

02 웹 프로그래밍 기본

Javascript, jQuery, Ajax

김범준 강사

1. 강의정보

과정개요	Javascript/jQuery 개념확립 및 이해 및 Join form page making 제작실습
학습목표	<ul style="list-style-type: none">- Javascript의 이해- jQuery의 이해
핵심역량	Javascript/jQuery 개념확립 및 문제해결력, Join form page making(로그인 페이지) 제작역량 습득
SW융합요소	<input checked="" type="checkbox"/> 이론 <input checked="" type="checkbox"/> 실습 <input type="checkbox"/> 토론 <input type="checkbox"/> 사례분석

2. 학습내용

주제	강의내용	강의방법	시간
Javascript의 이해	<Javascript-1> - 변수, 데이터, 명령문, 함수	이론강의	10
	<Javascript-2> - 배열, 객체, Event, class		
	<Javascript API> - Javascript DOM 객체 활용 - Data, Effect, 이벤트 처리기법		
jQuery의 이해	<jQuery 기초 문법> - \$ Function의 이해 및 함수체계 분석	이론강의	10
	<jQuery API> - jQuery Data분석 및 이해 - jQuery Effect 적용 사례 및 이해		
	<jQuery Ajax> - Ajax 기초 개념 - JSON 데이터 형식 개념 - Ajax 입출력 활용		
과정실습	<과정실습 진행> Weekly project : Join form page making	이론 및 실습강의	16
강의시간(h)			36

Javascript 1

Javascript 동작 환경

- Web browser에서 동작하는 Javascript
 - 주로 DOM(Document Object Model)을 사용해서 HTML 변경.
- Web browser를 벗어난 Javascript
 - Node.js를 사용하면 웹 브라우저 환경이 아니어도 Javascript 실행 가능.
 - Back-end, desktop 앱 및 mobile 앱 개발에 사용됨. (크로스 플랫폼)
 - CRA를 이용한 React.js 프로젝트를 생성 하면 Node.js 서버 환경으로 생성 됨.

Java와 다른 Javascript

- Java

- 컴파일 방식
- 주로 어플리케이션이나 **back-end** 개발
- C ,C++ ,C#과 유사하다.
- 컴파일 결과는 JVM에서 인터프리터 되는 **bytecode**
- JVM(Java Virtual Machine) 자바 가상머신.

- Javascript

- Netscape에서 LiveScript란 이름으로 개발.
- Java가 유행 해서 Javascript로 개명.
- MS에서 JScript 발표.
- VBScript, Actionscript, JScript, Javascript 등 유사한 스크립트가 생겨 남.
- Netscape가 ECMA script 표준 제안.
- 현재 ES6+알파 버전으로 발전 (2015년 이후)

- 형태가 비슷해 보이나 계보와 용도가 다름.

- 자바스크립트는 스크립트 언어 중엔 최강이다.
- NASA에서도 공식적으로 Javascript 사용. (드론 개발에도 Javascript가 사용된다.)

SPA(Single Page Application)

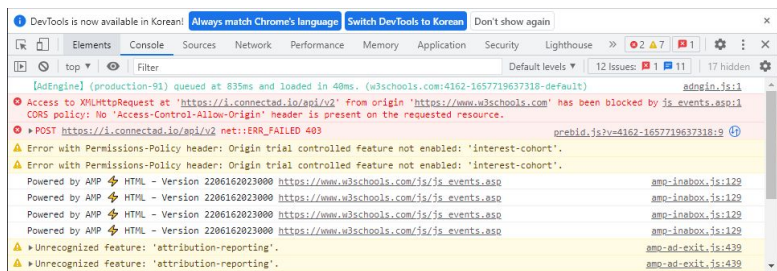
- SPA 단일 페이지 어플리케이션
 - 여러 페이지를 링크로 연결 하지 않기 때문에 불필요한 로드가 발생하지 않음.
 - Ajax 기술을 이용해서 서버와 **JSON** 문자열로 데이터를 주고 받는다.
 - 데이터가 변경 되는 부분만 바뀌기 때문에 모든 내용을 **Redirect** 할 필요가 없다.
 - Javascript 비동기 통신 관련 기술 기반.
- SPA Framework
 - React.js
 - Vue.js
 - Angular.js

Web Browser

- HTML 문서 내에서 **Body element**의 마지막 부분에 **script** 작성 권장.
- 개발자 도구로 콘솔 확인 가능.
- 브라우저마다 보여지는 결과가 다를 수 있다.
- 크로스 브라우징 체크 필수.

개발자 도구

- 크롬 브라우저에서 단축키 **F12** 또는 마우스 오른쪽 클릭 > 검사
- Elements 탭 : 태그 엘리먼트 구조 파악
- Console 탭 : log 출력
- Source : HTML 문서의 실제 소스
- Network : 서버에서 보내지는 데이터
- Application : Storage, Cache, Application 등의 데이터 저장 장치



<https://mainia.tistory.com/2393>

자바스크립트 이벤트

- inline 방식
 - 태그에 이벤트 속성을 이용해서 함수 실행문을 일일이 직접 작성
- internal 방식
 - `<script>` 태그 안에서 `document.getElementById()` 등의 메소드를 이용해서 DOM 선택.
 - DOM select를 이용해서 엘리먼트를 제어 한다.
- external 방식
 - 외부의 javascript 파일에 선언된 스크립트를 실행 파일에 불러와서 실행 한다.
- 이벤트 관련 예제
 - `console.log()` 확인
 - 배경색 변경
 - 내용 변경

DOM 엘리먼트 생성, 추가 예제

- DOM을 이용해서 문서의 엘리먼트를 생성, 추가, 삭제, 변경 가능.
 - `dom.onclick = function(event) { }`
 - `dom.createElement('태그')`
 - `dom.appendChild(newElement)`
- DOM 엘리먼트 생성, 추가 예제
 - Todo list
 - 요리 주문 목록 등.

문장과 블록

- 자바스크립트는 다른 언어에서 처럼 문장 뒤에 세미콜론(;)을 붙인다.
 - 세미콜론은 붙이지 않아도 상관 없다.
 - 어차피 압축파일로 만들어 질 때 세미콜론이 모두 제거 된다.
 - 요즘 언어인 **Kotlin, Swift, Python** 등도 세미콜론 사용이 자유롭다.
 - **naming convention** 협의 : 프로젝트 시작 전 네이밍 규칙 협의.
- 자바스크립트 실행은 위에서 밑으로 인터프리트 된다.
 - 실행 순서가 중요하다.
 - 객체가 로드 되기 전에 이벤트를 선언 하면 오류.
- 단문
 - 문장이 하나 - 중괄호 블록 표시 생략 가능.
- 복문
 - 두줄 이상 - 중괄호 블록 표시 생략 불가능.

식별자 명명 규칙 (Naming Convention)

- 의미 있는 이름을 사용한다.
- 예약어는 사용 할 수 없다.
- 대소문자 구분한다.
- 특수문자 포함 불가능
 - `_`, `$`는 특수 문자라도 사용 가능
- 첫 글자로 숫자 사용 불가능
- 낙타봉 표기법(Camel case) 권장
 - 뱀표기법(snake case)도 사용 가능 - 모두 소문자이거나 모두 대문자로 지어야 할 경우.
- 변수, 함수는 소문자로 시작
- 클래스는 대문자로 시작
- 상수는 모두 대문자 사용.

주석

- 코딩에 완성은 주석이다.
 - 소스코드도 언어이다.
 - 너무 장황하고 뻘한 주석은 오히려 가독성을 떨어지게 한다.
- 주석은 실행 되지 않는다.
- 미래의 자기 자신과 다른 개발 자를 위한 정보.
- 한 줄 주석
 - `//`
 - 편리하고 많이 사용 된다.
- 여러 줄 주석
 - `/* ... */`
 - 블록 주석
 - 중첩 안됨 주의

자바스크립트 자료 형(Data type)

- 원시 값(언어의 최고 로우레벨에서 직접 표현되는 불변 데이터)
 - Boolean 타입
 - Null 타입
 - Undefined 타입
 - Number 타입
 - BigInt 타입
 - String 타입
 - Symbol 타입
- Object 객체(속성의 컬렉션)
- 논리 식에서 `null`, `undefined`를 모두 `false`로 간주한다.
- `typeof`
 - 표현식의 타입을 확인 할 수 있다.

변수

- **Scope**
 - 변수가 유효한 범위
- **const**
 - 값이 변하지 않는 상수 선언.
 - **Scope**는 블록 단위
- **let**
 - 일반적인 변수 선언
 - **Scope**가 블록 단위이다.
 - 같은 블록에서 중복된 변수 선언 불가능.
- **var**
 - **Scope**가 함수 단위이다.
 - 같은 **Scope** 내에서 중복된 식별자 사용 가능.
 - **Hoisting**이 발생한다. (주의 해서 사용 할 것)

변수

- 자바스크립트는 변수 선언 시에 타입을 지정 하지 않는다. (dynamic type)
 - 변수에 대입 되는 값에 따라서 타입이 자동으로 바뀐다.
 - 반면 Typescript는 엄격한 타입.
 - 자유롭다 하더라도 엄격하게 사용하는 것이 바람 직 하다.
- 자바스크립트는 모든 데이터가 객체처럼 사용 된다.
 - 정수도 **number** 클래스가 존재 한다.
- 함수도 변수나 배열에 저장 가능하다.
 - 함수의 참조가 저장 되는 것.
- 데이터 처리를 위해 배열(**Array**)타입을 많이 사용.
- **Object** 및 **Class**
 - **Object**를 사용하고 있었는데 ES6부터 **class** 문법 지원.

연산자

- 연산자는 결합 방식과 우선 순위가 중요하다.
- 대입(할당)
 - =
 - 복합 대입 (연산 후 할당)
 - +=, -=, *=, /=
- 사칙연산
 - +, -, *, /, ++, --, %
- 비트 연산
 - |, &, ~, ^, <<
- 논리 연산
 - ||, &&, !
- 비교 연산
 - >, >=, <, <=

삼항 연산자(논리 연산)

- 조건 ? true일 때 : false일 때;
 - ?앞의 조건이 참이면 콜론(:)앞을 실행 하고 거짓이면 콜론 뒤를 실행 한다.
- Ternary 연산자
- 가난한 if~else문은 삼항 연산자로 사용 가능.
- 중첩하면 가독성이 떨어진다.

```
let isAdult = age > 19 ? "성인" : "미성년";
```

등호 (==, ===)

- 같은지를 비교하는 등호가 두 개이다.
- ==
 - 타입이 달라도 의미가 같으면 `true`
 - `"15" == 15`의 결과는 `true`
- ===
 - 타입과 의미가 모두 같아야 `true`
 - `"15" === 15`의 결과는 `false`
 - 엄격한 비교로 사용될 때 권장.

제어문

- JAVA나 C++에서 사용하는 제어문과 동일 하다.
- if
 - if ~ else
 - 다중 if ~ else
 - 중첩 if ~ else
- switch ~ case
- for
 - 중첩 for
- while
- do ~ while
- 보조 제어문
 - break
 - continue

제어문

- 어떤 기술을 사용 하든지 알고리즘의 기본은 제어문이다.
 - 최근 코딩 테스트에 언어 구분은 하지 않지만 알고리즘은 거의 동일하다.
- 제어문과 자바스크립트 배열 컬렉션은 밀접한 관계가 있다.
 - 제어문이 안 되면 배열을 제대로 사용하기 힘들다.
 - 배열 컬렉션에서 제공 되지 않는 기능은 직접 구현 해야 한다.
- 아무튼 제어문은 프로그래머에게 기본 소양이다.

난 수 발생기 (Random number)

- `Math.random()` 함수
 - 0보다 크고 1보다 작은 실수인 난 수 생성
 - ex) 0.76830903
- `Math.floor()` 함수
 - 소수점 이하 버림.
- `Math.random()`의 결과에 100을 곱하면 0~99사이의 난 수 발생.
 - `Math.floor(Math.random()*3)`의 결과는 0, 1, 2 중 하나.
 - `5 + Math.floor(Math.random()*3)`의 결과는 5, 6, 7 중 하나.

undefined, null, and, NaN

- undefined
 - 정의되지 않은 값 및 타입(아무런 객체로 참조하지 않음), `var name;`
- null
 - null 데이터, `var name=null;`
- Divide by zero
- Infinity 또는 -Infinity
- NaN (Not a Number)
 - 숫자가 아닌 것을 `parseInt()` 또는 `Number()`이 인자로 사용 할 때.

문자열에 저장된 숫자

- **string** 타입에서 + 연산자
 - Concatenation(연결)
- **number** 타입과 **string** 타입을 연산
 - 결과 의미가 달라 지므로 권장하지 않음
 - $10 + 5 + '10' == 1510$
 - $'10' + 5 + 10$ 의 결과는 다르다. 10510
- **parseInt()**
 - 문자열 numeric을 number 형으로 변화.

