

웹최신기술리뷰01

4일 과정: 간단하게 기술을 리뷰하는 형태로 진행

교 육 명	웹애플리케이션 최신 트렌드 과정
교 육 특 징	<ul style="list-style-type: none">▪ 현재 가장 핫한 웹프론트엔드 프레임워크인 React 와 백엔드 기술인 ASP.NET 웹폼으로 구현하는 과정▪ D3js, Threejs, VueJS, AngularJS, Javascript ES6/7 등에 관한 최신 기술들의 사례를 습득할 수 있는 과정 개발 도구는 비주얼스튜디오 코드, 비주얼스튜디오 2019 나 2022 를 사용
교 육 목 적	<ul style="list-style-type: none">▪ 최신 웹 개발 상황을 이해할 수 있다.▪ React 에 대해서 기본적인 학습과 함께 기본적인 패턴 및 구현과정에 대해서 알 수 있다▪ ASP.NET 웹폼을 사용해 작업하는 방법을 알 수 있다.▪ Javascript 기반 웹 프론트엔드 프레임워크를 사용해 반응형 웹사이트를 제작할 수 있다.
교	▪ 2022 년 02 월 7 일, 02 월 9,10,11 일 총 4 일(1 일

육 기 간	5 시간, 총 20 시간)	
교 육 장 소	<ul style="list-style-type: none"> 영우글로벌러닝 (서울특별시 학동로 171 삼익악기빌딩 2 층) 	
교 육 대 상	<ul style="list-style-type: none"> 최신 웹 개발 상황에 대한 기본적인 지식이 필요하신 분 웹 개발 프론트엔드 프레임워크에 관한 지식이 필요하신 분 	
교 육 내 용	1 일차	<p>최근 IT 기술의 트렌드 – 가속화되는 DT(디지털 트랜스포메이션)와 메타버스, De-fi, NFT, P2E 에 대한 이해</p> <p>웹기반 기술의 히스토리와 HTML5 소개와 이해</p> <ul style="list-style-type: none"> ✓ HTML 5 ✓ Document Object Model ✓ JavaScript 를 직접 사용하는 데모

	2 일차	<p>Front End – 웹기술의 앞단을 이해하기 위한 소개와</p> <ul style="list-style-type: none"> ✓ jQuery 의 소개와 데모 ✓ Single Page Web Application ✓ ReactJS 소개와 데모 ✓ AngularJS ✓ VueJS ✓ D3js ✓ threejs 	
	3 일차	<p>Back End</p> <ul style="list-style-type: none"> ✓ Node.js 소개와 데모 ✓ JSON 소개와 데모 ✓ Java 소개 ✓ Spring Boot 소개 ✓ Asp.Net 소개 	
	4 일차	<p>ASP.NET Core 를 사용한 간단한 웹앱을 만들어 보기</p> <ul style="list-style-type: none"> ✓ ASP.NET Core 소개 ✓ MSDN 을 사용한 학습방법 ✓ 간단한 웹 페이지를 생성해 보기 	
	5 일차	<p>파이썬을 사용한 웹 페이지를 만들어 보기</p> <ul style="list-style-type: none"> ✓ 데이터 사이언스와 인공지능 분야에서 활용도 <p>파이썬</p>	

- | | |
|--|---|
| | <ul style="list-style-type: none">✓ 파이썬 설치와 개발환경 셋팅✓ Django 소개와 사용하기 |
|--|---|

*웹브라우저별로 HTML5 지원 점수

<http://html5test.com/index.html>

*브라우저별 어떤 버전부터 기능이 지원되는지를
검색하는 경우

<http://caniuse.com/>

*웹브라우저 한국 점유율

<https://gs.statcounter.com/browser-market-share/all/south-korea/#monthly-202012-202201>

1. DemoScript 웹 빈 프로젝트를 만들고 script00.html을
추가해서 아래와 같이 작업

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
<title></title>
</head>
<body>
```

```

<br />
```

```

```

```
<script>
```

```
    var helpIcon2 =
document.getElementById("helpIcon2");
    helpIcon2.addEventListener("mouseover",
function () {
        window.alert('약간의문자열출력');
    });
```

```
</script>
```

```
</body>
```

```
</html>
```

1. script01.html에 아래와 같이 작업

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
```

```
<title></title>
```

```
</head>
```

```
<body>
```

```
<form>
```

```
<p>
```

```
<label>이름:
```

```
<input type="text" id="NameBox" /></label>
```

```

</p>
<input type="button" id="submit" value="전송" />
</form>
<div id="thankYouArea"></div>
<script type="text/javascript">
    function sayThankYou() {
        var userName =
document.getElementById("NameBox").value;
        var thankYou =
document.createElement("p");
        thankYou.textContent = "감사합니다. " +
userName;

document.getElementById("thankYouArea").appendC
hild(thankYou);
    }

document.getElementById("submit").addEventListener
("click", sayThankYou);
</script>

</body>
</html>

```

(JavaScript기본-수업실습 3장에
DemoJavaScriptFund프로젝트에 저장되어있다.)

1. 논리합(||) 과논리곱(&&)연산자에 대한데모

```

<script>
// alert (true && true) => alert(true); 으로연산됨
    alert(30 > 20 && 20 > 10);
</script>

```

2. 연산자와 수식 데모

```
<script>  
var radius = 10;  
var pi = 3.141592;  
  
    alert(2 * pi * radius);  
</script>
```

3. 간단한 형식 나열 데모

```
<!DOCTYPEhtmlPUBLIC"-//W3C//DTD XHTML  
1.0  
Transitional//EN""http://www.w3.org/TR/xhtml1/  
DTD/xhtml1-transitional.dtd">
```

```
<htmlxmlns="http://www.w3.org/1999/xhtml">  
<head>  
<title></title>  
<script>  
var stringVar = 'String';  
var numberVar = 200;  
var booleanVar = true;  
var functionVar = function () { };  
var objectVar = {};  
</script>
```

4. 바디에 동적 형태로 내용을 나열한다.

```
<!DOCTYPEhtmlPUBLIC"-//W3C//DTD XHTML  
1.0  
Transitional//EN""http://www.w3.org/TR/xhtml  
1/DTD/xhtml1-transitional.dtd">
```

```
<htmlxmlns="http://www.w3.org/1999/xhtml">  
<head>  
<title></title>  
<script>
```

```
    window.onload = function () {  
var list = '';
```

```
        list += '<ul>';  
        list += '    <li>Hello</li>';  
        list += '    <li>Javascript</li>';  
        list += '</ul>';
```

```
        document.body.innerHTML = list;  
    }  
</script>
```

```
</head>
```

```
<body>
```

```
</body>
```

```
</html>
```

5. 배열을 나열하고 undefined의 의미를 살펴보는 데모


```
<!DOCTYPEhtmlPUBLIC"-//W3C//DTD XHTML
1.0
Transitional//EN""http://www.w3.org/TR/xhtmll
1/DTD/xhtml1-transitional.dtd">
```

```
<htmlxmlns="http://www.w3.org/1999/xhtml">
<head>
<title></title>
<script>
```

```
//배열을선언
```

```
var array = [273, 'string', true, function ()
{ }, {}, [273, 100]];
```

```
//출력
```

```
alert(array);
```

```
//undefined
```

```
alert(typeof (variable));
```

```
//변수를선언했지만초기화하지않아도
```

```
undefined
```

```
var variable2;
```

```
alert(typeof (variable2));
```

```
</script>
```

```
</head>
```

```
<body>
```

```
</body>
</html>
```

6. 더하기 연산자의 경우 문자열로 연산될 수 있다.

```
<script>
//숫자와 문자열 자료형 변환
    alert('52 + 273');
    alert(52 + 273);
//전부 다 문자열로 처리된다.
    alert('52' + 273);
    alert(52 + '273');
    alert('52' + '273');
```

//더하기 연산자를 제외한 사칙 연산자는 숫자가 우선
된다.

```
    alert('52 * 273');
    alert(52 * 273);
    alert('52' * 273);
    alert(52 * '273');
    alert('52' * '273');
```

```
</script>
```

7. 불린 형식 데모:

```
<script>
//전부 false로 변환된다.
    alert(Boolean(0));
    alert(Boolean(NaN));
```

```
alert(Boolean(''));
alert(Boolean(null));
alert(Boolean(undefined));
```

//논리부정연산자를사용하면전부 true

```
alert(!Boolean(0));
alert(!Boolean(NaN));
alert(!Boolean(''));
alert(!Boolean(null));
alert(!Boolean(undefined));
```

</script>

8. 자동형식변환이일어나서문제가된다면

===을비교할경우사용한다.

<script>

//자료변환이자동으로일어나서모드 true 리턴

```
alert('' == false);
alert('' == 0);
alert(0 == false);
alert('273' == 273);
```

//자료형이다른것을확실하게구분하고싶다면일치
연산자사용

```
alert('' === false);
alert('' === 0);
alert(0 === false);
alert('273' === 273);
```

</script>

9. 조건문 데모

<script>

var date = new Date();

var hour = date.getHours();

if (hour < 11) {

 alert('아침먹을시간입니다');

 } else {

if (hour < 15) {

 alert('점심먹을시간입니다');

 } else {

 alert('저녁먹을시간입니다');

 }

 }

//좀더 쉽게 변경해보면

if (hour < 11) {

 alert('아침먹을시간입니다');

 } elseif (hour < 15) {

 alert('점심먹을시간입니다');

 } else {

 alert('저녁먹을시간입니다');

 }

</script>

10. 조건문 데모 switch 구문

<script>

```
var input =  
Number(prompt('숫자를 입력하세요', '숫자'));
```

```
switch (input % 2) {  
  case 0:  
    alert('짝수입니다.');
```

```
    break;
```

```
  case 1:
```

```
    alert('홀수입니다.');
```

```
    break;
```

```
  default:
```

```
    alert('숫자가 아닙니다.');
```

```
    break;
```

```
}
```

</script>

11. 함수 데모:

<script>

//이름이 없는 익명 함수여서 변수에 넣어서 사용한다.

```
var 함수 = function () {
```

```
  var output = prompt('숫자를 입력', '숫자');
```

```
  alert(output);
```

```
};
```

```
//코드를직접본다.  
    alert(함수);
```

```
//직접실행한다.  
함수();
```

```
</script>
```

12. 두번째함수데모: 두번째함수가실행됨

```
<script>  
//첫번째함수를두번째함수가덮어써버리기때문에  
두번째함수가실행됨  
function함수() { alert('함수 A'); }  
function함수() { alert('함수 B'); }  
함수();  
</script>
```

13. 익명함수와선언적함수비교

```
<script>  
//아래의익명함수를실행되지않는다.  
변수를선언하기이전에  
//변수를사용해서그렇다!  
함수();  
var함수 = function () { alert('함수 A'); }  
var함수 = function () { alert('함수 B'); }  
</script>  
  
<script>
```

//선언적함수의경우순서에관계없이실행된다.

```
함수();  
function 함수() { alert('함수 A'); }  
function 함수() { alert('함수 B'); }  
</script>
```

14. 매개변수와리턴값데모

```
<script>  
//function 함수이름(매개변수, 매개변수){  
//  함수코드  
//  return 리턴값;  
//}  
function f(x) { return x * x; }  
    alert(f(3));  
</script>
```

15. 클로저:

내부변수의값을그대로남겨두고사용할수있도록한다.

```
<script type="text/javascript">  
function outerFunction(name) {  
    //자바스크립트스스로아직지역변수 output을  
    //지우면안된다는것을인식하고남겨두므로정상  
    //출력된다. 이것을클로저라고한다.  
    var output = 'Hello' + name + '...';  
    return function() {  
        alert(output);  
    };  
}
```

```

    };
}
//괄호가 붙는 것에 주의한다.
    outerFunction('aaa')();
</script>

```

16. 내부변수의 값을 남겨두고 각각 사용하는 데모

```

<script>
function outerFunction(name) {
var output = 'Hello' + name + '...';
return function () {
return output;
};
}

```

```

//변수를 선언
var first = outerFunction('Javascript');
var second = outerFunction('jQuery');

```

```

//출력한다. 고유한 지역변수
output을 가지고 있는 것을
//알 수 있다.
    alert(first());
    alert(second());
</script>

```

17. 인코딩 데모:

```

<script>
//인코딩할 URL을 만듭니다.

```



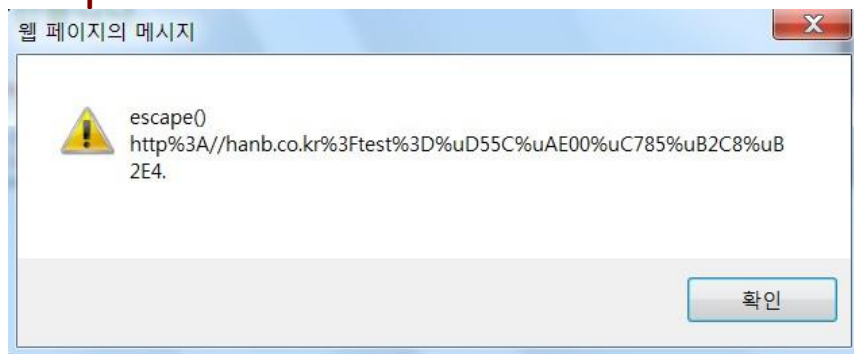
```

var URI = 'http://hanb.co.kr?test=한글입니다.';

//출력할문자열을만듭니다.
var output = '';
    output += 'escape()\n'
    output += escape(URI) + '\n\n';
//output += 'escapeURI()\n'
//output += escapeURI(URI) + '\n\n';
//output += 'escapeURIComponent()\n'
//output += escapeURIComponent(URI) + '\n\n';

    alert(output);
</script>

```



18. 평가함수데모:

```

<script>
//eval()함수에사용할문자열을만든다.
var willEval = '';
    willEval += 'var number = 10;';
    willEval += 'alert(number);';

//eval()함수를사용
    eval(willEval);

```

//eval()을통해실행된코드에서정의된변수도사용할 수있다.

```
        alert(number);  
</script>
```

19. NaN 과 inFinite를구분해야한다. 데모:

```
<script>  
//NaN은숫자지만자바스크립트가표현할수없는숫  
자  
//0으로숫자를나누면 infinity라는값이들어간다.  
var number1 = 10 / 0;  
var number2 = 10 / 'A';  
        alert(number1 + ' : ' + number2);  
//2가지를구분해서 true를리턴하는함수  
        alert(isFinite(number1) + ' : ' +  
isNaN(number2));
```

//주의해야할코드로 isNaN을꼭사용해야한다.

