
HTML에서 웹앱까지

8주차_01

한 동 대 학 교
김군오 교수

학습 목표: 자바스크립트 함수 알아보기

학습내용:

1. 함수 정의
2. 함수 호출
3. 배열 객체의 함수
4. 이벤트

자바스크립트 함수

- 함수의 정의
 - 특정 기능을 하는 구문(알고리즘, 로직)을 독립된 형태로 만드는 것
 - 변수에 저장할 수 있는 한가지 자료형
- 함수 구조 형식

```
var {함수명} = function() { // 저장할 코드 }
```

 - 축약 형식 :

```
function {함수명}() { // 저장할 코드 }
```
- 함수 실행
 - {함수명}() 로 저장된 코드를 실행

첫번째 함수 만들기

```
1 var sayHello
```

첫번째 함수 만들기

```
1  var sayHello = function() {  
2  
3  }
```

첫번째 함수 만들기

```
1  var sayHello = function() {  
2      console.log('Hello~');  
3  }
```

첫번째 함수 만들기

```
1  var sayHello = function() {  
2    |   console.log('Hello~');  
3  }  
4  
5  sayHello();  
6  sayHello();  
7  sayHello();
```

첫번째 함수 만들기

< undefined

```
> var sayHello = function() {  
    console.log('Hello~');  
}
```

```
sayHello();  
sayHello();  
sayHello();
```

Hello~

VM400:2

Hello~

VM400:2

Hello~

VM400:2

< undefined

함수 축약형식

```
1  var sayHello = function() {  
2    |   console.log('Hello~');  
3  }  
4  
5  // 같은 코드  
6  function sayHello() {  
7    |   console.log('Hello~');  
8  |   }  
    }
```

함수에 값 전달하기

- 함수에 저장된 코드를 실행할 때 값을 전달할 수 있음
- 매개변수 사용 함수 형식

function {함수명}(값1, 값2, ...) {}

- 값1, 값2, ... 매개변수라고 함
- 코드 블록 안에서 전달 받은 값 사용 가능
- 함수에 전달하는 값을 함수 인자라고 함

제곱을 구하는 함수 만들기

```
1  function square(x) {  
2  
3  }
```

제곱을 구하는 함수 만들기

```
1 function square(x) {  
2     console.log(x * x);  
3 }
```

제곱을 구하는 함수 만들기

```
1 function square(x) {  
2     console.log(x * x);  
3 }  
4  
5 square(2);  
6 square(5);  
7 square(0.1);
```

제곱을 구하는 함수 만들기

```
> function square(x) {  
    console.log(x * x);  
}
```

```
square(2);  
square(5);  
square(0.1);
```

4

VM416:2

25

VM416:2

0.010000000000000002

VM416:2

← undefined

값을 반환하는 함수

- 함수의 실행 결과를 반환
- 함수 코드 블록 안에서 `return {값}` 형식 사용
- 함수 형식

```
function {함수명}(값1, 값2, ...) {  
    ...  
    return {값}  
}
```

- 반환된 값은 다른 표현식들과 같은 취급

값을 반환하는 함수

- 예제
- 제곱을 구하는 함수의 값을 반환하도록 변경
- 2의 제곱의 제곱 구하기

값을 반환하는 함수

```
1 function square(x) {  
2     return x * x;  
3     // 실행되지 않음  
4     console.log(x);  
5 }  
6  
7 var value = square(4);  
8 console.log(value);  
9 console.log(square(square(2)));
```

값을 반환하는 함수

```
1 function square(x) {  
2     return x * x;  
3     // 실행되지 않음  
4     console.log(x);  
5 }  
6  
7 var value = square(4);  
8 console.log(value);  
9 console.log(square(square(2)));
```

