

# 3주차 JavaScript & jQuery 기초

2017.10. 14

조성현

# ▶ 지난 주 실습

## 〈자기소개 페이지 만들기〉

자신이 마음에 드는 레퍼런스 사이트를 찾아,  
적절하게 자신의 사이트로 만들기

필수 포함 사항

자신의 사진



## 이번 주 실습

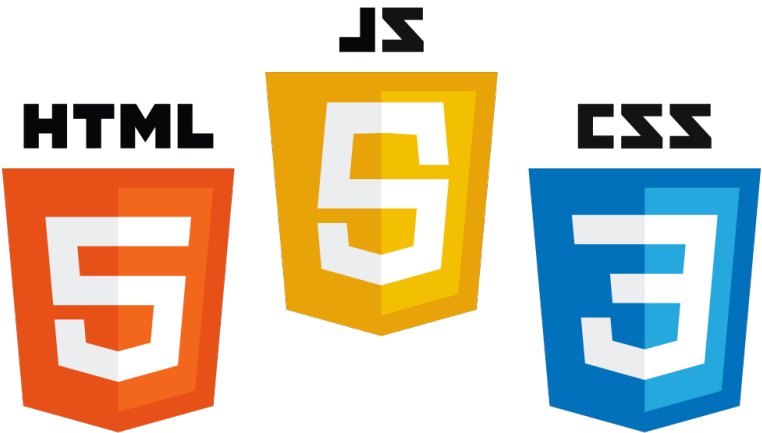
# 계산기 만들기

## 2가지 버전

1. 순수 JavaScript만으로 만들기
2. jQuery를 사용해서 만들기



# JavaScript란?



# JavaScript란?

웹 페이지상에서 문단, 제목, 표,  
이미지, 동영상등을 정의하고,  
그 구조와 의미를 부여하는 마크업 언어



# JavaScript란?



배경색, 폰트, 컨텐츠의 레이아웃등을 지정하여, HTML 컨텐츠를 꾸며주는 스타일 규칙 언어

# JavaScript란?

동적으로 콘텐츠를 바꾸고, 멀티미디어를 다루고, 움직이는 이미지 등  
웹 페이지를 꾸며주도록 하는 프로그래밍 언어

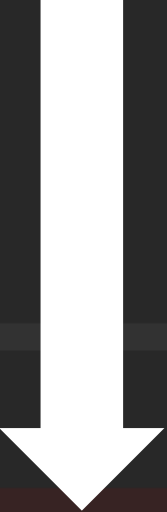


# JavaScript란?

## 특징

### 1. 인터프리터 언어

```
const Modal = new function () {  
  var that = this;  
  
  function off($ele) {  
    $ele.css('opacity', '0');  
    $ele.css('z-index', '-100');  
    $('#sunghyunWrap').css('overflow', 'scroll');  
    $('#sunghyunWrap').css('display', 'block');  
  };  
  
  this.on = function ($ele, offset) {  
    $ele.css('opacity', '1');  
    $ele.css('z-index', '1000');  
    $ele.css('top', offset);  
    $ele.css('margin-top', -offset);  
    $('#sunghyunWrap').css('overflow', 'hidden');  
    $('#sunghyunWrap').css('display', 'none');  
    $ele.css('overflow-y', 'auto');  
  
    $('#exitBtn').click(function () {  
      off($ele);  
      that.onModal1($('#modal1'));  
    });  
  };  
  
  this.offModal1 = function ($ele) {  
    $ele.css('overflow', 'hidden');  
    $ele.css('display', 'none');  
  };  
};
```





# JavaScript란?

## 특징

### 1. 인터프리터 언어

```
const Modal = new function () {  
  var that = this;  
  
  function off($ele) {  
    $ele.css('opacity', '0');  
    $ele.css('z-index', '-100');  
    $('#sunghyunWrap').css('overflow', 'scroll');  
    $('#sunghyunWrap').css('display', 'block');  
  };  
  
  this.on = function ($ele, offset) {  
    $ele.css('opacity', '1');  
    $ele.css('z-index', '1000');  
    $ele.css('top', offset);  
    $ele.css('margin-top', -offset);  
    $('#sunghyunWrap').css('overflow', 'hidden');  
    $('#sunghyunWrap').css('display', 'none');  
    $ele.css('overflow-y', 'auto');  
  
    $('#exitBtn').click(function () {  
      off($ele);  
      that.onModal1($('#modal1'));  
    });  
  };  
  
  this.offModal1 = function ($ele) {  
    $ele.css('overflow', 'hidden');  
    $ele.css('display', 'none');  
  };  
};
```

에러!