//아래의 == 비교는 false값이출력된다.

```
if (NaN == NaN) {  
        alert('Nan == Nan');  
    } else {  
        alert('Nan != Nan');  
    }  
</script>
```

20. parseInt(), parseFloat() 함수구분

```

<script>
var won = Number('1000원');
var dollar = Number('1.5$');
    alert(won + ' : ' + dollar);

var won2 = '1000원';
var dollar2 = '1.5$';
    alert(parseInt(won2) + ' : ' +
parseInt(dollar2));
    alert(parseFloat(won2) + ' : ' +
parseFloat(dollar2));
</script>

```

21. 자바스크립트에는 6가지 형식이 있다: 숫자, 문자열, 불리언, 함수, 객체, 정의되지 않은 자료형
22. 객체는 배열과 비슷하게 선언한다.

```

<script>
//객체를 선언
var product = {
    제품명: '망고',
    유형: '당절임',
    성분: '망고, 설탕',
    원산지: '필리핀'
};
</script>

```

두번째 데모를 보면 메서드가 추가된 것을 알 수 있다.

```
//객체의속성이가질수있는자료형
//      var object = {
//          number: 273,
//          string: 'Kim',
//          boolean: true,
//          array: [52, 273, 103, 32],
//          method: function () {

//      }
//      };
```

```
<script>
//객체를선언
var person = {
    name: '박문수',
    eat: function (food) {
        alert(this.name + '이 ' + food +
'를먹습니다.');
```

```
    }
};

//메서드를호출한다.
    person.eat('밥');
</script>
```

23. 객체의반복문

```
<script>
```

```

var product = {
    name: 'MS Visual Studio 2010',
    price: '750000',
    language: '한국어',
    supportOS: 'Win 32',
    subscription: true
};

var output = '';
for (var key in product) {
    output += key + ' : ' + product[key] +
'\n';
}
alert(output);
</script>

```

24. with 키워드를 사용하면 편하다.

```

<script>
var output = '';
var student = {
이름: '전우치',
국어: 100, 수학: 95,
영어: 90, 과학: 88
};

```

//with 키워드를 사용하면 객체를 명시할 필요없이 속성을 쉽게 사용한다.

```

with (student) {
    output += '이름: ' + 이름 + '\n';
    output += '국어: ' + 국어 + '\n';
    output += '영어: ' + 영어 + '\n';
    output += '수학: ' + 수학 + '\n';
    output += '과학: ' + 과학 + '\n';
};

alert(output);
</script>

```

25. 동적으로속성과메서드를추가한다.

```

<script>
//동적으로속성과메서드추가
var student = {};
    student.이름 = '박문수';
    student.취미 = '먹기';
    student.특기 = '프로그래밍';

//toString()메서드생성
    student.toString = function () {
var output = '';
for (var key in this) {
//toString()메서드는출력하지않게한다.
if (key != 'toString') {
        output += key + '\t' +
this[key] + '\n';

```

```

        }
    }
    return output;
};

    alert(student.toString());
    //속성을제거할때 delete키워드를사용한다.
    delete (student.장래희망)
    alert(student);
</script>

```

26. 배열에 객체를 추가해서 메서드로 처리한다.

```

<script>
// 학생 정보 배열을 만듭니다.
var students = [];
    students.push({ 이름: '윤인성', 국어: 87,
수학: 98, 영어: 88, 과학: 95 });
    students.push({ 이름: '연하진', 국어: 92,
수학: 98, 영어: 96, 과학: 98 });
    students.push({ 이름: '구지연', 국어: 76,
수학: 96, 영어: 94, 과학: 90 });
    students.push({ 이름: '나선주', 국어: 98,
수학: 92, 영어: 96, 과학: 92 });
    students.push({ 이름: '윤아린', 국어: 95,
수학: 98, 영어: 98, 과학: 98 });
    students.push({ 이름: '윤명월', 국어: 64,
수학: 88, 영어: 92, 과학: 92 });

```

```
students.push({ 이름: '김미화', 국어: 82,  
수학: 86, 영어: 98, 과학: 88 });  
students.push({ 이름: '김연화', 국어: 88,  
수학: 74, 영어: 78, 과학: 92 });  
students.push({ 이름: '박아현', 국어: 97,  
수학: 92, 영어: 88, 과학: 95 });  
students.push({ 이름: '서준서', 국어: 45,  
수학: 52, 영어: 72, 과학: 78 });
```

```
for (var i in students) {  
  //총점구하는메서드를추가  
  students[i].getSum = function () {  
    returnthis.국어 + this.수학 + this.영어 + this.과학;  
  };  
  //평균을구하는메서드를추가  
  students[i].getAverage = function () {  
    returnthis.getSum() / 4;  
  };  
}
```

```
//출력한다.  
var output = '이름\t\t총점\t\t평균\n';  
for (var i in students) {  
  with (students[i]) {  
    output += 이름 + '\t' + getSum() +  
      '\t' + getAverage() + '\n';  
  }  
}
```



```
}  
alert(output);
```

```
</script>
```

27. 객체를 생성해주는 함수를 만든다.

```
<script>
```

```
function makeStudent(name, korean, math,  
english, science) {
```

```
var willReturn = {
```

```
//속성
```

```
이름: name,
```

```
국어: korean,
```

```
수학: math,
```

```
영어: english,
```

```
과학: science,
```

```
//메서드
```

```
    getSum: function () {
```

```
    return this.국어 + this.영어 + + this.수학 +  
    this.과학;
```

```
    },
```

```
    getAverage: function () {
```

```
    return this.getSum() / 4;
```

```
    },
```

```
    toString: function () {
```

```
    return this.이름 + '\t' + this.getSum() + '\t' +  
    this.getAverage();
```

```
    }
```

```
        };  
    return willReturn;  
}
```

// 학생 정보 배열을 만듭니다.

```
var students = [];  
    students.push(makeStudent('박문수', 98,  
98, 90, 90));  
    students.push(makeStudent('전우치', 98,  
98, 90, 90));  
    students.push(makeStudent('김길동', 98,  
98, 90, 90));  
    students.push(makeStudent('이순신', 98,  
98, 90, 90));  
    students.push(makeStudent('김문수', 98,  
98, 90, 90));  
    students.push(makeStudent('문길동', 98,  
98, 90, 90));
```

//출력

```
var output = '이름\t\t총점\t\t평균\n';  
for (var i in students) {  
    output += students[i].toString() +  
'\n';  
}  
alert(output);  
</script>
```

28. 이번에는 생성자(new)를 사용해 본다.

<script>

// 생성자로 사용할 수 있도록 만든다.

```
function Student(name, korean, math, english, science) {
```

// 속성

```
this.이름 = name;
```

```
this.국어 = korean;
```

```
this.수학 = math;
```

```
this.영어 = english;
```

```
this.과학 = science;
```

// 메서드

```
this.getSum = function () {
```

```
return this.국어 + this.수학 + this.영어 + this.과학;  
};
```

```
this.getAverage = function () {
```

```
return this.getSum() / 4;  
};
```

```
this.toString = function () {
```

```
return this.이름 + '\t' + this.getSum() + '\t' +  
this.getAverage();
```

```
};
```

```
}
```

// 학생 정보 배열을 만든다. 생성자(new)를 사용한다.

```
var students = [];
    students.push(new Student('전우치', 99,
98, 98, 90));
    students.push(new Student('박문수', 99,
98, 98, 90));
    students.push(new Student('이길동', 99,
98, 98, 90));
    students.push(new Student('김길동', 99,
98, 98, 90));
    students.push(new Student('박길동', 99,
98, 98, 90));
    students.push(new Student('오달수', 99,
98, 98, 90));

var output = '이름\t총점\t평균\n';
for (var i in students) {
    output += students[i].toString() +
'\n';
}
alert(output);
</script>
```

프로그램은 다르더라도 변수의 개념은 같기에 미리 읽어보면 좋겠네요.

자바스크립트의 변수에 대한 이야기를 시작해봅니다.

먼저 자바스크립트의 변수는 일반적으로 var 라는 키워드로 선언을 합니다.

ES6 부터는 var 의 구조적 문제의 해결을 위해 let 과 const 가 추가되었습니다.

간단하게 비교한다면...

var 는 같은 변수명으로 중복 선언이 가능합니다.

let 은 같은 변수명으로 중복 선언이 안됩니다.

const 는 상수를 표현합니다. 즉 무조건 선언시 값을 할당해야 하고 이후 값을 변경할 수 없습니다.

예제 (Demo)

```
var firstVal = 0;
var firstVal = 1;

console.log("<<<<<<<<<<<<<firstVal>>>>>>>>>>>");
console.log(firstVal);
```

결과.

```
<<<<<<<<<firstVal>>>>>>>>  
1
```

위 예제처럼 firstVal 은 중복 선언이 가능하고 오류가
나지 않습니다.

이렇게 짧은 코드는 개발자가 쉽게 인지하겠지만...

공통 작업을 하는 경우 엄청 길고 많은 코딩이 이루어질텐데

var 로 중복된 변수를 선언해도 알기 힘들고 또한 잘못된 값이 전달될 수 있기에

문제가 발생하게 됩니다.

예제 (Demo)

```
let firstVal = 0;
let firstVal = 1;

console.log("<<<<<<<<<<firstVal>>>>>>>>>>");
console.log(firstVal);
```

결과.

```
let firstVal = 1;
```

```
  ^
```

```
SyntaxError: Identifier 'firstVal' has already been declared  
    at Object.compileFunction (node:vm:352:18)  
    at wrapSafe (node:internal/modules/cjs/loader:1031:15)  
    at Module._compile (node:internal/modules/cjs/loader:1098:18)  
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1168:10)  
    at Module.load (node:internal/modules/cjs/loader:978:32)  
    at Function.Module._load (node:internal/modules/cjs/loader:819:12)  
    at Function.executeUserEntryPoint [as runMain] (node:internal/main/run_main_module:17:47)
```

let 의 경우는 중복 선언이 안된다고 했습니다.

위 예제에서 보듯 let 으로 firstVal 을 같은 이름으로
중복 선언하고 실행하니

이미 선언된 식별자(변수)라고 오류를 표시합니다.

예제(Demo)

```
const firstVal;
```

```
console.log("<<<<<<<<<<firstVal>>>>>>>>>");  
console.log(firstVal);
```

결과.

```
const firstVal;
```

```
    ^^^^^^^^^
```

```
SyntaxError: Missing initializer in const declaration
    at Object.compileFunction (node:vm:352:18)
    at wrapSafe (node:internal/modules/cjs/loader:1031:15)
    at Module._compile (node:internal/modules/cjs/loader:1099:13)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1161:10)
    at Module.load (node:internal/modules/cjs/loader:981:32)
    at Function.Module._load (node:internal/modules/cjs/loader:822:14)
    at Function.executeUserEntryPoint [as runMain] (node:internal/main/run_main_module:17:47)
```

이번 예제는 상수 `const` 에 대한 부분입니다.

상수는 선언 시 무조건 값도 할당해야 합니다.

선언만 해서 초기 값이 빠졌다는 오류가 표시됩니다.

`const` 도 중복 선언이 안되며 `let` 과도 중복된 이름을 사용할 수 없습니다.

특히, `var` 는 중복 선언이 가능하지만 그건 `var` 끼리만 가능하고

`const` 나 `let` 과는 `var` 라도 중복 선언을 할 수 없습니다.

예제.


```
const firstVal = "init";
var firstVal = "b";

console.log("<<<<<<<<<<firstVal>>>>>>>>>");
console.log(firstVal);
```

결과.

```
var firstVal = "b";
  ^
SyntaxError: Identifier 'firstVal' has already been de
    at Object.compileFunction (node:vm:352:18)
    at wrapSafe (node:internal/modules/cjs/loader:1031:
    at Module._compile (node:internal/modules/cjs/load
    at Object.Module._extensions..js (node:internal/mo
    at Module.load (node:internal/modules/cjs/loader:9
    at Function.Module._load (node:internal/modules/cj
    at Function.executeUserEntryPoint [as runMain] (no
    at node:internal/main/run_main_module:17:47
```

const 에 firstVal 을 선언했기에 아무리 var 라도 중복된 이름을 선언하니 오류가 납니다.

자바스크립트의 변수는 초기에 값을 설정하지 않으면 "undefined"라는 값이 할당됩니다.

undefined 는 null 도 아니고 빈 값 ""도 아닌 그냥 undefined 입니다.

그래서 변수의 타입을 확인해봐도 undefined 로 나타납니다.

다음 예제로 확인해보겠습니다.

예제.

```
var abc;  
  
console.log(typeof abc);  
console.log(abc);
```

결과.

```
undefined  
undefined
```

신기하네요.

그런데 뭔가 이상한 점이 있지 않나요?

자바스크립트 변수를 선언할 때는 var, let, const 로 선언한다고 했습니다.

물론 위 선언문 없이 그냥 변수명만 써도 사용이 가능합니다.

그런데...

자바나 C 등의 프로그램은 변수를 선언할 때 타입으로 지정해서 하는데

자바스크립트는 그런 부분이 없습니다.

문자인지, 숫자인지, boolean 인지 선언하지 않아도 자동으로 캐스팅이 된다는 겁니다.

즉, 선언하고 값을 할당할 때 자동으로 값에 맞는 타입이 된다는 의미입니다.

예제.

```
var numStr = 1;  
console.log(typeof numStr);  
  
numStr = "a";  
console.log(typeof numStr);  
  
numStr = false;  
console.log(typeof numStr);  
  
numStr = null;  
console.log(typeof numStr);
```

결과.

```
number  
string  
boolean  
object
```

위 예제를 보면 numStr 를 선언하고 값을 바꿀 때마다 해당 타입이 변경되는 부분을 확인할 수 있습니다.

자바스크립트의 변수는 먼저 사용하고 그 뒤에 선언해도 사용이 가능합니다.

예제.

```
1 console.log(score);  
2  
3 score = 80;  
4 console.log(score);  
5 var score;  
6  
7 console.log(score);
```

결과.

```
undefined  
80  
80
```

위 예제를 보면 1 번 라인에 score 라는 변수의 값을 출력합니다.

결과는 undefined 입니다.

4 번의 출력 값은 80 이 나오고 7 번 또한 80 이 나옵니다.

변수는 3 번에서 할당했고 5 번에서 선언을 했습니다.

그냥 보기에 혼란스럽네요.

자바스크립트는 런타임 시점에 한 줄씩 순서대로 실행되는데,

변수 선언은 좀 다릅니다.

런타임 전 자바스크립트 엔진에서 먼저 선언된 변수들을 끌어올려(호이스팅)

실행을 하게 됩니다.

그 논리로 다시 위 예제를 보겠습니다.

```
1  console.log(score);  
2  
3  score = 80;  
4  console.log(score);  
5  var score;  
6  
7  console.log(score);
```

var score 가 5 번에 있기에 먼저 실행하고 인지하게 됩니다.

그럼 1 번 앞에 var score;가 오겠네요.

그리고 1 번이 실행되면 아직 값이 할당 안되었기에 undefined 로 할당돼서 결과가 나오게 됩니다.

그리고 3 번에서 80 으로 할당하게 됩니다.

