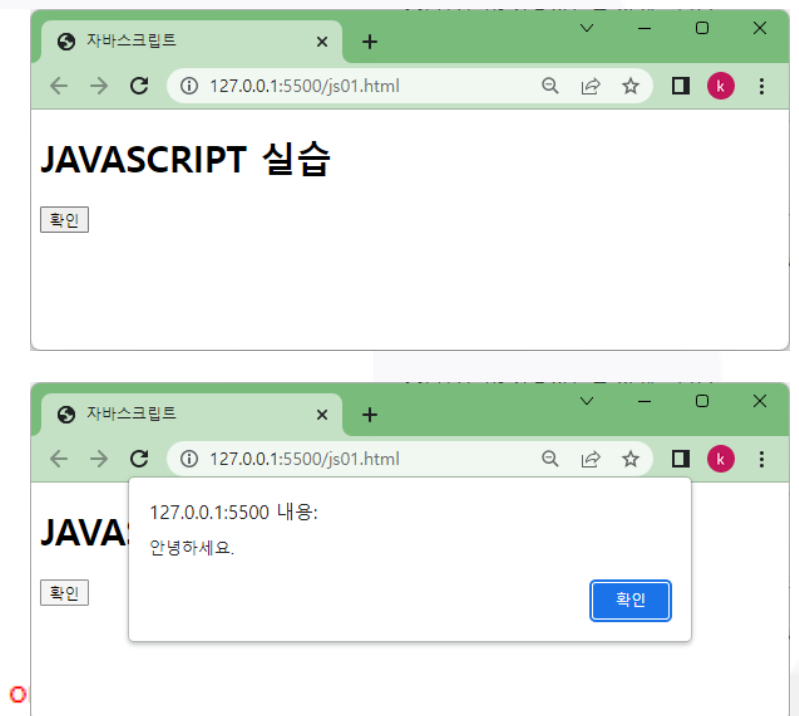
JavaScript 시작하기

- 내부 자바스크립트 코드로 작성
 - `<script>` 요소로 삽입
- 외부 자바스크립트 파일 삽입하여 작성
 - `<script src="">` 요소의 `src` 속성을 이용하여 외부 자바스크립트 파일 추가
 - 단, 작성된 외부 자바스크립트 파일의 확장자는 `.js`로 사용

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>자바스크립트</title>
  <script>
    alert("안녕하세요.");
  </script>
</head>
<body>
  <h1>JAVASCRIPT 실습</h1>
</body>
</html>
```



```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>자바스크립트</title>
  <script>
    function hello() {
      alert("안녕하세요.");
    }
  </script>
</head>
<body>
  <h1>JAVASCRIPT 실습</h1>
  <input type="button" value="확인" />
</body>
</html>
```



DOM(Document Object Model)

- 웹 페이지가 로드 되면 브라우저는 페이지의 문서 객체 모델을 생성
 - 문서의 구조화된 표현 (structured representation)을 제공하며 프로그래밍 언어가 DOM 구조에 접근할 수 있는 방법을 제공
- DOM CRUD(Create, Read, Update, Delete)
 - 페이지의 모든 HTML 요소 및 속성 변경, 추가, 제거
 - 페이지의 모든 CSS 스타일 변경
 - 페이지의 모든 기존 HTML이벤트에 반응하거나 새로운 HTML 생성
- JavaScript를 통해 DOM CRUD를 구현

JavaScript DOM 제어

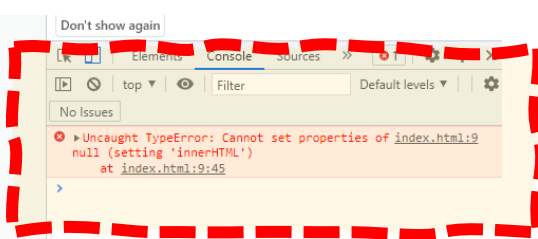
- DOM이 렌더링이 된 후 제어

- document.addEventListener("DOMContentLoaded", function(){

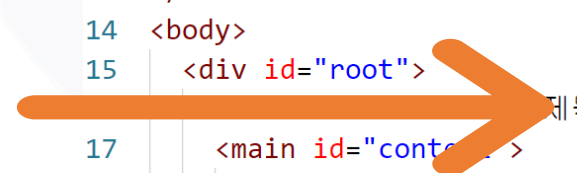
- 단 한번만 실행

