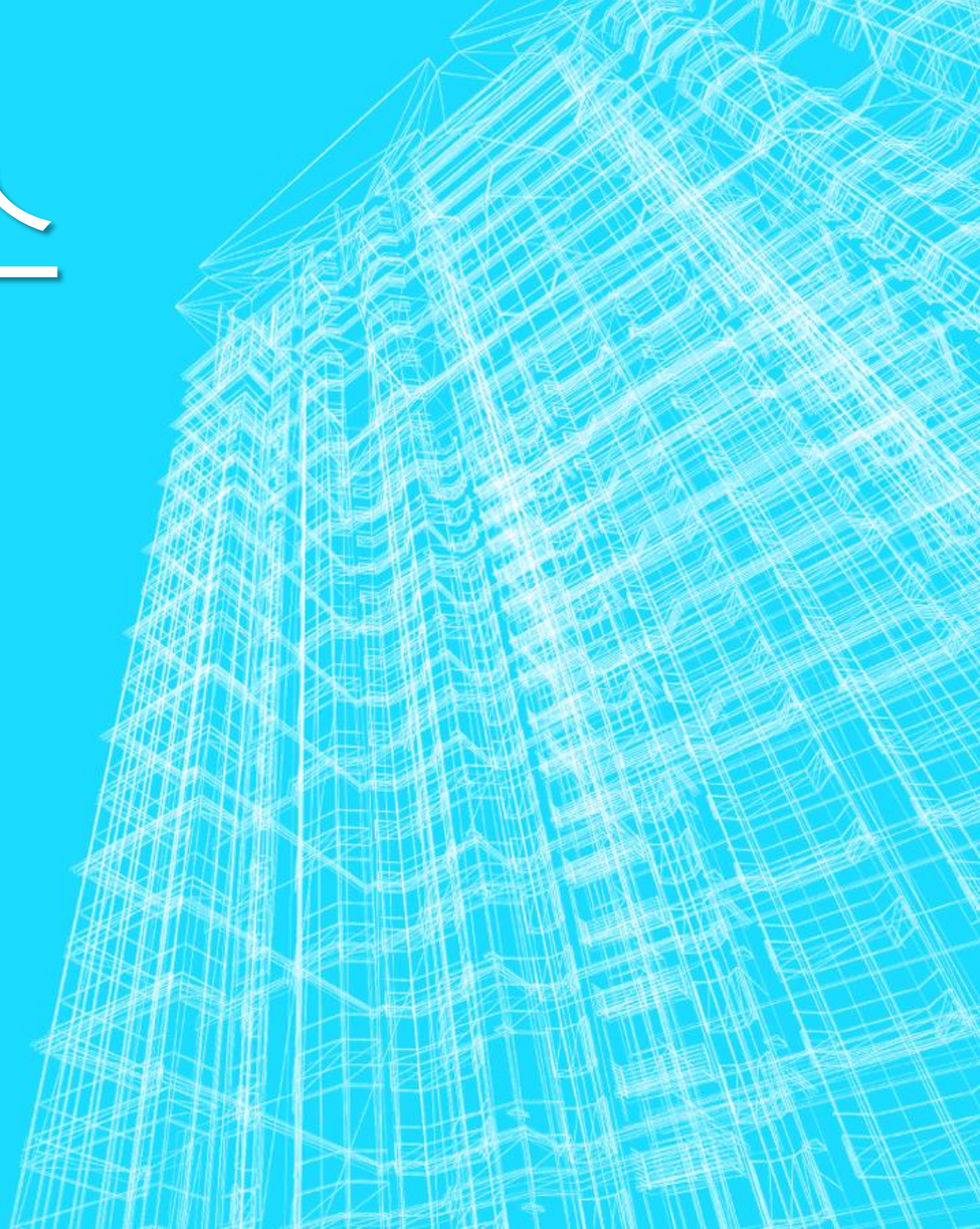


네트워크 서비스 프로토콜

3주차 1

소프트웨어학부
김형균 교수



주차별 수업 계획

01주차	9월 3일	교과목 소개	09주차	10월 29일	몽고디비 1
	9월 5일	교과목 소개		10월 31일	몽고디비 2
02주차	9월 10일	노드 시작하기	10주차	11월 5일	익스프레스로 SNS 서비스 만들기 1
	9월 12일	추석 휴무		11월 7일	익스프레스로 SNS 서비스 만들기 2
03주차	9월 17일	알아두어야 할 자바스크립트 1	11주차	11월 12일	웹 API 서버 만들기 1
	9월 19일	알아두어야 할 자바스크립트 2		11월 14일	웹 API 서버 만들기 2
04주차	9월 24일	노드 기능 알아보기 1	12주차	11월 19일	웹 소켓으로 실시간 데이터 전송하기 1
	9월 26일	노드 기능 알아보기 2		11월 21일	웹 소켓으로 실시간 데이터 전송하기 2
05주차	10월 1일	개천절 휴무	13주차	11월 26일	실시간 경매 시스템 만들기 1
	10월 3일	패키지 매니저		11월 28일	실시간 경매 시스템 만들기 2
06주차	10월 8일	익스프레스 웹 서버 만들기 1	14주차	12월 3일	과제발표 1
	10월 10일	익스프레스 웹 서버 만들기 2		12월 5일	과제발표 2
07주차	10월 15일	MySQL 1	15주차	12월 10일	기말고사
	10월 17일	MySQL 2		12월 12일	보충수업
08주차	10월 22일	중간고사			
	10월 24일	보충수업			