그러면 4 번에서 로그 출력 시 80 으로 나오게 됩니다.

5 번은 이미 1 번 위로 선언되었기에 7 번으로 가서 다시 80 을 출력하게 됩니다.

이렇게 자바스크립트를 실행하면 자바스크립트 엔진에서 먼저 선언된 변수들을

쪽 끌어올려서 실행을 하는데 이것을 호이스팅(hoisting)이라고 합니다.

다른 프로그래밍 언어와는 조금 다른 방식의 변수 관리였습니다.

이 부분을 이해하기 위해서는 반드시 샘플을 다양하게 만들어서

확실히 이해하고 간다면

실전에서 개발하는 경우 로직의 이해나 오류에 대응할 때 도움이 될 것입니다.

내일 또 코딩 전쟁터로 떠나야겠네요.

즐 코딩하세요~~

안녕하세요. 신기한 연구소입니다.

자바스크립트를 공부하면 Scope(스코프)를 만나게 되는데요.

그중 렉시컬 스코프(lexical scope)의 의미가 참~이해하기 힘들더군요.

자바스크립트 렉시컬 스코프 (lexical scope) - tiBoy -

그래서 이번 포스팅은 렉시컬 스코프(lexical scope)를
나름 쉽게 정리해보려고 합니다.

그럼 같이 살펴보겠습니다.

자바스크립트(javascript)는 변수, 함수 등을 선언하고
사용합니다.

기본적으로 함수와 변수를 어디에 포함되지 않고(전역지역) 선언하면 전역 함수, 전역 변수가 됩니다.

전역이라면 어디서든 접근이 가능하다는 의미입니다.

하지만 함수 내부에 선언된 변수와 내부 함수는 지역 변수, 함수가 됩니다.

즉 부모 함수의 밖에서는 접근할 수 없게 됩니다.

부모 함수안에 자식 함수를 만들고 그 자식 함수 안에 손자 함수를 만든다고 가정한다면

손자 -> 자식 -> 부모 -> 전역으로 접근해서 사용할 수 있게 됩니다.

같은 전역 지역에 선언된 함수 A, B 2 개가 있다면 서로 내부의 변수와 함수에 접근하거나 사용할 수 없습니다.

그냥 전역에 선언된 함수만 알 뿐이죠. 그 안의 변수와 함수는 다른 함수들이 알 수 없거든요.

예제 소스로 확인해보겠습니다.

```
1 var a = 1; //전역 지역의 전역변수
2
3 function funcA() { //전역 지역의 전역 함수
4     console.log(a);
5     return 2;
6 }
7
```



```
8  
9 console.log(a);  
10 console.log(funcA());
```

1 라인의 a 는 전역 지역에 선언된 전역 변수입니다.

3 라인의 funcA()는 전역 지역에 선언된 함수입니다.

4 라인의 a 는 전역 변수를 사용하고 있습니다.

실행 결과는 다음과 같습니다.

```
1 1  
2 1  
3 2  
4 CS
```

9 라인의 결과 1, 10 라인의 결과는 함수를 호출해서
4 라인의 1, 5 라인의 2 가 순서대로 출력되었습니다.

다음은 지역에 대한 예제입니다.

```
1 var a = 1; //전역 지역의 전역변수  
2  
3 function funcA() { //전역 지역의 전역 함수  
4     var a = 3;  
5     console.log(a);  
6     console.log(b);  
7     return 2;  
8 } CS
```

```

9 }
10
11 function funcB() { //전역 지역의 전역 함수
12     var b = 4;
13     console.log(a);
14     console.log(b);
15     return 2;
16 }
17
18 console.log(a);
19 console.log(b);
    console.log(funcA());

```

이 예제는 오류가 발생합니다.

바로 6 라인 때문입니다.

변수 b 는 전역 지역의 다른 함수 funcB 에 선언되어
있기에

funcA 에서는 접근할 수 없습니다.

오류를 수정해봅시다.

```

1 var a = 1; //전역 지역의 전역변수
2
3 function funcA() { //전역 지역의 전역 함수
4     var a = 3;
5     console.log(a);
6     return 2;
7 }
8
9 function funcB() { //전역 지역의 전역 함수 CS

```

```

10  var b = 4;
11  console.log(a);
12  console.log(b);
13  return 2;
14 }
15
16 console.log(a);
17 console.log(b);
18 console.log(funcA());

```

이 코드 또한 오류가 발생했습니다.

바로 17 라인 때문인데요.

변수 b 는 funcA 함수내 지역 변수이기에 전역에서 바라볼 수 없습니다.

안에서 밖으로 바라볼 수 있지만 밖에서는 안으로 못가는 상황이네요.

밖에서 (전역) 함수 내 선언된 변수(지역)을 바라볼 수 없다고 정리됩니다.

이 내용은 중요하니 잘 기억해두세요.

다시 오류를 수정해봅시다.

```

1  var a = 1; //전역 지역의 전역변수
2
3  function funcA() { //전역 지역의 전역 함수
4      var a = 3;
5      console.log(a);
6  }

```

```

7   return 2;
8 }
9
10 function funcB() { //전역 지역의 전역 함수
11     var b = 4;
12     console.log(a);
13     console.log(b);
14     return 2;
15 }
16
17 console.log(a);
18 console.log(funcA());
   console.log(funcB());

```

이제 오류 없이 실행이 됩니다.

결과는 어떻게 나올까요?

```

1 1
2 3
3 2
4 1
5 4
6 2CS

```

1 라인은 전역 변수 a 를 출력합니다. 그래서 1.

2 라인은 funcA 에서 지역 변수 a 를 출력합니다.
그래서 3.

3 라인은 funcA 의 return 으로 2.

4 라인은 funcB 에서 전역 변수 a 를 바라보기에
1 입니다.

주의할 점은 funcA 를 실행했고 a 가 3 으로 변할 거라
생각할 수 있지만 전역 변수 a 를 바라봅니다.

이 설명을 다시하면

자바스크립트는 변수와 함수를 호출 시점에 선언하고
사용하는 것이 아닌

최초 로딩되는 순간 전역 지역 기준으로 바라보게
됩니다.

즉, 1 라인의 변수 a, 함수 funcA, funcB 는 전역 지역의
변수와 함수로

프로그램 실행 전 선언 단계에서 바라보게 됩니다.

하지만 funcA 는 내부에 var a 라고 지역변수를
선언했기 때문에

전역 변수 a 가 아닌 지역 변수 a 를 바라보게 되는
것이며

funcB 에서 11 라인의 console.log(a)는 로딩 때 이미
전역 변수 a 를 바라보고

funcA 내부의 var a 는 바라볼 수 없기에

funcA 에서 var a = 3;으로 한다고 해서 전역 변수
a 값이 재할당 되지는 않습니다.

전역 기준으로 선언이 되면 해당 전역 객체들을 기본적으로 바라보게 되고

함수 내 지역 변수가 선언되면 전역 기준으로 서로 내부는 들어갈 수 없게 됩니다.

```
1 var a = 1; //전역 지역의 전역변수
2
3 function funcA() { //전역 지역의 전역 함수
4     var a = 3;
5     console.log(a);
6     return 2;
7 }
8
9 function funcB() { //전역 지역의 전역 함수
10    var b = 4;
11    console.log(funcA());
12    console.log(b);
13    console.log(a);
14    return 2;
15 }
16
17 console.log(funcB());
```

CS

그럼 위 예제를 다시 보겠습니다.

funcB 를 호출했습니다.

11 라인으로 내부에서 전역의 funcA()를 호출했습니다.

내부에서는 로딩 때 이미 선언된 함수이기에 위로 바라보면서 funcA 를 호출할 수 있습니다.

funcA 는 지역 변수 var a 를 할당했기에 전역 변수 a 의 값 1 이 아닌 3 을 출력합니다.

이렇게 다른 함수의 내부 변수 값을 받을 수 있게 됩니다.

그리고 return 2 를 출력하고

12 라인에서 b 값 4 를 출력합니다.

11 라인에서 funcA 를 호출해서 funcA 의 지역 변수 a 에 3 으로 할당했지만

13 라인은 전역 변수 a 의 값 1 을 출력합니다.

위 소스를 실행하면 자바스크립트 엔진은 먼저

전역 변수 a 와 함수 funcA, funcB 를 정의합니다.

그리고 funcA 와 funcB 또는 전역 지역에서는 기본적으로 a 를 찾으면 전역 변수 a 를 바라봅니다.

이미 정의를 했기 때문이지요. (lexical scope or static scope)

하지만 함수 내부에서 같은 이름으로 지역 변수를 선언한다면 그 값을 바라보게 됩니다.

안에서 밖의 변수와 함수를 바라볼 수 있지만 반대로 접근은 할 수 없습니다.

렉시컬 스코프(lexical scope)를 풀어서
정의해보겠습니다.

변수나 함수가 전역 환경에서 선언이 되었다면 렉시컬
스코프(lexical scope)는 전역이 됩니다.

변수나 함수가 함수 내부처럼 지역 환경에 선언이
되었다면 해당 변수나 함수의 렉시컬 스코프(lexical
scope)는

해당 함수의 지역이 됩니다.

이렇게 자바스크립트의 렉시컬 스코프를 나뉘
정리해봤습니다.

잘못된 정보나 혹은 다른 추가 정보가 있다면
정정하겠습니다.

즐 코딩하세요.

이번 포스팅은 자바스크립트의 중요한 개념 중 하나인
클로저(closure)에 대해 알아보니다.

사실 클로저(closure)는 자바스크립트에서만 사용하는
개념은 아니고 함수형 프로그램에서 사용하는

중요한 부분입니다.

자바스크립트 클로저 (Closure) - tiBoy -

단순히 책을 보고 이해하기 어려운 부분이 있기에 좀 더 쉽게 설명해보려고 합니다.

그럼 클로저(closure)가 무엇인지 알아보도록 합니다.

사전적 의미는 폐쇄입니다.

우선 자바스크립트의 클로저(closure)를 이해하기 전 알아야 할 개념이 있습니다.

바로 렉시컬 스코프(lexical scope)입니다.

미리 읽어보고 오시면 좋겠네요.

자바스크립트는 함수의 스포크를 정할 때 어디서 실행했는지가 아닌 어디에 정의했는지에 따라서

상위 스코프를 정한다고 했습니다. 바로 이것을 렉시컬 스코프(lexical scope)라고 합니다.

그럼 클로저(closure)를 먼저 정의해보겠습니다.

어떤 함수가 있습니다.

그 함수 안에는 내부 함수가 존재합니다.

그 함수는 내부 함수의 상위 함수라고 부르겠습니다.

내부 함수는 상위 함수의 지역 변수에 접근할 수 있습니다.

하지만 상위 함수는 내부 함수의 지역 변수에 접근할 수 없습니다.

```
1 function outerFunc(){  
2   var a = 1;  
3  
4   function innerFunc(){  
5     var b = 2;  
6     console.log(a + b);  
7   }  
8
```

CS

```
9 | console.log(a + b);  
10| }  
11|  
12| outerFunc();  
13|
```

위 예제를 보면 9 라인에서 오류가 납니다.

innerFunc 의 b 에 접근할 수 없기 때문입니다.

클로저는 이렇게 내부 함수가 외부 함수의 지역 변수에 접근해서 사용을 합니다.

그리고 외부 함수보다 내부 함수의 수명이 길어야 합니다.

즉 외부 함수를 호출 한 뒤 외부 함수는 수명을 다해서 사라지더라도

내부 함수는 렉시컬 스코프를 유지하면서 외부 함수의 변수를 계속 참조하고 있어야 합니다.

또한 외부 함수의 지역 변수에 바로 접근을 할 수 없지만 클로저를 통해서

접근 및 사용이 가능하게 됩니다.

그리고 그 외부 함수의 지역 변수의 값을 유지해 줍니다.

다음 예제로 확인해 봅니다.

클로저를 만드는 방법 중 하나입니다.

```
1 function outFunc(){
2     var a = 1;
3
4     function inFunc(){
5         return a++;
6     }
7
8     return inFunc;
9 }
10
11 var testFunc = outFunc();
12
13 console.log(testFunc());
14 console.log(testFunc()); cs
```

먼저 외부 함수 outFunc 를 선언하고

지역 변수 a 를 설정합니다.

이 지역 변수 a 는 현재 외부 함수에서만 사용 가능합니다.

이제 내부 함수 inFunc 를 선언하고

렉시컬 스코프 범위의 외부 함수의 지역 변수인 a 에 ++연산을 해서 접근합니다.

그리고 외부 함수는 내부 함수 자체를 객체로 리턴합니다.

11 라인을 보면 외부 함수 outFunc()를 변수 testFunc 에 담았습니다.

이제 이 전역 변수는 `outFunc()`의 리턴된 내부 함수 `inFunc`를 참조하게 됩니다.

13 과 14 라인을 실행하면 `outFunc`의 지역 변수 `a`를 `++`로 연산한 값을 받게 됩니다.

초기화되지 않고 연결해서 1, 2가 출력됩니다.

바로 이 내부 함수가 클로저입니다.

전역 변수 `testFunc()`에 담기는 상황에 `outFunc()`는 `return`으로 내부 함수 `inFunc()`를 넘겨주고

수명을 다하게 됩니다.

그럼 지역 변수 `a`도 같이 사라지게 되겠지요?

하지만 내부 함수에서 외부 함수의 지역 변수인 `a`를 참조하고 있고

이 범위를 렉시컬 스코프라 하는데

내부 함수가 외부 함수의 렉시컬 스코프를 갖고 있는 겁니다.

그래서 외부 함수의 수명이 다 했더라도 외부 함수의 지역 변수가

클로저(내부 함수)에 의해 참조되고 있으니 사라지지 않고 유지가 되는 겁니다.

자바에서 클래스 내부의 `private` 로 선언된 변수를 `getter/setter` 로 접근하는 방식과 비슷해 보입니다.

값을 숨겨두고 정의된 함수에서만 그 값을 컨트롤할 수 있는 기능이 자바스크립트에서는 클로저로 가능하겠네요.

클로저(`closure`)의 의미에 대해 살펴보았습니다.

이제 클로저를 어떤 방식으로 만들고 사용하면 좋을지 고민할 시간이네요.

다음 포스팅에서 이 부분을 다뤄볼 생각입니다.

이번 포스팅은 자바스크립트에서 원시 타입 중 숫자에 대해 같이 알아보도록 하겠습니다.

자바스크립트
원시 타입
Number, Na
- tiBoy -

자바스크립트에서는 원시 타입과 객체 타입이 있습니다.

자바스크립트에서는 총 6 가지의 원시 타입이 있습니다.

ES6 에서 나타난 심벌타입은 다음에 따로 확인하겠습니다.

그래서 이번 포스팅에서는 5 가지의 원시 타입 중 숫자 타입에 대해 알아봅니다.

자바스크립트에서 숫자 타입(number)는 정수, 실수를 구분하지 않습니다.