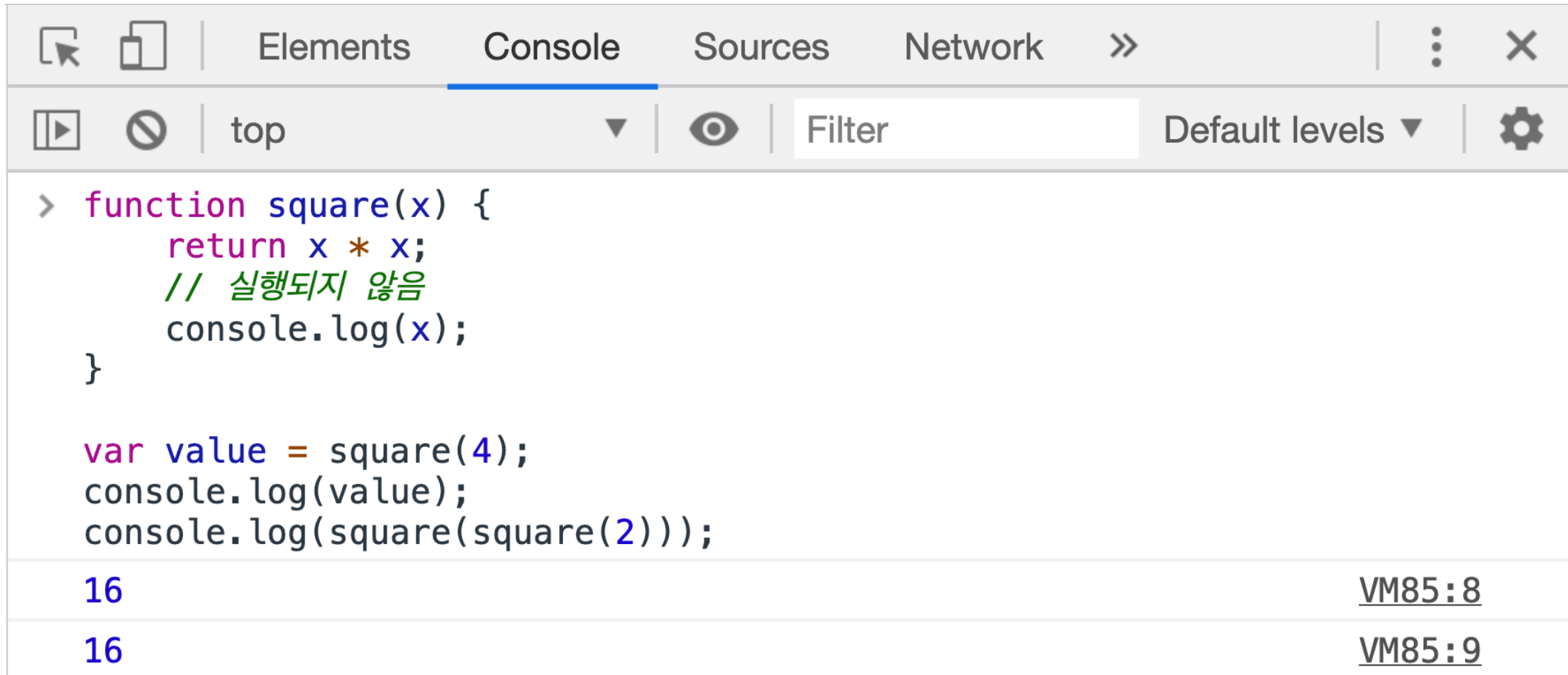
값을 반환하는 함수

```
1 function square(x) {  
2     return x * x;  
3     // 실행되지 않음  
4     console.log(x);  
5 }  
6  
7 var value = square(4);  
8 console.log(value);  
9 console.log(square(square(2)));
```

값을 반환하는 함수

```
1 function square(x) {  
2     return x * x;  
3     // 실행되지 않음  
4     console.log(x);  
5 }  
6  
7 var value = square(4);  
8 console.log(value);  
9 console.log(square(square(2)));
```

함수에 함수 전달하기



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays the following JavaScript code and its execution results:

```
> function square(x) {  
    return x * x;  
    // 실행되지 않음  
    console.log(x);  
}  
  
var value = square(4);  
console.log(value);  
console.log(square(square(2)));
```

The execution results are shown in a table below the code:

