

15 함수와 이벤트

15-1. 함수 알아보기

■ 함수(functions)

- 반복적으로 실행되는 구문들을 미리 정의하여 호출을 통해 수행하는 방법

- 프로그램의 모듈화와 코드의 재사용 용이
- 유지보수 용이

- 함수의 구분

- 내장 함수

자바스크립트 엔진에 포함되어 있어 자바스크립트가 기본적으로 제공하는 함수로 별도의 선언 없이 호출해 사용 가능

- 사용자 정의 함수

자바스크립트가 제공하지 않은 함수를 사용자가 직접 작성하는 함수

- 함수의 생성(선언)

- function 키워드를 사용해 사용자 함수를 생성함

- 선언 함수와 익명 함수가 있음

선언 함수 : 이름을 가지는 함수 (함수를 선언할 때 이름을 지정)

익명 함수 : 이름을 가지지 않는 함수

■ 선언 함수의 선언과 호출

• 선언 함수의 선언

- 함수를 선언한다는 것은 함수를 생성한다는 것을 말함

```
function 함수명() {  
    함수의 실행 문장들;  
}
```

• 선언 함수의 호출

- 함수의 이름으로 호출

```
함수명();
```

```
chapter15 > <> function1.html > ...  
1  <!DOCTYPE html>  
2  <html lang="en">  
3  <head>  
4  |   <meta charset="UTF-8">  
5  </head>  
6  <body>  
7  |   <script>  
8  |   |  
9  |       function addNumber() {  
10 |           let num1 = 20;  
11 |           let num2 = 30;  
12 |           let sum = num1 + num2;  
13 |           alert(sum);  
14 |       }  
15 |   </script>  
16 </body>  
17 |   <script>  
18 |       addNumber();  
19 |   </script>  
20 </html>
```

127.0.0.1:5500 내용:

50

확인

15-2. var를 사용한 변수의 특징

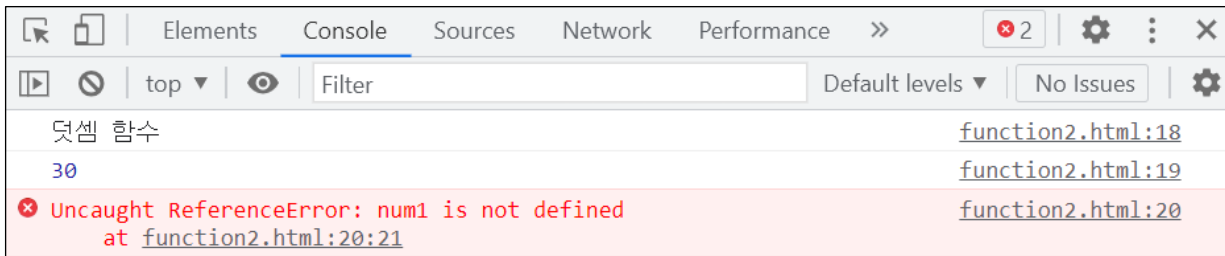
■ 변수의 적용 범위

• 지역 변수(Local Variable)

- 함수 내에서만 사용할 수 있는 변수
- ECMA5에서 선언 방법
var 키워드를 사용해 함수 내부에 선언

• 전역 변수(Global Variable)

- 문서 전체에서 사용할 수 있는 변수
- ECMA5에서 선언 방법
var 키워드를 사용해 함수 외부에 선언
함수 내부에서 선언할 경우 var 키워드 사용하지 않고 선언

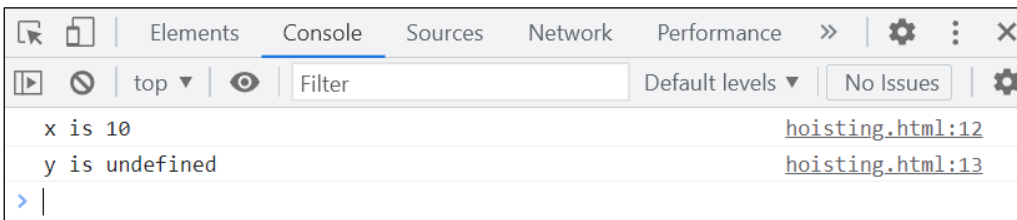


```
chapter15 > <> function2.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5  </head>
6  <body>
7    <script>
8      var str = "덧셈 함수"
9
10     function addNumber() {
11       var num1 = 10;
12       var num2 = 20;
13       sum = num1+num2;
14     }
15
16     addNumber();
17
18     console.log(str);
19     console.log(sum);
20     console.log(num1);
21   </script>
22 </body>
23 </html>
```

■ 변수 호이스팅(Hoisting)

- 변수를 선언하기 이전에 참조하는 것을 변수 호이스팅이라고 함
 - var 변수는 호이스팅을 허용함
선언하기 전에 참조하면 undefined로 지정되지만 오류는 발생하지 않음
 - 프로그램 실행에는 오류가 발생할 수 있음

프로그램 종료



```
chapter15 > <> hoisting.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  </head>
6  <body>
7  |   <script>
8  |       |
9  |       var x= 10;
10 |
11 |       function displayNumber() {
12 |           console.log("x is " + x);
13 |           console.log("y is " + y);
14 |           var y= 20;
15 |       }
16 |
17 |       displayNumber();
18 |
19 |       document.write("프로그램 종료");
20 |
21 |   </script>
22 </body>
23 </html>
```