그냥 하나의 number 타입으로 사용합니다.

```
1 var int = 100;
2 var minusInt = -100
3 var dbl = 200.25;
4
5 var bi = 0b100;
6 var otc = 00100;
7 var hex =0xa;
8 var hex2 =0x100;
9
10 console.log(int);
11 console.log(typeof int);
12 console.log(minusInt);
13 console.log(typeof minusInt);
14 console.log(dbl);
15 console.log(typeof dbl);
16
17 console.log(bi);
18 console.log(typeof bi);
19 console.log(otc);
```



```

20 console.log(typeof otc);
21 console.log(hex);
22 console.log(typeof hex);
23 console.log(hex2);

```

결과

```

1 100
2 number
3 -100
4 number
5 200.25
6 number
7 4
8 number
9 64
10 number
11 10
12 number
13 256

```

정수, 음의 정수, 실수, 2 진수, 8 진수, 16 진수
모두 typeof 로 확인한 결과 number 로
나옵니다.

다음은 Infinity 에 대해 알아보니다.

```

1 var a = 1000000000000000000;
2
3 console.log(a);
4 console.log(a ** 2);
5 console.log(a ** a);
6
7
8 console.log(a === Infinity);
9 console.log((a ** 2) === Infinity);

```

```
10 console.log((a ** a) === Infinity);
```

a 에 엄청 큰 숫자를 할당했습니다.

6 가지 로그를 출력했는데 어떤 결과가 나올까요?

```
1 1000000000000000000  
2 1e+32  
3 Infinity  
4 false  
5 false  
6 true
```

3 번 라인을 출력하면 a 값 그대로 출력됩니다.

4 번 라인은 너무 큰 값이 되어서 e 의 32 로 표현됩니다.

5 번 라인은 측정할 가치가 없을 정도로 너무 커서 Infinity(무한)으로 표기됩니다.

10 번 라인의 경우 Infinity 로 비교했을 때 true 가 됩니다.

```
1 var b = '3';
2
3 console.log(b * 3);
4 console.log(b * 'a');
5
6 console.log(typeof b);
7 console.log(typeof (b * 3));
8 console.log(typeof (b * 'a'));cs
```

위 예제를 보겠습니다.
b 에 문자 '3'을 할당했습니다.
결과를 보겠습니다.

```
1 9
2 NaN
3 string
4 number
5 number
```

cs

3 라인은 문자 '3'에 3 을 곱하니 결과 1 라인의 9 가 나왔습니다.

문자 '3'이 형변환 되었네요.

4 라인은 '3'에 'a'를 곱하니 NaN 이 나왔습니다.

NaN 은 Not a Number 의 약자로 숫자가 아니라는 의미입니다.

'3'의 타입을 확인하니 3 라인의 string 으로 출력됩니다.

'3' * 3 은 9 가 나오고 number 타입이며

'3' * 'a'는 NaN 이 나오는데 역시 number 타입입니다.

즉, NaN 은 number 라는 의미입니다. 잘 기억하세요.

지금까지 숫자 타입(number)에 대해 알아보았습니다.

자바스크립트에서는 숫자가 정수, 실수 구분 없이 사용되며

문자열로 된 숫자도 계산이 가능하고

숫자 형식으로 사용할 수 없으면 NaN 으로 나타납니다.

NaN 도 타입은 number 입니다.

코딩의 즐거움은 직접 만들어가는 겁니다.

즐코딩 하세요~

여러 가지 책을 보니 시간이 많이 부족하다는 생각이 드네요.

그래도 읽을 책이 많다는 건 행복한 일이 아닐까 싶네요.



자바스크립트
원시 타입
문자열 타입
- tiBoy -

오랜만에 자바스크립트 책을 펴 봅니다.

이제 하나씩 끝장을 내야지 째째 펴서 읽으니 기억이 가물거리네요.

이번 포스팅은 자바스크립트의 문자열 타입에 대해 알아봅니다.

보통 프로그래밍 언어는 문자열을 다룰 수 있습니다.

자바스크립트도 문자열을 당연히~ 다루겠지요?

어떻게 잘 다루면 되는지 같이 살펴보겠습니다.

문자열이란? 바로 여러분이 생각하는 바로 그 글자들입니다.


한글, 영어, 한자어 등등 다양한 언어들과

숫자, 기호 등 문자로 표현할 수 있는 모든 것일 말하는 것입니다.

유니코드 문자로 웬만한 문자는 모두
/Users/jongdeokkim/Desktop/ScriptES6_15.html 표현이
가능합니다. (UTF-16)

문자열 샘플을 같이 볼게요.

```
1 var str1 = "My name is Double quotation String\n";  
2 var str2 = 'My name is Single quotation String\n';  
3 var str3 = `My name is Backtick String`;  
4  
5 alert(str1 + str2 + str3);
```



문자열은 무엇으로 감싸줘야 합니다.

보통 Double Quotation (")을 사용하지만

자바스크립트는 무려 3 가지를 지원합니다.

",',`로 Double Quotation, Single Quotation, BackTick 이 세 가지입니다.

이렇게 문자열을 감싸지 않고 변수에 할당을 하면
그 문자열을 변수로 인식하고 할당되지 않았다면
오류가 발생합니다.

다음 예제로 확인해 봅니다.

```
JS index1.js > [❌] str3
1   var str1 =
2   var str2 =
3   var str3 =
4   var str4 = str;
5
6   alert(str1 + str2 + str3 + str4);
```

'str' 이름을 찾을 수 없습니다. 'str1'

any

Quick Fix... (Ctrl+.)

경고 라인이 나타나 마우스를 올려보니

'str' 이름을 찾을 수 없습니다. 'str1'을(를)
사용하시겠습니까?

라는 문구가 보입니다.

그렇다면 실제로 다음과 같은 출력을 문자열에 넣고
싶다면

어떻게 해야 할까요?

그는 어제 "난 엄청 배가 고프다고"라고 말했다.

또는,

그녀는 방금 '이 상황을 어찌 해결할까'라고 생각했다.

이렇게 문자열 안에 " 또는 '를 넣고 싶다면

다음과 같이 사용하면 된다.

```
1 var str1 = '그는 어제 "난 엄청 배가 고프다고" 라고 말  
2 했다.\n';  
3 var str2 = "그녀는 방금 '이 상황을 어찌 해결 할까?'라  
4 고 생각했다\n";  
alert(str1 + str2);
```

주의할 점은 "" 안에는 "로 ' 안에는 "'로 써야 오류가 발생하지 않는다.

당연한 거 아닌가?

사실 자바는 문자열을 String 이라는 객체 형태로 사용하지만

자바스크립트의 문자열은 원시형 타입입니다.

또한 값을 바꿀 수 없는 타입이다.

무슨 의미일까?

var vs let

기존에는 **var** 이란 예약어로 변수를 선언했었음.

이제는 구문적인 변수 영역 규칙(렉시컬 스코프)을 지원하도록 자바스크립트가 변했다.

예시를 통해 살펴보자.

```
> var x = "자바스크립트"

if(true){
  var x = "바꿔볼까?"
  console.log(x)
}

console.log(x)
바꿔볼까?
바꿔볼까?
```

기존 var 사용시

다른 언어를 사용했었던 사람이라면 신기할 수 있다.

두 번 if 문 안에서 선언한 x 가 결국은 global 변수와 같다는 이야기이다.

이제 **let** 으로 변수를 선언해보자.

```
> var x = "자바스크립트"

if(true){
  let x = "바꿔볼까?"
  console.log(x)
}
```

```
console.log(x)
```

```
바꿔볼까?
```

```
자바스크립트
```

let 으로 선언

아하! 뭐가 다른지 보인다.

이 코드를 통해 렉시컬 스코프를 지원한다는 의미가 무슨 의미인지 확실히 이해할 수 있다.

```
> let x = "자바스크립트"

if(true){
  let x = "바꿔볼까?"
  console.log(x)
}

console.log(x)
```

✖ Uncaught SyntaxError: Identifier 'x' has already been declared

그렇다면 이 코드가 왜 오류가 나는지도 쉽게 이해할 수 있다.

(내가 기존에 공부했던 자바스크립트가 정말 오래전꺼였구나.. 이제야 조금씩 윤곽이 잡히네 ㅎㅎ)

let 을 알려준 갓갓 형님.. 너무 감사합니다!!)

템플릿 문자열

기존 자바스크립트에서 문자열을 만들 때에는 '+' 기호를 통해 만들어야 했다.

다음 두 출력 코드를 보자.

```
var lastname = "Kim"  
var firstname = "Tae"  
var middlename = "Hyun"
```

```
console.log(lastname + ", " + firstname + " " +  
middlename) // 기존
```

```
console.log(`${lastname}, ${firstname} ${middlename}`)  
// ES6
```

(참고로 **ES6**의 템플릿 문자열을 사용하려면 ``` (백틱) 기호를 사용해야 해요! 백틱은 물결 문자와 같은 키보드 칸!)

```
> var lastname = "Kim"  
   var firstname = "Tae"  
   var middlename = "Hyun"  
  
   console.log(lastname + ", " + firstname + " " + middlename)  
  
   console.log(`${lastname}, ${firstname} ${middlename}`)
```

Kim, Tae Hyun

VM863:5

Kim, Tae Hyun

VM863:7

결괏값은 똑같다.

하지만 아래 **ES6** 코드가 훨씬 훨씬 좋다.

체감이 잘 안될 것 같은데 처음엔 나도 그랬다.

다음 코드를 보면 감탄할 것이다.

```
> console.log('기존의 자바스크립트  
다음줄에서 출력이 되기를..')
```

✖ Uncaught SyntaxError: Invalid or unexpected token

기존 방식대로 문자열을 만든다면 무조건 에러가 나는 코드이다.

하지만! B. U. T

```
> console.log(`새로운 자바스크립트  
다음줄에서 출력이 되기를..`)
```

```
새로운 자바스크립트  
다음줄에서 출력이 되기를..
```

???

이제 좀 느껴지는가?

두 코드의 차이는 '(작은따옴표)를 찍어주었느냐 `(백틱)를 찍어주었느냐 차이뿐인데..

지금이야 간단한 두 줄이지만 예를 들어 긴 메뉴판을 입력해야 된다고 가정해보자.

템플릿 문자열이 얼마나 편한지 상상할 수 있다.

이름부터 '템플릿'이라는 건 **C++**이나 **Java** 를 공부했던 사람이라면 어느 정도 예상은 했을 것이다.

그만큼 어렵지 않으나 굉장히 유용한 기술인 것 같다.

(개인적으로도 코드를 짤 때 굉장히 불편했던 기억이 있어서 더욱 유용한 느낌이다.)

화살표 함수

이 책을 공부하기 전에 Javascript 에 대해서 많은 사전 조사?를 했었는데 그때마다 등장했던 것이 화살표 함수였다.

↓

이 화살표 함수는 ES6 에 새롭게 추가된 기능이다.

이를 사용하면 **function** 키워드 없이도 함수를 만들 수 있고 **return** 을 사용하지 않아도 알아서 반환 값을 반환한다.

아래의 코드는 일반적인 함수 구문이다.

```
var funobject = function(name, cost) {  
    return `${name} 고객님의~ ${cost} 결제되었습니다!`  
}
```

이 함수를 화살표 함수로 바꿔보면 다음과 같다.

```
var funobject = (name, cost) => `${name} 고객님의~  
${cost} 결제되었습니다!`
```

```
console.log(funobject("전우치",1000));
```

이 두 함수는 호출 시 동일한 결과값을 출력한다.

궁금하다면.. 직접 해보는 걸로^^

두 코드를 봤을 때 화살표 함수로 짠 코드가 훨씬 간결하다.

또한 반환 값이 무엇인지 '='>' 기호로 알려주고 있기 때문에 위에서 언급했듯이 **return** 을 사용하지 않아도 된다.

다음으로 아래 코드를 한번 실행시켜보자.

```
var func1 = {  
  prop: ["안녕", "Hi", "Bonjour"],  
  country: ["한국", "미국", "프랑스"],  
  print: function(delay = 1000){  
    setTimeout(function() {  
      for(let i=0;i<3;i++){  
        console.log(`${this.prop[i]}은  
${this.country[i]}의 인사말이에요!`)  
      }  
    }, delay)  
  }  
}  
  
func1.print()
```



```

> var func1 = {
  prop: ["안녕", "Hi", "Bonjour"],
  country: ["한국", "미국", "프랑스"],
  print: function(delay = 1000){
    setTimeout(function() {
      for(let i=0;i<3;i++){
        console.log(`${this.prop[i]}은 ${this.country[i]}의 인사말
미에요!`)
      }
    }, delay)
  }
}
< undefined
> func1.print()
< undefined
✖ ▶ Uncaught TypeError: Cannot read property '0' of undefined VM445:7
   at <anonymous>:7:41

```

오류가 생긴다.

이유는 `this.prop [i]`에서 `prop` 를 인식하지 못하기 때문이다.

먼저 위의 코드를 잘 읽어보아야 한다. (내가 대충 읽었다가 이해하는데 시간을 많이 잡아먹었다..)

`func1` 의 `print` 프로퍼티를 보면 이 객체는 함수이구나를 알 수 있다.

이때 그냥 함수가 아니라 `setTimeout()`를 통해 시간을 지연시킨 뒤 출력하는 함수임을 알 수 있다.

이때 `this` 를 호출한 곳은 `print` 객체의 `setTimeout` 함수 안에서 호출된다.

자바스크립트를 처음 배우는 대부분의 사람들이 생각한 `this` 란 아마 `func1` 의 객체 영역을 의미했을 것이다.

그러나 정확하게 말하면 저 부분의 `this` 는 `func1` 이 아닌 `window` 객체를 의미한다.

우리가 의도한 대로 `func1` 객체를 `this` 가 가리키게 하려면 어렵지 않다.

화살표 함수를 사용하면 된다!

```
var func2 = {  
  prop: ["안녕", "Hi", "Bonjur"],  
  country: ["한국", "미국", "프랑스"],  
  print: function(delay = 1000){  
    setTimeout(() => {  
      for(let i=0;i<3;i++){
```

```

        console.log(`${this.prop[i]}은
        ${this.country[i]}의 인사말이에요!`)
    }
    }, delay)
}
}

> var func2 = {
  prop: ["안녕", "Hi", "Bonjour"],
  country: ["한국", "미국", "프랑스"],
  print: function(delay = 1000){
    setTimeout(() => {
      for(let i=0;i<3;i++){
        console.log(`${this.prop[i]}은 ${this.country[i]}의 인사말
        미에요!`)
      }
    }, delay)
  }
}

< undefined
> func2.print()
< undefined
안녕은 한국의 인사말이에요! VM541:7
Hi은 미국의 인사말이에요! VM541:7
Bonjour은 프랑스의 인사말이에요! VM541:7

```

두 코드의 차이점은 `function` 을 `()=>`로 바꿔준 것뿐이다.

