**자바스크립트 ( JavaScript )**

웹브라우저를 자바스크립트로 제어하는 방법에 대한 수업입니다.

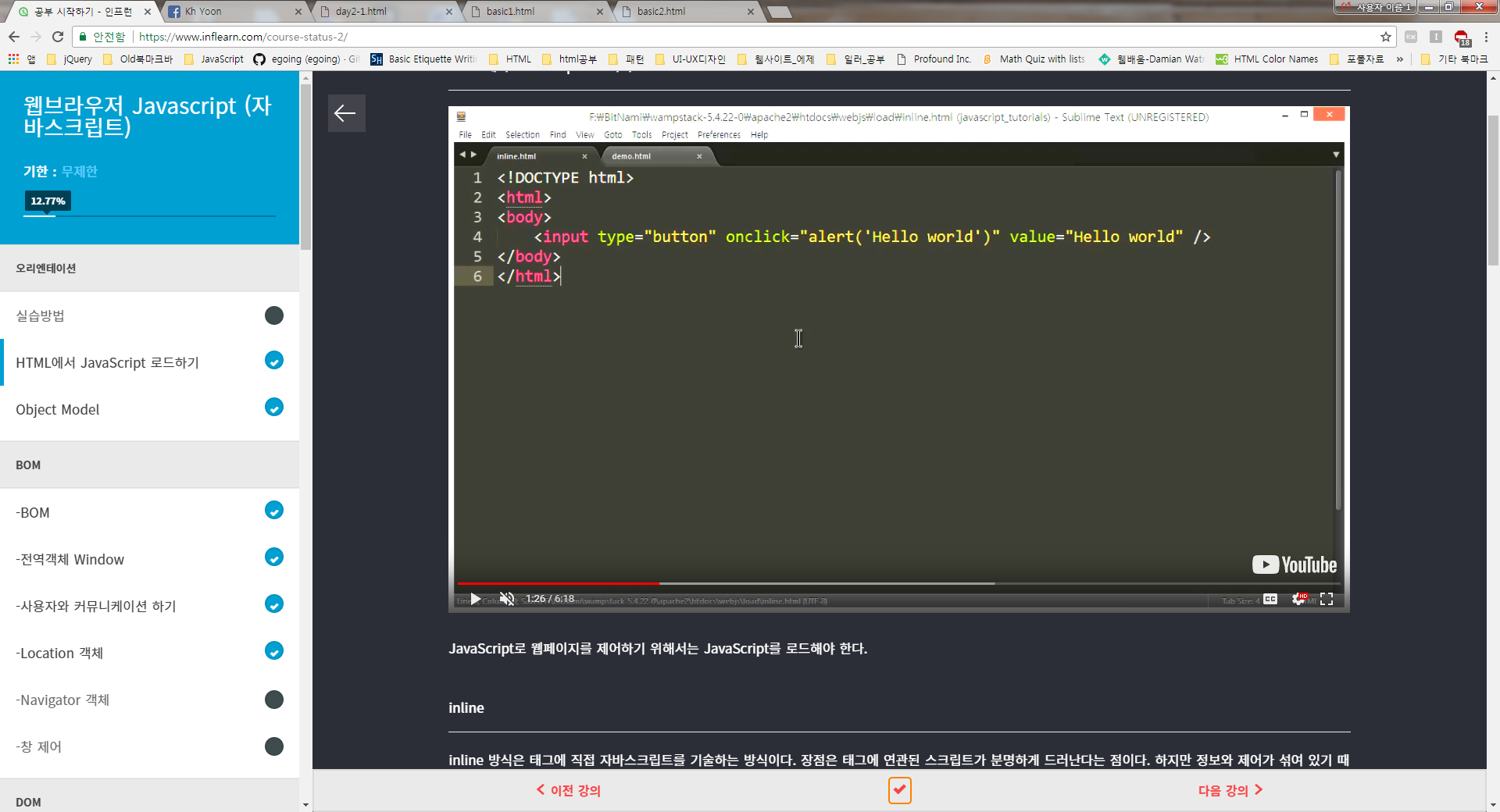
 순수한 자바스크립트로 웹브라우저를 제어하는 방법과 더 적은 코드로 더 강력한 효과를 얻을 수 있는 수단을 제공하는 각종 라이브러리를 사용할 것입니다.