```
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, user-scalable=no">
6   <title>자바스크립트</title>
7   <link rel="stylesheet" href="./styles/index.css">
8   <script>
9     document.getElementById("h1").innerHTML = "자바스크립트연습";
10  </script>
11 </head>
12 <body>
13   <div id="root">
14     <header><h1 id="h1">제목</h1></header>
15     <main id="content">
16       main내용
17     </main>
18   </div>
19 </body>
20 </html>
```

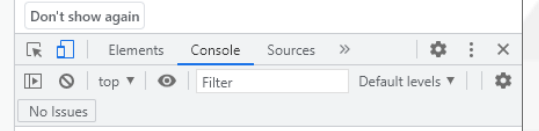
제목



```
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, user-scalable=no">
6   <title>자바스크립트</title>
7   <link rel="stylesheet" href="./styles/index.css">
8   <script>
9     document.addEventListener("DOMContentLoaded", function(){
10       document.getElementById("h1").innerHTML = "자바스크립트연습";
11     });
12 </script>
13 </head>
14 <body>
15   <div id="root">
16     <header><h1 id="h1">제목</h1></header>
17     <main id="content">
18       main내용
19     </main>
20   </div>
21 </body>
22 </html>
```



제목



화살표 함수

- ES6 추가
- 함수 표현식을 작성하기 위한 짧은 구문으로 함수명이 없는 경우
=> (팻애로우:fat arrow)를 사용하여 작성

```
/*  
document.addEventListener("DOMContentLoaded", function(){  
    document.getElementById("h1").innerHTML = "자바스크립트연습";  
});  
*/
```

```
/* 화살표 함수 적용 */  
document.addEventListener("DOMContentLoaded", () => {  
    document.getElementById("h1").innerHTML = "자바스크립트연습";  
});
```


DOM Create

- **createElement()**
 - document 내에 새로운 요소 생성
 - 생성된 요소는 변수에 저장
- **append()**
 - 새로 생성한 요소된 요소를 특정 노드에 연결
 - 여러 개의 노드 객체나 요소의 텍스트를 매개변수로 사용
- **appendChild()**
 - 선택한 요소 안에 자식요소를 추가
 - 단 하나의 노드 객체만 사용

```
const domAdd = () => {  
  const btn1 = document.createElement("button") ;  
  btn1.innerHTML = "버튼1" ;  
  btn1.id = "bt1"  
  btn1.className = "btc1"  
  document.getElementById("content").append(btn1);  
  
  const btn2 = document.createElement("button") ;  
  btn2.innerHTML = "버튼2" ;  
  btn2.id = "bt2"  
  btn2.className = "btc2"  
  document.getElementById("content").append(btn2);  
  
  const btn3 = document.createElement("button") ;  
  btn3.innerHTML = "버튼3" ;  
  btn3.id = "bt3"  
  btn3.className = "btc1"  
  document.getElementById("content").append(btn3);  
}
```

```
<body>  
  <header>...</header> flex  
  <main id="content"> flex == $0  
    <button id="bt1" class="btc1">버튼1</button>  
    <button id="bt2" class="btc2">버튼2</button>  
    <button id="bt3" class="btc1">버튼3</button>  
  </main>
```

DOM Read

- DOM 내의 특정 요소를 조회
- **document.querySelector(selectors)**
 - 제공한 선택자 또는 선택자 문치와 일치하는 문서 내 첫 번째 Element를 반환
- **document.querySelectorAll(selectors)**
 - 지정된 셀렉터 그룹에 일치하는 다큐먼트의 엘리먼트 리스트를 나타내는 정적(살아 있지 않은) NodeList 를 반환
 - 요소.forEach(함수)
- **document.getElementById(id)**
 - 주어진 문자열과 일치하는 id 속성을 가진 요소를 찾고, 이를 나타내는 Element 객체를 반환
- **document.getElementsByTagName(name)**
 - 주어진 태그명을 가진 요소를 찾고, 이를 나타내는 Element 객체를 반환