사실 이렇게 `this` 의 객체를 지정하는 데에 혼란이 있는 것에 대해 더그 크록포드(자바스크립트 핵심 가이드 저서)는 자바스크립트의 설계 오류라고 지적했다고 한다.

위의 내용을 정확히 이해하려면 자바스크립트의 `this` 바인딩을 공부해야 한다.

이전 포스팅에서 간단하게 다뤘던 기억이 있는데 벌써 가물가물 한 거 보니.. 다음에 다시 공부해서 포스팅해야겠다.

ES6 객체와 배열

구조 분해.. 분명 공부했고 클론 코딩을 진행하면서 정말 하루에 한 번씩 마주쳤던 개념이다.

그런데 솔직히 아직도 익숙하지가 않다..

developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment

구조 분해 할당 - JavaScript | MDN

구조 분해 할당 구문은 배열이나 객체의 속성을
해체하여 그 값을 개별 변수에 담을 수 있게 하는
JavaScript 표현식입니다. The source for this interactive
example is stored in a GitHub repository. If you'd like to
cont
developer.mozilla.org

구조 분해 할당에 대한 자세한 개념은 위의 링크를
통해 공부하는 게 가장 좋을 것이다.

객체 리터럴 개선

객체 리터럴 개념은 구조 분해 할당과 반대되는
경우이다.

객체 리터럴은 아래의 코드로 쉽게 이해 가능하다.

```
var name = "김태현"  
var code = "17"  
var major = "소프트웨어학"
```

```
var print = function() {  
    console.log(`${this.name}은 ${this.major}과  
    ${this.code}학번 입니다.`)  
}
```

```
var student = {name, code, major, print}
```

```
student.print()
```

```
> var name = "김태현"  
   var code = "17"  
   var major = "소프트웨어학"  
  
   var print = function() {  
       console.log(`${this.name}은 ${this.major}과 ${this.code}학번 입니다.`)  
   }  
  
   var student = {name, code, major, print}  
  
   student.print()
```

김태현은 소프트웨어학과 17학번 입니다.

VM1938:6

말 그대로 변수들을 하나로 합쳐서 객체로 만들어 준다.

스프레드 연산자

스프레드 연산자 '...'으로 표기하며 여러 가지 역할로 사용이 가능하다.

1. 배열 합치기

```
var atoc = ["A", "B", "C"]
```

```
var dtoe = ["D", "E"]
```

```
var atoe = [...atoc, ...dtoe]
```

```
console.log(atoe)
```

```
> var atoc = ["A", "B", "C"]
```

```
var dtoe = ["D", "E"]
```

```
var atoe = [...atoc, ...dtoe]
```

```
console.log(atoe)
```

```
▶ (5) ["A", "B", "C", "D", "E"]
```

2. 스프레드 연산자는 원본 배열을 복사한다(원본 배열을 건드리지 않는다)

```
> var atoc = ["A", "B", "C"]

var rev_atoc = atoc.reverse()
console.log(atoc)

▶ (3) ["C", "B", "A"]

< undefined

> var atoc = ["A", "B", "C"]

var rev_atoc = [...atoc].reverse()
console.log(atoc)

▶ (3) ["A", "B", "C"]

< undefined
```

원본 배열을 건드리지 않는다는 게 얼마나 좋은 건지는 코딩 좀 해봤다 하면 당연히 알겠죠..?

이 외에도 다양하게 응용이 가능하다.

필요할 때마다 찾아보면서 사용하면 좋을 것 같다.

클래스

이전까지 배워왔던 **C++**이나 **Java** 에서 클래스란 가장 중요하기도 하면서 나를 힘들게 했던 개념이다.

근데 처음 자바 스크립트를 공부할 때에 클래스란 개념이 자바 스크립트에는 존재하지 않는다고 해서 많이 놀랐다.

(놀람 반 기쁨 반 ㅎㅎ)

근데 **ES6** 에는 클래스 선언이 추가되었다고 한다.

물론 자바 스크립트의 작동 방식이 변한 것이 아니라 클래스란 선언을 추가함으로써 전통적인? 객체 지향을 사용해온 개발자들에게 이해하기 쉽게 바뀌었다고 한다.

다음 코드를 보자(Demo)

```
class Student {  
  constructor(name, age){  
    this.name = name  
    this.age = age  
  }  
  print(){  
    console.log(`${this.name} 학생의 나이는  
    ${this.age}살 입니다.`)  
  }  
}
```

```
const Taehyun = new Student("김태현", 23)
```

```
Taehyun.print()
```

너무너무 익숙한 코드이다.

지금까지 공부한 자바 스크립트 코드 중 가장 이해하기 쉬운 코드였다.

살짝 헤맨 부분은 **const** 로 선언해야 한다는 점..?

(왜 **const** 로 선언해야 하는지는 아직 찾지 못했어요ㅠㅠ)

class 이기 때문에, 당연히 상속 (**extends**)가 가능하고, **super** 를 사용할 수도 있다.

(이게 무슨 소린지 이해가 안 된다면.. 자바 강의를 꼭 듣기를 나름 유익함)

jQuery 소개와 데모

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
  <title></title>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/j
query.min.js"></script>
  <script type="text/javascript">
    $(document).ready(function () {
      $("h2").each(function () {
        this.style.color = "RED";
      });
    });
  </script>
</head>
<body>
  <h1>aaaa</h1>
  <h2>bbb</h2>
  <h1>ccc</h1>
  <h2>aaaddda</h2>

</body>
</html>

```

jQuery 로 입력된 값을 출력하는 데모

```

<!DOCTYPE html>
<html>
<head>

```

```
<title></title>
```

```
< <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

```
</head>
```

```
<body>
```

```
  <form>
```

```
    <p><label>Write name: <input type="text"
id="NameBox" /></label></p>
```

```
    <input type="button" id="submit" value="click to
submit" />
```

```
  </form>
```

```
  <div id="thankYouArea"></div>
```

```
  <script type="text/javascript">
```

```
    $(document).ready(function () {
```

```
      $("#submit").click(
```

```
        function () {
```

```
          var userName = $("#NameBox").val();
```

```
          $("#thankYouArea").replaceWith("<p>
```

```
Thank you " + userName +
```

```
"</p>");
```

```
        })
```

```
      });
```

```
    </script>
```

```
</body>
```

```
</html>
```

[jQuery 란?]

: **JavaScript** 를 편리하게 사용할 수 있도록
해주는 **Java Script Library** 이다.

- javascript 에 **jQuery** 를 쓰면 **웹 호환성**이 좋아진다!

jQuery 사용을 선언하기

```
<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/  
3.5.1/jquery.min.js"></script>
```

<head>와 </head> 사이에 삽입해준다.

jQuery 시작 하기

```
jQuery(document).ready(function(){  
    // process..  
});
```

```
$(document).ready(function(){  
    // process..  
});
```

\$(document).ready(callback 함수) = **jQuery** 와 **\$**는
같은 의미이다

document 가 준비된 시점에 (=메모리에 로딩 시에)
익명함수가 실행되고,

이 익명함수는 현 페이지에서 필요한 이벤트 행위를 일괄적으로 등록한다.

- jQuery 이벤트 메서드 중 하나

- 문서의 DOM 요소들을 조작 할 수 있는 시점에서 호출
- javascript 의 load 이벤트와 비슷

jQuery 기본 문법 -Selector (선택자)

- `$('a')` - page 내 모든 `<a>`
- `$('div a')` - page 내 `<div>` 하위에 있는 `<a>`
- `$('#test')` - id 가 test 인 태그
- `$('.btn')` - class 가 btn 인 모든 태그
- `$('tr:odd')` - `<tr>` 태그들 중 홀수번째들
- `$('tr:last tr:last')` - 문서내의 마지막 `<tr>`
- `$('b:contains('hi'))` - hi 를 content 로 가진 b 태그
- `$('div:has('ul')` - `` 을 가진 `<div>` 태그
- `$('input:checkbox')` - input 태그중 checkbox
- `$('td nth-child(2n))` - 2 의 배수 번째 `<td>`

Content/DOM 변경 및 조회 메소드

- **html()**
 - 선택요소의 html 내용을 가져옴. innerHTML 과 동일
- **html(value)**
 - 선택요소에 인수로 받은 value 를 넣는다.
 - value 내에 html 태그가 있는 경우 태그로 들어간다.
- **text()**
 - 선택요소 내의 html 태그 요소를 제외한 내용을 가져옴.
 - innerText innerText 와 동일.
- **text(value)**
 - 선택요소 내에 인수로 받은 value 를 넣는다.
 - value 내의 html 태그도 text 로 들어간다.
- **val()**
 - input 태그의 value 값을 조회
- **val(value)**
 - 인수로 받은 value 를 input 태그의 value 값을 설정

사용 예시

[목차] : 클릭하여 목차 이동

1. jQuery ready & 이벤트 등록 기본 - Tag

2. jQuery ready & 이벤트 등록 기본 - Input

3. Class Selector 이용 + Confirm 적용

4. 특정대상 골라서 event 주기 - id

5. class tag Selector 적용

6. Selector 대상이 여러 개일 경우

7. 이미지 조정하기

1. jQuery ready & 이벤트 등록 기본 - Tag

\$("#tag")로 이벤트 등록

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>jQuery basic</title>
<meta name="viewport" content="width=device-width,
initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
4.1/css/bootstrap.min.css">
```



```

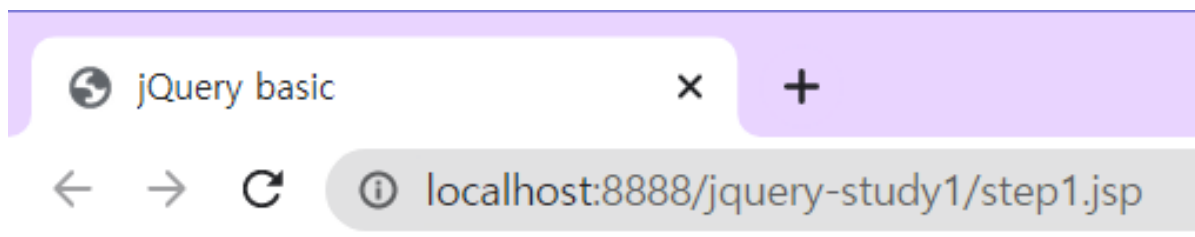
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/
3.5.1/jquery.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4
.1/js/bootstrap.min.js"></script>
<script type="text/javascript">
    //jQuery의 ready 함수는 document가 준비되는
    시점에 실행된다.
    /*   jQuery(document).ready(function() {
            alert("jquery");
        }); */
    // jQuery와 $는 같은 표현이다
    $(document).ready(function() {
        //alert("jquery"+
document.getElementById("ts"));

        //ready 시점에 현 페이지의 이벤트를 등
        록한다.
        // 이벤트 : a tag를 클릭했을 때, 자신의
        텍스트를 alert
        $("a").click(function () {
            alert($(this).text());
        })
    });
</script>
</head>
<body>
<div class="container">

```

```
<h3>jQuery Basic</h3> <br>
<span id="ts">test span</span>

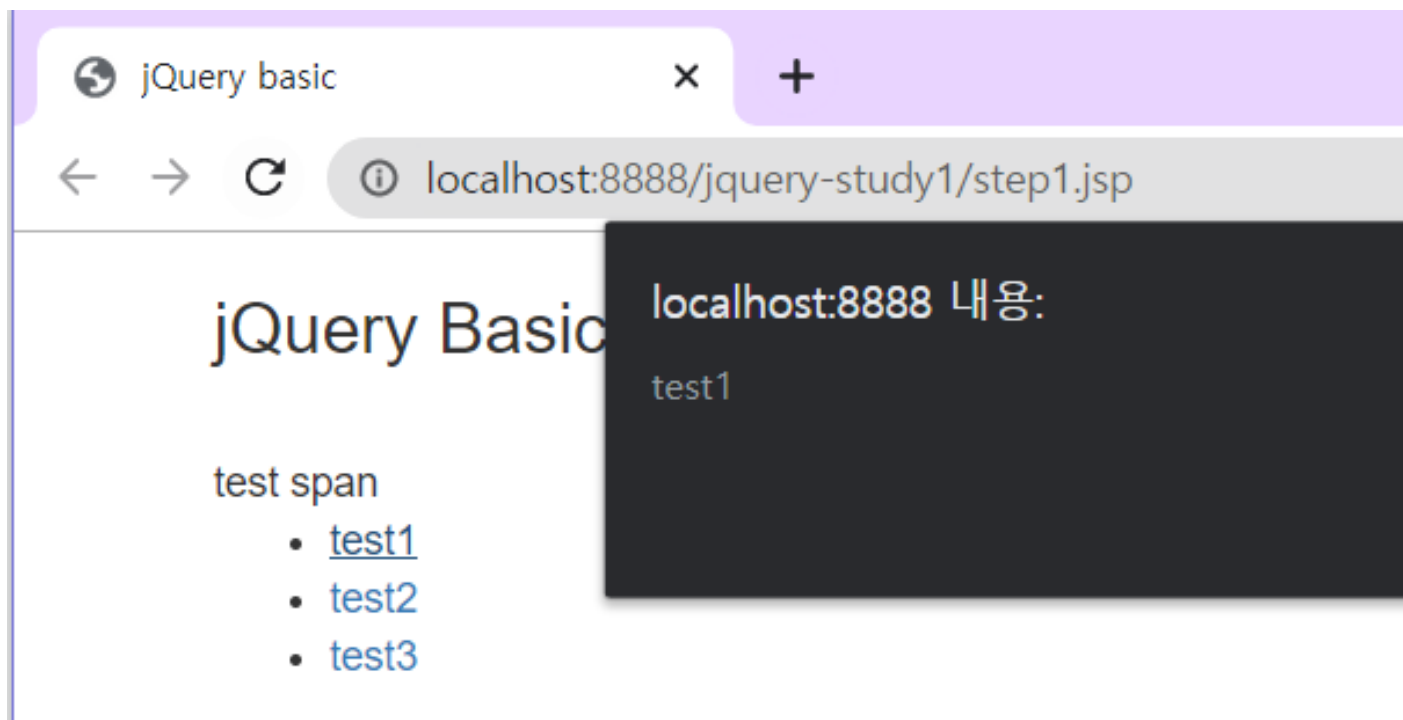
<ul>
  <li><a href="#">test1</a>
  <li><a href="#">test2</a>
  <li><a href="#">test3</a>
</ul>
<!-- <script type="text/javascript">
      alert(document.getElementById("ts"));
    </script>
-->
</div>
</body>
</html>
```



jQuery Basic

test span

- test1
- test2
- test3



2. jQuery ready & 이벤트 등록 기본 - Input

`$(":type")`로 이벤트 등록

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>jQuery Basic</title>
<meta name="viewport" content="width=device-width,
initial-scale=1">
```

```

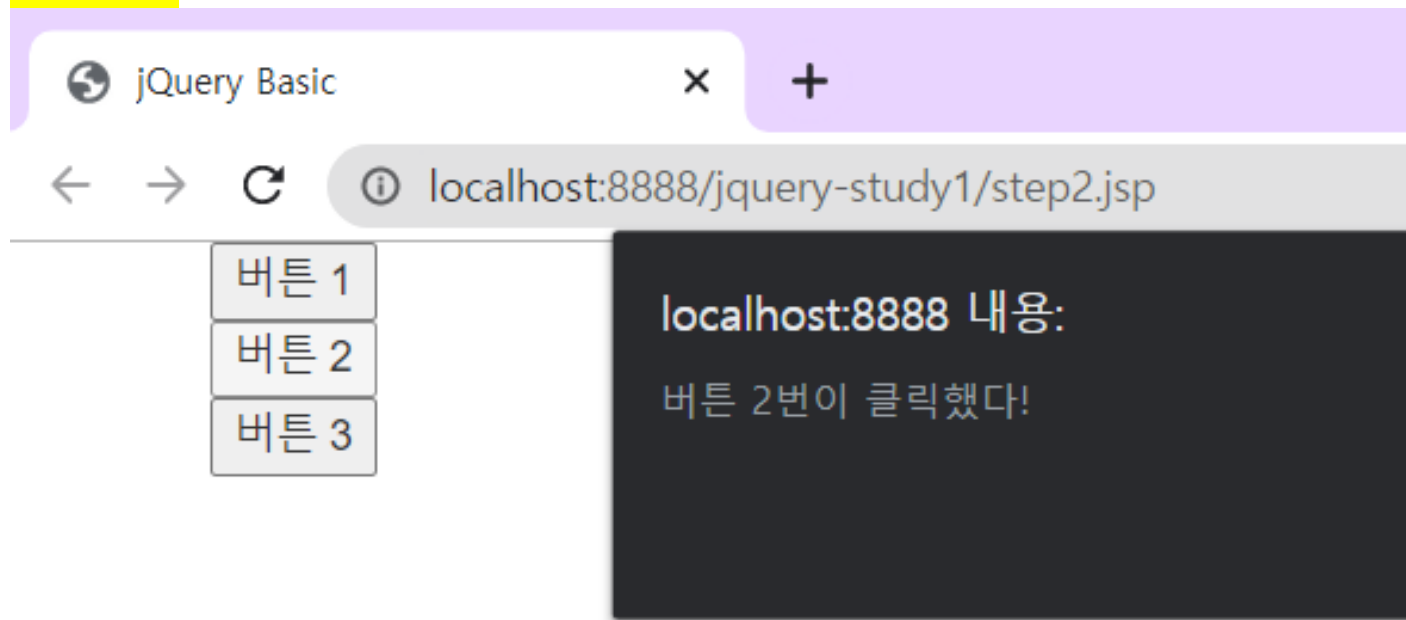
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
4.1/css/bootstrap.min.css">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/
3.5.1/jquery.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4
.1/js/bootstrap.min.js"></script>
<script type="text/javascript">
    //document 가 준비된 시점에 (=메모리에 로딩
    시에) 익명함수가 실행되고,
    //이 익명함수는 현 페이지에서 필요한 이벤트
    행위를 일괄적으로 등록한다.
    $(document).ready(function() {
        //이벤트 등록
        // : $(:button) -> jQuery 필터로 선택,
        type=button 을 선택한다.
        $("button").click(function() {
            alert($(this).val() + "번이 클릭했
다!");
        });
    });
</script>
</head>
<body>
<div class="container">
    <input type="button" value="버튼 1"><br>
    <input type="button" value="버튼 2"><br>