수업에 들어가며

- 복습문제
 - 객체 리터럴
- 이번 시간 학습목표
 - 객체의 메서드
 - DOM을 사용해 이벤트처리기 등록
 - HTML요소의 <input>, <output> 태그 사용
 - HTML요소의 innerHTML 프로퍼티 사용



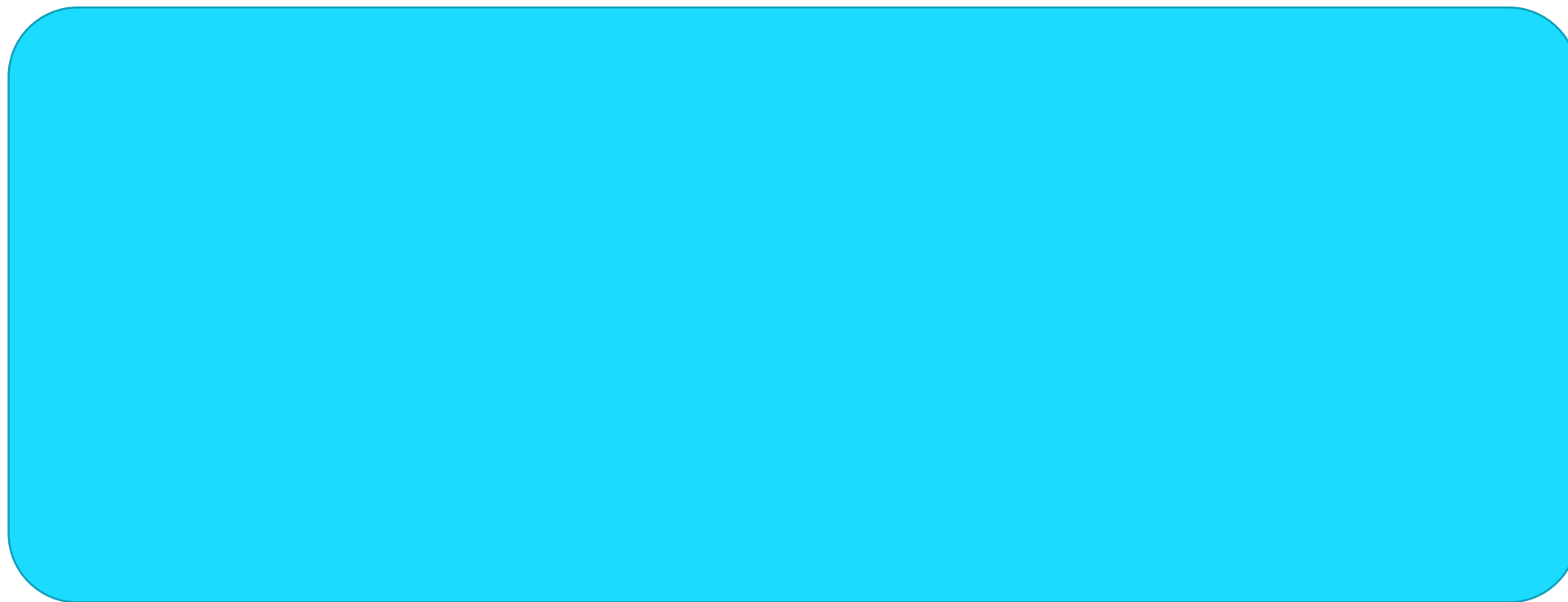
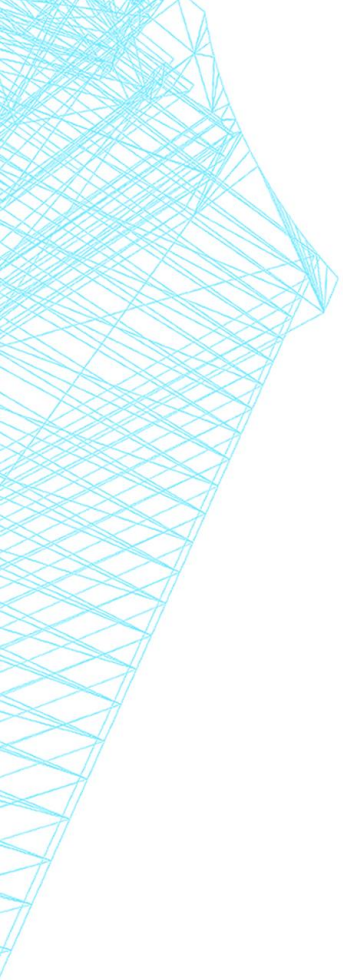
복습문제. 객체 리터럴

다음과 같이 원의 중심좌표와 반지름을 객체리터럴을 이용해 생성하는 객체를 선언하고 출력하시오.