```
// DOM 조회
const domRead = () => {
  const btc1 = document.querySelectorAll(".btc1");
  console.log(btc1);

  btc1.forEach((item, idx) => {
    console.log(idx);
    console.log(item.innerHTML);
    console.log(item.id);
  });
}
```

자바스크립트 연습

버튼1

버튼2

버튼3

DevTools is now available in Korean!

Always match Chrome's language Switch DevTools to Korean Don't show again

	Elements	Console	Sources	Network	Performance	
	top	Filter	Default levels	No Issues		
	NodeList(2) [button#bt1.btc1, button#bt3.btc1]					index.js:25
0	버튼1					index.js:28
bt1						index.js:29
1	버튼3					index.js:30
bt3						index.js:28
						index.js:29
						index.js:30

반복 for

- for
 - ES1 버전 부터 있었던 가장 전통적인 반복문
- for in
 - Object의 key를 순회하기 위해 사용되는 반복문
- forEach(함수)
 - Array를 순회하는 데 사용되는 Array의 메소드
 - 배열의 요소와 인덱스 모두에 접근
 - await을 루프 내부에 쓸 수 없음
 - 중간에 루프를 탈출할 수 없음
- for of 구문
 - ES6에 나온 가장 최신 기능
 - await을 사용
 - break continue를 사용
 - 키만 접근하거나, 혹은 키와 값 모두 접근하거나 하는 것이 모두 가능

```
//for 구문
console.log("for");
for(let i=0; i < class2.length ; i ++){
  console.log(class2[i]);
}
```

```
//for in 구문 : Object의 key를 순회하기 위해 사용되는 반복문
console.log("for in");
for(let i in class2){
  console.log(class2[i]) ;
}
```

```
//forEach(함수) 구문 : Array를 순회하는 데 사용되는 Array의 메소드
console.log("forEach");
class2.forEach((item,idx) => console.log(idx, item))
```

```
//for of 구문 : 이터러블한 객체를 모두 순회할 수 있는 강력한 반복문
console.log("for of");
for(let item of class2){
  console.log(item) ;
}
for(let [idx, item] of class2.entries()) {
  console.log(idx, item) ;
}
```

변수/상수 선언

- 선언

- 변수 선언 : let
- 상수 선언 : const

- 명명규칙

- 변수명에는 오직 문자와 숫자, 그리고 기호 \$와 _로 작성
- 첫 글자는 숫자가 될 수 없음
- 대소문자 구분

- 한번만 선언

- 이전 변수 선언

- 변수 선언 : var

- 호이스팅

- 코드 실행전에 변수나 함수의 선언이 저장되어 선언 구문이 파일의 최상단으로 끌어올려져 선언문보다 참조나 호출이 먼저 나와도 동작할 수 있음

```
console.log(x)
var x = 1
console.log(x)
```

undefined	js01.js:1
-----------	-----------

1	js01.js:3
---	-----------

변수를 let으로 수정하면 오류