■ var 변수의 재선언과 재할당

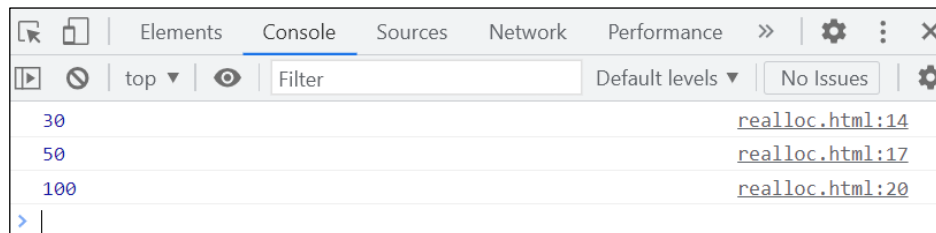
• 재선언

- 이미 선언되어 있는 변수를 다시 선언하는 것
var 키워드는 재선언을 허용함

• 재할당

- 다른 값을 가진 변수의 값을 다른 값으로 변경하는 것
var 키워드는 재할당을 허용함

프로그램 정상 종료



```
chapter15 > <> realloc.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  </head>
6  <body>
7  |   <script>
8  |       function addNumber(num1, num2) {
9  |           return num1 + num2;
10 |       }
11
12 |       var sum = addNumber(10, 20);    // 선언
13 |       console.log(sum);
14
15 |       sum = 50;                      // 재할당
16 |       console.log(sum);
17
18 |       var sum = 100;                 // 재선언
19 |       console.log(sum);
20
21 |       document.write("프로그램 정상 종료");
22 |   </script>
23 </body>
24 </html>
```


15-3. let과 const의 등장

■ ECMA6에서 var 키워드를 사용하지 않는 이유

- 함수 레벨 스코프(function-level scope)
 - 함수의 코드 블록을 스코프로 인정
let과 const는 블록 레벨 스코프(Block-level-scope)임
- var 키워드 생략 허용
 - 암묵적 전역 변수를 양산할 가능성이 큼
 - 현대 프로그래밍언어에서는 전역 변수의 사용을 최소화하고 있음
- 변수 재선언 허용
 - 의도하지 않은 변수 값의 변경이 일어날 가능성이 큼
let은 변수의 재선언을 허용하지 않음
- 변수 호이스팅
 - 변수를 선언하기 이전에 참조해도 undefined로 지정되며, 오류는 발생하지 않음
let과 const는 변수 호이스팅을 허용하지 않음(오류 발생)

■ 함수 레벨 스코프와 블록 레벨 스코프

• 함수 레벨 스코프(Function Level Scope)

- 함수 내에서 선언된 변수가 함수 내에서만 유효한 경우
var 키워드는 함수 레벨 스코프를 지원함
- 함수 레벨 스코프는 자주 사용되지 않음

```
for 구문 내의 i : 0  
for 구문 내의 i : 1  
for 구문 내의 i : 2  
for 구문 내의 i : 3  
for 구문 내의 i : 4  
for 구문 내의 i : 5  
for 구문 내의 i : 6  
for 구문 내의 i : 7  
for 구문 내의 i : 8  
for 구문 내의 i : 9  
for 구문 밖의 i : 10
```

```
chapter15 > <> function-level.html > ...  
1 <!DOCTYPE html>  
2 <html lang="en">  
3 <head>  
4 | <meta charset="UTF-8">  
5 </head>  
6 <body>  
7 | <script>  
8 | | function dispNumber(num) {  
9 | | |  
10 | | | for(var i=0; i<num; i++) {  
11 | | | | document.write("for 구문 내의 i : ", i, "<br>");  
12 | | | }  
13 | | | document.write("for 구문 밖의 i : ", i, "<br>");  
14 | | }  
15 | }  
16 |  
17 | dispNumber(10);  
18 |  
19 | </script>  
20 </body>  
21 </html>
```

- 블록 레벨 스코프(Block Level Scope)

- 모든 코드 블록(함수, if 문, for 문, while 문, try/catch 등)에서 선언된 변수는 코드 블록 내에서만 유효
- 코드 블록 외부에서는 참조할 수 없음
- 대부분의 프로그래밍 언어에서 사용

```
for 구문 내의 i : 0  
for 구문 내의 i : 1  
for 구문 내의 i : 2  
for 구문 내의 i : 3  
for 구문 내의 i : 4  
for 구문 내의 i : 5  
for 구문 내의 i : 6  
for 구문 내의 i : 7  
for 구문 내의 i : 8  
for 구문 내의 i : 9
```

```
chapter15 > <> block-level.html > html > body > script > dispNumber  
1  <!DOCTYPE html>  
2  <html lang="en">  
3  <head>  
4  |   <meta charset="UTF-8">  
5  </head>  
6  <body>  
7  |   <script>  
8  |   |   function dispNumber(num) {  
9  |   |   |  
10 |   |   |   for(let i=0; i<num; i++) {  
11 |   |   |   |   document.write("for 구문 내의 i : ", i, "<br>");  
12 |   |   |   |  
13 |   |   |   }  
14 |   |   |   document.write("for 구문 밖의 i : ", i, "<br>");  
15 |   |   }  
16 |   |  
17 |   |   dispNumber(10);  
18 |   |  
19 |   </script>  
20 </body>  
21 </html>
```

■ let 키워드

- ECMA6에서 변수를 선언하기 위해 사용되는 키워드
- 재정의 불가능

```
let data = "aaa";  
let data = "bbb";           // Uncaught SyntaxError: Identifier 'data' has already been declared
```

- 재할당은 가능

```
let data = "aaa";  
data = "bbb";
```

■ const 키워드

- ECMA6에서 상수를 선언하기 위해 사용되는 키워드
- 재정의 불가능

```
const data = "aaa";  
const data = "bbb";    // Uncaught SyntaxError: Identifier 'data' has already been declared
```