- 객체명 : circle
- 중심좌표 프로퍼티명 : center
- 반지름 프로퍼티명 : radius

- 출력형태

```
원의 중심좌표는 (1,2)
원의 반지름은 2.5
```



객체의 메서드

- 메서드는 객체가 가지고 있는 동작
- 메서드를 수행하기 위해서는 객체를 통해서 해당 메서드를 수행
- 함수와 메서드의 차이 : 함수는 그 동작을 수행하기 위해 객체에게 어떤 동작을 수행하라고 명령하지 않아도 된다. 그이유는 함수자체가 그 동작을 정의한 함수객체이기 때문에 자기 자신을 수행하는 것
- 형식)

```
메서드명 : function( ) {  
    실행명령 1;  
    :  
    return 리턴값;  
}
```



문제

앞서 생성한 circle 객체에 다음과 같이 메서드를 추가하고 출력하시오

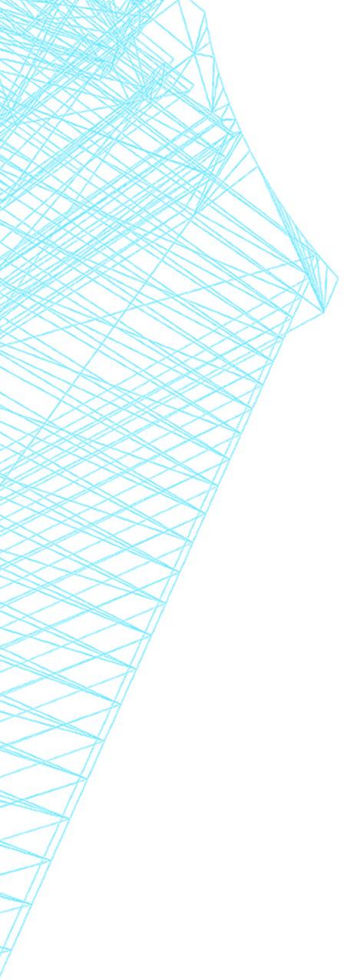
- 원의 면적 계산 메서드 : area
- 원의 둘레 계산 메서드 : round
- 원주율 상수 : Math.PI
- 출력형태

```
원의 중심좌표는 (1,2)
```

```
원의 반지름은 2.5
```

```
원의 면적은 19.63입니다.
```

```
원의 둘레는 15.71입니다.
```



```
var circle = {  
  center : {x:1.0, y:2.0} ,  
  radius : 2.5,
```

```
};
```

```
console.log("원의 중심좌표는 (" + circle.center.x + "," + circle.center.y + ")");  
console.log("원의 반지름은 " + circle.radius );
```




문제

앞서 생성한 circle 객체리터럴을 변경하지않고 원을 평행이동시키는 translate 메서드를 함수형태로 추가해서 다음과 같이 출력하시오

```
원의 중심좌표는 (1,2)  
(1,2)이동한 원의 중심좌표는 (2,4)
```

```
var circle = {  
  center : {x:1.0, y:2.0} ,  
  radius : 2.5,  
  area : function(){  
    return Math.PI * this.radius * this.radius;  
  },  
  round : function(){  
    return 2 * Math.PI * this.radius;  
  }  
};
```

```
console.log("원의 중심좌표는 (" + circle.center.x + "," + circle.center.y + ")");
```

```
console.log("(1,2)이동한 원의 중심좌표는 (" + circle.center.x + "," + circle.center.y + ")");
```



DOM을 사용해 이벤트처리기 등록

- DOM(Document Object Model)은 자바스크립트 등의 프로그램이 **HTML 요소를 조작할 수 있게 하는 인터페이스**
- DOM을 사용해 이벤트처리기 등록하는 전형적인 방법
 - 1) **window.onload** 를 사용해 HTML문서를 다 읽어들인 후 2)와3)을 실행
 - 2) `document.getElementById` 메서드를 사용해 특정 id속성 값을 가진 HTML요소의 객체를 가져온다.
 - 3) 요소 객체의 이벤트 처리기 프로퍼티에 이벤트 처리기로 동작할 함수를 등록한다.