16	<u>VM85:8</u>
16	<u>VM85:9</u>

함수를 인자로 전달받는 함수

- 함수의 인자로 함수명을 전달
- 함수 내부에서 전달받은 함수를 실행

함수를 인자로 전달받는 함수

```
1 function operator(x, y, func) {  
2     return func(x, y);  
3 }  
4  
5 function add(a, b) { return a + b; }  
6 function subtract(a, b) { return a - b; }  
7  
8 console.log(operator(4, 2, add));  
9 console.log(operator(4, 2, subtract));
```

함수를 인자로 전달받는 함수

```
1 function operator(x, y, func) {  
2     return func(x, y);  
3 }  
4  
5 function add(a, b) { return a + b; }  
6 function subtract(a, b) { return a - b; }  
7  
8 console.log(operator(4, 2, add));  
9 console.log(operator(4, 2, subtract));
```


함수를 인자로 전달받는 함수

```
1 function operator(x, y, func) {  
2     return func(x, y);  
3 }  
4  
5 function add(a, b) { return a + b; }  
6 function subtract(a, b) { return a - b; }  
7  
8 console.log(operator(4, 2, add));  
9 console.log(operator(4, 2, subtract));
```

함수를 인자로 전달받는 함수

```
1 function operator(x, y, func) {  
2     return func(x, y);  
3 }  
4  
5 function add(a, b) { return a + b; }  
6 function subtract(a, b) { return a - b; }  
7  
8 console.log(operator(4, 2, add));  
9 console.log(operator(4, 2, subtract));
```

함수를 인자로 전달받는 함수

```
1 function operator(x, y, func) {  
2     return func(x, y);  
3 }  
4  
5 function add(a, b) { return a + b; }  
6 function subtract(a, b) { return a - b; }  
7  
8 console.log(operator(4, 2, add));  
9 console.log(operator(4, 2, subtract));
```

함수를 인자로 전달받는 함수

```
> function operator(x, y, func) {  
    return func(x, y);  
}  
  
function add(a, b) { return a + b; }  
function subtract(a, b) { return a - b; }  
  
console.log(operator(4, 2, add));  
console.log(operator(4, 2, subtract));
```

6

VM119:8

2

VM119:9

함수를 인자로 전달받는 함수

console.log(operator(4, 2, add)); //8행



return add(4, 2); //2행



Add(4, 2) //5행
return 6;



return 6; //2행



console.log(6) //8행



6 출력

함수를 인자로 전달받는 함수

```
> function operator(x, y, func) {  
    return func(x, y);  
}  
  
function add(a, b) { return a + b; }  
function subtract(a, b) { return a - b; }  
  
console.log(operator(4, 2, add));  
console.log(operator(4, 2, subtract));
```

6

VM119:8

2

VM119:9

setTimeout

- 형식

setTimeout(함수, 지연 시간);

- 특정 시간이 흐른 후 코드를 실행하고 싶을 때 사용
- 첫번째 인자로 실행할 함수를 받고 두번째 인자로 지연될 시간을 밀리초(1/1000) 단위로 받음

setTimeout

```
1  <!DOCTYPE html>
2  <html>
3      <head><title>8-1</title></head>
4      <body>
5          <p id="text">쿠폰번호: 123456</p>
6          <script>
7              var element = document.getElementById('text');
8              setTimeout(function() {
9                  element.innerText = '삭제됨';
10             }, 3000);
11          </script>
12      </body>
13  </html>
```


setTimeout

```
1  <!DOCTYPE html>
2  <html>
3      <head><title>8-1</title></head>
4      <body>
5          <p id="text">쿠폰번호: 123456</p>
6          <script>
7              var element = document.getElementById('text');
8              setTimeout(function() {
9                  element.innerText = '삭제됨';
10             }, 3000);
11          </script>
12      </body>
13  </html>
```

setTimeout

```
1  <!DOCTYPE html>
2  <html>
3      <head><title>8-1</title></head>
4      <body>
5          <p id="text">쿠폰번호: 123456</p>
6          <script>
7              var element = document.getElementById('text');
8              setTimeout(function() {
9                  element.innerText = '삭제됨';
10             }, 3000);
11          </script>
12      </body>
13  </html>
```

setTimeout

쿠폰번호: 123456

삭제됨

Array 객체의 함수

함수명	설명
Array.forEach	배열의 요소를 인자로 전달받은 함수를 배열의 길이만큼 호출
Array.map	forEach 와 비슷하지만 전달받은 함수의 반환 결과들로 이루어진 새 배열을 반환
Array.filter	forEach 와 비슷하지만 전달받은 함수의 반환 결과가 참인 요소들 로만 이루어진 새 배열을 반환