HTML에서 JavaScript 로드하기

Javascript 로 웹페이지를 제어하기 위해서는 자바스크립트를 로드해야 한다

**inline**

inline 방식은 태그에 직접 자바스크립트를 기술하는 방식이다. 장점은 태그에 연관된 스크립트가 분명하게 드러난다는 점이다. 알아보기 쉽지만 정보와 제어문장이 함께 있는 것이 단점이다



교재 276쪽

**내부 스크립트**

script를 head 태그에 위치시킬 수도 있다. 하지만 이 경우는 오류가 발생한다.

**교재274쪽**

**외부파일로**

태그를 만들어서 여기에 자바스크립트 코드를 삽입하는 방식이다. 장점은 html 태그와 js 코드를 분리할 수 있다는 점이다.

**자바스크립트의 위치**

<head>와 </body>전에 삽입

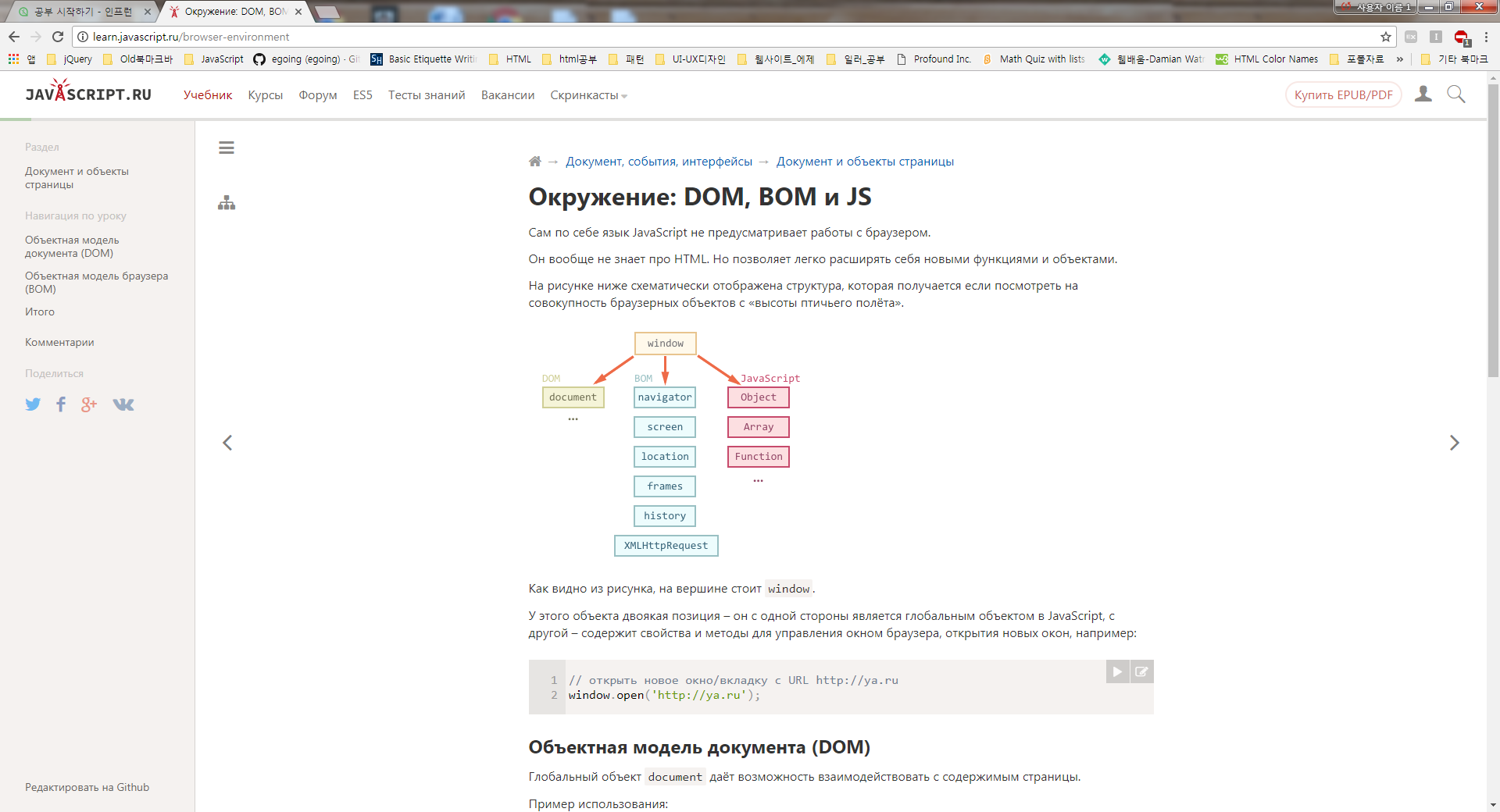
변수와 상수

자료형

객체화란?

웹브라우저의 구성요소들은 하나하나가 객체화되어 있다. 자바스크립트로 이 객체를 제어해서 웹브라우저를 제어할 수 있게 된다. 이 객체들은 서로 계층적인 관계로 구조화되어 있다. BOM과 DOM은 이 구조를 구성하고 있는 가장 큰 틀의 분류라고 할 수 있다.이 관계를 그림으로 나타내면 아래와 같다. (출처 : <http://learn.javascript.ru/browser-environment>)

자바스크립트로 브라우저를 제어하기 위해서는 자바스크립트로 제어할 수 있는 무엇이 있어야 하는데 이 ‘무엇’을 총칭해서 오브젝트 (object, 객체 )라고 하며 브라우저의 구성요소들은 이미 이런 오브젝트화 즉 객체로 만들어서 제공해 되고 있다



자바스크립트로 브라우저를 제어하기 위해서 브라우저의 요소들은 객체화가 되어 있다. 이 객체화란 아래의 의미와 같다.

Img 태그가 자바스크립트오 제어가능한 형태 즉 오브젝트 이어야 한다

이런 오브젝트는 이미 만들어져 있고,. 브라우저에서 읽어내려가는 과정에서 각각의 태그들에 해당하는 객체들을 불러와서 그 객체의 속성이라든가 메소드를 호출하는 동작을 통해서 그 객체를 제어하는 것이다 그리고 객체를 제어한다는 것을 그 객체가 가리키고 있는 태그을 제어한다는 것이다.