DOM을 사용해 이벤트처리기 등록

1) **WINDOW.ONLOAD** 를 사용

- DOM에서 이벤트처리기 등록하는 가장 큰 목적은 HTML코드와 자바스크립트 코드를 분리하는 것
- 이를 구현하기 위해 script요소를 head요소의 자식요소를 배치
- DOM을 사용하면 body요소의 바깥에서 body요소 안에 있는 HTML요소를 조작할 수 있음
- 이벤트 처리기를 등록하는 작업의 실행 시점은 HTML문서 전체를 읽어들이는 이후로 미루게 됨
- 이를 위해 window객체의 onload 프로퍼티에 이벤트 처리기를 등록하는 작업을 수행하는 초기 설정 함수를 정의함
- **window.onload = function() { ... };**
- 따라서 이 코드에 의해 웹브라우저가 HTML문서 전체를 모두 읽어 들인 후에 우변의 함수를 실행하게 됨



DOM을 사용해 이벤트처리기 등록

2) `Document.getElementById()`

- 이 메서드는 주어진 문자열과 일치하는 id 속성을 가진 요소를 찾고, 이를 나타내는 `Element` 객체를 반환
- ID는 문서 내에서 유일해야 하기 때문에 특정 요소를 빠르게 찾을 때 유용
- ID가 없는 요소에 접근하려면 `Document.querySelector()`를 사용
- 형식)
 - `document.getElementById(id);`
 - 매개변수(id) : 찾으려는 요소 ID. ID는 대소문자를 구분하는 문자열로, 문서 내에서 유일해야 합니다. 즉, 하나의 값은 하나의 요소만 사용할 수 있습니다.
 - 반환 값 : 주어진 ID와 일치하는 DOM 요소를 나타내는 `Element` 객체. 문서 내에 주어진 ID가 없으면 `null`.

DOM을 사용해 이벤트처리기 등록

2) `Document.getElementById()`

```
<html>
<head>
  <title>getElementById 예제</title>
  <script>
    function changeColor(newColor) {
      var elem = document.getElementById('para');
      elem.style.color = newColor;
    };
  </script>
</head>
<body>
  <p id="para">어떤 글</p>
  <button onclick="changeColor('blue');">blue</button>
  <button onclick="changeColor('red');">red</button>
</body>
</html>
```

어떤 글

blue

red

어떤 글

blue

red

어떤 글

blue

red



DOM을 사용해 이벤트처리기 등록

3) 이벤트 처리기 프로퍼티에 동작함수 등록

이벤트 처리를 위한 동작을 수행할 이벤트 처리기 등록

형식) 요소 객체명.이벤트처리기 프로퍼티 = 실행함수명

```
button.onclick = displayTime;
```

이벤트 프로퍼티	이벤트 발생시점	사용태그 or 객체
onAbort	이미지 읽는 것을 취소할 때	img
onBlur	포커스를 잃어버렸을 때	모든 Form요소, body, window객체
onChange	Form요소의 값이 변했을 때	INPUT TYPE=text, Select, Textarea
onClick	마우스를 클릭했을 때	A, Area, Input Type=button, Checkbox, Radio, Reset, Submit, Document객체
onDbClick	마우스를 더블 클릭했을 때	A, Area, Input Type=button, Checkbox, Radio, Reset, Submit, Document객체
onError	이미지나 문서를 불러올 때, 에러가 났을 때	img, window객체
onFocus	포커스를 받을 때	모든 Form요소, body, window객체
onKeyDown	Key가 눌려졌을 때	A, Area, Img, Textarea, document객체
onKeyPress	Key를 눌렀다 놓았을 때	A, Area, Img, Textarea, document객체
onKeyPress	Key를 눌렀다 놓았을 때	A, Area, Img, Textarea, document객체
onKeyUp	Key를 놓았을 때	A, Area, Img, Textarea, document객체
onLoad	문서의 로딩이 완료되었을 때	Img, Body, Frameset, Window객체
onMouseDown	마우스 버튼을 눌렀을 때	A, Area, Input Type=button, document객체
onMouseup	마우스 버튼을 놓았을 때	A, Area, Input Type=button, document객체
onMouseMove	마우스가 움직일 때	
onMouseout	해당 영역을 벗어났을 때	A, Area
onMouseover	해당 영역에 마우스를 놓았을 때	A, Area
onReset	Reset 버튼을 눌렀을 때	form
onResize	Frame이나 윈도우의 크기가 변경될 때	Window객체
onSelect	해당 Field를 선택했을 때	Input Type=text, textarea
onSubmit	Submit 버튼을 눌렀을 때	Form
onUnload	해당 문서를 빠져 나갈 때	Body, Frameset, Window객체

시각을 콘솔에 표시하기

×

+

←

→

↺

📄 파일 | C:/test/javascript/modern_javascript/ch06/6-1.html

click

🔍 📄 | Elements Console Sources Network

🎬 ⛔ | top ▼ | 👁 | Filter

현재 시각은 2019. 9. 15. 오후 5:40:07 입니다.

