

리액트를 활용한 웹 시스템 개발 과정

한국정보기술연구원 X 스나이퍼팩토리





강사 정동훈

현) 주식회사 세이프홈즈 대표이사

- 청년창업사관학교 우수 졸업

전) 대한민국 육군 대위

- 응용 프로그램 및 웹 개발, PM 6년

- 육군 3사관학교 암호학과

- SW협회 다수 프론트엔트 React 기초 강의

- 2019공개 SW대회 대상

- 미 PMP(Project Management Professional)자격

2강

HTML/CSS/JS

Whatever you want, Make it real.

강사 정동훈

React와 가까워지기

{

HTML, CSS, JS 기초 학습

}

React를 사용해야 하는 이유



1교시

HTML, CSS, JS 상관관계

HTML
웹 문서 뼈대
웹 요소를 보여주는 정해진
약속

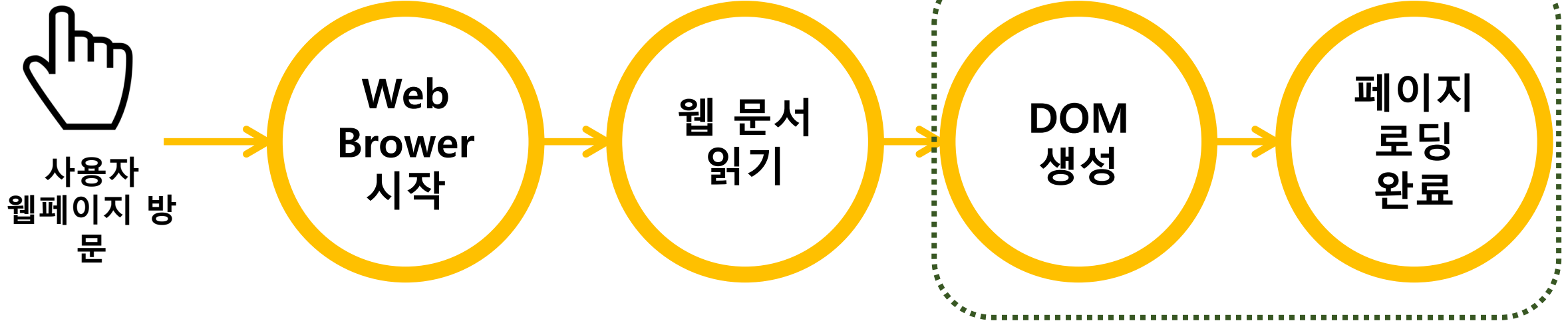
CSS
뼈대를 꾸며주는 CSS

JS
사용자의 액션



HTML, JS, CSS 작동 순서

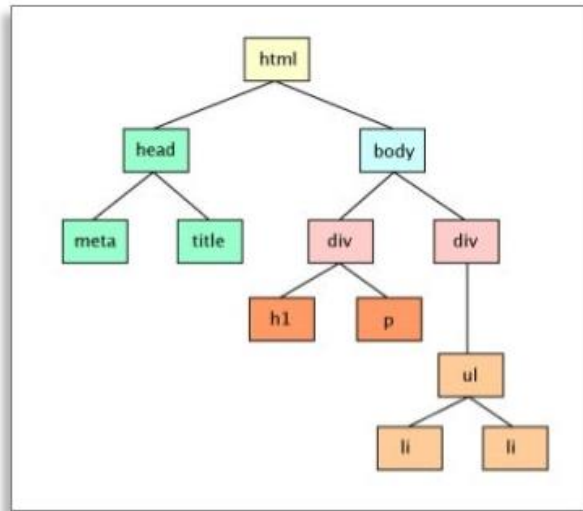
HTML을 객체 형태로 변환 → Javascript 로 접근 가능



CSS 적용 화면
최종 표시

DOM(Document Object Model) 이란?

※ DOM은 넓은 의미로 웹 브라우저가 HTML 페이지를 인식하는 방식을 의미합니다.



http://finearts.fontbonne.edu/tech/web/script/js_DOM.html

문서 객체가 생성되는 방식?