- 재할당 불가능

```
const data = "aaa";  
data = "bbb";          // Uncaught TypeError: Assignment to constant variable.
```

- 선언할 때 반드시 초기화해야 함

```
const data              // Uncaught SyntaxError: Missing initializer in const declaration
```

■ let 키워드와 const 키워드의 사용

- 일반적으로 자바스크립트에서는 const를 주로 사용함
 - 값이 재할당될 수 있는 경우에는 let 키워드를 사용함
- 호이스팅은 허용하지 않음
- 가급적 지역 변수를 사용
 - 과도한 전역 변수는 프로그램 수행 과정에서 오류를 발생시킬 수 있음

15-4. 재사용할 수 있는 함수 만들기

■ 매개변수와 인수

• 매개변수가 있는 함수의 정의문

- 함수를 호출할 때 실행에 사용되는 값을 함수에 전달할 수 있음
값을 전달받기 위해 지정한 변수를 매개변수라고 함
함수에 전달하는 값을 인수라고 함
- 인수는 함수를 호출할 때 전달하며, 함수의 매개변수를 통해 전달됨

• 매개변수가 있는 함수의 정의

```
function 함수명( 매개변수1, 매개변수2, ... 매개변수n ) {  
    자바스크립트 코드  
}
```

• 함수의 호출을 통한 인수 전달

```
함수명( 인수1, 인수2, ..., 인수n )
```

■ return 문

• 함수의 수행 결과를 반환

- 함수를 호출한 위치로 반환

• return 문이 있는 함수 정의

```
function 함수명( [매개변수1, 매개변수2, ... 매개변수n] ) {  
    자바스크립트 코드  
    return 반환값  
}
```

• 반환값 추출

```
변수 = 함수명( [인수 리스트] )
```

chapter15 > <> addNumber-1.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5  </head>
6  <body>
7    <script>
8      function addNumber(num1, num2){
9        let sum = num1 + num2;
10       return sum;
11     }
12     let result = addNumber(2, 3);
13
14     document.write("두 수를 더 한 값 : " + result);
15
16   </script>
17 </body>
18 </html>
```

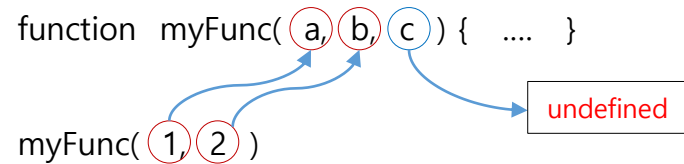
두 수를 더 한 값 : 5

■ 함수의 매개 변수와 전달되는 값의 관계

- 자바스크립트는 함수의 매개변수 개수와 함수로 전달되는 값이 달라도 오류가 발생하지 않음

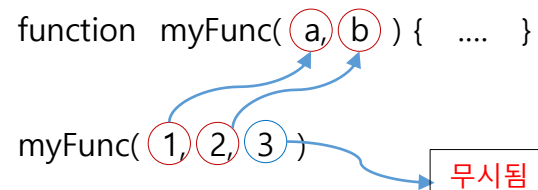
- 함수의 매개변수 개수 > 함수로 전달되는 값의 수

왼쪽부터 매개변수와 값이 1:1 대응되며, 나머지 매개변수는 undefined로 지정됨



- 함수의 매개변수 개수 < 함수로 전달되는 값의 수

왼쪽부터 매개변수와 값이 1:1 대응되며, 나머지 값은 무시됨



■ 기본 매개변수

- 매개 변수에 값이 전달되지 않을 경우, 해당 매개변수에 기본 값을 지정
 - 값이 전달되지 않은 매개변수가 undefined 자료형을 가지는 것이 아니라 지정된 기본 값을 가지게 하는 것을 말함

- 기본 매개 변수의 지정

- ECMA5

함수 본체 내에서 별도로 지정

- ECMA6

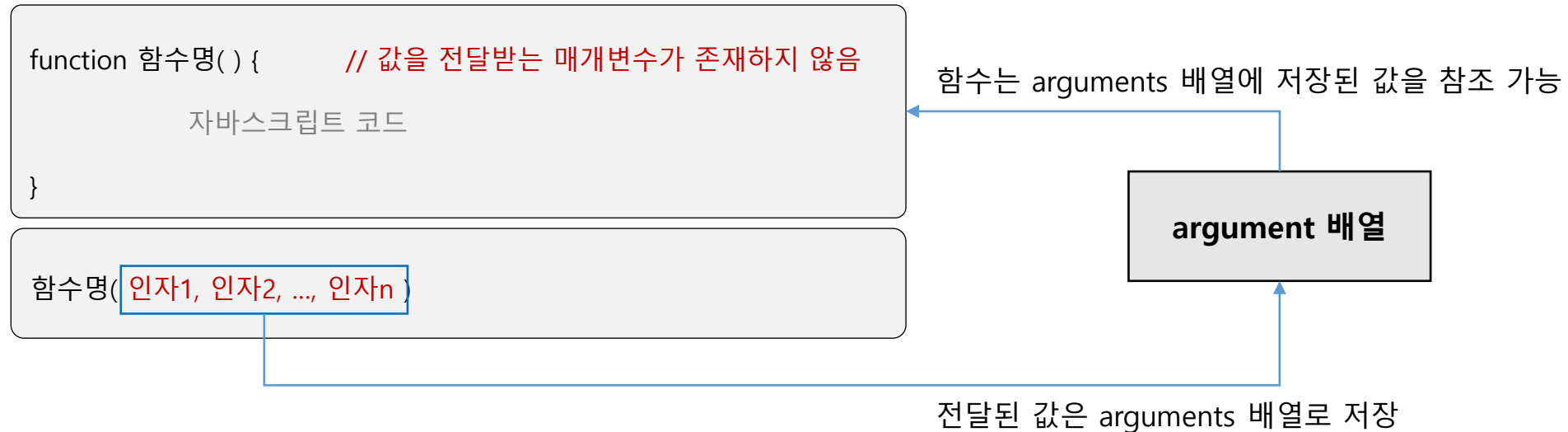
함수의 인자에서 지정

70
210
160