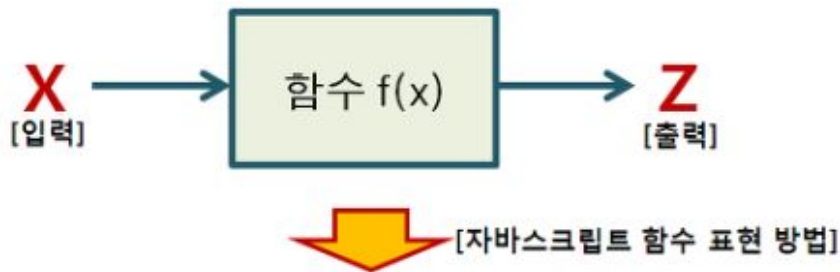
함수

- 자바스크립트의 함수와 수학에서의 함수의 다른 점
 - 자바스크립트 함수는 기능이란 의미가 더 강함.
 - 프로그래밍의 함수는 일의 묶음으로 기능의 재 활용이 가능 하다.
 - 변화가 필요한 경우 매개 변수를 이용해서 다른 값 사용.
- 자바스크립트의 함수는 기능이고 이름은 동사로 시작.
 - camel case 권장
- 프로그램 상에서 중복되는 코드 부분을 함수로 분리하여 구성.
 - 동일한 코드를 여러 번 작성 하는 것은 효율이 떨어진다.
 - 동일한 코드가 하나의 프로그램에 여러 번 반복 된다면 잠재적 문제를 일으키게 된다.
 - 유지보수도 어렵게 된다.

함수

- 함수 선언
 - function으로 시작
 - 매개변수와 return 사용 가능
 - 매개변수나 return 없이 기능만 구현 가능.
- 함수의 선언과 호출
 - 함수를 호출 해야 작동 한다.
 - 필요 할 때 여러 번 다시 호출 가능.
- 기본적인 구조

```
function 함수명(매개변수, 매개변수 ...) {  
    실행문;  
    return 반환값;  
}
```



```
function kwang(x)  -- ① 함수 이름  
{  
    b= x + 12;      -- ② 연산식 (스크립트)  
    return b;       -- ③ 출력 값  
};  
  
kwang(7);          -- ④ 함수 실행
```

[그림 1] 자바스크립트 함수

Call by Value

- arguments

- 자바스크립트 함수는 호출 될 때 **arguments** 객체가 자동 생성 된다.
 - ES6부터는 **arguments** 객체 대신 나머지 매개변수(...) 권장.
- 매개변수 : 함수를 호출 할 때 넘겨지는 **Parameter** 데이터를 저장 할 변수.
- 자바스크립트 함수의 매개 변수는 **arguments** 객체에 담겨서 전달된 데이터이다.

- 자바스크립트의 **arguments**는 **call by value** 방식이다.

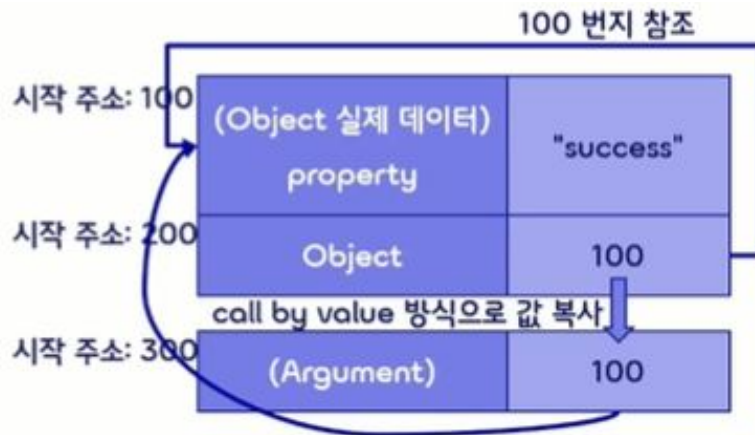
- 값에 의한 호출 (호출 할 때 넘겨 준 데이터가 매개변수로 복사 된다.)
- 함수 안에서 전달받은 **argument**를 변경 해도 복사된 값이므로 원본 데이터는 변하지 않는다.

- **Object**를 참조하는 변수의 값은 메모리상의 주소인 **pointer** 형

- 인스턴스를 참조하는 변수는 해당 인스턴스(객체)의 메모리상의 주소를 참조하는 값이다.
- 함수의 **argument**에서도 해당 객체의 위치를 참조 하고 있는 형태.
- 매개변수에 새로운 데이터를 저장 하는 것은 값을 바꾸는 것이므로 원본은 변함이 없다.
- 그러나 매개 변수가 참조하는 객체를 변경 하면 원본의 결과도 바뀐다.
- 이 원리는 자바 언어에서도 동일 하다.

Object Argument 전달

- 전달 받은 argument로 object의 property 변경 가능.
 - Call by value 방식으로 Object의 reference 복사해서 매개 변수로 전달.
 - 매개 변수도 같은 Object를 참조 하고 있기 때문에 함수 내부에서 원본 데이터 변경 가능.



Closure

- 함수가 호출될 때 **lexical**(어휘) **environment**(환경)라고 불리는 상황 (**context**) 정보를 생성하고 메모리에 유지.
- 다른 상황과 분리하여 닫혀 있는 상태 제공.
- 쉽게 생각해서 전달 된 매개 변수 값이 유지 되는 새로운 함수를 생성 해서 리턴 해 주는 형태.
- 리턴 받은 함수를 실행 하면 처음 생성 될 때 매개변수로 전달 한 값이 그대로 유지 된다.

```
1 "use strict"
2
3 const counter = closureCounter()
4 const counter100 = closureCounter(100)
5
6 function closureCounter(initValue = 0) {
7   let count = initValue
8   return function addCount() {
9     return ++count
10  }
11 }
12
13 console.log(counter()) // 1
14 console.log(counter100()) // 101
15
16 console.log(counter()) // 2
17 console.log(counter100()) // 102
```

콜백 함수 (Call Back)

- 호출 된 함수의 기능이 모두 끝난 후 실행 할 함수를 인자로 넘겨 주는 것.
- 코드의 흐름에 따라 프로그래머가 함수 호출 시점을 지정.
- **Call(호출) back(뒤로)**
 - 호출 한 함수가 완료 되면 인자로 전달 받은 함수를 꺼꾸로 호출 해 준다.
 - 예를 들어 외부 페이지의 엘리먼트를 로드 할 경우 로드가 끝나기 전에 이벤트 핸들러가 선언 된다면 제대로 실행 되지 않는다.
 - 이때 로드가 완전히 완료 되면 이벤트 핸들러가 실행 되도록 콜백 함수로 구현 할 수 있다.
- **setTimeout()**이나 **setInterval()**에 사용 되는 함수도 콜백 함수이다.
 - 지정한 시간이 모두 지나면 실행 되는 함수를 인자로 사용 한다.
 - 취소는 **clearTimeout()**, **clearInterval()** 함수를 사용 한다.

콜백 함수 (Call Back)

- 콜백 함수는 적당히 사용하면 유용하지만 남발 하면 콜백 헬에 빠진다.
 - 콜백 헬(지옥): 콜백 함수가 과다하게 중첩 되어 가독 성이 떨어 지는 상태.
- 콜백 헬 문제 해결 방법
 - 흐름 제어
 - Promise
 - Async
- Promise
- Async

재귀 호출 (Recursive call)

- 함수가 자기 자신을 호출하는 방식으로 동작
- 모든 재귀 호출 구현은 **while** 반복문과 **stack** 구조로 구현 가능 함.

Event

- 이벤트 핸들러
- 자바스크립트에서 자주 사용되는 이벤트
 - click
 - mousedown
 - mouseup
 - mousemove
 - keydown
 - keyup
 - wheel
 - load
- 사용 예
 - `document.onload = function(event) { }`
 - `button.onclick = function() { }`
 - `button.addEventListener('click', function() {})`

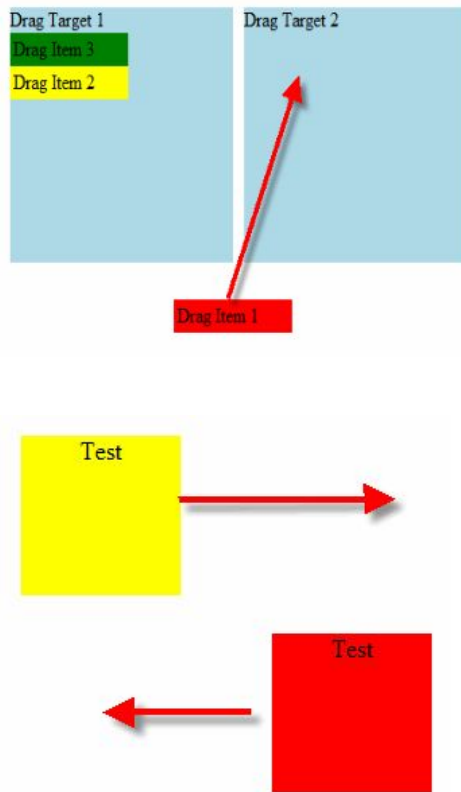
형식	설명
<code>button.onclick = function() {}</code>	이벤트 다중 추가 불가능.
<code>button.addEventListener('click', function() {})</code>	이벤트 다중 추가 가능.

과정 실습

자바스크립트 실습

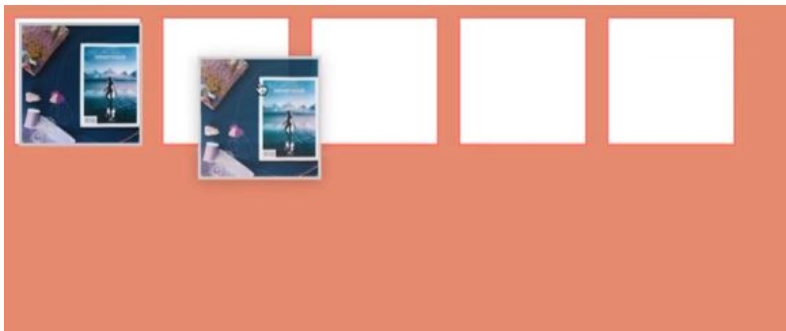
- 드래그 앤 드롭 기능 구현 (1)

```
function DragMove(o,e){  
    if (oDragItem==null) return;  
  
    if(!e) var e = window.event;  
    var x = e.clientX + document.body.scrollLeft - document.body.clientLeft - iClickOffsetX;  
    var y = e.clientY + document.body.scrollTop - document.body.clientTop - iClickOffsetY;  
  
    with(oDragItem.style){  
        zIndex = 1000;  
        position="absolute";  
        left=x;  
        top=y;  
    }  
}
```



자바스크립트 실습

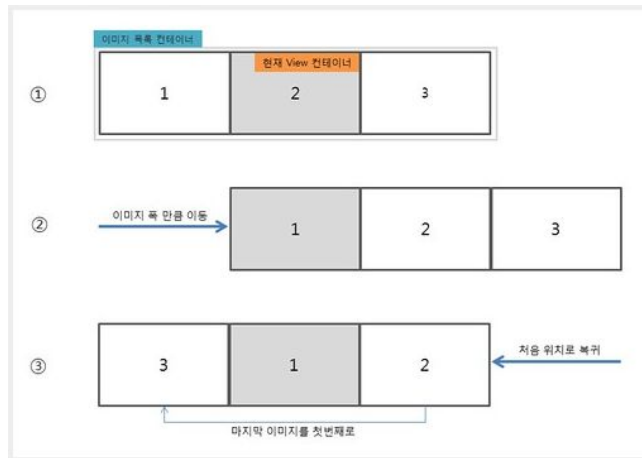
- 드래그 앤 드롭 기능 구현 (2)



```
1 const fill = document.querySelector('.fill');
2 const empties = document.querySelectorAll('.empty');
3
4 // Fill Listeners
5 fill.addEventListener('dragstart', dragStart);
6 fill.addEventListener('dragend', dragEnd);
7
8 // Loop through empties and call drag events
9 for(const empty of empties) {
10   empty.addEventListener('dragover', dragOver);
11   empty.addEventListener('dragenter', dragEnter);
12   empty.addEventListener('dragleave', dragLeave);
13   empty.addEventListener('dragover', dragOver);
14 }
15
16 // Drag Functions
17 function dragStart() {
18   this.className += ' hold';
19   setTimeout(() => (this.className = 'invisible'), 0);
20 }
21
22 function dragEnd() {
23   this.className = 'fill';
24 }
```

자바스크립트 실습

- 카로셀 이미지 슬라이더 구현



```
function swapNodes(){  
  
    // Element 노드 선택  
    firstChild = setElementNodeNext(list.firstChild);  
    lastChild = setElementNodePre(list.lastChild);  
  
    // 마지막 목록을 처음으로 이동하여 목록을 순환시킴  
    list.insertBefore(lastChild, firstChild);  
  
    // 첫째 element node 선택  
    firstChild = setElementNodeNext(list.firstChild);  
  
    // css 슬라이더 위치 초기화(CSS transition 중단)  
    list.className = "";  
  
    // transitionend 이벤트 리스너 초기화  
    list.removeEventListener("transitionend", swapNodes);  
  
}
```

Javascript 2

Object

- 논리적으로 연관 있는 것들의 묶음.
 - 객체는 중괄호 `{}` 사용하여 정의
- Property 접근
 - 마침표(`.`) 사용
 - 대괄호 및 문자열 인덱스 사용
 - `saram['name'] == saram.name`
- Function도 가질 수 있음

```
let saram = {  
  name : '홍길동',  
  greeting : function() {  
    return "hello " + this.name;  
  }  
}
```


prototype

- 원형, 기초 형태, 원래의 초기 모델
 - rapid prototyping - 요구 사항을 대략 먼저 만들어 보는 것.
- 다른 객체로부터 상속 가능하게 한다.
- 자바스크립트 **function**은 **prototype**이라는 **property**가 기본 포함 되어 있다.
 - **constructor** : **function**의 생성자 함수를 참조한다.
 - **__proto__** : 함수의 **prototype**
 - 크롬 개발자 도구에서 **console.dir(object)**으로 확인 가능.
- **prototype chain**
 - 상속 관계에 따라서 **prototype**이 확장 됨.