1. 정적으로 문서 객체를 생성 : 웹 브라우저가 HTML 페이지에 적혀 있는 태그를 읽으면 생성하는 것
2. 동적으로 문서 객체를 생성 : 원래 HTML 페이지에 없던 문서객체를 JavaScript를 이용해서 생성

{HTML} : Hyper Text Markup Language

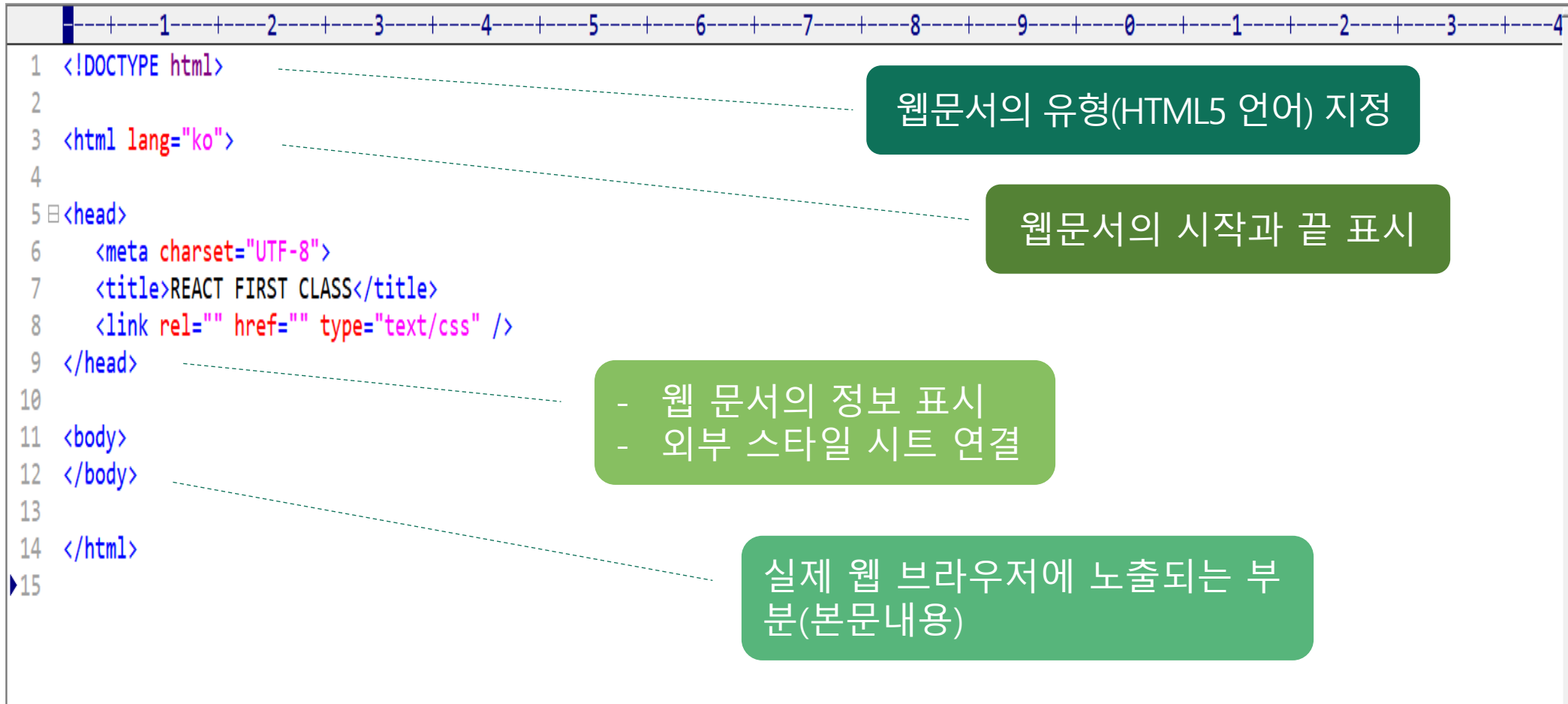
웹 페이지를 만드는 데 사용되는 마크업 언어



HTML을 사용하여 텍스트, 이미지, 비디오 등과 같은 콘텐츠를 구성하고 구조화 할 수 있으며, 이를 통해 웹 페이지를 디자인하고 브라우저에 표시할 수 있습니다.