```
✖ Uncaught js01.js:1
ReferenceError: Cannot access
'x' before initialization
at js01.js:1:13
```

호이스팅(Hoisting)

- 코드에 선언된 변수 및 함수를 코드의 상단으로 끌어올려서 해당 변수 및 함수 유효 범위의 최상단에 선언하는 것
 - 자바스크립트 엔진은 코드를 실행하기 전 실행 컨텍스트를 위한과정에서 모든 선언(var, let, const, function, class)을 스코프에 등록하여 코드 실행 전 이미 변수선언/함수선언이 저장되어 있기 때문에 선언문보다 참조/호출이 먼저 나와도 오류 없이 동작
 - let, const, class를 이용한 선언문을 호이스팅이 발생하지 않는 것처럼 동작
 - 스코프의 시작에서 변수의 선언까지 일시적 사각지대(Temporal Dead Zone; TDZ)에 빠지기 때문
 - var 키워드는 선언과 함께 undefined로 초기화되어 메모리에 저장되는데 let과 const는 초기화되지 않은 상태로 선언만 메모리에 저장
 - TDZ의 영향을 받지 않는 구문 : var, function, import 구문

자료형

- 동적 타입 (dynamically typed) 언어
 - 자료의 타입은 있지만 변수에 저장되는 값의 타입은 언제든지 바꿀 수 있는 언어
- 자료형
 - 숫자 : 정수, 부동 소수점 숫자 등의 숫자
 - 문자열 : 작은따옴표(')나 큰따옴표(")로 묶인 문자열
 - 불린형: true, false
 - 배열: 대괄호로 묶이고 쉼표로 구분 된 여러 값을 포함하는 단일 객체
 - 객체: 복잡한 데이터 구조를 표현
 - null : 알 수 없는 null 값을 위한 독립 자료형
 - undefined : 할당되지 않은 undefined 값을 위한 독립 자료형
- 자료형 확인 : typeof(변수명)

DOM Update

- 요소 값 변경
- innerHTML
 - 요소 내의 HTML 소스를 가져옴
- innerText
 - style이 적용된 웹 브라우저에서 보이는 텍스트를 가져옴
 - 만약 style.display = 'none'이면 값을 가져오지 못함
- textContent
 - style에 상관없이 요소의 텍스트를 가져옴

```
//DOM 수정
const domUpdate = () => {
  document.querySelector("#bt1").innerHTML = "<a href='http://naver.com'>버튼1</a>" ;
  document.querySelector("#bt2").addEventListener("click", ()=>{ console.log("click button2")})
}
```

```
<!DOCTYPE html>
<html lang="ko">
  <head>...</head>
  <body>
    <header>...</header>
    <main id="content">
      <button id="bt1" class="btc1">
        <a href="http://naver.com">버튼1</a>
      </button>
      <button id="bt2" class="btc2">버튼2</button>
      <button id="bt3" class="btc1">버튼3</button>
    </main>
```

DOM 이벤트 추가하기

```
addEventListener(type, listener);  
addEventListener(type, listener, options);  
addEventListener(type, listener, useCapture);
```

//DOM이 로드된 후 실행

```
document.addEventListener("DOMContentLoaded", () => {  
  const imgIcon1 = document.querySelector("#imgIcon1")  
  const imgIcon2 = document.querySelector("#imgIcon2")  
  const imgIcon3 = document.querySelector("#imgIcon3")
```

```
  imgIcon1.addEventListener("mouseover", changeView1) ;  
  imgIcon2.addEventListener("mouseover", changeView2) ;  
  imgIcon3.addEventListener("mouseover", changeView3) ;
```

```
})
```

```
imgIcon1.addEventListener("mouseover", ()=>changeView(1)) ;  
imgIcon2.addEventListener("mouseover", ()=>changeView(2)) ;  
imgIcon3.addEventListener("mouseover", ()=>changeView(3)) ;
```

- **addEventListener()**

- 한 요소에 여러 이벤트 처리를 연결

- type

- 수신할 이벤트 유형을 나타내는 대소문자 구분 문자열
- <https://developer.mozilla.org/ko/docs/Web/Events>

- listener

- 지정한 이벤트를 수신할 객체
- 이벤트가 발생하면 처리할 함수

- options (Optional)

- 이벤트 수신기의 특징을 지정할 수 있는 객체

- useCapture (Optional)

- 이벤트 대상의 **DOM** 트리 하위에 위치한 자손 **EventTarget**으로 이벤트가 전달되기 전에, 이 수신기가 먼저 발동돼야 함을 나타내는 불리언 값

DOM Delete

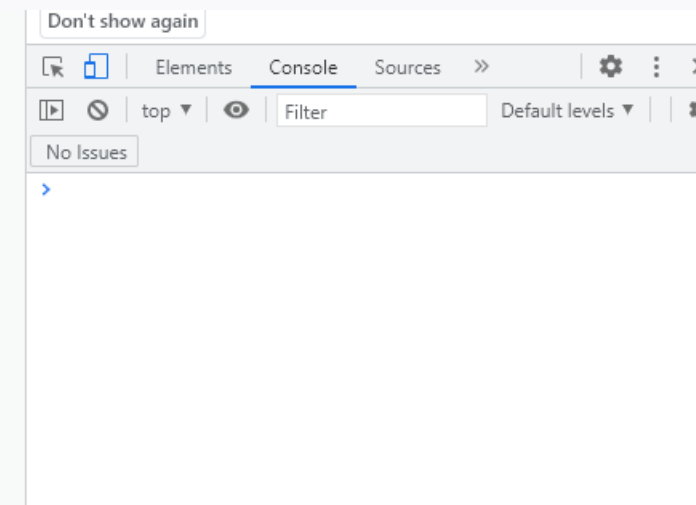
- remove()

–요소 제거