class

- ES6부터 지원 됨.
- 이전엔 **function**를 이용해서 클래스 생성자 함수 선언
- **class**는 **Object**를 위한 템플릿이라고 할 수 있다.(관련 항목을 묶는 것)
- **new** 연산자를 이용해서 객체를 생성한다.
 - `const saram = new Saram("홍길동", "대졸")`
- **constructor()** 생성자 함수가 별도로 정해져 있다.
- **Class**로 일관되게 비슷한 형태의 **instance**가 생성 됨.
 - 생성된 **instance**들은 각각 다른 **property**값을 가진다.
- **static**
 - 클래스 멤버
 - 클래스 명으로 바로 접근

Getters And Setters

- **getter** 메소드 : 프로퍼티를 읽기 위한 용도.
 - `get` keyword로 지정
- **setter** 메소드 : 프로퍼티에 값을 쓰기 위한 용도.
 - `set` keyword로 정의
 - 일반적으로 잘못된 값이 들어가지 않도록 **validation** 과정 추가.
- 예시

```
class Saram {  
    constructor(name, age) {  
        this.name = name; this.age = age  
    }  
    set name(name) {  
        this.name = name  
    }  
    get name() {  
        return "name is " + this.name  
    }  
}  
new Saram("홍길동", 25).name
```

상속

- Saram에서 상속 받아서
 - Student 클래스 생성
- Super
 - Base class인 Saram
- getInfo() method
 - method Overriding

배열 (Array) 컬렉션

- 리스트 기능을 가진 배열. 대괄호 [] 사용.
- 가능한 같은 타입의 요소들로 배열 구성하는 것을 권장 함.
 - Javascript 타입은 dynamic이어서 다른 타입 요소로 배열 구성이 가능 함.
- length 프로퍼티
 - 배열의 길이
- 속성도 배열기호 []를 사용해서 접근 가능
 - `window['name'] == window.name == this.name`

Array Methods

- 자바스크립트 배열 컬렉션
 - 기본적으로 Array에 포함된 다양한 method를 제공한다.
 - forEach - 기본 iteration
 - map - 요소에 새 매핑
 - filter - 검색 후 조건에 맞는 값으로 새 배열 생성
 - reduce - 배열을 돌면서 한 값을 계산 (총합, 평균 등)
 - push / pop - 맨뒤
 - splice - 위치, 삭제, 추가
 - slice - 잘라서 새 배열 생성
 - findIndex - 위치 찾기
 - every - 모든 요소가 조건 만족을 하는지
 - some - 조건 만족하는 요소가 있는지

ES6 Arrow Function

- Arrow function
 - 기존 `function` 키워드 보다 간결하게 함수 표현 가능.
 - 불필요한 보일러 플레이트가 없어서 사용하기 편리하다.
 - Arrow function은 익명 함수이므로 **Callback** 함수로 많이 사용 됨.
- 배열 처리
 - 자바스크립트 배열은 컬렉션 라이브러리가 합쳐진 형태로 아주 많이 사용 된다.
 - Arrow function과 함께 사용하면 상당한 시너지 효과를 낼 수 있다.

ES6 Arrow Function

- 함수를 간결하게 작성 가능
- Anonymouse(익명) function
 - 이름을 붙이지 않은 함수
- Arrow(화살표) 사용
 - function 대신 => 사용
- 기본 문법
 - (매개변수) => { 실행 처리 문장 }
- 매개변수가 없는 경우
 - () => { 실행 처리 문장 }
- 매개변수가 한 개인 경우 괄호 생략 가능
 - 매개변수 => { 실행 처리 문장 }

ES6 Arrow Function

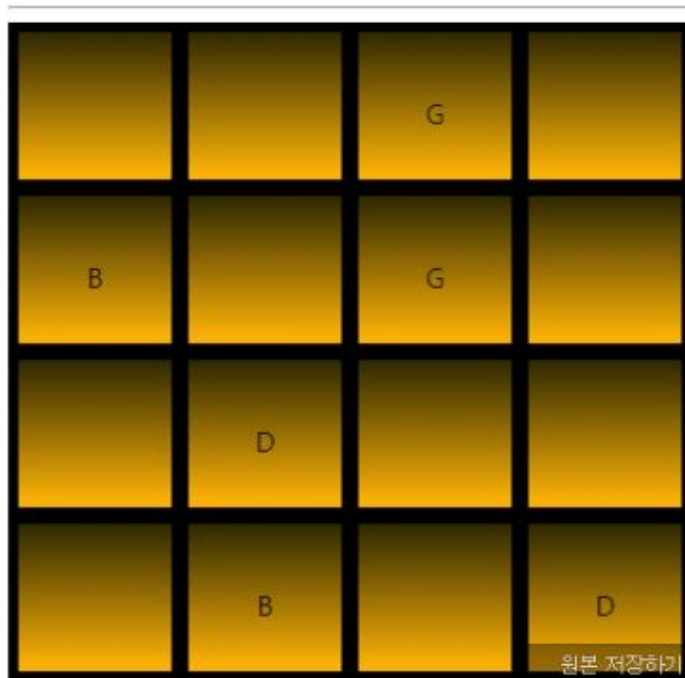
- 실행 처리문이 **return** 한 줄 뿐인 경우는 중괄호 블록 생략 가능.
 - (매개변수, 매개변수) => 반환 값(또는 식)
 - (a, b) => a+b
- 객체를 반환 할 때는 괄호가 필요하다.
 - (매개변수, 매개변수) => (객체)
- **setTimeout()**이나 **setInterval()** 함수에 사용 되는 콜백 함수도 **Arrow function**으로 처리 할 수 있다.

과정 실습

자바스크립트 실습

- 짝 맞추기 퍼즐 구현

Puzzle



자바스크립트 실습

- 햄버거 하우스 구현



자바스크립트 실습

⋮ 낱말 맞추기 퍼즐 게임 ⋮

B	R	O	T	H	E	R

문제 : 형제

brother

확인

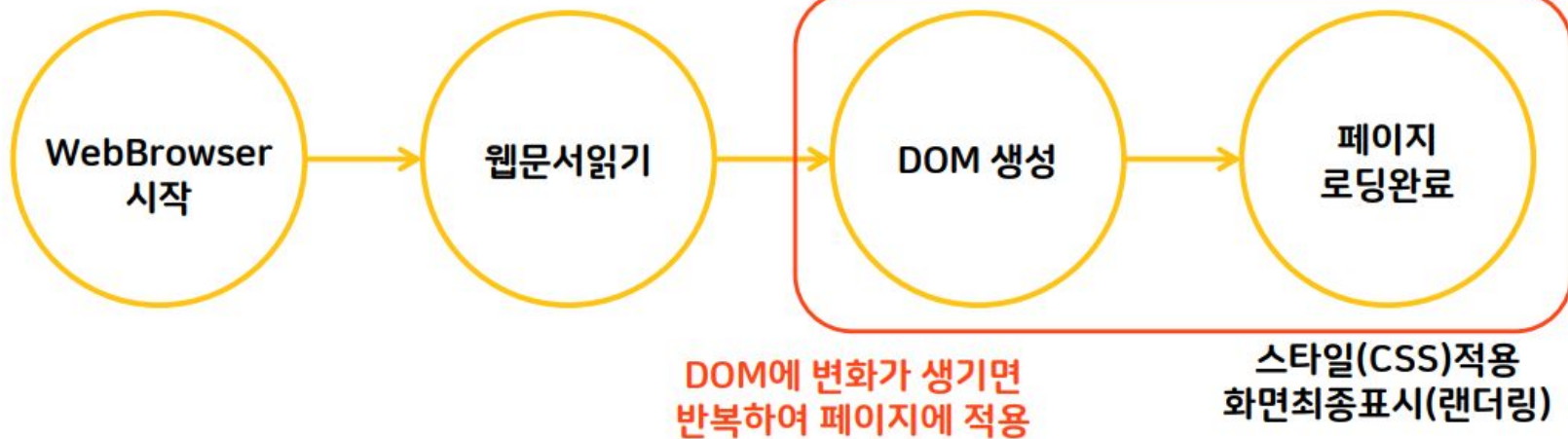
결과 : 땡땡!

Javascript API

Javascript 문서 실행 순서

사용자가 웹 페이지 방문

HTML을 모두 객체형태로 변환
-> Javascript로 접근가능



Javascript DOM 객체

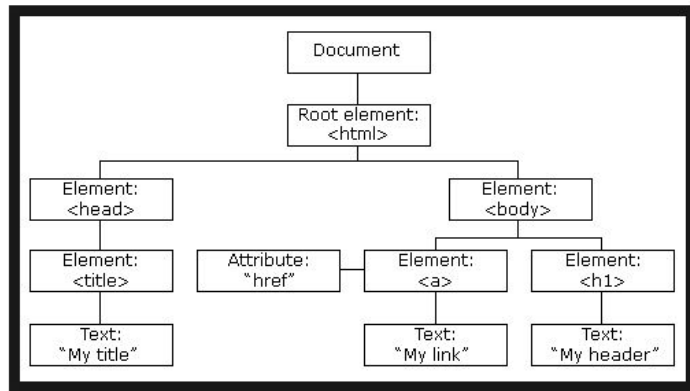
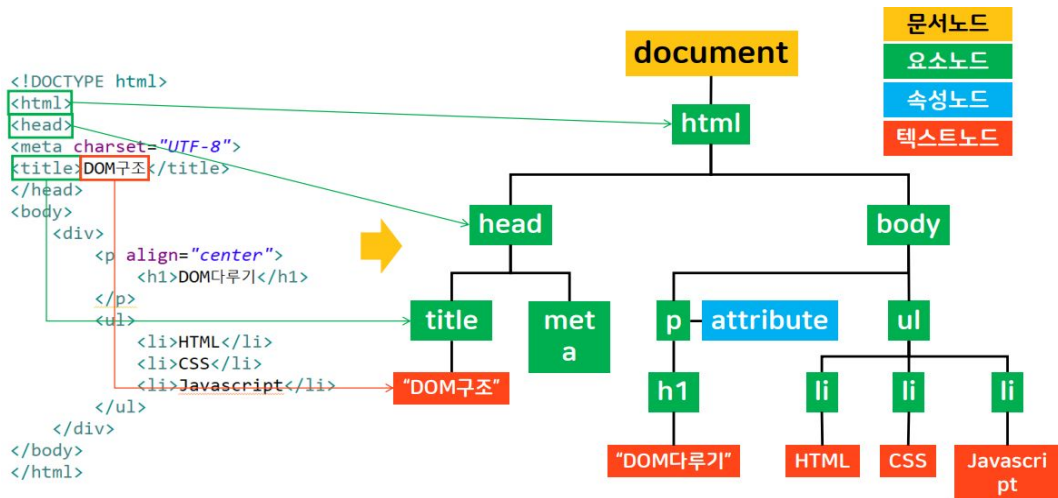
- 동적인 HTML 문서 구현
 - DOM selector - Event handler : 이벤트에 따라서 문서 요소의 속성 변경.
- DOM객체의 멤버 속성 확인 : `console.dir(DOM객체)`
 - 이벤트 속성, `style`, `title` 등을 확인 가능.
 - HTML 요소를 DOM 객체 형태로 참조해서 이벤트 핸들러를 추가한다.
 - 이벤트가 발생하면 특정 연산 하거나 요소를 변경, 추가, 제거 할 수 있다.

DOM(Document Object Model)

- HTML 문서
 - 단순 텍스트 파일이다.
 - 자바스크립트 프로그램 안에서 제어 하기 위해서는 DOM 객체로 변환 해야 함.
 - HTML 문서 내의 **text**를 읽어서 **DOM tree** 형태로 구성 한다.
 - 처리된 객체는 다시 파일로 렌더링 해 준다.
- Tree 구조
 - 최상위 **root**로 부터 분화 되어 내려 가는 형태. (최상위 객체는 **window**)
 - 실제로는 부모 요소 안에 자식 요소가 들어가는 형태.
 - 연결 된 **Node**들 간에 부모(**parent**) - 자식(**child**) 관계 형성. 형제(**sibling**)
- DOM
 - **element**
 - **style**
 - **content**
 - **event**

DOM(Document Object Model) Tree

- DOM은 계층적인 Tree 구조이다



The HTML DOM Tree of Objects <출처 : w3schools>

Data

- `localStorage.setItem("name", "John Doe");`
- `localStorage.getItem("name");`
- 저장소 개체 속성 및 메소드

Property/Method	Description
<u><code>key(n)</code></u>	Returns the name of the <i>n</i> th key in the storage
<u><code>length</code></u>	Returns the number of data items stored in the Storage object
<u><code>getItem(keyname)</code></u>	Returns the value of the specified key name
<u><code>setItem(keyname, value)</code></u>	Adds that key to the storage, or update that key's value if it already exists
<u><code>removeItem(keyname)</code></u>	Removes that key from the storage
<u><code>clear()</code></u>	Empty all key out of the storage

- Web Strage API 관련 메소드

Property	Description
<u><code>window.localStorage</code></u>	Allows to save key/value pairs in a web browser. Stores the data with no expiration date
<u><code>window.sessionStorage</code></u>	Allows to save key/value pairs in a web browser. Stores the data for one session

JSON 데이터 형식 개념

CSV

- 각 항목을 쉼표로 구분해 데이터를 표현
- 파일 용량이 작아 많은 양의 데이터 전송 시 유용하나, 가독성이 떨어짐

예

홍길동, hjd@test.com, 충남 천안시

XML

- HTML처럼 태그를 이용하여 데이터를 표현
- 가독성이 좋으나, 정보를 나타내는 태그 때문에 파일의 용량이 커짐

예

```
<name>홍길동</name>
<email>hjd@test.com</email>
<address>충남 아산시</address>
```