구문분석 중지

# JavaScript란?

특징

## 2. 클라이언트 스크립트 언어



# JavaScript 사용하기

HTML, CSS와 연결하기

외부 파일 연결

```
<script type="text/javascript" src="script2.js"></script>
```

# JavaScript 사용하기

## 변수 선언

```
var a = 0;
var b = "";
var c = [];
var d = {};
```

.....▶ Number

.....▶ String

.....▶ Array

.....▶ Object

# JavaScript 사용하기

console.log()

```
var a = 0;  
console.log(a);
```

# JavaScript 사용하기

비교연산자

==

VS

===

# JavaScript 사용하기

## 비교연산자

==

VS

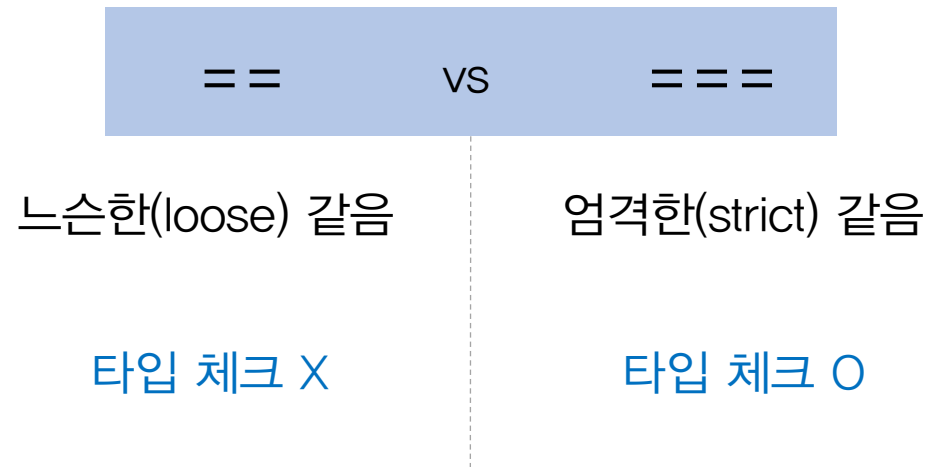
===

느슨한(loose) 같음

엄격한(strict) 같음

# JavaScript 사용하기

## 비교연산자





# JavaScript 사용하기

## 비교연산자

```
var a = 0;  
var b = "0";  
  
console.log(a == b);  
console.log(a === b);
```

# JavaScript 사용하기

## 비교연산자

```
var a = 0;  
var b = "0";
```

```
console.log(a == b); .....> true  
console.log(a === b); .....> false
```

# JavaScript 사용하기

함수 Function

```
function 함수이름 ( 매개변수1, 매개변수2, ...) {  
  
    //함수  
  
    ...  
    // return 값;  
  
}
```

# JavaScript 사용하기

## 함수 Function

Function Declaration (함수선언) vs Function Expressions (함수표현)

```
function foo() {  
    console.log("Hello");  
}
```

```
var foo = function () {  
    console.log("hello");  
};
```

# JavaScript 사용하기

## Function Declaration (함수선언)

- Function Statement 함수 문장
- 함수의 정의를 나타낸다.

```
function foo() {  
    console.log("Hello");  
}
```

# JavaScript 사용하기

## Function Declaration (함수선언)

코드블럭 자체는 실행 가능한 코드가 아니라는 것

- Function Statement 함수 문장
- 함수의 정의를 나타낸다.

```
function foo() {  
    console.log("Hello");  
}
```

# JavaScript 사용하기

## Function Declaration (함수선언)

코드블럭 자체는 실행 가능한 코드가 아니라는 것

- Function Statement 함수 문장
- 함수의 정의를 나타낸다.

```
function foo() {  
    console.log("Hello");  
}
```



```
> function foo() {return 'hello';}  
undefined  
>
```

# JavaScript 사용하기

## Function Expression (함수표현)

- Function Literal
- 변수나 데이터구조에 할당되어지고 있음을 의미.

```
//anonymous function expression
var foo = function() {
    console.log("hello");
};

//named function expression
var foo = function foo() {
    console.log("hello");
};

//self invoking function expression
(function foo() {
    console.log("hello");
})();
```



# JavaScript 사용하기

## Function Expression (함수표현)

- 특정변수에 할당되거나, 즉시 실행가능한 코드

```
//anonymous function expression
var foo = function() {
    console.log("hello");
};

//named function expression
var foo = function foo() {
    console.log("hello");
};

//self invoking function expression
(function foo() {
    console.log("hello");
})();
```