HTML은 웹 개발의 기초 중 하나이며, 주로 CSS, JavaScript와 함께 사용됩니다.

{HTML} : Hyper Text Markup Language



```
1 <!DOCTYPE html>
2
3 <html lang="ko">
4
5 <head>
6   <meta charset="UTF-8">
7   <title>REACT FIRST CLASS</title>
8   <link rel="" href="" type="text/css" />
9 </head>
10
11 <body>
12 </body>
13
14 </html>
15
```

웹문서의 유형(HTML5 언어) 지정

웹문서의 시작과 끝 표시

- 웹 문서의 정보 표시
- 외부 스타일 시트 연결

실제 웹 브라우저에 노출되는 부분(본문내용)

{HTML} : Hyper Text Markup Language

시맨틱 요소(**Semantic Elements**) : 의미가 있는 요소

시맨틱 요소를 고려하여 만든 html은 tag의 이름만 보고도 tag 가 html 안에서 어떠한 역할을 수행하는지 브라우저와 개발자 모두 쉽게 알 수 있다.



시맨틱 요소를 고려하여 HTML을 제작하면
웹 접근성을 높일 수 있습니다.

웹 접근성을 높이면 웹페이지에 일반적인 방법으로
접근할 수 없는 사람들(시각장애인)들도
원활한 소통이 가능하도록 도울 수 있습니다.

이 레이아웃 이외에도 여러가지 종류가 있지만
기본적인 구조는 비슷합니다.

* Header와 Main, Footer의 위치는 고정

HEADER

SECTION

{HTML} : Hyper Text Markup Language



HTML 태그들을 올바르게 사용해야 하는 이유는 웹 표준을 준수하기 위해서 입니다.

HTML 태그를 올바르게 사용하면 좋은 점?

1. 웹 페이지의 구조와 의미를 명확하게 합니다.
2. 가독성이 향상됩니다.
3. 가독성이 높은 코드는 유지보수가 용이하며, 코드 수정이나 업데이트 작업시에도 문제가 발생할 확률이 적어집니다.
4. 크로스 브라우징 이슈(각 브라우저에서 웹 페이지가 다르게 보이는 문제)를 최소화 할 수 있습니다.
☒ 쉽게 말해, CSS를 입히지 않은 상태에서도 내용이 잘 읽혀야 합니다.

{HTML} HTML의 기본 태그

태그명	의미	태그명	의미
<html>	웹 문서 유형을 html로 지정	<h1> </h1> <h2> </h2> ... <h6> </h6>	제목(내용)의 크기
<head>	모든 html 태그의 최상의 태그	<p>	문단 나눔
<meta>	메타 데이터 입력	 	줄바꿈
<title>	문서의 제목	<hr>	horizontal rule (수평선)
<body>	문서의 본문	 	순서가 있는 목록
<a href src="#" />	하이퍼 링크 삽입	 	순서가 없는 목록

{HTML} HTML의 기본 태그

태그명	의미		
<table>	테이블 만들기		
	<caption>	표 제목	
	<tr>	<td>	셀 삽입
		<th>	셀 삽입 (진하게 삽입, 컬럼명 입력할 때 주로 쓰임.)
		rowspan="2"	2개의 행 합치기
		colspan = "2"	2개의 열 합치기
	<thead>	표의 제목	
	<tbody>	표의 본문	

{HTML} HTML의 기본 태그 및 식별자

태그명	태그 의미	식별자	식별자 의미
<form>	.웹 페이지의 입력 양식 .내용 입력 후, submit 버튼을 누르면 백엔드 서버에 양식이 전달	name	폼의 이름
		action	폼 데이터가 전송되는 백엔드 url
		method	폼 전송 방식 (GET / POST)

{HTML} HTML의 기본 태그 및 식별자

태그명	태그 의미	식별자	식별자 의미
<input>	다양한 타입의 input요소를 이용하여 사용자로부터 입력을 받을 수 있음.	id	고유 식별을 목적으로 하는 경우 사용
		class	여러 영역에서 재사용 가능한 식별자
		name	form 컨트롤 요소 값을 서버로 전송 시 사용하는 식별자

{HTML} : Hyper Text Markup Language

HTML에서 **ID**와 **CLASS**는 둘 다 HTML 요소에 대한 식별자(identifier)를 지정하는 데 사용됩니다.