JSON

- **JavaScript에서 사용하는 객체 형태로 데이터를 표현**
- 가독성이 좋고, 파일 용량이 작은 편이어서 Ajax에서 주로 사용하는 데이터 표현 방식
- 객체, 배열, 문자열, 숫자, 불린, null만 사용 가능하며, 문자열은 큰따옴표를 사용해야 함
- 데이터 양이 많아질 경우 데이터의 추출 속도가 떨어짐

예

```
{
  name: "홍길동",
  email: "hjd@test.com",
  address: "충남 천안시"
}
```

JSON

- JSON 자체는 문자열
- 데이터는 이름/값 쌍.
- 데이터는 쉼표로 구분.
- 중괄호는 객체를 보유.
- 대괄호는 배열을 포함.

```
const myObj = {name: "John", age: 31, city: "New York"};
const myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

```
const myJSON = '{"name":"John", "age":31, "city":"New York"}';
const myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

Effect

- CSS transition 활용.
- CSS animate 활용.
- setInterval 메서드
- 콜백 함수 활용

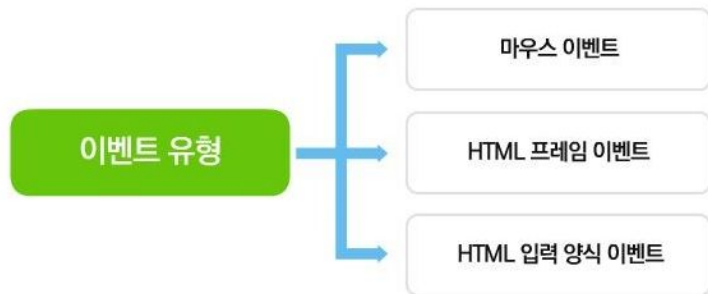
```
<p><button onclick="myMove()">Click Me</button></p>
```

```
<div id = "container">  
  <div id = "animate"></div>  
</div>
```

```
<script>  
function myMove() {  
  let id = null;  
  const elem = document.getElementById("animate");  
  let pos = 0;  
  clearInterval(id);  
  id = setInterval(frame, 5);  
  function frame() {  
    if (pos == 350) {  
      clearInterval(id);  
    } else {  
      pos++;  
      elem.style.top = pos + "px";  
      elem.style.left = pos + "px";  
    }  
  }  
}  
</script>
```

이벤트 처리 기법

- 이벤트 리스너 - 이벤트 처리기
- 이벤트 주도형 프로그래밍



Event	Handler	이벤트 발생 시기
focus	onFocus	입력양식을 선택해 포커스가 주어졌을 때
blur	onBlur	포커스가 폼의 입력 양식을 벗어났을 때
select	onSelect	입력 양식에서 한 필드를 선택했을 때
change	onChange	입력 양식에서 값이 바뀌었을 때 발생
load	onLoad	해당 페이지가 로딩 되었을 때(처음 읽힐 때)
unload	onUnload	해당 페이지를 빠져 나갈 때
mousemove	onMouseMove	해당 영역에서 마우스 포인터가 움직일 때
mouseover	onMouseover	해당 영역에 마우스 포인터가 올라갈 때
mouseout	onMouseout	해당 영역에서 마우스 포인터가 빠져 나갈 때
mousedown	onMouseDown	해당 영역을 마우스로 클릭할 때
mouseup	onMouseup	해당 영역을 마우스로 클릭했다 땔 때
click	onClick	해당 영역을 마우스로 클릭할 때
keydown	onKeyDown	해당 영역에서 키보드를 눌렀을 때
keyup	onKeyUp	해당 영역에서 키보드를 눌렀다 땔 때
keypress	onKeyPress	해당 영역에서 키보드를 계속 누르고 있을 때
submit	onSubmit	입력 양식의 내용을 전송 할 때
reset	onReset	입력 양식의 내용을 초기화 할 때

addEventListener 메서드

- `target.addEventListener(type, listener, useCaptuer);`
 - `target` : 이벤트 리스너를 등록 할 DOM 노드
 - `type` : 이벤트 유형
 - `listener` : 이벤트 발생 시 처리 할 콜백 함수.
 - `useCapture` : 이벤트 단계, 캡처링/버블링(default) 단계 설정.
- 장점
 - 같은 요소의 같은 이벤트에 여러 이벤트 리스너 추가 가능.
 - 버블링 캡처링 설정 가능.
 - `removeEventListener`, `stopPropagation`, `preventDefault` 등의 메서드로 이벤트 전파를 정밀하게 제어 가능.
 - HTML 요소를 포함한 모든 DOM 노드에 이벤트 리스너 등록 가능.

Event 객체

```
function (event) => {  
    event.target.style.backgroundColor = "orange";  
}
```

- 이벤트 핸들러에는 **event** 객체가 자동 생성 된다.
- **event** 객체를 통해서 이벤트의 **target** 요소에 관련된 정보 사용 가능.
- **event** 인자를 간단히 줄여서 **e**로 사용하는 것이 관례.
- 발생 이벤트의 유형에 따라서 **event** 객체의 속성이 다르다.

Event 객체 공통 속성

프로퍼티	값 타입	설명
type	문자열	이벤트 이름('click', 'mousemove' etc.)
target	요소 객체	이벤트가 발생한 요소
currentTarget	요소 객체	처리를 담당하는 이벤트 리스너가 등록된 요소 객체
eventPhase	정수	이벤트 전파 단계(1: 캡처링, 2: 타겟, 3: 버블링)
timeStamp	정수	이벤트 발생 시각(1970/1/1.00:00:00부터 경과한 밀리초)
bubbles	논리값	버블링 단계인지를 뜻하는 값
cancelable	논리값	preventDefault()로 기본 이벤트를 취소할 수 있는지 여부 표현
defaultPrevented	논리값	preventDefault()로 기본 작업이 취소되었는지를 나타냄
isTrusted	논리값	해당 이벤트가 사용자의 액션에 의해 생성되었는지를 뜻함

Mouse Event 객체

프로퍼티	값 타입	설명
screenX, screenY	정수	클릭한 위치의 화면 좌표(컴퓨터 화면의 좌상단 꼭짓점이 원점)
clientX, clientY	정수	클릭한 위치의 윈도우 좌표(표시 영역의 좌상단 꼭짓점이 원점)
pageX, pageY	정수	클릭한 위치의 문서 좌표(문서의 왼쪽 위 꼭짓점이 원점)
offsetX, offsetY	정수	이벤트가 발생한 요소의 상대 좌표(요소의 왼쪽 위 꼭짓점이 원점)
altKey	논리값	Alt가 눌렸는지를 뜻함
ctrlKey	논리값	Ctrl이 눌렸는지를 뜻함
shiftKey	논리값	Shift가 눌렸는지를 뜻함
detail	정수	이벤트의 자세한 정보: 마우스 이벤트의 경우 클릭 횟수
button	정수	눌린 마우스의 버튼(0: 왼쪽 버튼, 2: 휠 버튼, 3: 오른쪽 버튼)
relatedTarget	객체	mouseover 이벤트에서는 마우스가 떠난 노드 mouseout 이벤트에서는 마우스가 들어온 노드

Keyboard Event 객체

프로퍼티	값 타입	설명
altKey	논리값	Alt가 눌렸는지를 뜻하는 논리값
ctrlKey	논리값	Ctrl이 눌렸는지를 뜻하는 논리값
shiftKey	논리값	Shift가 눌렸는지를 뜻하는 논리값
metaKey	논리값	Meta 키가 눌렸는지를 뜻하는 논리값 맥은 Command, 윈도우는 window 키
key	문자열	눌린 키의 DOMString
keyCode	정수	눌린 키의 키 코드
code	문자열	눌린 키가 키보드에서 차지하는 물리적 위치를 뜻하는 문자열

Event 단계

1. 캡처링 단계

- ♪ 이벤트가 Window 객체에서 출발해서 DOM 트리를 타고 이벤트 타깃까지 전파
- ♪ 이 단계에 반응하도록 등록된 이벤트 리스너는 이벤트가 발생한 요소에 등록된 이벤트 처리기나 이벤트 리스너보다 먼저 실행됨
- ♪ addEventListener 메서드의 useCapture 매개변수에 true를 인자로 전달하면 캡처링 단계에 이벤트 리스너 실행
- ♪ 이벤트 타깃이 이벤트를 수신하기 전에 이벤트를 빼돌리는(캡처하는) 단계라는 뜻에서 캡처링 단계라고 함

※ *Event Target* 이벤트 타깃

이벤트가 발생한 요소를 이벤트 타깃이라고 함

이벤트를 발생시키는 것을 이벤트를 발사(*Fire*)한다고 표현하기도 함

3. 버블링 단계

- ♪ 이벤트가 이벤트 타깃에서 출발해서 DOM 트리를 타고 Window 객체까지 전파
- ♪ 거품이 올라오는 것처럼 이벤트가 DOM 트리 아래에서부터 위로 올라온다는 뜻에서 버블링 단계라고 함
- ♪ useCapture를 false로 설정하면 버블링 단계에서 이벤트 리스너 실행
- ♪ 같은 요소의 캡처링 단계와 버블링 단계에 각각 이벤트 리스너 등록 가능
- ♪ 같은 요소의 같은 이벤트, 같은 단계에 반응하게 등록된 이벤트 처리기와 리스너가 있다면 처리기가 먼저 실행되고 리스너가 등록된 순서에 따라 순서대로 실행됨

※ *focus, blur* 이벤트는 해당 요소에만 필요한 이벤트이므로 버블링 발생 X

2. 타깃 단계

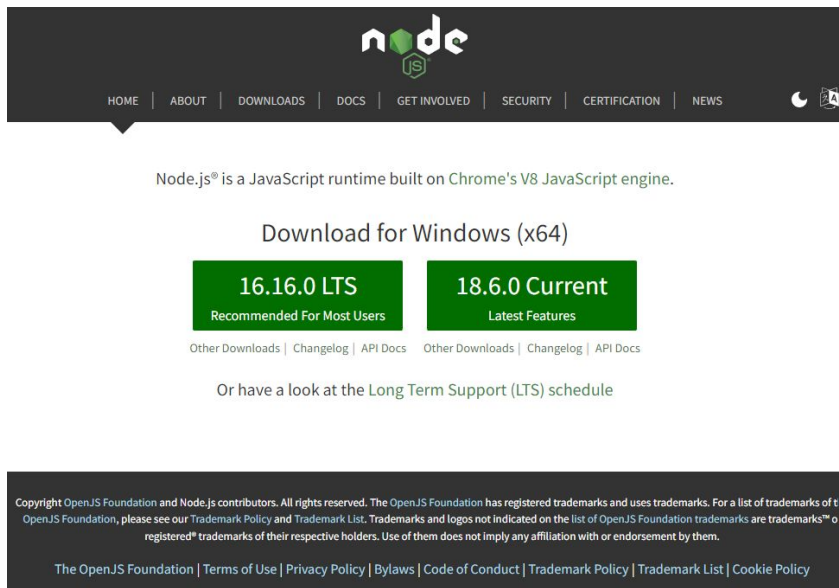
- ♪ 이벤트가 이벤트 타깃에 전파되는 단계
- ♪ 이벤트 타깃에 등록된 이벤트 처리기나 이벤트 리스너는 이 시점에 실행
- ♪ 이벤트 처리기의 경우 타깃 단계와 버블링 단계에서만 실행됨

이벤트 리스너 함수 활용

- 이벤트 전파 취소
 - `event.stopPropagation();`
- 이벤트 전파 일시 정지
 - `event.stopImmediatePropagation();`
- 기본 동작 취소
 - `event.preventDefault();`
- `bind` 메서드를 사용하면 이벤트 핸들러의 **this**를 지정 가능하다.
- `Person.prototype.handleEvent` - 이벤트 핸들러 함수 대신 사용 가능한 객체.
- ES6의 화살표 함수는 객체가 초기화 되는 시점에 **this**가 결정 됨.
- 함수를 반환 하는 함수를 이벤트 리스너로 등록. - 코드의 가독성이 좋아 짐.

Node.js express 서버

- Node.js는 구축 플랫폼이다.
- [크롬의 자바 런타임](#) 빠르고 쉽게 확장 가능한 네트워크 응용 프로그램을 구축.
- Node.js는 분산 장치에서 실행하는 데이터 집약적 인 실시간 애플리케이션에
- 가볍고 효율적으로 완벽하게 이벤트 중심, 논 블로킹 (non-blocking) I / O 모델을 사용합니다.
- express 프레임워크



node

HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for Windows (x64)

16.16.0 LTS
Recommended For Most Users

18.6.0 Current
Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#) [Other Downloads](#) | [Changelog](#) | [API Docs](#)

[Or have a look at the Long Term Support \(LTS\) schedule](#)

Copyright OpenJS Foundation and Node.js contributors. All rights reserved. The OpenJS Foundation has registered trademarks and uses trademarks. For a list of trademarks of the OpenJS Foundation, please see our [Trademark Policy](#) and [Trademark List](#). Trademarks and logos not indicated on the list of OpenJS Foundation trademarks are trademarks™ or registered® trademarks of their respective holders. Use of them does not imply any affiliation with or endorsement by them.