```
//DOM Delete
const domDelete = () => {
  const mainE = document.querySelector("#bt")

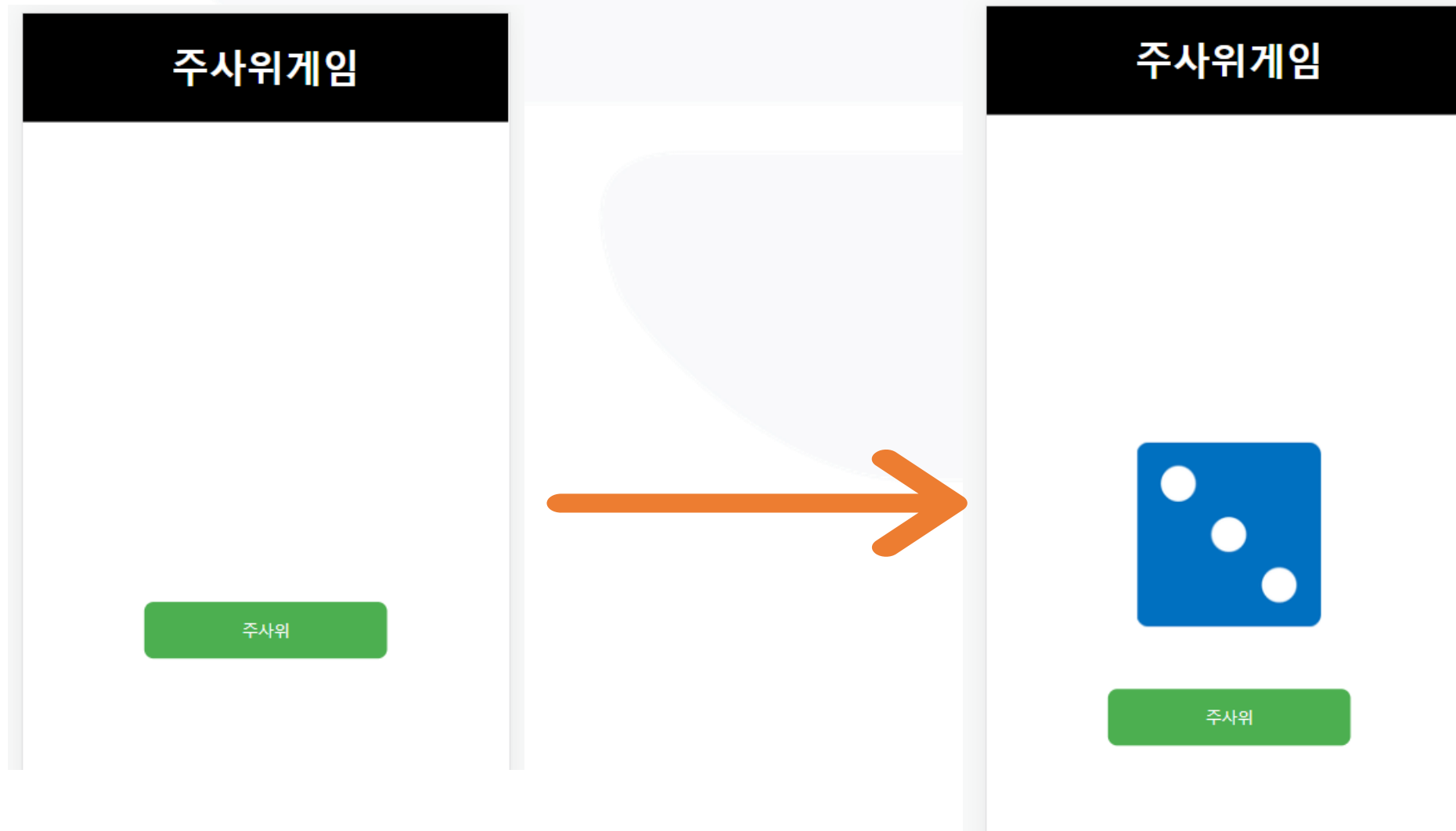
  mainE.remove();
}
```

자바스크립트연습



해결문제

- 주사위 버튼을 누르면 임의의 주사위 번호가 나오도록 작성



연산자

비교연산자	이름	예제
===	엄격 일치 (정확히 같은가?)	<code>5 === 2 + 4 // false</code> <code>'Chris' === 'Bob' // false</code> <code>5 === 2 + 3 // true</code> <code>2 === '2' // false</code> 숫자와 문자열은 다름 Copy to Clipboard
==	동등연산	<code>0 == false // true</code> <code>" == false // true</code> 다른 피연산자를 비교할 때 피연산자를 숫자형으로 바꾸기 때문에 발생
!==	불일치 (같지 않은가?)	<code>5 !== 2 + 4 // true</code> <code>'Chris' !== 'Bob' // true</code> <code>5 !== 2 + 3 // false</code> <code>2 !== '2' // true</code> 숫자와 문자열은 다름 Copy to Clipboard
<	미만	<code>6 < 10 // true</code> <code>20 < 10 // false</code> Copy to Clipboard
>	초과	<code>6 > 10 // false</code> <code>20 > 10 // true</code>

산술연산자	이름	예제
+	더하기	<code>6 + 9</code>
-	빼기	<code>20 - 15</code>
*	곱하기	<code>3 * 7</code>
**	거듭제곱	<code>2 ** 4</code>
%	나머지	<code>10 % 2</code>