ID 하나의 요소에 대해 고유한 식별자를 지정합니다.

즉, 문서 내에서 단 하나의 요소만 해당 ID를 가질 수 있습니다

CLASS 하나 이상의 요소에 대해 같은 식별자를 지정합니다.

즉, 문서 내에서 여러 요소가 동일한 **CLASS**를 가질 수 있습니다.

따라서 ID는 단일 요소의 식별에 사용되고, CLASS는 여러 요소의 그룹화 및 스타일링에 사용됩니다.

{CSS} : Cascading Style Sheet

※ HTML이 웹 문서의 뼈대를 만드는 것이라면, CSS는 색상이나 크기, 이미지 등 디자인 요소 담당

① 내부 스타일 시트

```
<head>
<style>
body {
  background-color: red;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>
  ...
</body>
```

② 외부 스타일 시트를 import 하여 사용

```
<link rel="stylesheet" type="text/css" href="mystyle.css">
<link rel="stylesheet" type="text/css" href="http://cdn.site.com/css/mystyle.css">
```

③ 인라인 스타일 시트 : html 파일에서 스타일 직접 지정

```
<h1 style="color:blue; margin-left:30px;">This is a heading</h1>
```

{CSS} : CSS 우선 적용 순위

※ 같은 요소에 다른 CSS 속성이 중복 설정되어 있을 때, 우선 적용되는 순서

우선순위 1 : 속성값 뒤에 !important가 붙어 있는 속성

예시)

```
<div>
  <p>안녕하세요. 정동훈입니다.</p>
</div>
<style>
  p{color: blue!important;}
  p{color: red;}
</style> </body>
```



안녕하세요. 정동훈입니다.

우선순위 2 : 인라인 스타일로 적용되어 있는 속성

우선순위 3 : 선택자에 id가 쓰인 속성

우선순위 4 : class, attribute, pseudo-class로 지정한 속성

우선순위 5 : 태그 이름으로 지정한 속성

우선순위 6 : 부모 요소에 의해 상속된 속성

{CSS} : 기본 디자인 요소 Review

속성명	의미	속성명	의미
background-color	배경색	margin	바깥 여백 설정
background-image	배경 이미지 정의	padding	안쪽 여백 설정
border	경계선 스타일 정의	color	글자색을 변경
border-radius	테두리 둥글게 처리	text-align	글자 수평 중앙 정렬
letter-spacing	text 자간 설정	display	none: 해당 부분을 숨김 block: 해당 부분을 보임

{CSS} : 기본 선택자 Review

Selector	예제	설명
#id	#firstname	id가 firstname 인 요소를 선택한다.
.class	.intro	class가 intro인 요소를 선택한다.
*	*	모든 요소를 선택한다.
element	p	모든 <p> 요소를 선택한다.
element, element	div, p	모든 <p>와 <div> 요소를 선택한다.
element > element	div > p	부모가 <div>요소인 모든 <p> 요소를 선택한다.
element+element	div+p	<div> 요소 바로 뒤에 배치되는 첫번째 <p> 요소를 선택한다.