[The OpenJS Foundation](#) | [Terms of Use](#) | [Privacy Policy](#) | [Bylaws](#) | [Code of Conduct](#) | [Trademark Policy](#) | [Trademark List](#) | [Cookie Policy](#)

Node.js express 프레임워크

- <http://expressjs.com/en/guide/routing.html>

```
const http = require('http');
const express = require('express');
const app = express();
const cors = require('cors');

// public 폴더 static 설정.
app.use(express.static('public'));

app.get("/", (req, res)=> {
  res.end("<h1>Hello Nodejs world</h1>");
});
```

```
const server = http.createServer(app);
server.listen(3000, ()=>{
  console.log("run on Server : http://localhost:3000");
});
```


Javascript Ajax

1. XMLHttpRequest 객체 생성
2. 콜백 함수 정의
3. XMLHttpRequest 객체 열기
4. 서버에 요청 보내기

```
// Create an XMLHttpRequest object
const xhttp = new XMLHttpRequest();

// Define a callback function
xhttp.onload = function() {
    // Here you can use the Data
}

// Send a request
xhttp.open("GET", "ajax_info.txt");
xhttp.send();
```

파일 업로드 미리 보기 (서버 사이드)

- https://www.w3schools.com/nodejs/nodejs_uploadfiles.asp

- Node.js 서버에 처리 되는 소스 코드

```
app.post('/fileupload', (req, res) => {  
    var form = new formidable.IncomingForm();  
    form.parse(req, async function (err, fields, files) {  
        //console.log('1 >>> PARSE!', fields);  
        //console.log(">>> filetoupload.length : ", files.filetoupload.length);  
        for(var i=0; i<files.filetoupload.length; i++) {  
            var oldpath = files.filetoupload[i].filepath;  
            var newpath = 'C:/Users/User/upload/' +  
files.filetoupload[i].originalFilename;  
            await fs.rename(oldpath, newpath, function (err) {  
                if (err) throw err;  
                res.write('File uploaded and moved!');  
                res.end();  
            });  
        }  
    });  
});
```

파일 업로드 미리보기 (클라이언트 사이드)

```
var fileInput = document.getElementById("imageSelector");
function readImage(input) {
  if(input.files && input.files[0] ) {
    const reader = new FileReader();

    reader.onload = (e)=>{
      let readDataURL = e.target.result;
      const previewImageDiv = document.getElementById('preview-image');
      previewImageDiv.innerHTML = ``;
    }

    reader.readAsDataURL(input.files[0]);
  }
}
fileInput.onChange = (e) => {
  readImage(e.currentTarget);
};
```

업로드 Ajax 버튼 이벤트핸들러 (클라이언트 -> 서버)

- FormData 객체 사용

```
const formElement = document.querySelector("form");
const formData = new FormData(formElement);
const request = new XMLHttpRequest();
request.open("POST", "submitform.php");
formData.append("serialnumber", serialNumber++);
request.send(formData);
```

https://developer.mozilla.org/en-US/docs/Web/API/FormData/Using_FormData_Objects

jQuery Ajax 이용 방식 예

```
e.preventDefault();
var form = $('#fileUploadForm')[0];
// Create an FormData object
var data = new FormData(form);

$.ajax({
  type: "POST",
  enctype: 'multipart/form-data',
  url: "emp/upload",
  data: data,
  processData: false,
  contentType: false,
  cache: false,
  timeout: 600000,
  success: function (data, status, xhr) {
    // 업로드 성공 후 화면 처리
  },
  error: function (e) {
    console.log("ERROR : ", e);
    $("#readBtn").prop("disabled", false);
    alert("fail");
  }
});
```

E-Mail 전송

- https://www.w3schools.com/nodejs/nodejs_email.asp

```
var nodemailer = require('nodemailer');

var transporter = nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: 'youremail@gmail.com',
    pass: 'yourpassword'
  }
});
```

```
var mailOptions = {
  from: 'youremail@gmail.com',
  to: 'myfriend@yahoo.com',
  subject: 'Sending Email using Node.js',
  text: 'That was easy!'
};

transporter.sendMail(mailOptions, function(error, info){
  if (error) {
    console.log(error);
  } else {
    console.log('Email sent: ' + info.response);
  }
});
```

Canvas

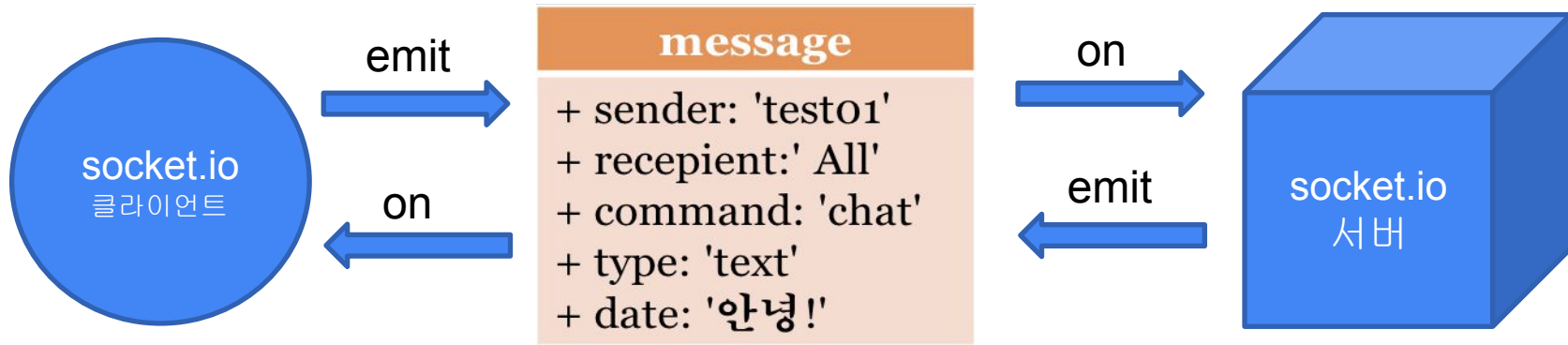
- https://www.w3schools.com/graphics/canvas_intro.asp
- <https://www.javascripttutorial.net/web-apis/javascript-canvas/>
- https://developer.mozilla.org/ko/docs/Web/API/Canvas_API/Tutorial

```
var ctx = document.getElementById("myCanvas").getContext("2d");  
console.log(ctx);
```

```
var x = 0;  
var interval = setInterval(function(){  
    x += 3;  
    if(x>=500) {  
        clearInterval(interval);  
    }  
    ctx.fillStyle = "#ffffff";  
    ctx.fillRect(x-10, 50, 100, 80);  
    ctx.fillStyle = "#ff0000";  
    ctx.fillRect(x, 50, 100, 80);  
},30);
```

Socket.io

- message 객체 속성
- [Socket.IO](#)
- [Get started | Socket.IO](#)



Socket.io - Server

```
const http = require('http');
const express = require('express');
const app = express();
const cors = require('cors');
const socketio = require('socket.io');
const server = http.createServer(app);
server.listen(3000, ()=>{
  console.log("run on Server : http://localhost:3000");
});
```

```
console.log(socketio);
var io = socketio.listen(server);
```

```
io.sockets.on('connection', function(socket) {
  socket.emit('news', {hello:'world'});
  socket.on('my other event', function(data) {
    console.log(data);
  });
});
```


Socket.io - Client

```
<script src="http://localhost:3000/socket.io/socket.io.js"></script>
<script>
const socket = io.connect('http://localhost:3000');
socket.on('connect', function(data) {
    console.log("서버 소켓과 연결 됨!");

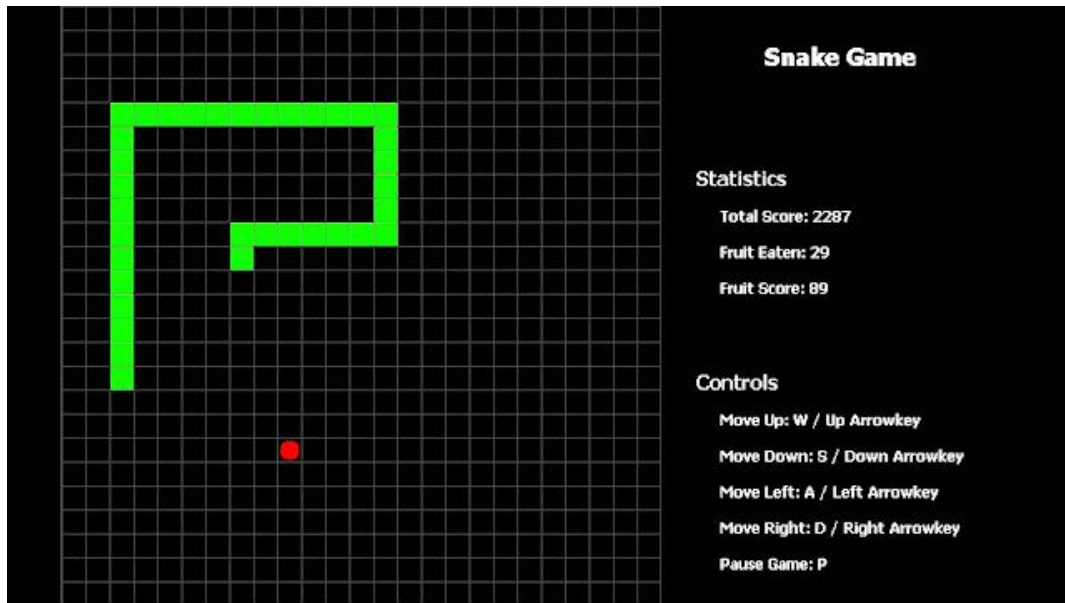
    socket.on('news', function(data){
        console.log("server data : " + data);

        socket.emit("client-message", "ok");
    });
});
</script>
```

과정 실습

자바스크립트 실습

- Snake 게임 구현



자바스크립트 실습

- Ajax 기능 구현 - 이메일 전송 및 목록 생성

Ajax 연습

이름
부서
직책
이메일
사진 선택된 파일 없음

새 사원 추가

선택 삭제

| 검색:

```

app.post('/fileupload', (req, res) => {
  var form = new formidable.IncomingForm();
  form.parse(req, function (err, fields, files) {
    console.log(">>>>> (1) ", fields);
  });

  form.on("end", function () {
    console.log(">>>>> (2) ");
    console.log("파일 갯수 : ", this.openedFiles.length);
    for(var i=0; i<this.openedFiles.length; i++) {
      let file = this.openedFiles[i];
      //console.dir(file);
      var oldpath = file.filepath;
      //var newpath = 'C:/Users/User/upload/' + file.originalFilename;
      var newpath = __dirname + '/public/upload/' + file.originalFilename;
      // 파일을 다른 드라이브로 이동 할 경우 rename 대신 copyFile 사용.
      fs.copyFile(oldpath, newpath, function (err) {
        if (err) throw err;
        res.writeHead(200, {"Content-Type":"text/html; charset=UTF-8"});
        res.write("<h2>upload file received!</h2>");
        res.end();
      });
      // Error: EXDEV: cross-device link not permitted
      // https://techoverflow.net/2023/03/14/how-to-fix-nodejs-error-exdev-cross-device-link-not-permitted-rename/
    }
  });
});

```

자바스크립트 실습

- 전자 칠판 구현

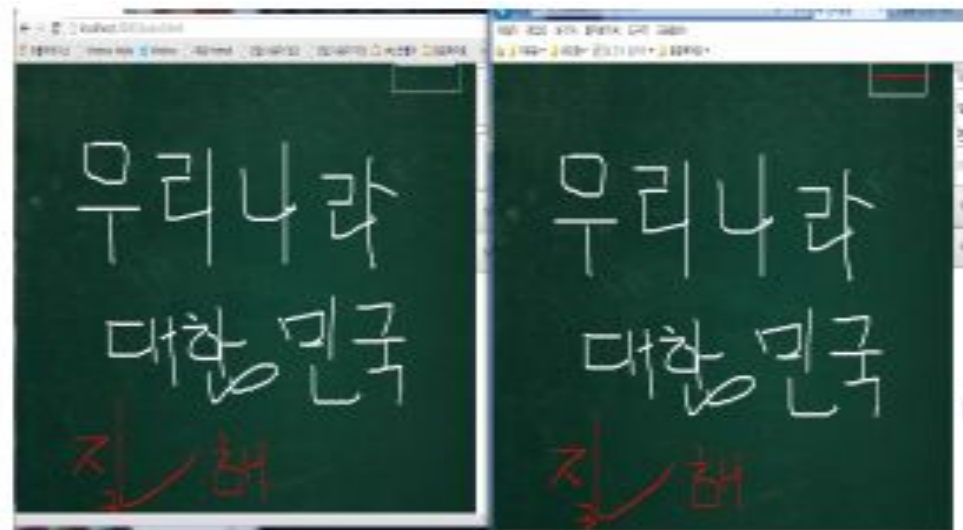
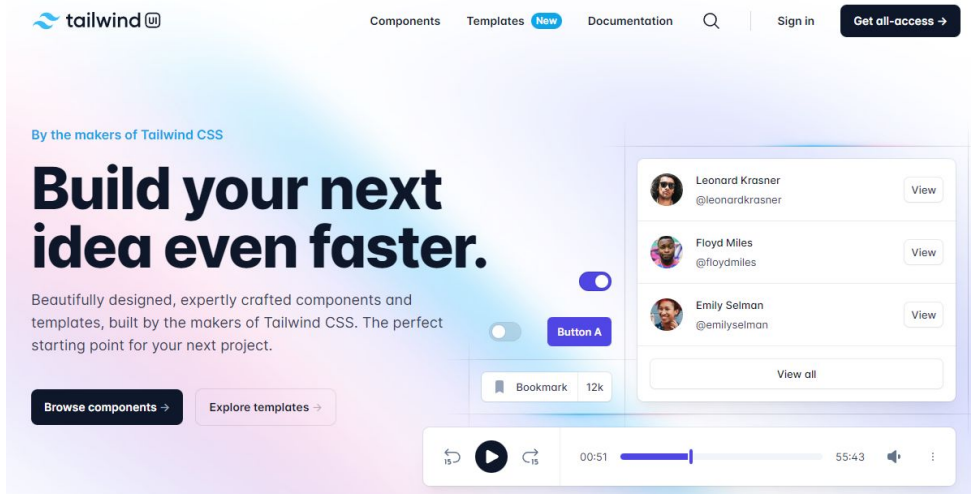


그림 5. 전자칠판
Fig. 5. Electronic blackboard

CSS 프레임 워크 - Tailwind

- 부트스트랩에 비해 자유도가 높다.
- 디자이너에게 적합.
- [Tailwind UI - Official Tailwind CSS Components & Templates](#)
- [Tailwind CSS + Next.js 적용.. : 네이버블로그 \(naver.com\)](#)
- [Tailwind CSS 사용법, 장점과 단점 \(tistory.com\)](#)



以衆小不勝 爲大勝也

이 중 소 불 승 위 대 승 야

여러 작은 패배가 있음으로써 큰 승리를 거둘
수 있다.