/	나누기	<code>10 / 5</code>



- 일치연산자 ===를 사용하여 null과 undefined를 비교하면 거짓
- 동등연산자 ==를 사용하여 null과 undefined를 비교하면 참
- undefined를 다른 값과 비교하면 안됨

DOM CSS 제어

- **style 속성**
 - CSS 속성 값을 수정
- **getComputedStyle()**
 - CSS속성값을 가지고 옴
- **setProperty(속성, 값)**
 - style 값 설정하기
- **getPropertyValue(속성)**
 - style 값 가져오기
- **item(인덱스)**
 - style값 가져오기, 인자로 index를 사용
- **removeProperty(속성)**
 - style 값 삭제하기

```
const showGameOk = () => {  
  const bt1 = document.getElementById("bt1") ;  
  const bt2 = document.getElementById("bt2") ;  
  const inNum = document.getElementById("inNum") ;  
  const dice = document.getElementById("dice") ;  
  
  //주사위 이미지 숨기기  
  dice.style.display = 'block' ;  
  
  //주사위 버튼 보이기  
  bt1.style.display = 'block' ;  
  
  //확인 버튼 숨기기  
  bt2.style.display = 'none';  
  
  //번호 선택 보이기  
  inNum.style.display = 'none';  
}
```


템플릿 문자열

- 백틱(`)으로 감싸면 문자열 안에 변수 사용 가능
- \${변수}

```
let htmlTag = `<img src='./images/${n}.png'>`
```

해결문제

- 랜덤으로 생성된 주사위 번호를 맞추는 게임을 작성