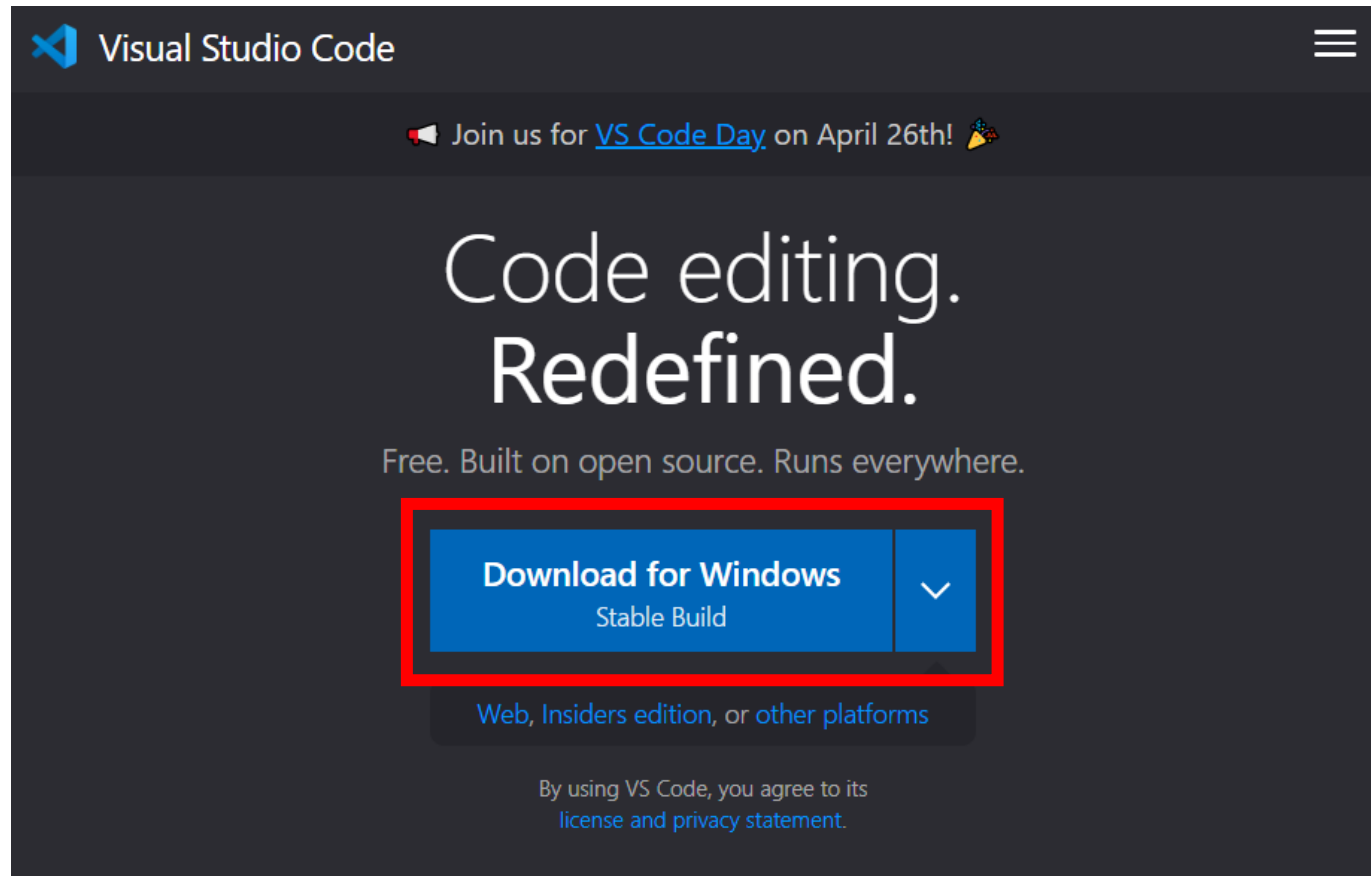
{CSS} : 자주쓰는 CSS Pseudo-class Selector

Selector	예제	설명
:active	a:active	활성 링크를 선택
:hover	a:hover	마우스 오버가 되는 링크를 선택
:visited	a:visited	방문한 모든 링크 선택
:first-child	p:first-child	부모의 첫 번째 자식인 모든 <p> 요소를 선택
:nth-child(n)	p:nth-child(2)	부모의 두 번째 자식인 모든 <p> 요소를 선택