현재 시각은 2019. 9. 15. 오후 5:40:08 입니다.

현재 시각은 2019. 9. 15. 오후 5:40:32 입니다.

현재 시각은 2019. 9. 15. 오후 5:40:33 입니다.

2 현재 시각은 2019. 9. 15. 오후 5:40:34 입니다.

현재 시각은 2019. 9. 15. 오후 5:40:35 입니다.

2 현재 시각은 2019. 9. 15. 오후 5:40:36 입니다.

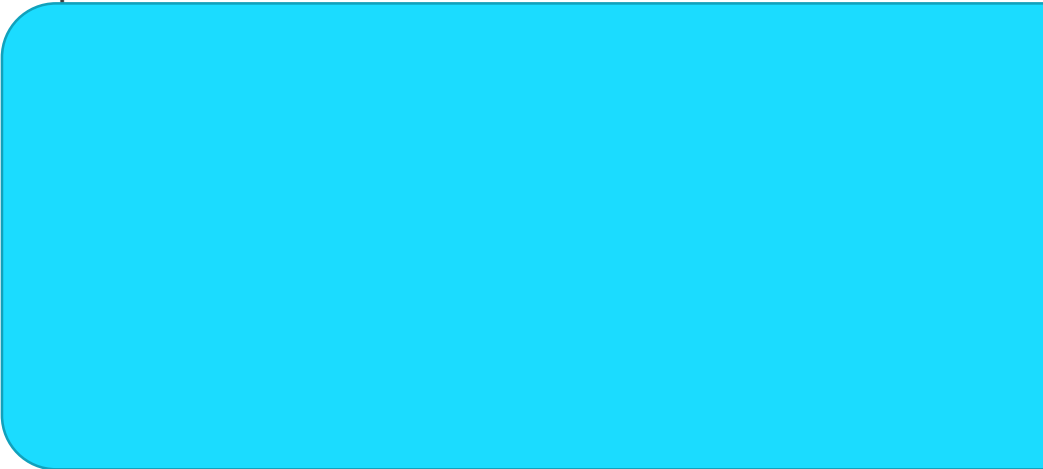
현재 시각은 2019. 9. 15. 오후 5:40:38 입니다.

>

일반적인 이벤트 처리 방식

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>시각을 콘솔에 표시하기</title>
</head>
<body>
  <script>
    function displayTime() {
      var d = new Date();
      console.log("현재 시각은 " + d.toLocaleString() + " 입니다.");
    }
  </script>
  <input type="button" value="click" onclick="displayTime()">
</body>
</html>
```


DOM을 사용해 이벤트처리기

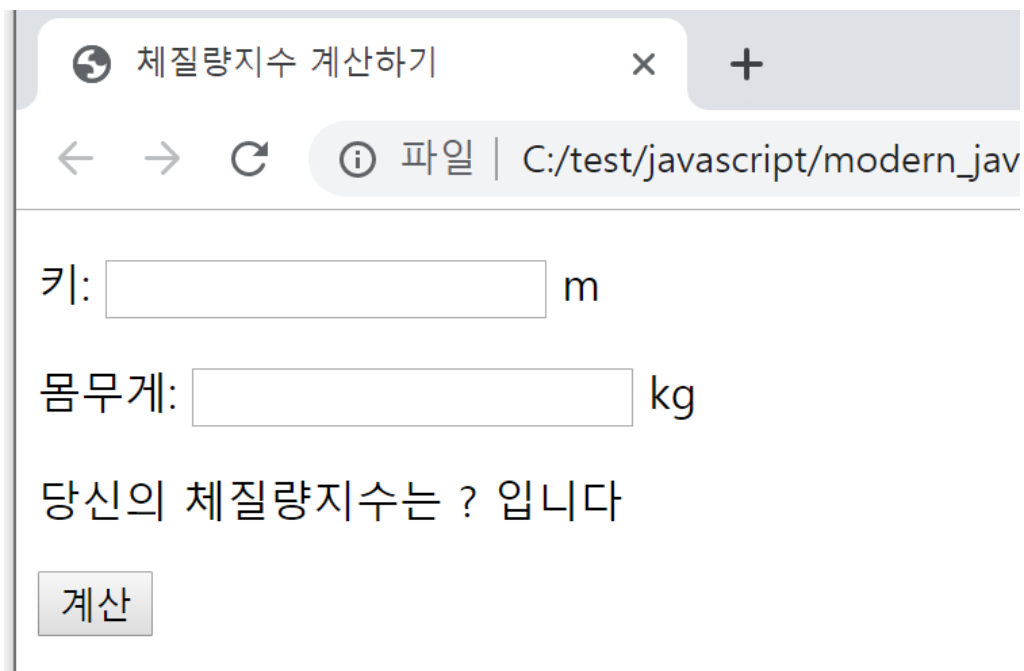
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>시각을 콘솔에 표시하기</title>
  <script>
    
  </script>
</head>
<body>
  <input type="button" value="click" id="button">
</body>
</html>
```