```

```

        <input type="button" value="버튼 3"><br>
    </div>
</body>
</html>

```



3. Class Seletor 이용 + Confirm 적용

`$(".class")`로 이벤트 등록

: 다음과 네이버는 이벤트 적용이 되고, 구글은 바로 이동이된다.

```

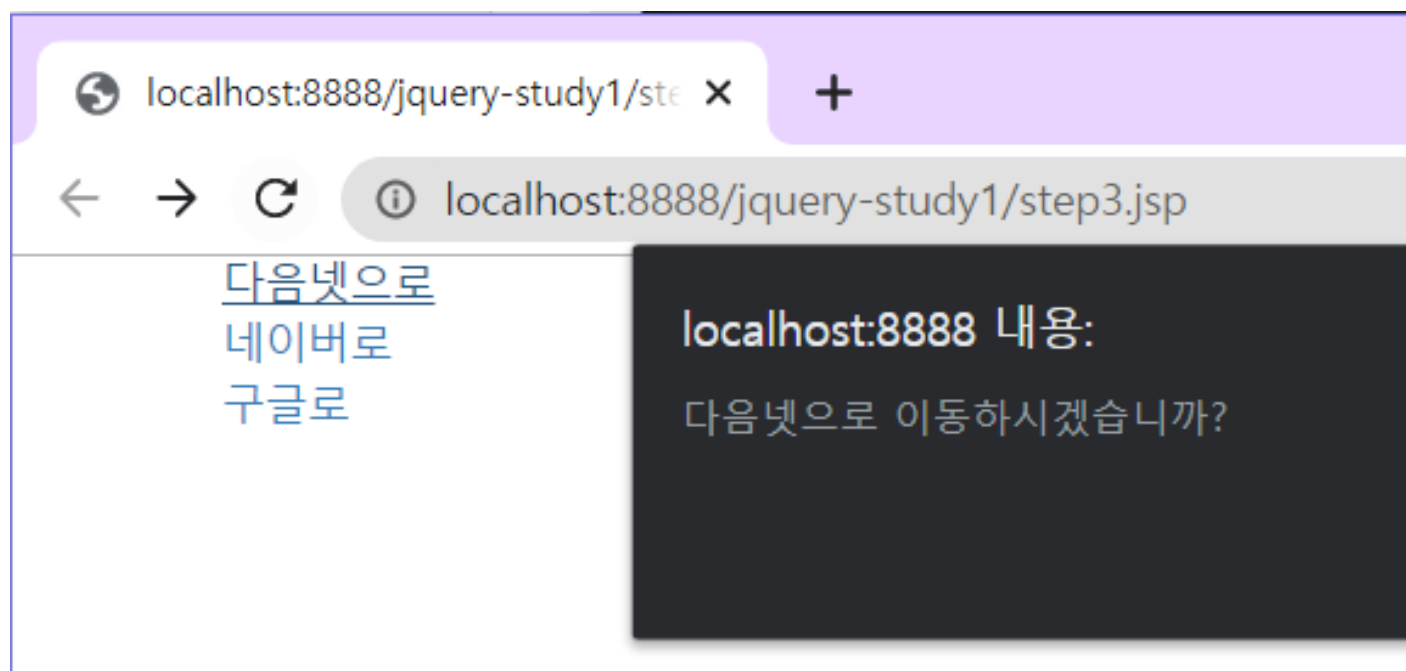
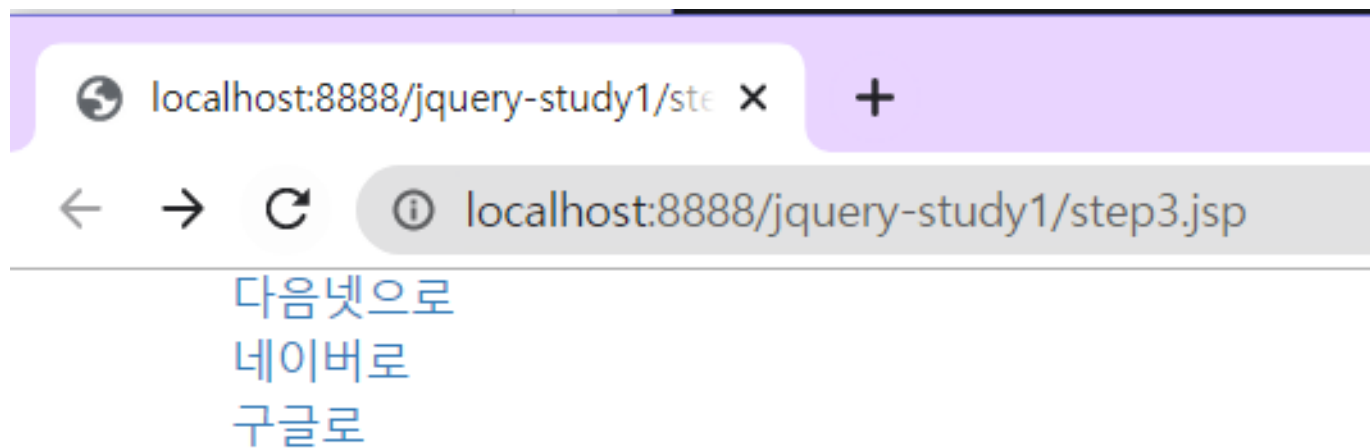
<%@ page language="java" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>

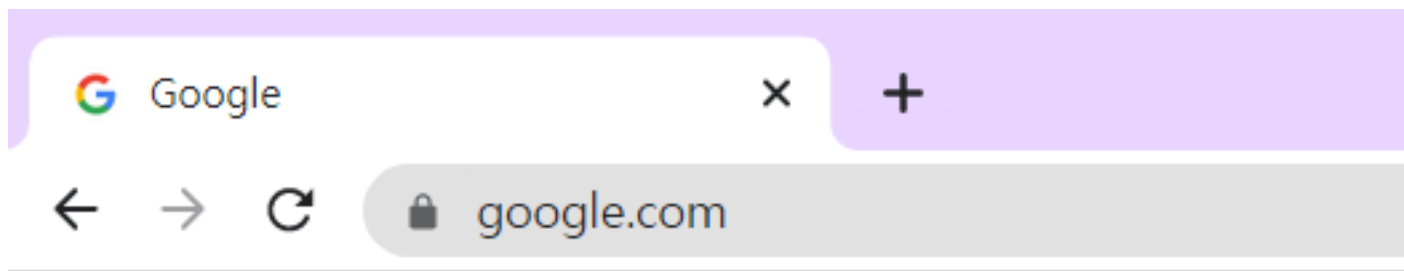
```

```
<head>
<meta charset="UTF-8">
<title></title>
<meta name="viewport" content="width=device-width,
initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
4.1/css/bootstrap.min.css">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/
3.5.1/jquery.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4
.1/js/bootstrap.min.js"></script>
<script>
    $(document).ready(function() {
        $(".ct").click(function () {
            //alert("클릭");

            //취소누르면 return
            return confirm($(".this").text()+" 이
동하시겠습니까?");
        });
    });
</script>
</head>
<body>
<div class="container">
```

```
<a href="http://daum.net" class="ct">다음넷으  
로</a><br>  
<a href="http://naver.com" class="ct">네이버  
로</a><br>  
<a href="http://google.com">구글로</a><br>  
</div>  
</body>  
</html>
```





4. 특정대상 골라서 event 주기 - id

`$("#id")`로 이벤트 등록

> text 적용 시

```
$("#id").text($(this).val());
```

> html tag 적용 시

```
$("#id").html("<html tag>" + $(this).val() + "</html tag>");
```

```
<%@ page language="java" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>jQuery Basic</title>
<meta name="viewport" content="width=device-width,
initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
4.1/css/bootstrap.min.css">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/
3.5.1/jquery.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4
.1/js/bootstrap.min.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
```

//id 가 btn2 인 대상을 선택해서, 이벤트 적용

```
$("#btn2").click(function() {  
    alert($(this).val());  
});
```

//id 가 btn3 인 대상을 선택해서, 이벤트 적용

// : 자신의 value 를 영역에 출력

```
$("#btn3").click(function() {  
    //alert($(this).val());
```

```
//$("#resultView").text($(this).val());
```

//innerHTML 형식으로 html tag 적용 : html()

```
$("#resultView").html("<font  
color=blue>" + $(this).val() + "</font>");  
});
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<div class="container">
```

```
<br><br>
```