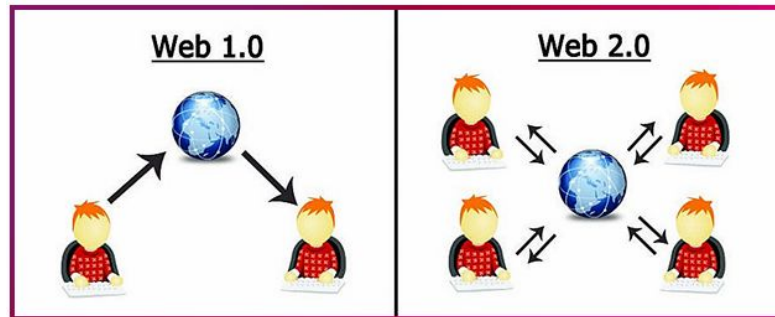
-《莊子장자》〈外篇외편 秋水추수〉-

jQuery 기초 문법

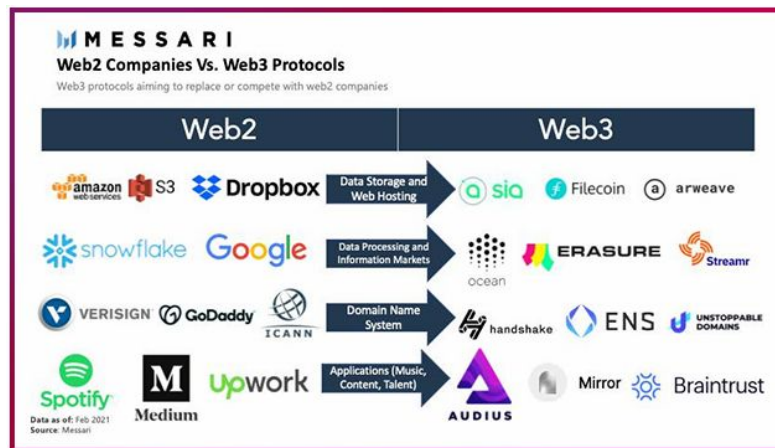
WEB 3.0



우리는 웹 2.0 시대에 살고 있습니다. 페이스북과 인스타그램, 유튜브와 틱톡, 넷플릭스와 같은 플랫폼 중심의 서비스 시대입니다. 아마존, 쿠팡과 같은 이커머스는 물론 모바일 뱅킹과 음악 스트리밍 서비스에 이르기까지 우리 일상생활 대부분을 차지하는 IT 기반 서비스는 웹 2.0 시대를 대표합니다.



웹 1.0과 2.0의 차이점



웹 2.0 기업과 웹 3.0 프로토콜 (출처: Messari)

웹의 전환기

웹 페이지와 사용자간의 원활한 소통



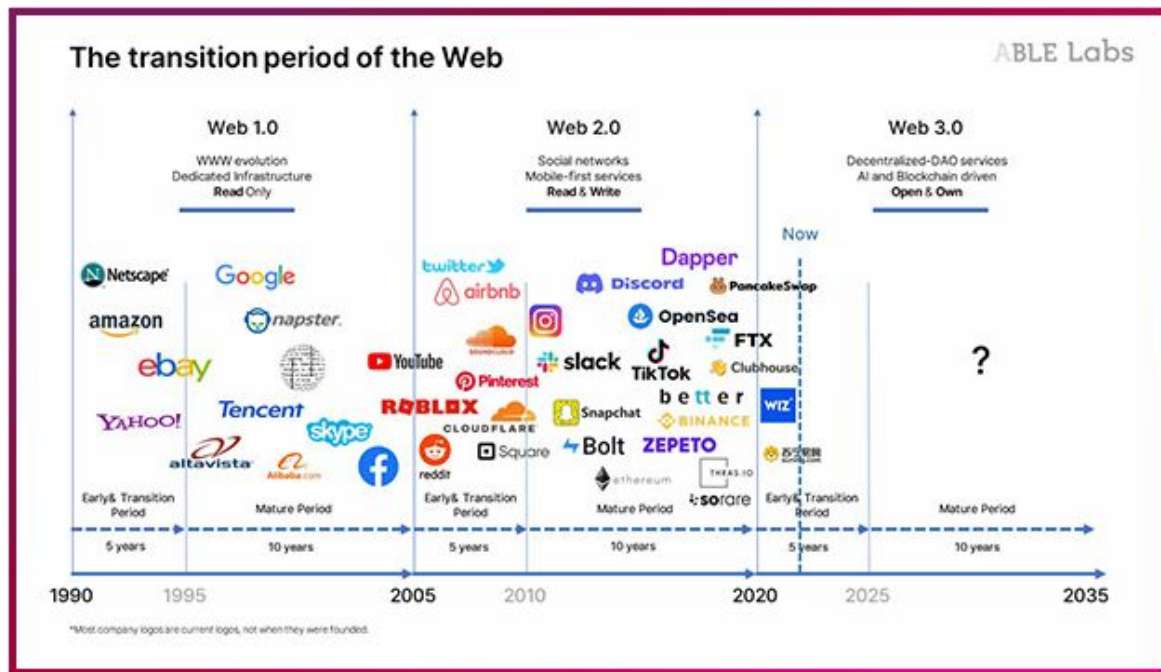
WEB 2.0, Ajax, RIA



새로운 UI, UX의 필요



많은 코드, 복잡한 코드, 크로스 브라우징



웹의 전환기 (출처: Able Labs)

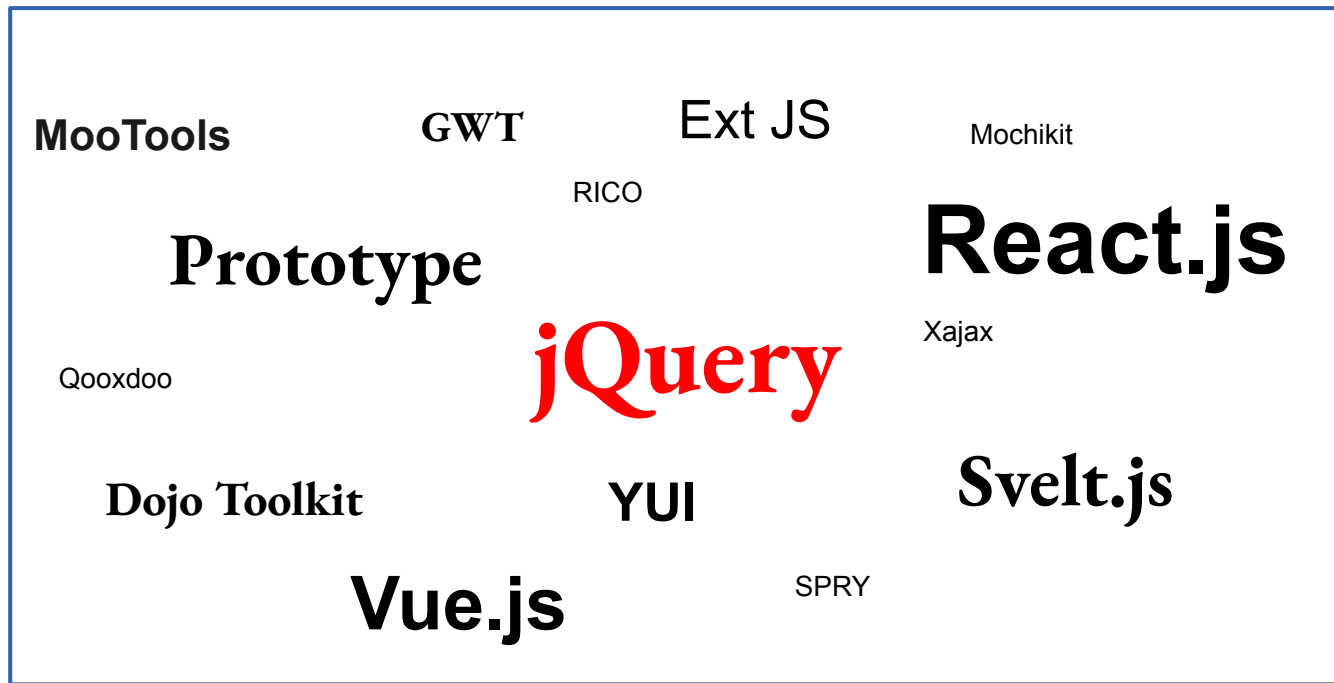
자바스크립트 라이브러리

- 정의
 - 하나의 서브루틴이나 자주 재사용 가능한 함수들이 모여 있는 파일.
- 특징
 - 크로스 브라우저 (Cross-Browser) 지원.
 - 용이한 Ajax 관련 작업.
 - DOM(Document Object Model)을 효과적으로 다루기 위한 Selector와 Event 핸들러 등 반복적으로 사용 되는 유용한 함수를 다수 제공.
 - 복잡한 구조를 가진 UI를 손쉽게 구현.
 - UI 구현에 필요한 대부분의 컴포넌트를 API화 하여 제공.
- 장/단점
 - 이해 하기 쉽고 간결한 코드 구사, 생산성 향상, 중복 작업 감소.
 - 별도로 라이브러리 사용법 이해, 덩치가 큰 파일들을 추가로 로드 하는 부담.

자바스크립트 라이브러리 유형

- JavaScript Helper Library
- User Interface / Component Library
- Complication Library

자바스크립트 라이브러리 종류



자바스크립트 라이브러리 트렌드 비교



선택 시 고려 사항

- 어떠한 웹 어플리케이션을 개발할 것인가?
- 어떠한 코딩 스타일(취향)인가?
- **대세를 따라서...**

바람을 읽어야 하는데 파도만 봤다. (영화 감상)

jQuery 라이브러리

- John Resig씨에 의해 작성된 가볍고 빠른 자바스크립트 프레임워크 라이브러리
- 자바스크립트 생산 편의성을 제공, 코드가 단순하고 간결
- DOM과 관련한 반복문과 Ajax의 복잡성을 기발하게 처리
- 코어 외의 추가적인 기능(특화 라이브러리)들은 플러그인의 형태로 제공
- Prototype에도 jQuery의 편리한 코딩 룰을 여러 부분에 도입하고 있는 실정
- CSS 표준을 따르는 jQuery의 DOM 선택기능은 사용하기 쉽기로 유명



jQuery 라이브러리

- 2005년 “John Resig”에 의해 디자인된 자바스크립트 라이브러리
- 자바스크립트의 코드를 단순하고 간결한 상태로 개발이 가능
- 동일한 코드의 반복, 복잡하고 많은 코드로 개발되던 기존의 작업에 비해 여러 가지 효과나 이벤트를 간단한 함수호출만으로 쉽고 빠르게 개발 가능
- 가장 강력한 자바스크립트 라이브러리
 - 간단한 자바스크립트 작업 수행
- 문서 객체 처리 기능
 - 웹 문서 내 객체 접근
- CSS나 Xpath 같은 직관적 방법
 - 웹 문서 내 객체를 쉽게 변경 가능
 - 웹 문서 내 사용자 액션을 변경 가능
- 다양한 기능 제공
 - 애니메이션 기능 추가 가능, Ajax 기능 사용 가능
 - 브라우저 호환성 유지



John Resig

John Resig is best known as an expert in the JavaScript programming language and the creator of the most popular JavaScript library in the world: [jQuery](#). He's created numerous JavaScript projects that continue to be integral parts of modern day web development. He's also the author of the popular JavaScript books [Secrets of the JavaScript Ninja](#) and [Pro JavaScript Techniques](#).



John is a Principal Architect at [Khan Academy](#) where he works to provide a free education to everyone, everywhere. He's worked on many aspects of the product: math exercises, mobile, accessibility, internationalization, build systems, and performance. His largest project was the creation of the [Computer Programming](#) learning environment.

<https://johnresig.com/>

jQuery 라이브러리 특징

- CSS Selector
 - HTML 문서의 구조를 명료 하면서도 읽기 쉬운 형태로 표현 및 사용 가능
- Plug-In Architecture
 - 중복되는 기능과 코드가 엉키는 등의 **Feature Creep**을 피하고 창의적인 산출물의 공유가 가능
 - 이미 개발된 많은 플러그인 을 쉽고 빠르게 이용 가능
- Method Chaining
 - 여러 개의 기능을 한 줄에 나열, 임시 변수 사용을 최소화하여 불필요한 코드 반복 지양
- Cross Browser
 - 각각의 브라우저에 발생하는 이벤트, 객체 처리방법에 따라 여러 개의 함수 또는 여러 번의 분기가 없이 jQuery에서 제공하는 함수 또는 문장으로 간단히 해결.

CDN이란?