```
chapter15 > <> function-initoal.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  </head>
6  <body>
7  |   <script>
8  |       function multiple(a, b = 5, c = 10) {    // b = 5, c = 10으로 기본값 지정
9  |           return a * b + c;
10 |       }
11
12 |       let result1 = multiple(5, 10, 20); // a = 5, b = 10, c = 20
13 |       document.write(result1, "<br>");
14
15 |       let result2 = multiple(10, 20);      // a = 10, b = 20, c = 10(기본값)
16 |       document.write(result2, "<br>");
17
18 |       let result3 = multiple(30);          // a = 30, b = 5(기본값), c = 10(기본값)
19 |       document.write(result3, "<br>");
20 |   </script>
21 </body>
22 </html>
```

■ argument 매개 변수

- 자바스크립트의 모든 함수가 기본적으로 가지는 변수
 - 일반적으로 함수의 매개변수와 함수에 전달되는 값은 1:1 대응 되어야 함
 - 함수의 매개변수가 없는 상태에서 값을 전달할 수도 있음
- 전달되는 값은 arguments라는 배열로 저장됨
 - 함수는 arguments 배열의 요소를 참조해 전달된 값을 사용할 수 있음



chapter15 > <> function-arg.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  </head>
6  <body>
7  |   <script>
8  |   |
9  |   |   function sum() {
10 |   |   |   let sum = 0;
11 |   |   |
12 |   |   |   document.write("매개변수의 수 : ", arguments.length , "<br>" )
13 |   |   |   for( let i=0; i<arguments.length; i++ ) {
14 |   |   |   |   sum += arguments[i];
15 |   |   |   |   }
16 |   |   |   document.write("합계 : ", sum);
17 |   |   }
18 |   |
19 |   |   sum(10, 20, 30);
20 |   |
21 |   </script>
22 </body>
23 </html>
```

매개변수의 수 : 3
합계 : 60

15-5. 함수 표현식

■ 익명 함수

- 선언 함수와는 다르게 이름을 가지지 않는 함수
 - 함수 자체가 식이므로 함수를 변수에 할당할 수도 있고 다른 함수의 매개변수로 사용할 수도 있음
- 익명 함수의 선언
 - 함수 참조 변수의 값으로 이름을 가지지 않는 함수를 지정

```
함수참조변수 = function( ) {  
    함수의 실행 문장들;  
}
```

```
<script>  
  let myFunc = function( ) {  
    const output = prompt("숫자를 입력하세요.", "숫자");  
    alert(output);  
  }  
</script>
```

- 익명 함수의 호출
 - 함수 참조 변수 이름을 호출

```
함수참조변수명( );
```

```
myFunc( );
```


chapter15 > <> function-anno.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  </head>
6  <body>
7  |   <script>
8  |   |
9  |   |   let sum = function(a, b){
10 |   |   |   return a + b;
11 |   |   |
12 |   |   }
13 |   |   document.write("함수 실행 결과 : " + sum(10, 20) );
14 |   |
15 |   </script>
16 </body>
17 </html>
```

함수 실행 결과 : 30

■ 함수의 재정의

- 같은 이름의 함수를 다른 코드의 내용으로 재정의

- 선언 함수는 새로 정의된 내용으로 대체됨

```
<script>  
    function myFunc( ) { alert('한국'); }  
    function myFunc( ) { alert('대한민국'); }  
    myFunc();  
</script>
```

// 출력됨

- 익명 함수는 재정의 할 수 없음

```
<script>  
    let myFunc = function( ) { alert('한국'); }  
    let myFunc = function( ) { alert('대한민국'); }  
    myFunc();  
</script>
```

// 오류 발생

■ 호이스팅(hoisting)

- 선언 함수는 호이스팅을 지원

- 함수 정의보다 호출이 먼저 나와도 정상적으로 함수를 실행함

```
<script type="text/javascript">  
  myFunc();  
  function myFunc( ) { alert('대한민국'); }  
</script>
```

// 정상 출력

- 익명 함수는 호이스팅을 지원하지 않음

- 함수 정의보다 호출이 먼저 나오면 오류 발생

```
<script type="text/javascript">  
  myFunc();  
  let myFunc = function( ) { alert('대한민국'); }  
</script>
```

// 오류 발생

■ 함수를 다른 함수의 매개변수에 전달

- 함수도 다른 함수의 매개변수로 전달될 수 있음
 - 이때 전달되는 함수를 콜백 함수(Callback Function)라고도 함

```
0 : 함수 호출  
1 : 함수 호출  
2 : 함수 호출  
3 : 함수 호출  
4 : 함수 호출  
5 : 함수 호출  
6 : 함수 호출  
7 : 함수 호출  
8 : 함수 호출  
9 : 함수 호출
```