프로젝트 환경 설정 - VS(Visual Studio) code 설치

1. VS code

① <https://code.visualstudio.com/> 에서 설치 파일 실행



HTML, CSS 실습

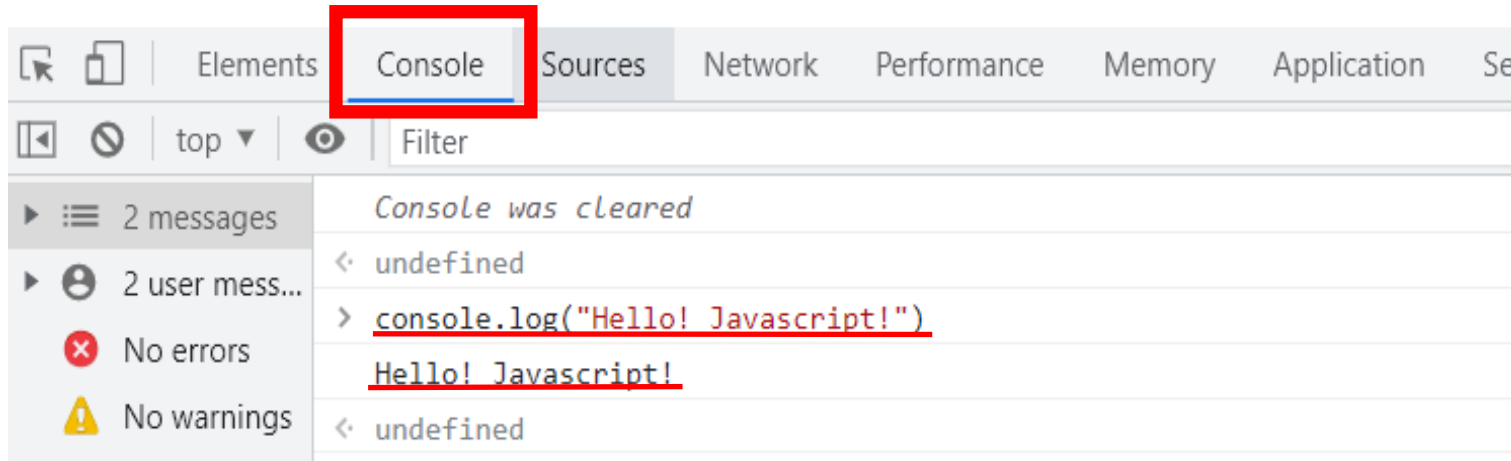


2교시

{JavaScript} Hello! Javascript!

※ Javascript를 실행 할 수 있는 가장 간단한 방법!

- ① 브라우저(크롬)을 실행 시킨 후, 개발자 도구(F12) 실행
- ② console 탭에 가서 Javascript 코드 실행



- ③ 로그 값을 찍는 `console.log("Hello! Javascript!");` 실행 → 디버깅 시 유용
`console.log(1+2+3+4+5);` 실행

{JavaScript} HTML element를 다뤄보자.

함수	설명
getElementById(id)	특정 ID를 가진 요소 조회
getElementsByName(name)	Name 속성을 가진 요소 조회
getElementsByTagName(tagName)	Tag Name 기준으로 요소 조회
getElementsByClassName(className)	특정 class를 가진 요소 조회
innerText	요소 안에 있는 글자를 가지고 올때 사용
HTMLElement.innerHTML	태그 내 새로운 요소 객체나 내용을 저장 및 조회할 때 사용
HTMLInputElement.value	input 태그의 value값을 저장 및 조회 할때 사용

{JavaScript} DOM element를 다뤄보자.

DOM element	함수	예제
생성	document.createElement	const newTag = document.createElement("div");
조회	document.querySelector (첫번째 element 반환)	document.querySelector("input");
	document.querySelectorAll (해당하는 모든 element 반환)	document.querySelectorAll(".children")
삽입	appendChild	parentElement.appendChild(newElement)
삭제	removeChild (자식 element 지정하여 삭제)	parentElement.removeChild();
	remove (해당 element 삭제)	targetElement.remove();

{JavaScript} ES6 (ECMAScript)

- ① ES6(ECMAScript6)는 자바스크립트의 버전 중의 하나 (2015년)
- ② 자바스크립트의 표준, 규격을 나타냄
- ② 새로운 자바스크립트 버전이 나올때마다 새로운 문법이 소개됨.
(ES7, ES8...)
- ③ 브라우저 버전마다 지원되는 자바스크립트 버전이 다름
→Babel:최신 버전의 자바스크립트가 구버전 브라우저에서도 실행 될 수 있도록 함.