HTML요소의 <INPUT>, <OUTPUT> 태그 사용

- <input> 태그는 사용자에게 입력을 받을 수 있는 필드를 생성하는 것, </input> 태그가 없음
 - **type (필수속성)**
 - ✓ 어떤 타입으로 입력을 받을 것인가 속성값으로 결정
 - ✓ <input type="text">
 - ✓ 속성값은 text외에 password, radio, checkbox 등
 - ✓ number 지정시 숫자만 입력이 가능, 숫자가 아닌 다른 타입은 표시되지 않음
 - value 속성 : 입력태그의 초기값을 설정
 - Id 속성 : 하나의 페이지 안에서 이름이 유일해야하고, 화면을 구성하는 모든 것들에게 따로따로 접근할 때 이용
- <output> 태그는 스크립트 등에 의해 수행된 계산의 결과나 사용자의 액션에 의한 결과를 나타낼 때 사용

HTML요소의 <INPUT>, <OUTPUT> 태그 사용

- 다음 화면과 같이 HTML 페이지를 구성해보자.



체질량지수 계산하기

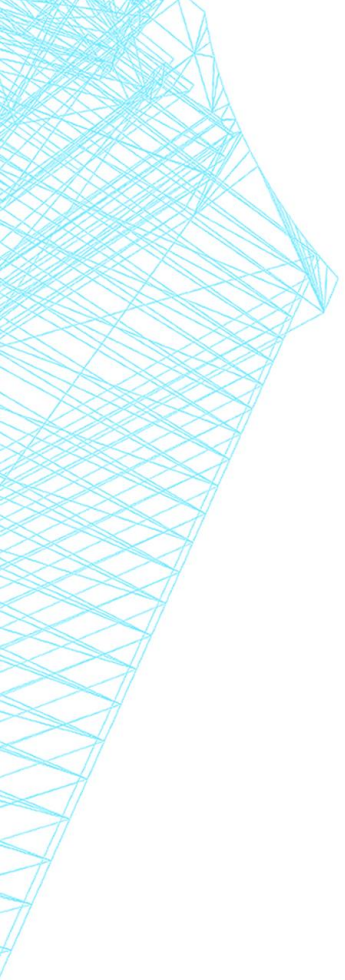
← → ↻ ⓘ 파일 | C:/test/javascript/modern_jav

키: m

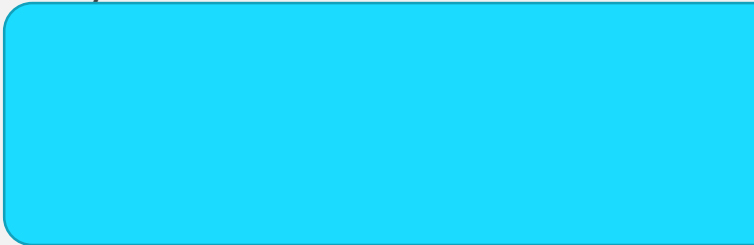
몸무게: kg

당신의 체질량지수는 ? 입니다

계산



```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>체질량지수 계산하기</title>
  <script>

  </script>
</head>
<body>
  
</body>
</html>
```

document.getElementById().value 사용

- getElementById() 함수를 사용해 웹 페이지에 있는 엘리먼트의 객체를 가져옴
- 이 함수는 엘리먼트의 데이터를 직접 잡아내는 대신 자바스크립트 객체를 사용해서 HTML 필드 자체를 제공함
- 따라서 각 필드의 value 프로퍼티를 통해 데이터에 접근할 수 있음.
- 형식) document.getElementById(id명).value

```
var h = parseFloat(document.getElementById("height").value);
```

```
<body>
  <p>키: <input type="number" id="height"> m</p>
  <p>몸무게: <input type="number" id="weight"> kg</p>
  <p>당신의 체질량지수는 <output id="bmi">?</output> 입니다</p>
  <input type="button" id="button" value="계산">
</body>
```