Array 객체의 함수

함수명	설명
Array.forEach	배열의 요소를 인자로 전달받은 함수를 배열의 길이만큼 호출
Array.map	forEach 와 비슷하지만 전달받은 함수의 반환 결과들로 이루어진 새 배열을 반환
Array.filter	forEach 와 비슷하지만 전달받은 함수의 반환 결과가 참인 요소들 로만 이루어진 새 배열을 반환

Array 객체의 함수

함수명	설명
Array.forEach	배열의 요소를 인자로 전달받은 함수를 배열의 길이만큼 호출
Array.map	forEach 와 비슷하지만 전달받은 함수의 반환 결과들로 이루어진 새 배열을 반환
Array.filter	forEach 와 비슷하지만 전달받은 함수의 반환 결과가 참인 요소들 로만 이루어진 새 배열을 반환

Array 객체의 함수

함수명	설명
Array.forEach	배열의 요소를 인자로 전달받은 함수를 배열의 길이만큼 호출
Array.map	forEach 와 비슷하지만 전달받은 함수의 반환 결과들로 이루어진 새 배열을 반환
Array.filter	forEach 와 비슷하지만 전달받은 함수의 반환 결과가 참인 요소들 로만 이루어진 새 배열을 반환

Array.forEach

```
1 var balls = [ '축구공', '야구공', '배구공', '농구공' ];
2 balls.forEach(throwBall);
3
4 function throwBall(ball) {
5     // 공을 던지는 함수
6     console.log(ball + '을(를) 던졌다!');
7 }
```


Array.forEach

```
1 var balls = [ '축구공', '야구공', '배구공', '농구공' ];
2 balls.forEach(throwBall);
3
4 function throwBall(ball) {
5     // 공을 던지는 함수
6     console.log(ball + '을(를) 던졌다!');
7 }
```

Array.forEach

```
1 var balls = [ '축구공', '야구공', '배구공', '농구공' ];  
2 balls.forEach(throwBall);  
3  
4 function throwBall(ball) {  
5     // 공을 던지는 함수  
6     console.log(ball + '을(를) 던졌다!');  
7 }
```

Array.forEach

```
1 var balls = [ '축구공', '야구공', '배구공', '농구공' ];
2 balls.forEach(throwBall);
3
4 function throwBall(ball) {
5     // 공을 던지는 함수
6     console.log(ball + '을(를) 던졌다!');
7 }
```

Array.forEach

```
> var balls = [ '축구공', '야구공', '배구공', '농구공' ];  
balls.forEach(throwBall);  
  
function throwBall(ball) {  
    // 공을 던지는 함수  
    console.log(ball + '을(를) 던졌다!');  
}
```

축구공을(를) 던졌다!

VM204:6

야구공을(를) 던졌다!

VM204:6

배구공을(를) 던졌다!

VM204:6

농구공을(를) 던졌다!