브라우저에서 만들어져 있는 객체중에 document라는 객체가 있다.

이 docdument라는 객체에는 getElementsByTagName 이라는 메소드가 있다 그래서 img 라는 태그들 제어학 위해서는

Document 객체를 불러내고 이 객체가 가진 메소드 중에 태그이름으로 제어할 수 있는 기능을 가진 getElementsByTagName를 불러내고, 어떤 태그를 제어할지알려준다. 그래서 즉

document.getElementsByTagName(‘img’) 가 된다.

콘솔창에

Var imgs = document.getElementsByTagName(‘img’);

내용을 변수안에 넣는다. 그러나 태그 이름은 여러가 있을 수 있기 때문에 배열로 되어 있다. 따라서 삽입된 이미지를 알아보기 위해서는

imgs[0] 배열로 표현해야 한다.

imgs[0]는 img 태그를 말하므로 img 태그가 갖고 있는 style 속성을 불러내야 한다. 즉 imgs[0].style 이라고 적는다. 여기서 style라는 속성은 img태그의 css를 말하므로 해당 그림의 너비를 변경하기 위해서는 imgs[0].style.width=’400px’; 입력한다. 그러면 브라우저에서 해당 img 태그의 너비가 주어진 크기로 바뀌어 있다

또한 브라우저는 window라는 객체를 통해서 제어할 수 있다 우리가 아까 사용했던 alert라는 메소드가 있다.

우리가 경고창을 사용하고자 한다면 아래와 같이 사용할 수 있다.

window.alert(‘경고메시지’) 또는

alert(‘경고메시지’);

BOM(Browser Object Model)이란 웹브라우저의 창이나 프래임을 추상화해서 프로그래밍적으로 제어할 수 있도록 제공하는 수단이다. BOM은 전역객체인 Window의 프로퍼티와 메소드들을 통해서 제어할 수 있다. 이것은 브라우저를 통해서 새창을 연다거나 현재창의 url 을 알아낸다거나 할 때 사용할 수 있다.