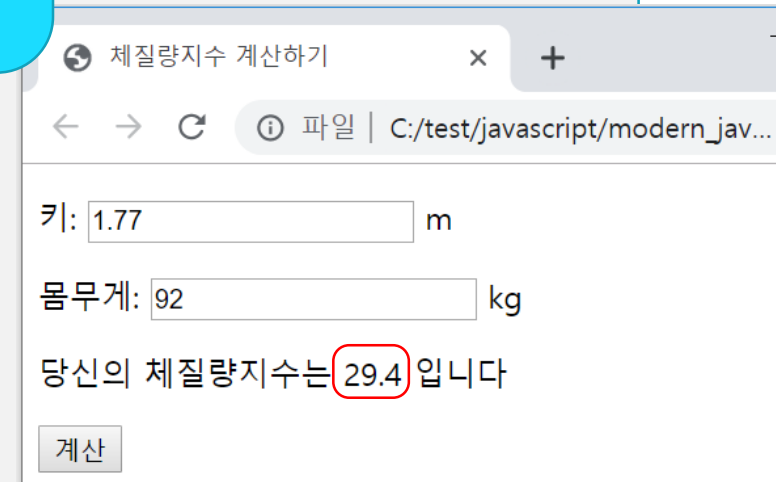

HTML요소의 innerHTML 프로퍼티 사용

- 요소객체의 **innerHTML** 프로퍼티
 - 그 HTML요소의 내용을 가리키며
 - 이로써 HTML요소의 내용을 읽거나 쓸수 있다.

```
var bmi = document.getElementById("bmi");  
bmi.innerHTML = (w/h/h).toFixed(1);
```

```
<body>  
  <p>키: <input type="number" id="height"> m</p>  
  <p>몸무게: <input type="number" id="weight"> kg</p>  
  <p>당신의 체질량지수는 <output id="bmi">?</output> 입니다</p>  
  <input type="button" id="button" value="계산">  
</body>
```

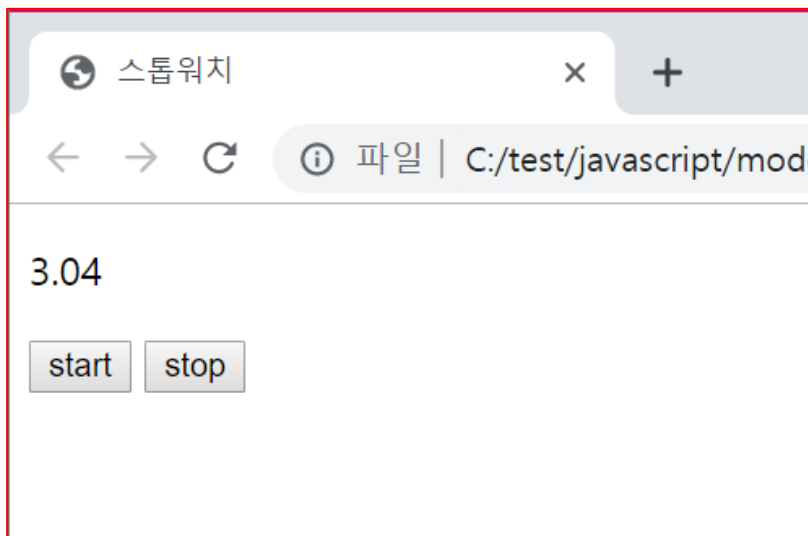
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>체질량지수 계산하기</title>
  <script>
    [Redacted JavaScript Code]
  </script>
</head>
<body>
  <p>키: <input type="number" id="height"> m</p>
  <p>몸무게: <input type="number" id="weight"> kg</p>
  <p>당신의 체질량지수는 <output id="bmi">?</output> 입니다</p>
  <input type="button" id="button" value="계산">
</body>
</html>
```



The screenshot shows a web browser window with the title "체질량지수 계산하기". The address bar shows the file path "C:/test/javascript/modern_jav...". The page content includes three input fields: "키: 1.77 m", "몸무게: 92 kg", and "당신의 체질량지수는 29.4 입니다". The value "29.4" is circled in red. Below the inputs is a "계산" button.

복습문제

- start버튼을 누르면 0.01 초마다 경과한 시간을 표시
- 타이머함수 활용
 - setInterval()함수 : 지정한 시간마다 반복실행하는 함수
 - clearInterval()함수 : 타이머를 중지하는 함수



타이머 함수 예제

5초마다 경고 메시지를 콘솔에 출력하는 예제

```
var time = 0;
playAlert = setInterval(function() {
    time+=5;
    console.log(time,"초 경과");
    if(time==20) clearInterval(playAlert);
}, 5000);
```

```
C:\Program Files\nodejs\node.exe --ir
Debugger listening on ws://127.0.0.1:
For help, see: https://nodejs.org/en/
Debugger attached.
5 초 경과
10 초 경과
15 초 경과
20 초 경과
```



```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>스톱워치</title>
  <script>
    (function() {
      // 스톱워치 관련 JavaScript 코드가 여기에 들어갑니다.
    })();
  </script>
</head>
<body>
  <p id="display">0.00</p>
  <input id="start" type="button" value="start">
  <input id="stop" type="button" value="stop">
</body>
</html>
```


ES2015+ P.56

ES2015 이전에는 var로 변수를 선언

- ES2015부터는 const와 let이 대체
- 가장 큰 차이점: 블록 스코프(var은 함수 스코프)