# JavaScript 사용하기

## Function Expression (함수표현)

- 자기호출함수 (self invoking function)
- 해석과 동시에 실행되는 코드블럭

```
//self invoking function expression
(function foo() {
    console.log("hello");
})();
```

# JavaScript 사용하기

## Function Expression (함수표현)

- named function expression
- 변수에 이름있는 함수를 할당한 것.

```
//named function expression
var foo = function foo() {
  console.log("hello");
};
```

```
var foo = function A() {
  A(); // 실행가능
};

A(); // Syntax Error
```



```
> (function (){return 'hello';})();
"hello"
>
```

# 호이스팅 (hoisting)

인터프리터가 자바스크립트 코드를 해석함에 있어서 Global영역에 선언된 코드블럭을 최상의 Scope로 끌어올리는 것

# 호이스팅 (hoisting)

인터프리터가 자바스크립트 코드를 해석함에 있어서 Global영역에 선언된 코드블럭을 최상의 Scope로 끌어올리는 것



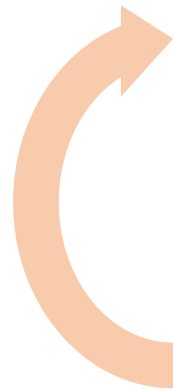
선언문은 항상 자바스크립트 구동 시 **가장 최우선으로 해석된다.**

# 호이스팅 (hoisting)

인터프리터가 자바스크립트 코드를 해석함에 있어서 Global영역에 선언된 코드블럭을 최상의 Scope로 끌어올리는 것



선언문은 항상 자바스크립트 구동 시 **가장 최우선으로 해석된다.**



```
foo();  
  
function foo () {  
    console.log("hello");  
}  
  
// > hello
```

# 호이스팅 (hoisting)

```
// 예제 1 : 함수선언에서의 호이스팅
foo();

function foo () {
    console.log("hello");
}
// > hello
```

```
//예제 2 : 함수표현에서의 호이스팅
foos();

var foos = function () {
    console.log("hello");
};
// > Syntax Error
```

# 호이스팅 (hoisting)

Quiz

```
function foo() {  
  function bar() {  
    console.log("hello");  
  }  
  
  return bar();  
  
  function bar() {  
    console.log("world");  
  }  
}  
  
foo();
```



# 호이스팅 (hoisting)

Quiz

```
function foo() {  
  function bar() {  
    console.log("hello");  
  }  
  
  return bar();  
  
  function bar() {  
    console.log("world");  
  }  
}  
  
foo();
```

정답 : world

# 호이스팅 (hoisting)

## Quiz

```
function foo() {  
  function bar() {  
    console.log("hello");  
  }  
  
  return bar();  
  
  function bar() {  
    console.log("world");  
  }  
}  
  
foo();
```

=

```
foo();  
  
function foo () {  
  function bar () {  
    console.log("hello");  
  }  
  return bar();  
  function bar() {  
    console.log("world");  
  }  
}
```

정답 : world

# 호이스팅 (hoisting)

Quiz

```
function foo(){  
  var bar= function() {  
    console.log('hello');  
  };  
  
  return bar();  
  
  var bar = function() {  
    console.log('world');  
  };  
}  
  
foo();
```

# 호이스팅 (hoisting)

Quiz

```
function foo(){  
  var bar= function() {  
    console.log('hello');  
  };  
  
  return bar();  
  
  var bar = function() {  
    console.log('world');  
  };  
}  
  
foo();
```

정답 : hello

# 호이스팅 (hoisting)

Quiz

```
function foo(){  
  var bar= function() {  
    console.log('hello');  
  };  
  
  return bar();  
  
  var bar = function() {  
    console.log('world');  
  };  
}  
  
foo();
```

```
function foo(){  
  var bar= undefined;  
  var bar= undefined;  
  
  bar= function() {  
    console.log('hello');  
  };  
  
  return bar();  
}
```

정답 : hello

## 호이스팅 (hoisting)

```
function foo(){  
    return bar();  
    var bar= function() {  
        console.log('hello');  
    };  
    var bar = function() {  
        console.log('world');  
    };  
}  
  
foo();
```



# getElementBy..()

```
document.getElementById("id Selector");
```

```
document.getElementsByClassName("class Selector");
```