```
chapter15 > <> function-callback1.html > ...  
1  <!DOCTYPE html>  
2  <html lang="en">  
3  <head>  
4  |   <meta charset="UTF-8">  
5  </head>  
6  <body>  
7  |   <script>  
8  |  
9  |       function callTenTimes ( myFunc ) {  
10 |           for (let i = 0; i < 10; i++) {  
11 |               document.write(i," : ");  
12 |               myFunc();  
13 |           }  
14 |       }  
15 |       function callback() {  
16 |           document.write('함수 호출<br>');  
17 |       };  
18 |       callTenTimes(callback);  
19 |  
20 |   </script>  
21 </body>  
22 </html>
```

- 익명 함수의 매개변수 전달

```
chapter15 > <> function-callback2.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  </head>
6  <body>
7  |   <script>
8  |   |
9  |   |   function callTenTimes ( myFunc ) {
10 |   |   |   for (let i = 0; i < 10; i++) {
11 |   |   |   |   document.write(i, ' : ');
12 |   |   |   |   myFunc();
13 |   |   |   }
14 |   |   }
15 |   |
16 |   |   callTenTimes( function() {
17 |   |   |   document.write('함수 호출<br>');
18 |   |   |   }
19 |   |   );
20 |   |
21 |   </script>
22 </body>
23 </html>
```

```
0 : 함수 호출
1 : 함수 호출
2 : 함수 호출
3 : 함수 호출
4 : 함수 호출
5 : 함수 호출
6 : 함수 호출
7 : 함수 호출
8 : 함수 호출
9 : 함수 호출
```

■ 즉시 실행 함수

- 호출하지 않아도 브라우저에 의해 실행되는 함수
 - 선언 함수나 익명 함수는 선언 후 호출을 통해 실행됨
- 즉시 실행 함수의 형식

```
( function( ) {  
    함수의 실행 문장들;  
} ( ));
```

```
( function( 매개변수 ) {  
    함수의 실행 문장들;  
}( 인자 ));
```

chapter15 > <> function-imm1.html > ...

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  </head>
6  <body>
7  |   <script>
8  |       (function() {
9  |           var userName = prompt("이름을 입력하세요.");
10 |           document.write("안녕하세요?" , userName , "님!");
11 |       }());
12 |
13 |   </script>
14 </body>
15 </html>

```

127.0.0.1:5500 내용:

이름을 입력하세요.

확인

취소

안녕하세요?홍길동님!

chapter15 > <> function-imm2.html > ...

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  </head>
6  <body>
7  |   <script>
8  |       (function(a, b){ // 함수 선언을 위한 매개변수
9  |           sum = a + b;
10 |       }(100, 200)); // 마지막에 함수 실행을 위한 인수
11 |
12 |       document.write("함수 실행 결과 : " + sum);
13 |
14 |   </script>
15 </body>
16 </html>

```

함수 실행 결과 : 300

화살표 함수

- ECMA6 버전부터 제공되는 함수
 - 익명 함수를 간단하게 선언할 수 있음
익명 함수에만 사용할 수 있음
- 화살표 함수의 형식

참조변수 = (매개변수) => { 함수 내용 }

안녕하세요?
홍길동님. 안녕하세요.
30

```
chapter15 > <> function-allow.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5  </head>
6  <body>
7    <script>
8
9      // 매개변수가 없는 경우
10     const result1 = () => { document.write("안녕하세요?<br>") };
11     result1();
12
13     // 매개변수가 하나인 경우
14     const result2 = (user) => { document.write(user, "님. 안녕하세요.<br>")}
15     result2("홍길동");
16
17     // 매개변수가 2개인 경우
18     let result3 = (num1, num2) => { return num1+num2 }
19     document.write(result3(10, 20), "<br>");
20
21   </script>
22 </body>
23 </html>
```


15-6. 이벤트와 이벤트 핸들러

■ 이벤트와 이벤트 핸들러

• 이벤트 (events)

- 마우스를 클릭하거나 키보드의 입력과 같이 사용자가 수행한 행위로 특정 상황이 발생하는 것
[예] 마우스의 클릭(click), 키보드 입력(keypress), 문서의 로드(load) 등
- 자바스크립트에서는 이벤트 이름 앞에 on을 붙여 표현
[예] 마우스의 클릭(onClick), 키보드의 입력(onKeyPress), 문서의 로드(onLoad) 등

• 이벤트 핸들러 (event handlers)

- 특정 이벤트가 발생했을 때 이를 처리하게 위해 수행되는 루틴

on이벤트 = 이벤트핸들러

- 주로 메서드나 함수와 함께 사용
특정한 이벤트 핸들러의 발생에 따른 처리를 메서드나 함수로 수행

표 15-1 마우스 이벤트

종류	설명
click	사용자가 HTML 요소를 클릭할 때 이벤트가 발생합니다.
dblclick	사용자가 HTML 요소를 더블클릭할 때 이벤트가 발생합니다.
mousedown	사용자가 요소 위에서 마우스 버튼을 눌렀을 때 이벤트가 발생합니다.
mousemove	사용자가 요소 위에서 마우스 포인터를 움직일 때 이벤트가 발생합니다.
mouseover	마우스 포인터가 요소 위로 옮겨질 때 이벤트가 발생합니다.
mouseout	마우스 포인터가 요소를 벗어날 때 이벤트가 발생합니다.
mouseup	사용자가 요소 위에 놓인 마우스 버튼에서 손을 뗄 때 이벤트가 발생합니다.

표 15-2 키보드 이벤트

종류	설명
keydown	사용자가 키를 누르는 동안 이벤트가 발생합니다.
keypress	사용자가 키를 눌렀을 때 이벤트가 발생합니다.
keyup	사용자가 키에서 손을 뗄 때 이벤트가 발생합니다.

표 15-3 문서 로딩 이벤트

종류	설명
abort	문서가 완전히 로딩되기 전에 불러오기를 멈췄을 때 이벤트가 발생합니다.
error	문서 가 정확히 로딩되지 않았을 때 이벤트가 발생합니다.
load	문서 로딩이 끝나면 이벤트가 발생합니다.
resize	문서 화면 크기가 바뀌었을 때 이벤트가 발생합니다.
scroll	문서 화면이 스크롤되었을 때 이벤트가 발생합니다.
unload	문서에서 벗어날 때 이벤트가 발생합니다.

표 15-4 폼 이벤트

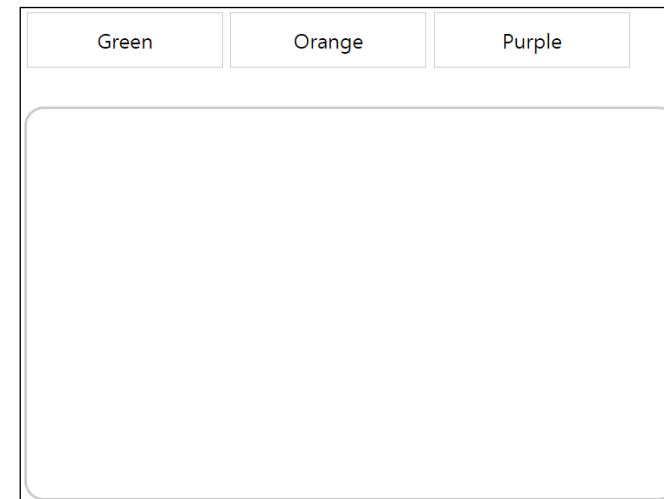
종류	설명
blur	폼 요소에 포커스를 잃었을 때 이벤트가 발생합니다.
change	목록이나 체크 상태 등이 변경되면 이벤트가 발생합니다. <input>, <select>, <textarea> 태그에서 사용합니다.
focus	폼 요소에 포커스가 놓였을 때 이벤트가 발생합니다. <label>, <select>, <textarea>, <button> 태그에서 사용합니다.
reset	폼이 리셋되었을 때 이벤트가 발생합니다.
submit	submit 버튼을 클릭했을 때 이벤트가 발생합니다.

chapter15 > css > # function.css > ...