```
if (true) {  
  var x = 3;  
}  
console.log(x); // 3
```

```
if (true) {  
  const y = 3;  
}  
console.log(y); // Uncaught ReferenceError: y is not defined
```

기준: 함수 스코프(function() {}이 스코프의 기준점)

- 다른 언어와는 달리 if나 for, while은 영향을 미치지 못함
- const와 let은 함수 및 **블록({})에도 별도의 스코프를 가짐**

CONST, LET

const는 상수

- 상수에 할당한 값은 다른 값으로 변경 불가
- 변경하고자 할 때는 **let**으로 변수 선언
- 상수 선언 시부터 초기화가 필요함
- 초기화를 하지 않고 선언하면 에러

```
const a = 0;  
a = 1; // Uncaught TypeError: Assignment to constant variable.  
let b = 0;  
b = 1; // 1
```

```
const c; // Uncaught SyntaxError: Missing initializer in const declaration
```

템플릿 문자열

문자열을 합칠 때 + 기호때문에 지저분함

- ES2015부터는 ` (백틱) 사용 가능
- 백틱 문자열 안에 \${변수} 처럼 사용

```
var num1 = 1;  
var num2 = 2;  
var result = 3;  
var string1 = num1 + ' 더하기 ' + num2 + '는 \'' + result + '\'';  
console.log(string1); // 1 더하기 2는 '3'
```

```
const num3 =1;  
const num4=2;  
const result2=3;  
const string2=`${num3} 더하기 ${num4}는 '${result2}'`;  
console.log(string2);
```

1 더하기 2는 '3'

객체 리터럴

- ES5 시절의 객체 표현 방법
 - 속성 표현 방식에 주목

```
var sayNode = function() {  
  console.log('Node');  
};  
var es = 'ES';  
var oldObject = {  
  sayJS: function() {  
    console.log('JS');  
  },  
  sayNode: sayNode,  
};  
oldObject[es + 6] = 'Fantastic';  
  
oldObject.sayNode(); // Node  
oldObject.sayJS(); // JS  
console.log(oldObject.ES6); // Fantastic
```



- ES5+ 훨씬 간결한 문법으로 객체 리터럴 표현 가능
 - 객체의 메서드에 :function을 붙이지 않아도 됨
 - { sayNode: sayNode }와 같은 것을 { sayNode }로 축약 가능
 - [변수 + 값] 등으로 동적 속성명을 객체 속성 명으로 사용 가능

```
const sayNode = function(){  
  console.log('Node');  
};  
const es = 'ES';  
const newobject = {  
  sayJS(){  
    console.log('JS');  
  },  
  sayNode,  
  [es + 6]: 'Fantastic',  
};  
newobject.sayNode();  
newobject.sayJS();  
console.log(newobject.ES6);
```

화살표 함수

- add1, add2, add3, add4는 같은 기능을 하는 함수
 - function 선언 대신 => 기호로 함수 선언
 - add2: add1을 화살표 함수로 나타낼 수 있음
 - add3: 함수의 본문이 return만 있는 경우 return 생략
 - add4: return이 생략된 함수의 본문을 소괄호로 감싸줄 수 있음
 - not1과 not2도 같은 기능을 함

```
function add1(x, y) {  
  return x + y;  
}
```

```
const add2 = (x, y) => {  
  return x + y;  
};
```

```
const add3 = (x, y) => x + y;
```

```
const add4 = (x, y) => (x + y);
```

```
function not1(x) {  
  return !x;  
}
```

```
const not2 = x => !x;
```

화살표 함수

화살표 함수가 기존 function() {}을 대체하는 건 아님(this가 달라짐)

- logFriends 메서드의 this 값에 주목
- forEach의 function의 this와 logFriends의 this는 다름
- that이라는 중간 변수를 이용해서 logFriends의 this를 전달

```
var relationship1 = {
  name: 'zero',
  friends: ['nero', 'hero', 'xero'],
  logFriends: function() {
    var that = this; // relationship1을 가리키는 this를 that에 저장
    this.friends.forEach(function(friend) {
      console.log(that.name, friend);
    });
  },
};
relationship1.logFriends();
```


NEW 연산자 이용해 객체 생성하기

```
function Card(suit, rank){  
    this.suit = suit;  
    this.rank = rank;  
}
```

```
var card1 = new Card("하트","A");  
console.log(card1);
```

```
> function Card(suit, rank){  
    this.suit = suit;  
    this.rank = rank;  
}  
  
var card1 = new Card("하트","A");  
console.log(card1);  
  
▶ Card {suit: "하트", rank: "A"}  
◀ undefined  
> |
```