# getElementBy..()

```
<body>
<div id="parent-id">
  <p>hello word1</p>
  <p class="test">hello word2</p>
  <p >hello word3</p>
  <p>hello word4</p>
</div>
<script>
  var parentDOM = document.getElementById("parent-id");

  //test is not target element
  var test=parentDOM.getElementsByClassName("test");
  console.log(test); //HTMLCollection[1]

  //hear , this element is target
  var testTarget=parentDOM.getElementsByClassName("test")[0];
  console.log(testTarget); //<p class="test">hello word2</p>
</script>
</body>
```



# addEventListener()

마우스 클릭, 키보드 입력 등의 이벤트를 듣는(listen) 역할을 하는 method

기존에 존재하는 이벤트 핸들러에 덮어쓰기 없이 요소에 이벤트 핸들러를 첨부하는 것

```
element.addEventListener(event, function, useCapture);
```

# ▶ addEventListener()

```
element.addEventListener(event, function, useCapture);
```



이벤트의 종류 ( ex. click, mousedown...)

# ▶ addEventListener()

```
element.addEventListener(event, function, useCapture);
```



이벤트가 발생 했을 때, 호출할 함수

# addEventListener()

```
element.addEventListener(event, function, useCapture);
```



선택적으로 작성

# addEventListener()

```
element.addEventListener("click", function(){ alert("Hello World!"); });
```

```
element.addEventListener("click", myFunction);  
  
function myFunction() {  
    alert ("Hello World!");  
}
```

# addEventListener()

같은 요소에 이미 존재하는 이벤트의 덮어쓰움없이 여러 이벤트를 추가 할 수 있음.

```
element.addEventListener("click", myFunction);  
element.addEventListener("click", mySecondFunction);
```



# innerHTML

HTML요소에 접근하여 값을 바꾸는 기능

폭스테일



팍스테일

```
element.addEventListener("click", myFunction);  
  
function myFunction(){  
    document.getElementById("#hi").innerHTML = "팍스테일";  
}
```



엘리먼트를 선택하는 강력한 방법과  
선택된 엘리먼트들을 효율적으로 제어할 수 있는 다양한 수단을 제공하는  
자바스크립트 라이브러리

자주 사용하는 코드들을 재사용할 수 있는 형태로  
가공해서 프로그래밍 효율을 높여주는 코드들





# jQuery

## 장단점

- 장점 : 멀티 브라우저 지원
- 단점 : 새로운 API학습, 퍼포먼스 문제

# jQuery

## 연결하기

- CDN 호스트 사용

```
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
```

- 직접 내려받아 사용

```
<script src="jquery-1.10.2.js"></script>
```

# jQuery

Selector

```
$( “선택대상” ).메소드();
```

# jQuery

## Selector

`$( “선택대상” ).메소드();`

`$( “.class” ).메소드();`

`$( “#id” ).메소드();`

`$( “tag” ).메소드();`

# jQuery

## Event 처리

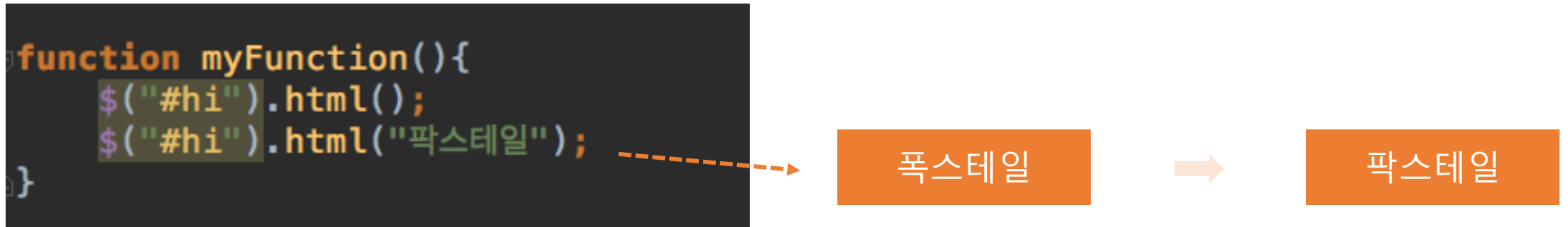
```
$( “선택대상” ).click(function() {  
    ...;  
});
```

# jQuery

html()

HTML요소에 접근하여 값을 바꾸는 기능

JavaScript의 innerHTML과 같은 역할



# 이번 주 실습

## “계산기 만들기”

### 2가지 버전

1. 순수 JavaScript만으로 만들기
2. jQuery를 사용해서 만들기

### 〈필요 기능〉

- 계산기 UI구성
- 사칙연산
- = 이후에 연산 계속 가능
- 기본적인 계산기와 다른 논리에 맞지 않는 버그 불가!