- CDN은 Content Delivery Network 의 약자
- 사용자에게 간편하게 콘텐츠 제공하는 방식 의미
 - 구글, 마이크로소프트, jQuery측에서 사용자가 jQuery를 사용하기 편하게 콘텐츠 제공
- jQuery 파일명
 - ○○.js 파일
 - Uncompressed 버전
 - ○○.min.js 파일
 - ○○.min.js 파일은 Minified 버전 (용량이 다섯 배 이상 차이)
 - Minified 버전은 파일의 용량을 최소화하려고 압축한 파일
- 오프라인에서 jQuery 사용
 - 반드시 다운받아 사용

jQuery CDN

- jQuery 라이브러리 다운로드
 - <https://jquery.com/download/>
- jQuery CDN
 - <https://releases.jquery.com/>
- 간단한 CDN URL
 - <https://code.jquery.com/jquery.js>

```
/*!
 * jQuery JavaScript Library v1.11.1
 * http://jquery.com/
 *
 * Includes Sizzle.js
 * http://sizzlejs.com/
 *
 * Copyright 2005, 2014 jQuery Foundation, Inc. and other contributors
 * Released under the MIT license
 * http://jquery.org/license
 *
 * Date: 2014-05-01T17:42Z
 */
(function( global, factory ) {
    if ( typeof module === "object" && typeof module.exports === "object" ) {
        // For CommonJS and CommonJS-like environments where a proper window is present,
        // execute the factory and get jQuery
        // For environments that do not inherently possess a window with a document
        // (such as Node.js), expose a jQuery-making factory as module.exports
        // This accentuates the need for the creation of a real window
        // e.g. var jQuery = require("jquery")(window);
        // See ticket #14549 for more info
        module.exports = global.document ?
            factory( global, true ) :
            function( w ) {
                if ( !w.document ) {
                    throw new Error( "jQuery requires a window with a document" );
                }
                return factory( w );
            };
    } else {
        factory( global );
    }
}(
    // Pass this if window is not defined yet
    typeof window !== "undefined" ? window : this, function( window, noGlobal ) {

        // Can't do this because several apps including ASP.NET trace
        // the stack via arguments.caller.caller and Firefox dies if
        // you try to trace through "use strict" call chains. (#13335)
        // Support: Firefox 18+
    })
```

<script src="https://code.jquery.com/jquery.js"></script>

\$ 함수의 이해

- \$는 식별자이다.
- 자바스크립트는 \$ 또는 _를 식별자로 사용 가능 함.
- `var $ = jQuery;`

```
(function( global, factory ) {  
    if ( typeof module === "object" && typeof module.exports === "object" ) {
```

--- 중 략 ---

```
        return jQuery;  
    });
```

- `$(this).hide()`- 현재 요소를 숨깁니다.
- `$("p").hide()`- 모든 `<p>` 요소를 숨깁니다.
- `$(".test").hide()`- `class="test"`인 모든 요소를 숨깁니다.
- `$("#test").hide()`- `id="test"`인 요소를 숨깁니다.

선택기 (Selector = 선택자)

- HTML 문서 내의 모든 DOM 요소/태그 접근 가능.
 - 단일 또는 다중 요소 선택 가능
- \$('선택기')
 - 선택기로 읽어오는 개체의 반환값은 jQuery에서 사용 할 수 있는 개체.
- CSS에서의 문법을 그대로 계승
 - `div a:div` 태그 내의 `a`요소 모두를 의미

`$("선택기")` 또는 `jQuery("선택기")`

선택기 (Selector = 선택기)

- 태그 이름으로 요소 가져오기
 - \$('p') 선택기
 - 모든 <p> 요소 가져 옴
 - \$('a') 선택기
 - 모든 <a> 요소 가져 옴
- 다중 태그 선택 : 콤마 구분
 - \$('p, a, span')
 - 모든 p, a, span 태그 요소 읽어 옴

jQuery 선택기

Syntax	Description
<code>\$("*")</code>	Selects all elements
<code>\$(this)</code>	Selects the current HTML element
<code>\$(".p.intro")</code>	Selects all <code><p></code> elements with <code>class="intro"</code>
<code>\$("p:first")</code>	Selects the first <code><p></code> element
<code>\$("ul li:first")</code>	Selects the first <code></code> element of the first <code></code>
<code>\$("ul li:first-child")</code>	Selects the first <code></code> element of every <code></code>
<code>\$("[href]")</code>	Selects all elements with an href attribute
<code>\$("a[target='_blank']")</code>	Selects all <code><a></code> elements with a target attribute value equal to <code>"_blank"</code>
<code>\$("a[target!='_blank']")</code>	Selects all <code><a></code> elements with a target attribute value NOT equal to <code>"_blank"</code>
<code>\$(":button")</code>	Selects all <code><button></code> elements and <code><input></code> elements of <code>type="button"</code>
<code>\$("tr:even")</code>	Selects all even <code><tr></code> elements
<code>\$("tr:odd")</code>	Selects all odd <code><tr></code> elements

CSS 선택기

태그	특정 태그 모두 가져오기
태그.class	특정 태그내의 특정 클래스 가져오기
태그#id	특정 태그내의 특정 id 가져오기
요소1 요소2	모든 자식 요소 가져오기
요소1 > 요소2	바로 아래 나오는 자식 요소
요소[특성]	attribute가 있는 요소 가져오기
요소[특성=값]	속성과 값이 일치하는 요소
요소[특성*=값]	~가 들어가는
요소[특성\$=값]	~로 끝나는
요소[특성^=값]	~로 시작하는
요소[특성1=값1][특성2=값2]	2개의 특성과 값을 모두 만족하는 요소
요소:nth-child(n)	부모 요소를 기준으로 n번째 위치한 요소
요소:first-child	부모 요소를 기준으로 첫번째 자식 요소
요소:last-child	부모 요소를 기준으로 마지막 자식 요소
요소:only-child	부모 요소를 기준으로 자식 요소가 딱 하나인 요소
요소:empty	부모 요소를 기준으로 자식을 갖지 않는 요소
요소:not(Selector)	다른 선택기를 제외한 요소

jQuery 전용 선택기

:first	첫번째 요소
:last	마지막 요소
:even	짝수번째 요소(n-1번째, 0번째부터 시작, 즉 0번째가 짝수)
:odd	홀수번째 요소
:nth(n)	n번째 요소
:eq(index)	주어진 인덱스에 해당하는 요소
:gt(index)	주어진 인덱스보다 큰 요소
:lt(index)	주어진 인덱스보다 작은 요소
:visible	CSS 속성의 display가 none이 아닌 요소
:hidden	CSS 속성의 display가 none인 요소
:parent	부모 요소
:contains('Text')	Text를 포함하는 요소

ready 핸들러

- ready 메서드
 - DOM 로드(이미지 로드 전) 후 실행할 콜백(callback)
- 함수 호출

기본 문법

```
$(document).ready(function() {  
    // DOM이 모두 로드된 후 실행할 코드 입력  
});
```

축약 문법

```
$(function() {  
    // DOM이 모두 로드된 후 실행할 코드 입력  
});
```

ready 핸들러

- Script는 DOM 요소가 준비가 될 때 까지 실행 대기
 - window.onload와 유사.
 - window.onload는 이미지를 포함하여 모두 로드될 때까지 기다림.
 - 스크립트 실행 지연.
 - ready 핸들러는 DOM tree가 만들어질 때까지만 대기.
 - jQuery가 이러한 작업을 모두 신경 씀.
 - 개발자는 그냥 사용만 하면 됨.

```
$(document).ready(function() {  
    실행문;  
});
```

jQuery 함수 체계

`$(document).ready();`

– jQuery의 진입점(Entry Point)

- 자바스크립트 익명 메서드(람다식) 호출

– `$(document).ready(function() { ... });`

- `$()` 로 CSS의 선택기와 모든 DOM 요소 접근

– `$("선택기").XXX메서드();`

- 익명(Anonymous)함수/무명함수

– `function() {}`

- `$()`와 `jQuery()`는 동일한 코드다.

– 둘 중 하나 사용 가능

jQuery 함수 체계

- `$('#id')`는 jQuery 개체를 반환한다.
- 만약, `document.getElementById()`와 같은 기능을 jQuery 표현하고자 한다면?

`$('#id')[0]`

또는

`$('#id').get(0)`

jQuery에 함수 추가(사용자 정의 Plugin)

- \$에 정의된 유틸리티 함수
- jQuery에 확장 집합에서 동작하는 메서드(커멘드)

```
(function($){  
    $.fn.[플러그인명] = function() {  
        // 여기에 플러그인.  
  
        return this;  
    }  
})(jQuery)
```

또는

```
$.fn.[플러그인명] = function() {  
    // 여기에 플러그인.  
  
    return this;  
}
```

jQuery 메서드 체인 형태로 만들기 위해서 반드시 **this**를 반환 해 주어야 한다.

jQuery에 함수 추가(사용자 정의 Plugin)

```
<h1>hello world</h1>
```

```
<script src="http://code.jquery.com/jquery.js"></script>
```

```
<script>
```

```
    $.fn.changeColor = function(color) {
```

```
        $(this).css('color',color);
```

```
        return this;
```

```
    }
```

```
</script>
```

```
<script>
```

```
    $('h1').changeColor('red').fadeOut('slow');
```

```
</script>
```

과정 실습

JQuery 실습

```
localStorage.setItem("name", "John Doe");  
localStorage.getItem("name");
```

로컬스토리지 설명

김범준의 가전 랜드

번호	제품명	가격	수량	장바구니담기
1	냉장고	300	1 ▼	장바구니에추가
2	세탁기	200	1 ▼	장바구니에추가
3	오디오	100	1 ▼	장바구니에추가
4	텔레비	150	1 ▼	장바구니에추가
5	압력솥	50	1 ▼	장바구니에추가
6	에어콘	100	1 ▼	장바구니에추가
7	건조기	100	1 ▼	장바구니에추가
8	전자렌지	20	1 ▼	장바구니에추가
9	에어프라이기	30	1 ▼	장바구니에추가

장바구니

냉장 2개 600원

총합계: 600원

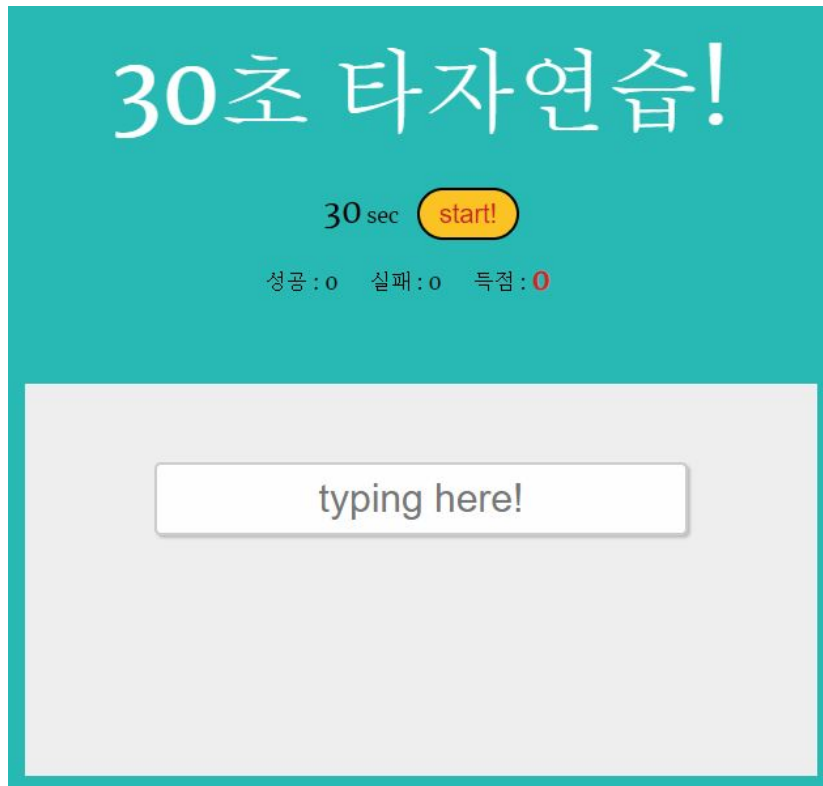
JQuery 실습

- 계산기



JQuery 실습

- 30초 타자연습



jQuery 실습

- jQuery를 이용한 슬라이드 메뉴 구현



Boots



Oxfords



Loafers

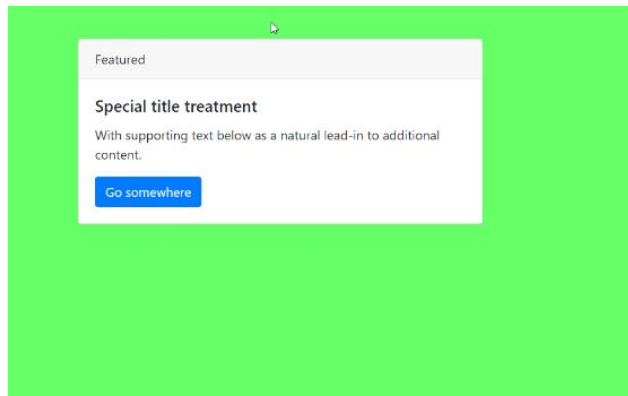
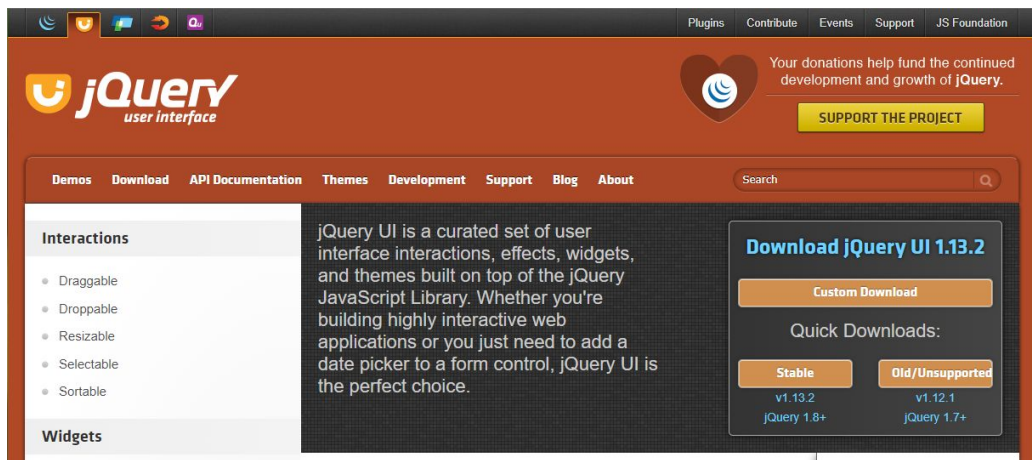