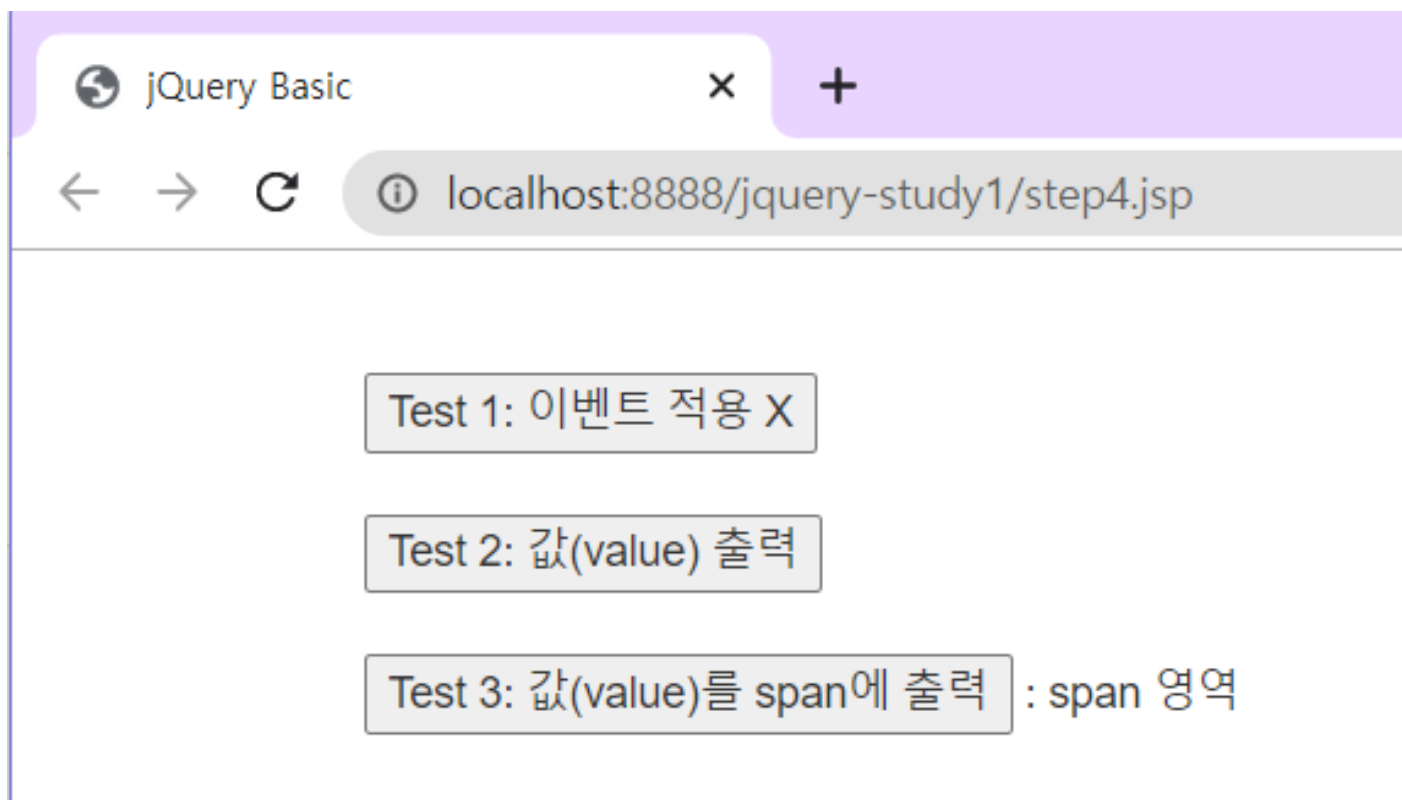
```
<input type="button" value="Test 1: 이벤트 적용 X"><br><br>
```

```

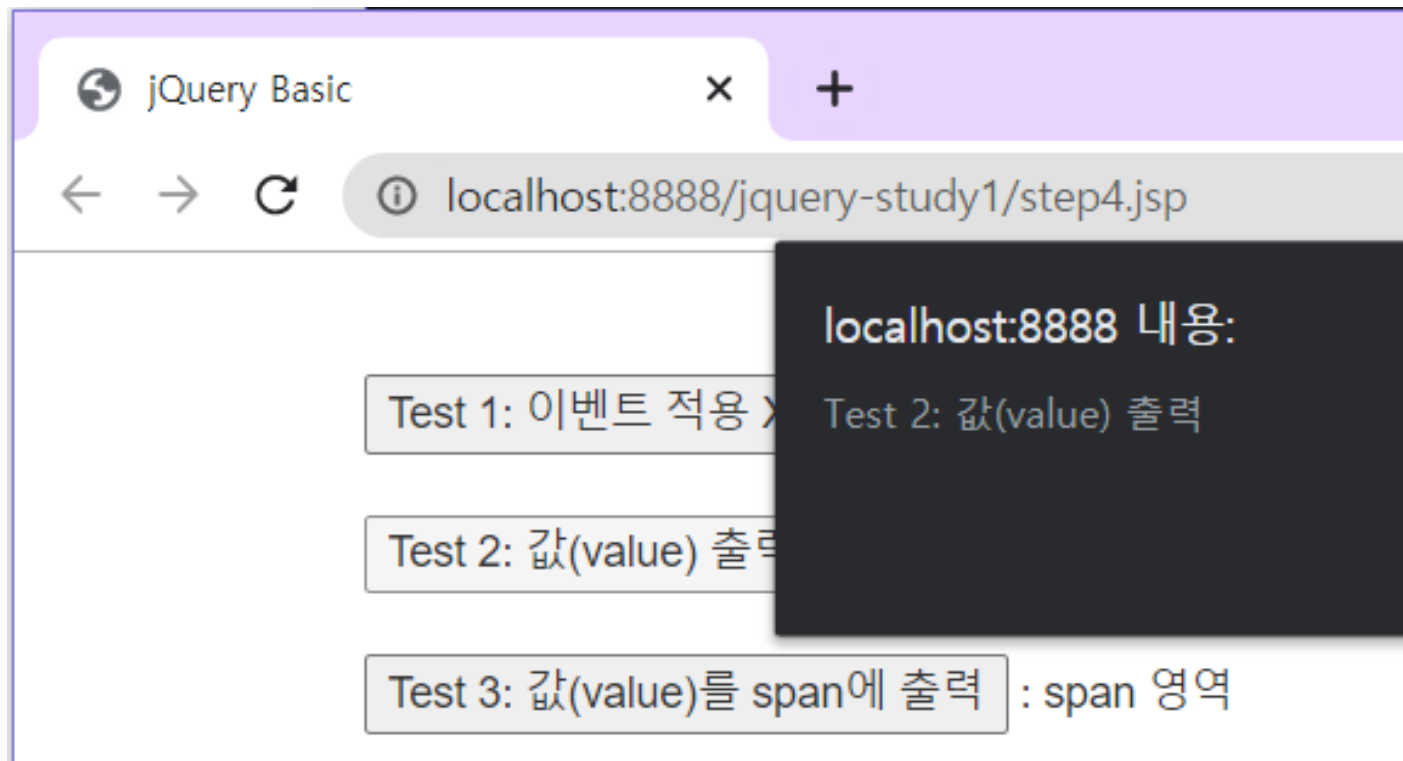
        <input type="button" value="Test 2: 값(value)
출력" id="btn2"><br><br>
        <input type="button" value="Test 3: 값(value)
를 span 에 출력" id="btn3"> :
    <span id="resultView">span 영역</span>
</div>
</body>
</html>

```

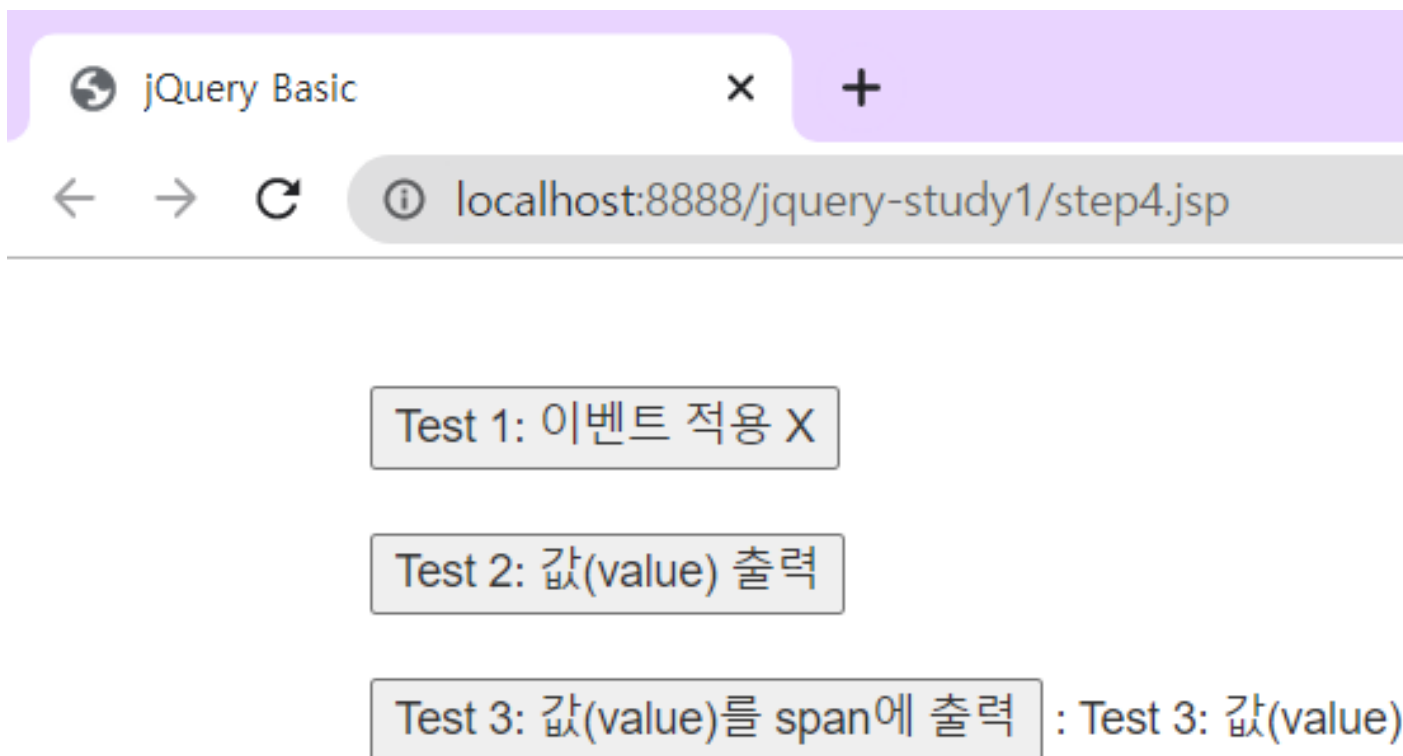
> 1 번 클릭하면, 아무 이벤트 X



> 2 번 클릭하면 alert 이 뜬



> 3 번 클릭하면 span 에 내용 나옴



5. class tag Selector 적용

`$(".class tag")`로 이벤트 등록

> class = food 의 li tag 에만 이벤트 적용

```
<%@ page language="java" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
    .a{
        background-color: yellow;
    }
    .b{
        background-color: lime;
    }
</style>
<title>jQuery CSS 선택</title>
<meta name="viewport" content="width=device-width,
initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
4.1/css/bootstrap.min.css">
```

```

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/
3.5.1/jquery.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4
.1/js/bootstrap.min.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        // food class 이하의 li 태그들을 대상으
로
        // 이벤트를 적용
        $(".food li").click(function() {
            alert($(this).text());
        });
    });
</script>
</head>
<body>
<div class="container">
<ol class="a food">
    <li>바나나</li>
    <li>사과</li>
    <li>포도</li>
</ol>

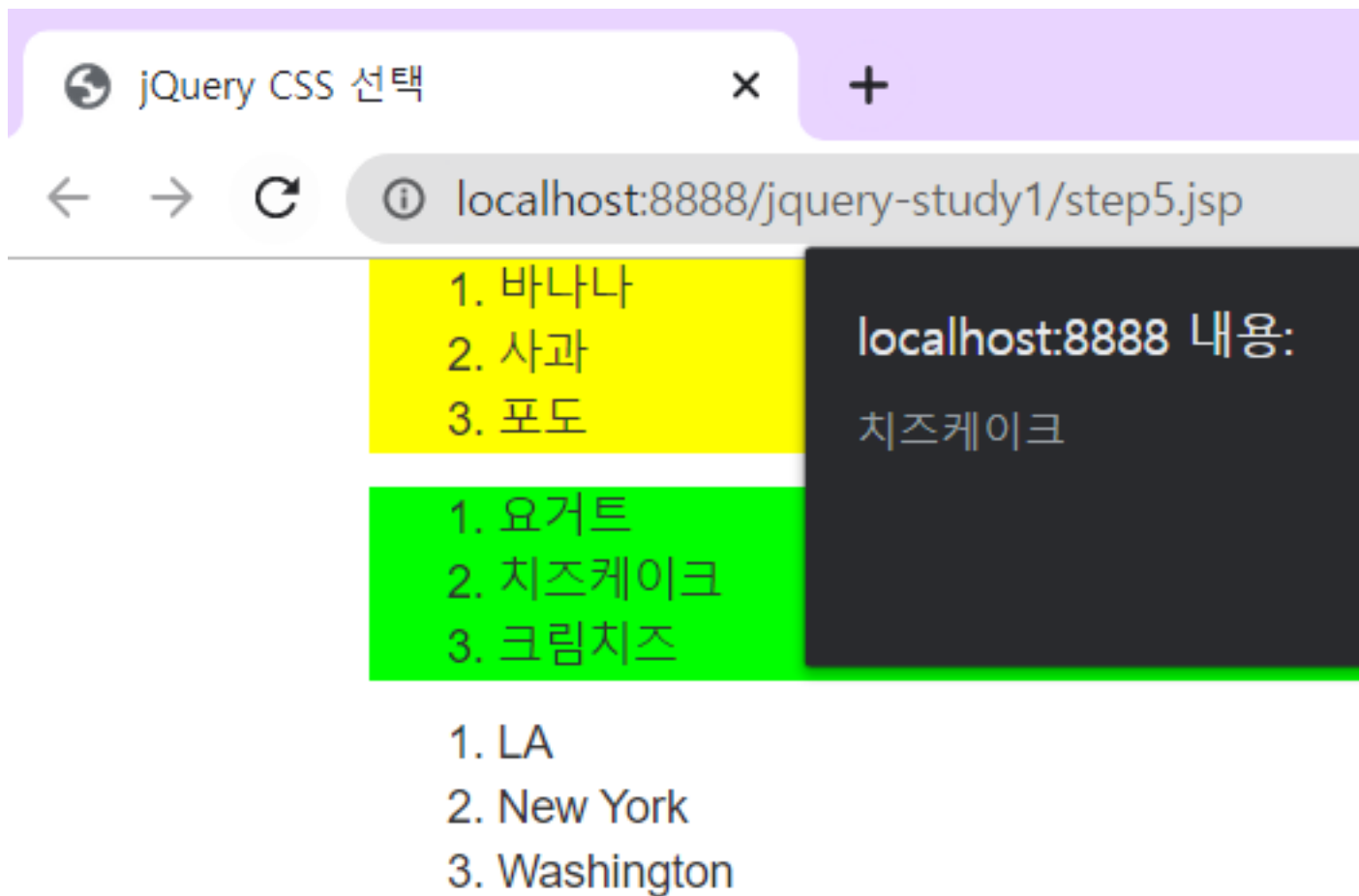
<ol class="food b">
    <li>요거트</li>

```

```
    <li>치즈케이크</li>  
    <li>크림치즈</li>  
</ol>
```

```
<ol>  
    <li>LA</li>  
    <li>New York</li>  
    <li>Washington</li>  
</ol>  
</div>  
</body>  
</html>
```

food class 에 속하는 **li** 를 클릭했을 때만, **alert** 반응이 있다.