```
1  a:link, a:visited {
2      color: black; text-decoration: none;
3  }
4  ul {
5      list-style: none;
6      width: 500px;
7      margin: 10px auto;
8      padding: 0;
9  }
10 li {
11     display: inline-block;
12     width: 120px;
13     border: 1px solid #ccc;
14     padding: 10px 15px;
15     font-size: 16px; text-align: center;
16 }
17 #result {
18     width: 500px; height: 300px;
19     margin: 30px auto;
20     border: 2px solid #ccc;
21     border-radius: 15px;
22 }
23 p {
24     width: 80%;
25     padding: 10px;
26     line-height: 2em;
27 }
```

chapter15 > <> event1.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <link rel="stylesheet" href="css/function.css">
6  </head>
7  <body>
8
9      <ul>
10         <li><a href="#" onclick="alert('버튼을 클릭했습니다.')">Green</a></li>
11         <li><a href="#" onclick="alert('버튼을 클릭했습니다.')">Orange</a></li>
12         <li><a href="#" onclick="alert('버튼을 클릭했습니다.')">Purple</a></li>
13     </ul>
14     <div id="result"></div>
15
16 </body>
17 </html>
```



15-7 DOM을 이용한 이벤트 처리

■ 문서 객체 모델(DOM:Document Object Model)

- 문서 객체
 - 요소(태그)를 자바스크립트에서 이용할 수 있는 객체로 만든 경우를 의미
 - 문서 객체의 집합을 문서 객체 모델(DOM)이라고 함
- 표준화된 방법으로 문서 객체에 접근할 수 있는 방법을 제공하기 위해 사용
 - [예] 자바스크립트를 사용해 문서 객체에 접근할 수 있으며, 속성과 스타일을 지정할 수 있음
- 자바 스크립트 모델에서 요소를 가져오는 방법
 - getElementById()
 - getElementsByTagName()
 - querySelector()
 - querySelectotAll()

- 웹 요소를 변수로 지정하고 선언 함수를 적용하는 방법

chapter15 > <> querySelector-2.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  </head>
6  <body>
7
8  |   <button id="change">CHANGE</button>
9  |   <h2>한국교통대학교</h2>
10
11 |   <script>
12 |       let changeBtn = document.querySelector("#change");
13 |       changeBtn.onclick = changeColor;
14 |
15 |       function changeColor() {
16 |           document.querySelector("h2").style.color = "#f00";
17 |       }
18 |   </script>
19
20
21 </body>
22 </html>
```

CHANGE

한국교통대학교

CHANGE

한국교통대학교

- 웹 요소를 변수로 지정하지 않고 선언 함수를 적용하는 방법

chapter15 > <> querySelector-2.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  </head>
6  <body>
7  |
8  |   <button id="change">CHANGE</button>
9  |   <h2>한국교통대학교</h2>
10 |
11 |   <script>
12 |       let changeBtn = document.querySelector("#change").onclick = changeColor;
13 |
14 |       function changeColor() {
15 |           document.querySelector("h2").style.color = "#f00";
16 |       }
17 |   </script>
18 |
19 |
20 </body>
21 </html>
```