위의 그림에서 어떤 것을 하는지 알 수 있다.

Window 객체는 모든 객체가 속해있는 객체로 전역객체이며 창 또는 프레임을 의미한다.

예)

function func(){

alert('Hello?');

}

func();

window.func();

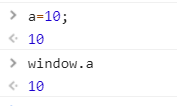
func();와 window.func();는 모두 실행이 된다. 모든 전역변수와 함수는 사실 window 객체의 프로퍼티다. 객체를 명시하지 않으면 암시적으로 window의 프로퍼티로 간주된다. 자바스크립트에서 모든 객체는 기본적으로 전역객체의 프로퍼티(속성)임을 알 수 있다.

여기서 window.func();

Window는 객체 .은 속성 가리키며 따라서 func라는 속성을 가져오는데 ( )를 갖고 있으므로 함수형태로 되어 있다. 우리는 이러한 것을 메소드라고 부른다

그래서 우리가 만드는 함수이건, 내장함수이건 이것들은 모두 window객체의 메소드이다. 뿐만 아니라 변수도 window객체에 속해 있다.

만약 console창에다



같은 결과가 된다. 전역변수라는 것은 window라는 객체의 프로퍼티, window 라는 객체의 메소드를 만드는 것이다.

내장함수 ( Standard Built-in Object ) 는자바스크립트가 기본적으로 가지고 있는 객체들이다. 내장 객체는 프로그래밍하는데 기본적으로 필요한 도구들이다. 자바스크립트가 다음과 같은 내장 객체를 가지고 있다.

* Object
* Function
* Array
* String
* Boolean
* Number
* Math
* Date
* RegExp

이제 우리는 내장객체가 제공하지 못하는 기능이 있다면 우리가 만들어서 사용할 수도 있더, 이것을 사용자정의 함수라고 한다

이때 prototype 이라 것이 등장한다 .

**Prototype**

prototype이란 무엇인가? 한국어로는 원형으로 prototype은 말 그대로 객체의 원형이라고 할 수 있다. 함수는 객체다. 그래서 생성자(new)라는 것을 통해서 새로운 객체를 만들 수 있다.

객체는 프로퍼티를 가질 수 있는데 prototype이라는 프로퍼티는 그 용도가 약속되어 있는 특수한 프로퍼티다. prototype에 저장된 속성들은 생성자를 통해서 객체가 만들어질 때 그 객체에 연결된다. 다시 설명하자면,

자바스크립트는 프로토타입 기반 언어라고 불립니다. 자바스크립트 개발을 하면 빠질 수 없는 것이 프로토타입이다. 자바스크립트는 함수(function)와 new를 통해 정의될 수 있다

예)

function Person() {}

Person.prototype.eyes = 2;  
Person.prototype.nose = 1;

var kim = new Person();  
var park = new Person():

console.log(kim.eyes); // => 2  
console.log(kim.nose); // => 1

console.log(park.eyes); // => 2  
console.log(park.nose); // => 1

자바스크립트의 프로토타입 상속(prototypal inheritance) 이라는 확장과 객체의 재사용을 가능하게 해준다. 자바스크립트에서는 객체의 원형인 **프로토타입을 이용한 클로닝(Cloning: 복사)과 객체특성을 확장해 나가는 방식을 통해 새로운 객체를 생성**해 낸다.

다른 말로 프로토타입 기반 언어는 객체 원형인 프로토타입을 이용하여 새로운 객체를 만들어냅니다. 이렇게 생성된 객체 역시 또 다른 객체의 원형이 될 수 있습니다.

function Person(){}

var joon = new Person();

var jisoo = new Person();

Person.prototype.getType = function (){

return "인간";

};

console.log(joon.getType()); // 인간

console.log(jisoo.getType()); // 인간

joon.getType = function (){

return "사람";

};

console.log(joon.getType()); // 사람

console.log(jisoo.getType()); // 인간

jisoo.age = 25;

console.log(joon.age); // undefined

console.log(jisoo.age); // 25

**함수**

함수는 function 이라는 키워드를 사용합니ㅏ. 함수 이름은 사용자가 직접 정의하여 만들면 되는데 이때 변수 이름 작성시와 같은 규칙에 의해 만들면 됩니다.