{JavaScript} 변수와 상수

변수(variable) : 데이터 값을 넣을 수 있는 저장소이며, 바뀔 수 있는 값을 의미

상수(constant) : 값이 변하지 않는 변수

		키워드	재할당	재선언
<div> <div>잘 안쓰는 추세</div> <div>ES6 추가</div> </div>	변수 (구 JS)	var	O	O
	변수	let	O	X
	상수	const	X	X

{JavaScript} 변수와 상수

<pre>let value = 1; console.log(value); //1 value = 2; console.log(value); //2</pre>	<pre>let value = 1; let value = 2 ; //오류</pre>	<pre>const a = 1; a = 2; //오류</pre>	<pre>const a = 1; const a = 2; //오류</pre>
재할당 가능	재선언 불가능	재할당 불가능	재선언 불가능
let (변수)		const (상수)	

{JavaScript} 조건문

```
const a = 10;
if(a>15){
  console.log('a가 15 큼니다.');
```

} else {

```
  console.log('a가 15보다 크지 않습니다.');
```

}

if - else 문

```
const device = 'iphone';

switch(device) {
  case 'iphone' :
    console.log('아이폰입니다.');
```

case 'ipad' :

```
  console.log('아이패드입니다.');
```

default :

```
  console.log('갤럭시입니다.');
```

}

switch-case 문

{JavaScript} 함수 : 특정 코드를 하나의 명령으로 실행 할 수 있게 해주는 기능

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

```
const sum = add(1,2);  
console.log(sum);
```

기본 함수 정의 및 호출

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

```
const sum = add(1,2);  
console.log(sum);
```

★ ES6
추가

화살표 함수 정의 및 호출

- ① **function** 키워드 대신에 **=>** 문자 사용
- ② => 좌측에는 함수의 파라미터, 우측에는 코드 블록
- ③ `const add = (a,b) => a+b;`
로 `return` 없이 함수를 줄여 정의 할 수 있음.

{JavaScript} 함수 화살표 함수를 쓰면 안되는 경우

```
const cat = {  
  name: 'meow';  
  callName: () => console.log(this.name);  
}  
  
cat.callName(); // undefined
```

객체 內 메소드

```
const Foo = () => {};  
const foo = new Foo()  
  
// TypeError: Foo is not a constructor
```

생성자 함수

```
button.addEventListener('click', function() {  
  console.log(this); // button 엘리먼트  
  this.innerHTML = 'clicked';  
});
```

addEventListener()의 콜백함수

{JavaScript} 객체

- ① 관련된 데이터와 함수의 집합
- ② 객체 생성 시, {} 안에 key-value 형태로 값을 넣어줌.
- ③ 객체 내, 함수를 정의할 수 있음. → 객체 내 함수를 정의할 때 화살표 함수는 쓸 수 없다.

```
const ironMan= {  
  name : '토니스타크',  
  actor : '로버트 다우니 주니어',  
  alias : '아이언맨',  
  talk : function(){  
    console.log('Hello'+this.name+'.');  
  }  
}
```

```
console.log(ironMan.name);  
console.log(ironMan.actor);  
console.log(ironMan.alias);  
ironMan.talk();
```

객체 생성 및 사용

{JavaScript} 객체 this

```
const ironMan= {  
  name : '토니스타크',  
  actor : '로버트 다우니 주니어',  
  alias : '아이언맨',  
  talk : function(){  
    console.log('Hello'+this.name+'.');  
  }  
}
```

```
console.log(ironMan.name);  
console.log(ironMan.actor);  
console.log(ironMan.alias);  
ironMan.talk();
```

객체 內 this

- ① 지금 동작하고 있는 객체를 가르킴.
- ② 예제에서 this 는 ironMan 객체와 동일

객체 생성 및 사용

{JavaScript} 객체 비구조화 할당

* 객체 비구조화 할당은 ES6에서 추가

- ① 객체 비구조화 할당은 객체 구조 분해라고도 함
- ② 객체 비구조화 할당은 짧고 보기 좋게 코드를 작성하기 용이
- ③ 객체 안에 함수를 넣을 때, 화살표 함수로 선언하면 안됨.

```
const ironMan= {
  name : '토니스타크',
  actor : '로버트 다우니 주니어',
  alias : '아이언맨'
}
```

```
console.log(ironMan.name);
console.log(ironMan.actor);
console.log(ironMan.alias);
```