CHANGE

한국교통대학교

CHANGE

한국교통대학교

- 익명 함수를 직접 지정하는 방법

```
chapter15 > <> querySelector-3.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  </head>
6  <body>
7
8  |   <button id="change">CHANGE</button>
9  |   <h2>한국교통대학교</h2>
10
11 |   <script>
12 |       |
13 |       |   let changeBtn = document.querySelector("#change").onclick = function () {
14 |       |       |   document.querySelector("h2").style.color = "#f00";
15 |       |       |   }
16 |   </script>
17
18 </body>
19 </html>
```

CHANGE

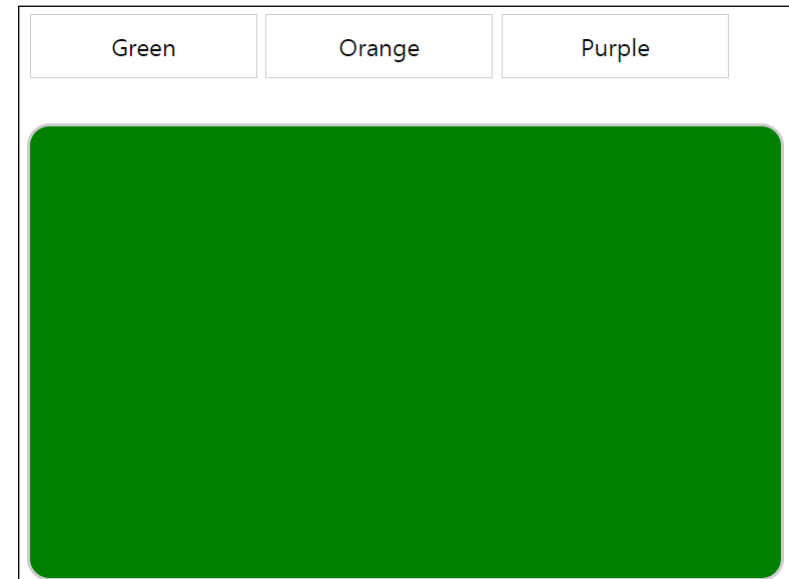
한국교통대학교

CHANGE

한국교통대학교

chapter15 > <> event2.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <link rel="stylesheet" href="css/function.css">
6  </head>
7  <body>
8
9      <ul>
10         <li><a href="#" onclick="changeBg('green')">Green</a></li>
11         <li><a href="#" onclick="changeBg('orange')">Orange</a></li>
12         <li><a href="#" onclick="changeBg('purple')">Purple</a></li>
13     </ul>
14     <div id="result"></div>
15
16     <script>
17         function changeBg(color) {
18             var result = document.querySelector('#result');
19             result.style.backgroundColor = color;
20         }
21     </script>
22
23 </body>
24 </html>
```



chapter15 > css > # event.css > ...

```
1  #item {
2      position:relative;
3      width:500px; height:auto;
4      padding:15px 20px;
5      margin:auto;
6  }
7  button {
8      background-color: rgba(255,255,255,0.7);
9      padding:5px;
10     border:1px solid #ccc;
11     font-size:0.8em;
12 }
13 .over {
14     position:absolute;
15     left:30px;
16     bottom:30px;
17 }
18 .detail {
19     width:400px;
20     text-align:left;
21     line-height:1.8;
22     display:none;
23 }
```

chapter15 > <> detail-1-result.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <link rel="stylesheet" href="css/event.css">
6  </head>
7  <body>
8      <div id="item">
9          
10         <button class="over" id="open" onclick="showDetail()">상세 설명 보기</button>
11         <div id="desc" class="detail">
12             <h4>등심붓꽃</h4>
13             <p>북아메리카 원산으로 각지에서 관상초로 흔히 심고 있는 귀화식물이다. 길가나 잔디밭에서 흔히 볼 수 있다. 아주 작은 씨앗을 무수히
14             많이 가지고 있는데 바람을 이용해 씨앗들을 날려보내거나, 뿌리줄기를 통해 동일한 개체들을 많이 만들어 냄으로써 번식한다. </p>
15             <button id="close" onclick="hideDetail()">상세 설명 닫기</button>
16         </div>
17     </div>
18
19     <script>
20         function showDetail() {
21             document.querySelector('#desc').style.display = "block"; // 상세 설명 부분을 화면에 표시
22             document.querySelector('#open').style.display = "none"; // '상세 설명 보기' 단추를 화면에서 감춤
23         }
24         function hideDetail() {
25             document.querySelector('#desc').style.display = "none"; // 상세 설명 부분을 화면에서 감춤
26             document.querySelector('#open').style.display = "block"; // '상세 설명 보기' 단추를 화면에 표시
27         }
28     </script>
29 </body>
30 </html>
```



등심붓꽃

북아메리카 원산으로 각지에서 관상초로 흔히 심고 있는 귀화식물이다. 길가나 잔디밭에서 흔히 볼 수 있다. 아주 작은 씨앗을 무수히 많이 가지고 있는데 바람을 이용해 씨앗들을 날려보내거나, 뿌리줄기를 통해 동일한 개체들을 많이 만들어 냄으로써 번식한다.