함수의 호출은 함수의 이름을 불러서 호출한다. 이름을 불러줄때에는 함수이름 뒤에 ()를 붙인다.

1. 선언적 함수는 function 함수이름() { }
2. 익명함수는 var 변수 = function 함수이름() { }

선언적 함수와 익명함수의 기능적인 부분에는 차이가 없지만 중요한 것은 함수를 작성하는 위치입니다. 익명함수는 반드시 함수를 호출하기 전에 작성되어야 합니다

▶ 함수인자(arguments and parameters)를 활용한 함수확장

함수를 선언할 때 ( ) 안에 들어가는 여러가지 변수를 매개 변수(arguments and parameters)라고 합니다.

<script>

function sum(a,b){

var c=a+b;

console.log(c);

}

sum(10,20);

// 페이지를 로드한 후에 console창에서 결과를 확인한다

</script>

실행해 보기 : sum (15,30) // 45출력 , sum(50,40) // 90출력

▶ 함수종료 및 리턴 값

함수를 실행하고 난 후에 함수의 결과를 얻어 사용하려면 return을 사용하여 결과값을 전달한다.

return 변수 // 함수를 호출한 곳으로 보내고 싶은 값을 전송하고 함수를 종료한다

return // 함수를 종료한다 returen false 로도 표현한다

예제1)

<script>

function sum(a,b){

var c=a+b;

return;

}

sum(10,20);

// 페이지를 로드한 후에 console창에서 결과를 확인한다

</script>

예제2)

<script>

function sum(a,b){

var c=a+b;

return c;

}

var x = sum(10,20);

document.write(x);

</script>

▶변수의 재정의

<script>

// 변수를 선언하고 값을 입력한다

var f1 = '김치찌개';

var f1 = '라면';

var f1 = '냉면';

// 출력합니다.

alert(f1);

</script>

위의 예제에서 변수를 alert 로 바꾼 후 출력에서 alert(alert ) 하면 오류가 난다

*<script>*

*// 변수를 선언하고 값을 입력한다*

*var alert = '김치찌개';*

*var alert = '라면';*

*var alert = '냉면';*

*// 출력합니다.*

*// alert(alert);*

*</script>*

▶증감연산자 연습1

<script>

var number =10;

alert(number++);

alert(++number);

alert(number--);

alert(--number);

</script>

▶continuer키워드

<script>

var output =0;

for(var i=1; i<=10; i++){

if (i%2==1) {

continue;

// 홀수이면 현재 반복을 중지하고 다음 반복을 수행한다

}

output+=1;

}

alert(output);

</script>

▶자료형 검사

<script>

alert(typeof ('String')); //문자열

alert(typeof (365)); // 숫자

alert(typeof (true)); // 불리언

alert(typeof(function(){})); // 함수

alert(typeof ({})); // 객체

alert(typeof (alpha)); // undefined

</script>

▶시간을 이용한 조건 분기

<script>

var date = new Date();

var hour = date.getHours();

switch (true){

case hour<11:

alert('오전입니다');

break;

case hour<15:

alert('오후입니다');

break;

default:

alert('곧 집에 갈 시간입니다');

}

</script>

▶삼항연산자

<script>

var input =prompt('숫자를 입력하세요');

var number = Number(input);

(number > 0)? alert('자연수입니다') :alert('자연수가 아닙니다');

</script>

</body>

▶ 짧은 조건문

<script>

true || alert('실행될까요... 1');

false ||alert('실행될까요... 2');

// 좌변이 참이면 우변을 실행하지 않습니다 .

// (불리언 표현식) || (불리언 표현식이 거짓일 때 실행한 문장)

</script>

<script>

var input = Number(prompt('숫자를 입력하세요'));

input%2 ==0 || alert('홀수입니다');

input%2 ==0 && alert('찍수입니다');

//논리곱 && 연산자는 양 변이 모두 참일때만 참을 나타낸다.

// (불리연 표현식) && ( 불리언 표현식이 참일때 실행할 문장 )

</script>