일반 객체 사용

```
const ironMan= {
  name : '토니스타크',
  actor : '로버트 다우니 주니어',
  alias : '아이언맨'
}
```

 ES6
추가

```
const {name, actor, alias} = ironMan;
console.log(name);
console.log(actor);
console.log(alias);
```

```
const onChange = (event) => {
  const {target:{value}} = event;
  console.log(value);
}
//console.log(value) == console.log(event.target.value)
```

객체 비구조화 할당

{JavaScript} 배열 선언

- ① 배열 : 이름과 인덱스로 참조되는 정렬된 값의 집합
- ② element(요소) : 배열을 구성하는 각각의 값
- ③ index(인덱스) : element의 위치를 가리키는 숫자

```
let arr = ['zero','one','two'];
```

```
for(let i=0; i<arr.length ; i++){
  console.log(arr[i]);
}
```

대괄호를 사용하여 만드는 방법

```
let arr = new Array();
```

```
arr[0] = 'zero';
arr[1] = 'one';
arr[2] = 'two';
```

```
for(let i=0; i<arr.length ; i++){
  console.log(arr[i]);
}
```

Array() 생성자 함수로 배열을 생성하는 방법

배열 선언 방법

{JavaScript} 배열 데이터 타입 및 추가

- ① js의 배열은 서로 다른 데이터 타입을 담을 수 있다.
- ② 배열의 크기는 동적으로 변경 될 수 있다.

<pre>let arr = [1234, 'test', true];</pre>	<pre>let arr = [1234, 'test', true]; arr.length = 5; arr[5] = 'apple' arr.push('banana');</pre> <div data-bbox="891 896 2344 1025"> <div>1234</div> <div>'test'</div> <div>true</div> <div>undefined</div> <div>undefined</div> <div>'apple'</div> <div>'banana'</div> </div>
서로 다른 데이터 타입을 담을 수 있다.	배열의 크기는 동적으로 변경 될 수 있다.
배열 데이터 타입 및 추가	

{JavaScript} 반복문

<pre>for (let i = 0; i < 10; i++) { console.log(i); }</pre>	<pre>let i = 0; while (i < 10) { console.log(i); i++; }</pre>	<pre>let numbers = [10, 20, 30, 40, 50]; for (let number of numbers) { console.log(number); }</pre>
for 문	while 문	for ... of 문
반복문		

{JavaScript} 객체의 반복문 (1) 객체의 배열 정보를 알 수 있는 함수

```
const doggy = {  
  name: '멍멍이',  
  sound: '멍멍',  
  age: 2  
};
```

```
console.log(Object.entries(doggy));  
console.log(Object.keys(doggy));  
console.log(Object.values(doggy));
```

```
▼ [Array[2], Array[2], Array[2]]  
  ▼ 0: Array[2]  
    0: "name"  
    1: "멍멍이"  
  ▼ 1: Array[2]  
    0: "sound"  
    1: "멍멍"  
  ▼ 2: Array[2]  
    0: "age"  
    1: 2
```

```
▶ ["name", "sound", "age"]
```

```
▶ ["멍멍이", "멍멍", 2]
```

① **Object.entries**: [[키, 값], [키, 값]] 형태의 배열로 변환

② **Object.keys**: [키, 키, 키] 형태의 배열로 변환

③ **Object.values**: [값, 값, 값] 형태의 배열로 변환

반복문

{JavaScript} 객체의 반복문 (2) 배열 정보를 이용한 반복문

```
const doggy = {  
  name: '멍멍이',  
  sound: '멍멍',  
  age: 2  
};  
  
for (let key in doggy) {  
  console.log(`${key}: ${doggy[key]}`);  
}
```

for ... in 문

반복문

name: 멍멍이

sound: 멍멍

age: 2

JS 실습





감사합니다. 😊