

## 4. 함수

4-1. 함수의 기본 형태

4-2. 매개변수 (Parameter)

4-3. 결과 반환 (return)

4-4. 화살표 함수 (Arrow Function)

4-5. Closure

## ■ 함수

- 작업을 수행하기 위한 명령문의 집합
- 프로그램의 기본 구성 요소
- 기본 형태

04-01-기본형태.html

```
function name(param1, param2) {  
    // body ...  
    return;  
}
```

- 하나의 함수는 하나의 작업을 수행하도록 설계

```
function printHello() {  
    console.log('Hello');  
}  
printHello();  
  
function log(message) {  
    console.log(message);  
}  
log('Hello@');
```

## ■ 함수 (Parameter)

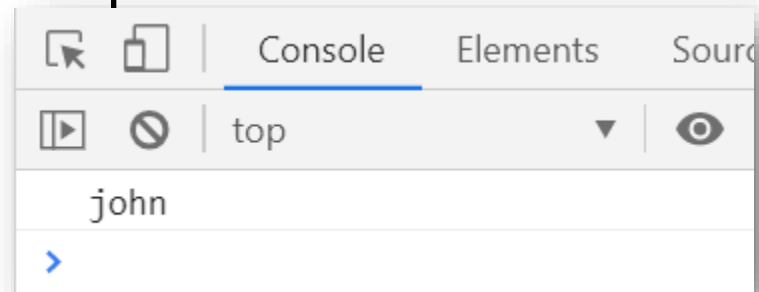
`function changeName(name= ...){}`

... 가

### ● 기본 자료형 : 값 전달

04-02-parameter.html

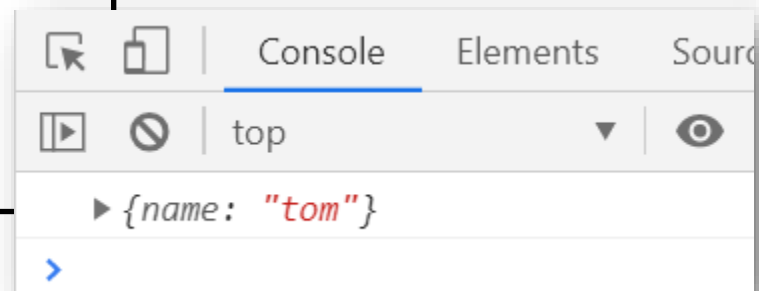
```
function changeName(name) {  
  name = 'tom';  
}  
const name = 'john';  
changeName(name);  
console.log(name);
```



### ● 참조 자료형 (객체) : 참조(주소) 전달

```
function changeNameByObject(obj) {  
  obj.name = 'tom';  
}  
const user = { name: 'john' };  
changeNameByObject(user);  
console.log(user);
```

가



## ■ 함수 (Parameter)

### ● 참조 자료형 사용 오류

04-03-참조자료형오류.html

```
const data = [
  { 'name': 'A', 'score': 77 },
  { 'name': 'B', 'score': 88 },
  { 'name': 'C', 'score': 66 },
  { 'name': 'D', 'score': 55 },
];

function show(data){
  let code = '<ol>';
  data.forEach(value => {
    code += `<li>${value.name},
    ${value.score}, ${value.pass}</li>`;
  });
  code += '</ol>';

  document.querySelector('#list').innerHTML = code;
}

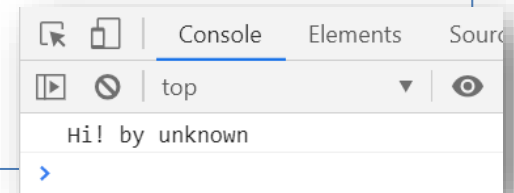
function check(data) {
  const result = [];
  const person = {};
  data.forEach(value => {
    person['name'] = value.name;
    person['score'] = value.score;
    person['pass'] = value.score > 60 ? true : false;
    result.push(person);
  });
  return result;
}
```