6. Selector 대상이 여러 개일 경우

`$("selector, selector .. ")`로 이벤트 등록

> 1. id 가 "testBtn"일 경우, 클릭하면 자신의 value 를 alert 으로 출력

> 2. class 가 "student"인 경우, 클릭하면 id 가 result 인 영역에 자신의 text 출력

```
<%@ page language="java" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title></title>
<meta name="viewport" content="width=device-width,
initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
4.1/css/bootstrap.min.css">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/
3.5.1/jquery.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4
.1/js/bootstrap.min.js"></script>
<script type="text/javascript">
    $(document).ready(function () {

        //1. id가 testBtn 일 경우, 클릭하면 자
        신의 value 를 alert 으로 출력
        $("#testBtn").click(function() {
            alert($(this).val());
        });
    });
}
```

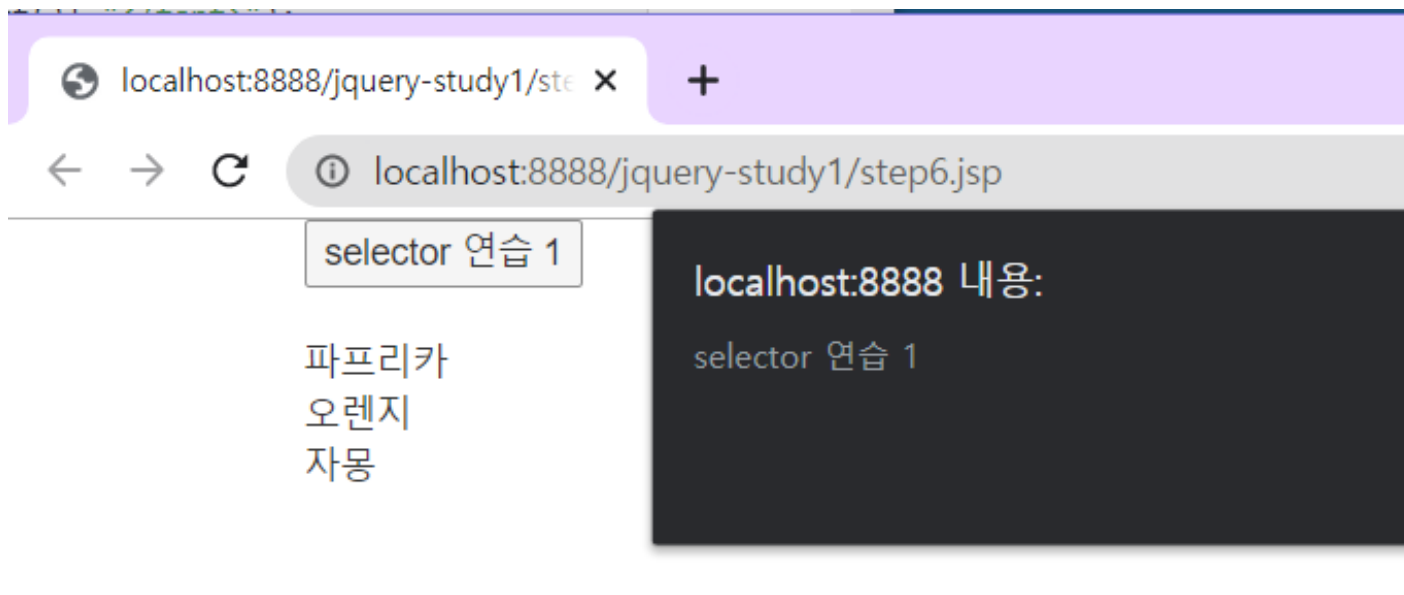
```

        //2. class 가 "student"인 경우, 클릭하면
        id가 result 인 영역에 자신의 text 출력
        // 셀렉터를 , 로 이어서 둘 다 출력되
        도록 함
        $(".student, .teacher").click(function()
        {
            //alert($(this).text());
            $("#result").html("<font
color=blue>" + $(this).text() + "</font>");
        });
    });
</script>
</head>
<body>
<div class="container">
    <input type="button" value="selector 연습 1"
id="testBtn"><br><br>

    <div class="student">파프리카</div>
    <div class="teacher">오렌지</div>
    <div class="student">자몽</div><br>
    <hr>
    <div id="result"></div>
</div>
</body>
</html>

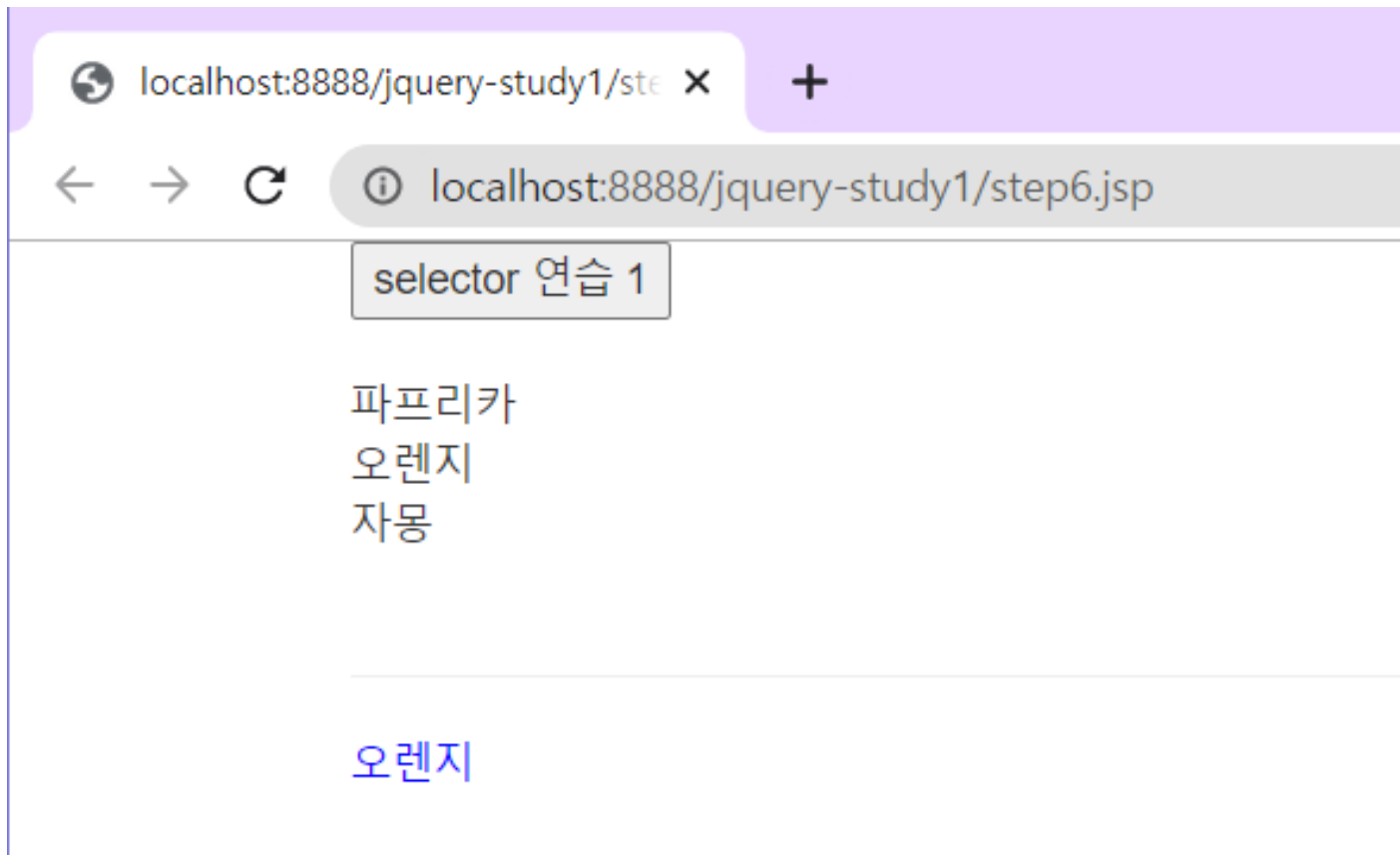
```

id 가 testBtn 인 버튼을 클릭했을 때, alert 반응이 온다.



**class 가 student 에 속하는 div(파프리카, 자몽)를
클릭했을 때만, div result 에 출력이 된다.**

(but, ', ' 를 통해 teacher class 도 출력이 되도록했다!)



7. 이미지 조정

```
<%@ page language="java" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title></title>
<meta name="viewport" content="width=device-width,
initial-scale=1">
```

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
4.1/css/bootstrap.min.css">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/
3.5.1/jquery.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4
.1/js/bootstrap.min.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        //사진 감추기
        $("#hideBtn").click(function() {
            //hide(시간, 후속작업)
            $("#imgView").hide(2000, function()
{
                $("#stateInfo").html("<font
size=7 color='lime'"
                +"밀크 없
다!"+"</font>");
            });
        });
        //사진 보기
        $("#showBtn").click(function() {
            $("#imgView").show(2000);
            $("#stateInfo").html("<font size=7
color='lime'">
```

```

        +"밀크 있다
~~"+"</font>");

    });

});
</script>
</head>
<body>
<div class="container">
    
    <input type="button" value="사진 감추기"
id="hideBtn">
    <input type="button" value="사진 보기"
id="showBtn">

    <hr>

    <span id="stateInfo">
        <font size="7" color="green">밀크 있
다!</font>
    </span>
</div>
</body>
</html>

```

> 1_1. '사진 감추기': 사라지는 중

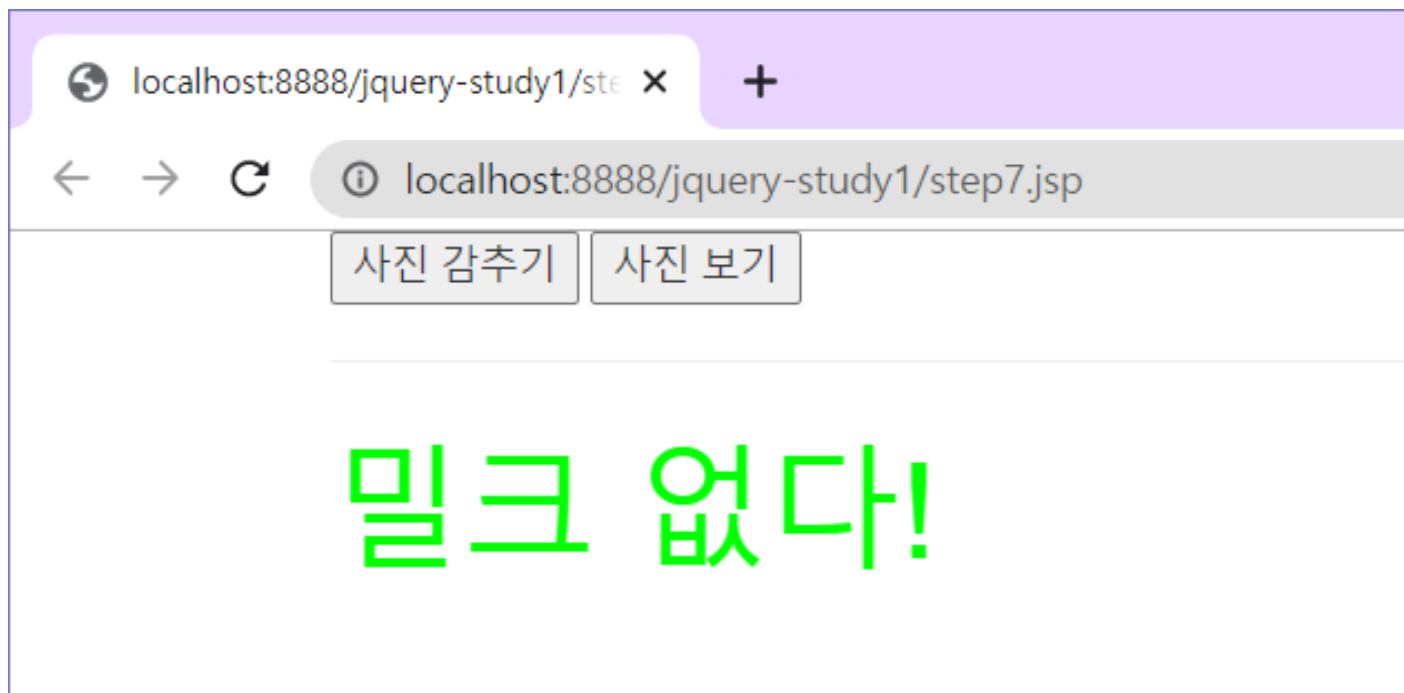


사진 감추기

사진

밀크 없다!

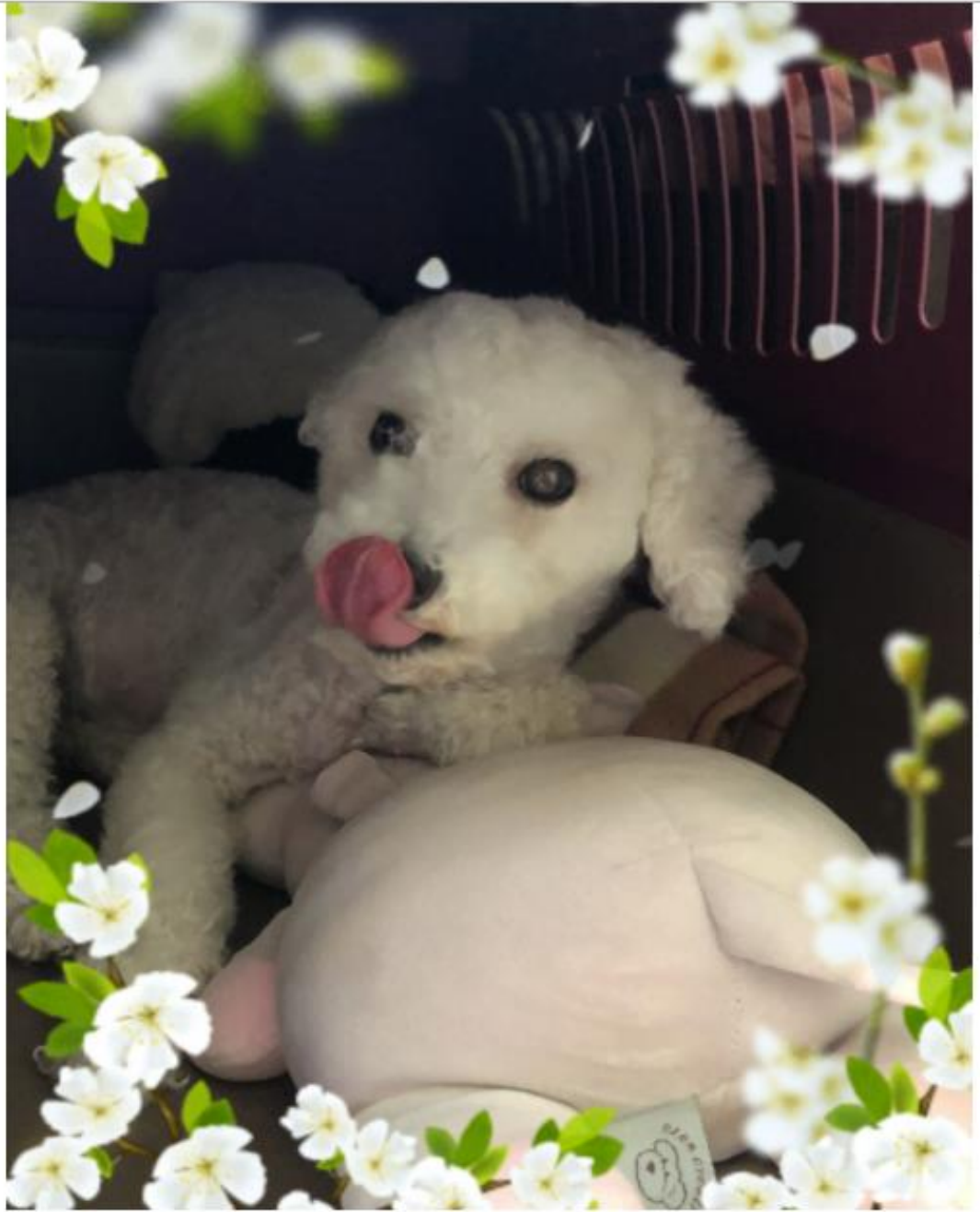
> 1_2. 사진이 감춰진 후, 후속 이벤트



> 2. '사진 보기'

localhost:8888/jquery-study1/ste x +

localhost:8888/jquery-study1/step7.jsp



밀크 없다!