VM204:6

Array.map

```
1  var numbers = [ 1, 2, 3, 4, 5 ];  
2  var twice = numbers.map(function(value, index) {  
3      |    return value * 2;  
4  });  
5  
6  console.log(twice);
```

Array.map

```
1 var numbers = [ 1, 2, 3, 4, 5 ];
2 var twice = numbers.map(function(value, index) {
3     |     return value * 2;
4     | });
5
6 console.log(twice);
```

Array.map

```
1  var numbers = [ 1, 2, 3, 4, 5 ];  
2  var twice = numbers.map(function(value, index) {  
3      |     return value * 2;  
4  });  
5  
6  console.log(twice);
```

Array.map

```
1 var numbers = [ 1, 2, 3, 4, 5 ];  
2 var twice = numbers.map(function(value, index) {  
3     |   return value * 2;  
4     | });  
5  
6 console.log(twice);
```


Array.map

```
> var numbers = [ 1, 2, 3, 4, 5 ];  
   var twice = numbers.map(function(value, index) {  
       return value * 2;  
   });
```

```
console.log(twice);
```

```
► (5) [2, 4, 6, 8, 10]
```

VM47:6

Array.filter

```
1 var numbers = [ 1, 2, 3, 4, 5 ];  
2 var even = numbers.filter(function(value, index) {  
3     |     return value % 2 == 0;  
4 });  
5  
6 console.log(even);
```

Array.filter

```
1 var numbers = [ 1, 2, 3, 4, 5 ];
2 var even = numbers.filter(function(value, index) {
3   | return value % 2 == 0;
4 });
5
6 console.log(even);
```

Array.filter

```
> var numbers = [ 1, 2, 3, 4, 5 ];  
   var even = numbers.filter(function(value, index) {  
       return value % 2 == 0;  
   });
```

```
console.log(even);
```

```
► (2) [2, 4]
```

VM98:6

자바스크립트 이벤트

- 사용자의 입력을 처리
- HTML 태그 마다 다양한 이벤트가 발생됨
- click, focus, blur 등
- 이벤트 처리기는 이벤트 발생 시 실행할 함수를 전달받음

이벤트 처리기 등록

- 자바스크립트로 html 요소를 선택한 뒤 addEventListener 함수를 호출
- 형식

addEventListener(이벤트이름, 실행할 함수)

첫 번째 인자-이벤트의 이름

두 번째 인자-이벤트가 발생했을 때 실행할 함수

Click 이벤트

- HTML 요소를 마우스로 클릭하거나 터치 디바이스에서 터치시 발생
- 마우스 이벤트 중 하나로 mousedown, mouseup 이벤트 발생 후에 발생됨
- 버튼 뿐만 아니라 모든 HTML 요소에서 이벤트가 발생되고 이벤트 처리기를 등록할 수 있음

Click 이벤트

```
<body>
  <button id="button">클릭</button>
  <p id="count">0회 클릭</p>
  <script>
    var button = document.getElementById('button');
    var element = document.getElementById('count');

    var count = 0;
    button.addEventListener('click', function() {
      count++;
      element.innerText = count + '회 클릭';
    });
  </script>
</body>
```


Click 이벤트

```
<body>
  <button id="button">클릭</button>
  <p id="count">0회 클릭</p>
  <script>
    var button = document.getElementById('button');
    var element = document.getElementById('count');

    var count = 0;
    button.addEventListener('click', function() {
      count++;
      element.innerText = count + '회 클릭';
    });
  </script>
</body>
```

Click 이벤트

```
<body>
  <button id="button">클릭</button>
  <p id="count">0회 클릭</p>
  <script>
    var button = document.getElementById('button');
    var element = document.getElementById('count');

    var count = 0;
    button.addEventListener('click', function() {
      count++;
      element.innerText = count + '회 클릭';
    });
  </script>
</body>
```

Click 이벤트

```
<body>
  <button id="button">클릭</button>
  <p id="count">0회 클릭</p>
  <script>
    var button = document.getElementById('button');
    var element = document.getElementById('count');

    var count = 0;
    button.addEventListener('click', function() {
      count++;
      element.innerText = count + '회 클릭';
    });
  </script>
</body>
```

Click 이벤트

클릭

0회 클릭

클릭

3회 클릭

요약

- 함수 정의
- 함수 호출
- 배열 객체의 함수
- 이벤트

차시 예고

- 8-2 : 자바스크립트 조건문 알아보기
 - if 문
 - if ~ else 문
 - if ~ else if ~ else 문
 - Switch문
 - break문

강의를 마치겠습니다
수고하셨습니다

8주차_03 자바스크립트 함수