Sneakers



jQuery UI

- <https://jqueryui.com/>



<http://www.w3big.com/try/try.php?filename=jqueryui-example-draggable>

<https://shxrecord.tistory.com/165>

<https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=bgpoilkj&logNo=221220885520>

jQuery 플러그인

- <https://www.jqueryscript.net/>

Demo

Account #	First Name	Last Name	Age	Total	Discount	Difference	Date
A102	Bruce	Evans	22	\$13.19	11%	-100.9	Jan 18, 2007 9:12 AM
A33	Clark	Evans	18	\$15.89	44%	-26	Jan 12, 2003 11:14 AM
A42a	Bruce	Evans	22	\$13.19	11%	0	Jan 18, 2007 9:12 AM
A255	Bruce	Jones	33	\$13.19	25%	+12	Dec 10, 2002 5:14 AM
A42b	Peter	Parker	28	\$9.99	20.9%	+12.1	Jul 6, 2006 8:14 AM
A1	Bruce	Almighty	45	\$153.19	44.7%	+77	Jan 18, 2001 9:12 AM

TIP! Sort multiple columns simultaneously by holding down the Shift key and clicking a second, third or even fourth column header!

<https://mottie.github.io/tablesorter/docs/>

jQuery API

jQuery Data 분석 및 이해

- Ajax
 - Asynchronous Javascript and XML.
 - 자바스크립트 비동기 서버 요청 기술.
- JSON
 - 예전엔 Ajax 기술에서 XML로 데이터를 처리 했지만 최근에는 JSON으로 처리.
 - `JSON.stringify()`
 - `JSON.parse()`
- Axios

jQuery Data 분석 및 이해

- data()
 - `$(selector).data(key, value)`
 - 선택한 엘리먼트에 "속성:값"을 "key:value"형태로 저장.
- 데이터 접근
 - `map` 구조이므로 저장된 `value`를 `key`로 접근 가능.
 - 실제 엘리먼트에는 `data-속성="값"` 형식으로 저장 됨.
- 데이터 삭제
 - `$(selector).removeData(key)`

jQuery Effect 적용

- jQuery 라이브러리에서 제공하는 애니메이션 기능.
 - hide()
 - show()
 - fadeIn()
 - fadeOut()
 - slideDown()
 - slideUp()
 - slideToggle()
 - animate()

과정 실습

jQuery 실습

- WebSocket 또는 Socket.io를 활용한 채팅 귓속말 기능.

login
+ id: 'test01'
+ password: '12345'
+ alias: '방탄소년단'
+ today: 좋은날

채팅 클라이언트

localhost	3000	연결하기
-----------	------	------

test01	*****
방탄소년단	좋은날

로그인	로그아웃
-----	------

보내는 사람 ID	test01
받는 사람 ID	test02
메세지 데이터	Hello^^
전송	

결과:

사용자 :	user2	Login	Disconnect
받는이 :	All		
메세지 :	All		Send
	user1		
	user2		

```
>>> open
...
<<< user2님이 입장하였습니다!
```

jQuery 실습

- SPA (Single page application) 구현



SPA 구조와 기존 Web구조

이름

부서

직책

사진 KakaoTalk_20...90108774.png

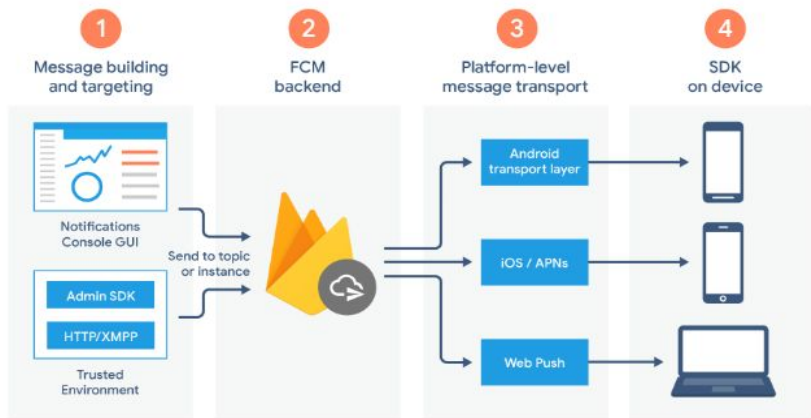
| 검색:

<input type="checkbox"/>	번호	사진	사원명	부서	직급	수정	삭제
<input type="checkbox"/>	1040		홍길동	개발	대리	<input type="button" value="edit"/>	<input type="button" value="delete"/>

(c)comstudy21.or.kr

jQuery 실습

- firebase 클라우드를 이용한 작품 배포
- <https://www.youtube.com/watch?v=bJ-33ANIScE>



<https://firebase.google.com/docs/cloud-messaging/fcm-architecture?hl=ko>

Firestore 활용한 SPA CRUD 구현

- 샘플 페이지 :
 - <https://coms21lab.web.app/>
- 자료 문서:
 - <https://docs.google.com/presentation/d/1oXC8dA5NZBvaXKHrO1kF6LkTaPxP8OfvhMVX782Tn2g/edit?usp=sharing>
- 소스코드
 - <https://cafe.naver.com/comstudy21/14002>

Comstudy21
please remember me
I want to remain in your memory no matter how many years pass.
Your own photo sharing service
Anyone is free to share their photos and brief information on this site.

ObjectID: 5U3dedF0ROBX4jRS1QXj


ID:



NAME:

EMAIL:

PHOTO: 선택된 파일 없음

PASSWORD:


more information.

NO	PHOTO	ID	NAME	EMAIL	DELETE
0		aa2	aa2	aa2@aa.com	<input type="button" value="비밀번호"/> <input type="button" value="DELETE"/>
1		ssubie	써비	crinsader@naver.com	<input type="button" value="비밀번호"/> <input type="button" value="DELETE"/>

教子採薪

교자채신 : 자식에게 땀나무를
해 오는 법을 가르치라

Give a man a fish, and you feed him for a day.
Teach a man to fish, and you feed him for a lifetime.

**“생선을 주기보다는
고기 잡는 법을 알려줘라!”**



jQuery Ajax

jQuery Ajax 기초 개념

- **AJAX**는 전체 페이지를 다시 로드하지 않고도 서버와 데이터를 교환하고 웹 페이지의 일부를 업데이트하는 기술.
- jQuery에서는 유용한 Ajax 함수를 많이 제공 한다.



https://www.w3schools.com/jquery/jquery_ajax_intro.asp

jQuery Ajax 메소드

메소드	설명
<code>\$.ajax()</code>	비동기식 Ajax를 이용하여 HTTP 요청을 전송함.
<code>\$.get()</code>	전달받은 주소로 GET 방식의 HTTP 요청을 전송함.
<code>\$.post()</code>	전달받은 주소로 POST 방식의 HTTP 요청을 전송함.
<code>\$.getScript()</code>	웹 페이지에 스크립트를 추가함.
<code>\$.getJSON()</code>	전달받은 주소로 GET 방식의 HTTP 요청을 전송하여, 응답으로 JSON 파일을 전송받음.
<code>.load()</code>	서버에서 데이터를 읽은 후, 읽어 들인 HTML 코드를 선택한 요소에 배치함.

GET / POST

- 목록이나 데이터 요청은 **GET**으로 입력, 수정, 삭제 등은 **POST**로 요청

특징	GET 방식	POST 방식
캐시화(cached)	캐시될 수 있음.	캐시되지 않음.
브라우저 히스토리	히스토리에 쿼리 문자열이 기록됨.	히스토리에 기록되지 않음.
데이터 길이	데이터의 길이가 URL 주소의 길이 이내로 제한됨. (익스플로러에서 URL 주소가 가질 수 있는 최대 길이는 2,083자이며, 이 중에서 순수 경로 길이는 2,048자까지만 허용됨)	제한 없음.
데이터 타입	오직 ASCII 문자 타입의 데이터만 전송할 수 있음.	제한 없음.
보안성	데이터가 URL 주소에 포함되어 전송되므로, 아무나 볼 수 있어 보안에 매우 취약함.	브라우저 히스토리에도 기록되지 않고, 데이터가 따로 전송되므로, GET 방식보다 보안성이 높음.

jQuery Ajax 입/출력 활용

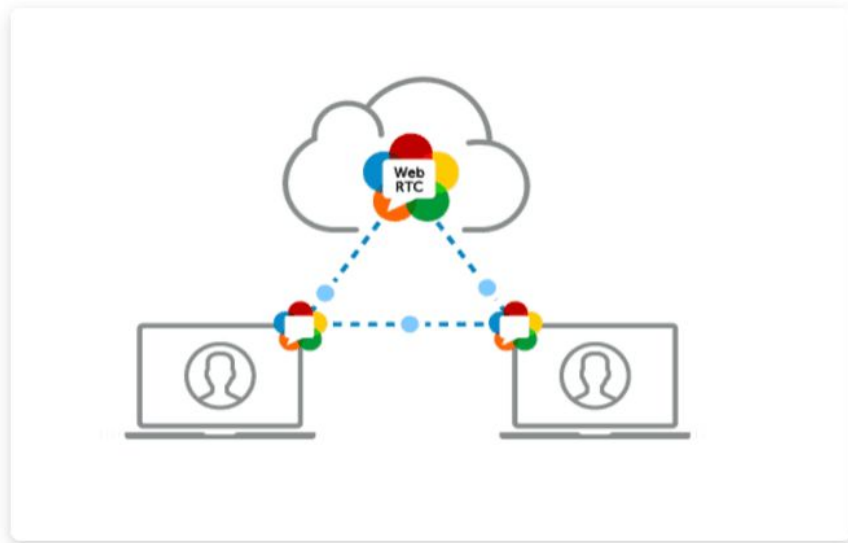
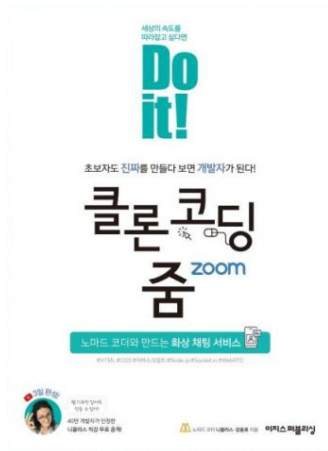
- 서버에서 받은 JSON 데이터를 파싱 해서 목록으로 만든다.
 - 방법 1 : `html()` 함수.
 - 방법 2 : `append()` 함수.
 - 방법 3 : `underscore` 템플릿 활용.



과정 실습

jQuery 실습

- WebRTC를 활용한 화상 채팅 구현
 - <https://nomadcoders.co/noom>



<https://www.wenyanet.com/opensource/ko/607f95c3c4641e0b9e2affdf.html>

<https://webrtc.org/getting-started/media-devices>

<https://aren227.me/how-to-use-mediadevices/>

다음은 React

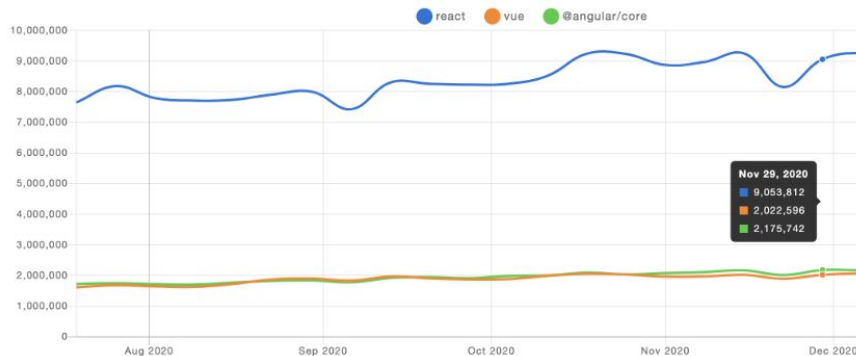
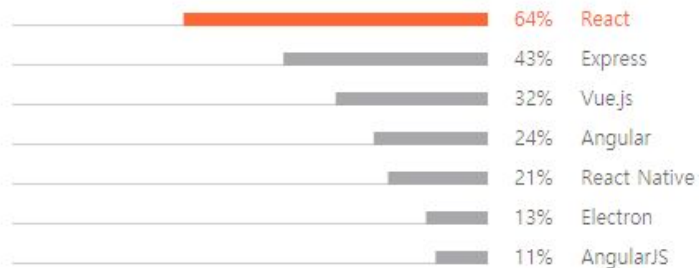


그림 4 지난 1년간 Angular vs React vs Vue.js NPM 다운로드 수 비교(출처: npm trends)

그림 3 IntelliJ 등의 유명한 개발 IDE를 개발하는 JetBrains사의 2020년도 개발자 에코시스템의 현황 설문조사