상세 설명 닫기

■ 기본 이벤트 지정과 인라인 이벤트 지정

• 기본 이벤트 지정

- 문서 객체를 생성하고 생성된 문서 객체에 이벤트를 연결하는 방법

이전에 사용했던 방식이지만 현재에서도 많이 사용

하나의 이벤트에 하나의 이벤트 리스너만 연결 할 수 있음

- [예]

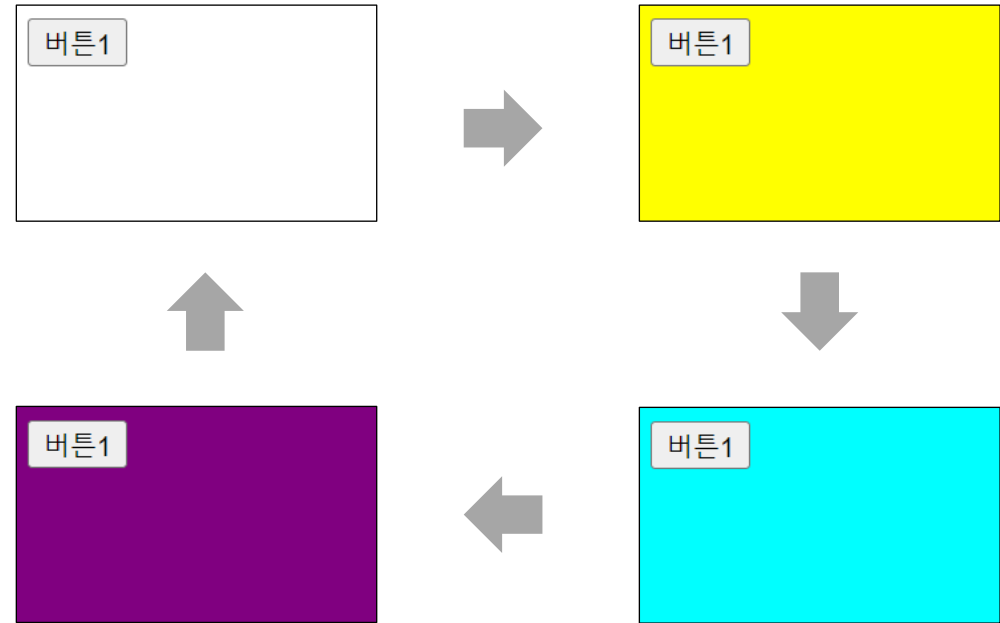
```
<script>
  const header = document.querySelector('header');
  header.onclick = function () {
    alert('클릭');
  };
</script>
```

• 인라인 이벤트 지정

- html 태그 내에 직접 이벤트를 지정하는 방식

```
<button id="btn-1" onclick="alert('클릭')"> 버튼 </button>
```

```
chapter15 > <> event-level0.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <script>
6      const color=["white","yellow","aqua","purple"];
7      let i=0;
8
9      window.onload = function() {
10        const btn = document.querySelector("#btn1");
11        btn.onclick = function(){
12          colorBg();
13        }
14      }
15
16      function colorBg(){
17        i++;
18        if(i>=color.length) {
19          i=0;
20        }
21        const bodyTag=document.querySelector("body");
22        bodyTag.style.backgroundColor=color[i];
23      }
24
25    </script>
26  <body>
27    <button id="btn1">버튼1</button><p>
28  </body>
29  </html>
```



■ 표준 이벤트 모델

- 모든 브라우저 적용될 수 있도록 표준화된 이벤트 지정 방식
- 하나의 문서 객체가 하나 이상의 이벤트 리스너를 가질 수 있는 방법을 제공
 - Level0의 기본 이벤트 모델이나 인라인 이벤트 모델은 한번에 하나의 이벤트 리스너만 가질 수 있음
- 인터넷 익스플로러 9버전 이상과 모든 브라우저에서 사용 가능
 - 인터넷 익스플로러 8 이하 버전에서는 인터넷 익스플로러 이벤트 모델을 사용해야 함
이전에는 IE8 버전 이하를 고려해 프로그래밍하였음
최근 IE8 이하의 버전이 거의 사용되지 않으므로 고려하지 않고 프로그래밍을 수행함
- 이벤트의 연결과 해제
 - 이벤트의 연결
addEventListener() 메서드를 사용
 - 이벤트의 해제
removeEventListener() 메서드를 사용

■ addEventListener()

- 이벤트를 연결하기 위한 메서드

```
요소.addEventListener( "이벤트이름", 이벤트 핸들러 );
```

- 이벤트 이름

- 이벤트 속성을 사용하지 않고 이벤트 이름을 사용
- DOM Level 0에서와는 달리 이벤트 이름 앞에 on을 붙이지 않음

- 이벤트 핸들러

- 익명 함수, 선언 함수, 화살표 함수로 구현 가능

■ removeEventListener()

- 문서에 연결된 이벤트를 제거할 때 사용되는 메서드

```
요소.removeEventListener( "이벤트이름", 이벤트 핸들러 );
```

chapter15 > <> event-level1.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <style>
6          div#box{
7              width:100px; height:100px;
8              background-color:■ #FC9;
9          }
10     </style>
11     <script>
12
13         window.onload = function() {
14
15             let box = document.querySelector("#box");
16             let txt = document.querySelector("#txt");
17             let btn = document.querySelector("#btn");
18
19             function over() {
20                 txt.innerHTML="마우스를 올렸습니다";
21             }
22             box.addEventListener('mouseover', over);
23
24             function out() {
25                 txt.innerHTML="마우스를 내렸습니다";
26             }
27             box.addEventListener('mouseout', out);
```

```
28
29         function clk() {
30             txt.innerHTML="마우스를 클릭했습니다";
31         }
32         box.addEventListener('click', clk);
33
34         function del_clk() {
35             box.removeEventListener('click', clk);
36         }
37         box.addEventListener('click', del_clk);
38
39     }
40
41     </script>
42 <body>
43     <div id="box"></div><p>
44     <div id="txt"></div><p>
45     <input type="button" id="btn" value="클릭 이벤트 제거"></body>
46 </html>
```

수고하셨습니다