## ■ 함수 (Parameter)

### ● 기본값 지정

04-04-default.html

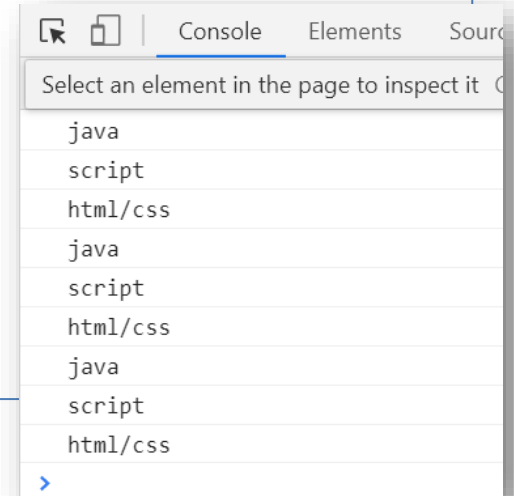
```
function showMessage(message, from='unknown') {  
    console.log(`${message} by ${from}`);  
}  
showMessage('Hi!');
```



### ● 가변인자 (Rest Parameter)

04-05-rest-parameter.html

```
function printAll(...args) {  
    for(let i = 0; i < args.length; i++) {  
        console.log(args[i]);  
    }  
    for(const arg of args) {  
        console.log(arg);  
    }  
    args.forEach((arg) => console.log(arg));  
}  
printAll('java', 'script', 'html/css');
```



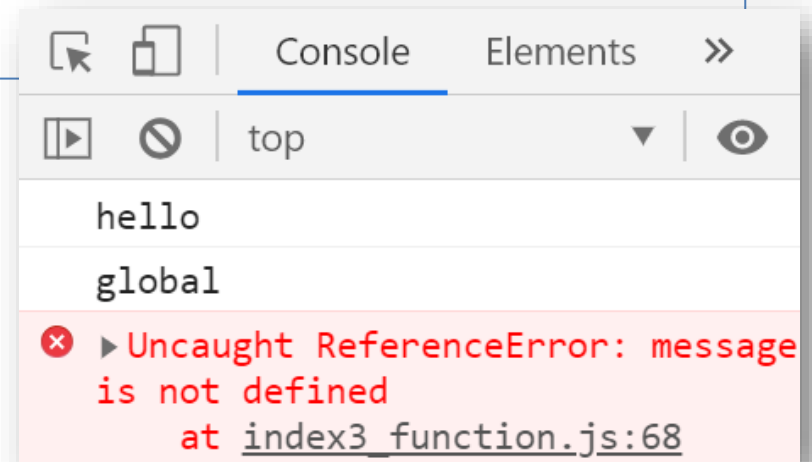
## ■ 함수 (Parameter)

### ● 변수의 유효범위

04-06-global-local.html

```
let globalMessage = 'global'; // global variable
function printMessage() {
  let message = 'hello';
  console.log(message); // local variable
  console.log(globalMessage);
}
printMessage();
console.log(message);
```

현재 범위에서 선언되지 않아서 오류 발생

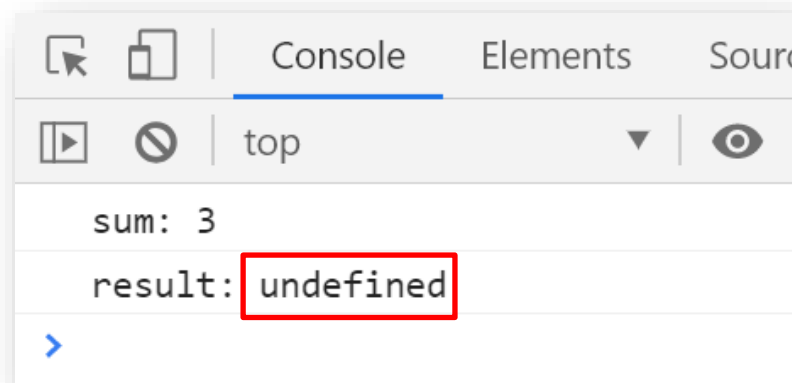


## ■ 함수 (Parameter)

### ● 결과 반환 (return)

04-07-return.html

```
function sum(a, b) {  
  return a + b;  
}  
const result = sum(1, 2); // 3  
console.log(`sum: ${sum(1, 2)}`);  
  
function nothing(a, b) {  
  
}  
console.log(`result: ${nothing(1, 2)}`);
```



## ■ First-class Function

- 함수를 변수처럼 처리

```
const func = function() {};
```

- 변수에 값으로 할당 (\*)

```
const show = func;
```

- 함수를 다른 함수의 인자(argument)로 사용

```
function action(f) {  
}  
action(func);
```

- 다른 함수를 함수의 반환(return) 값으로 사용 가능

```
function outer() {  
  return function inner() {  
  }  
}
```



## ■ First-class Function

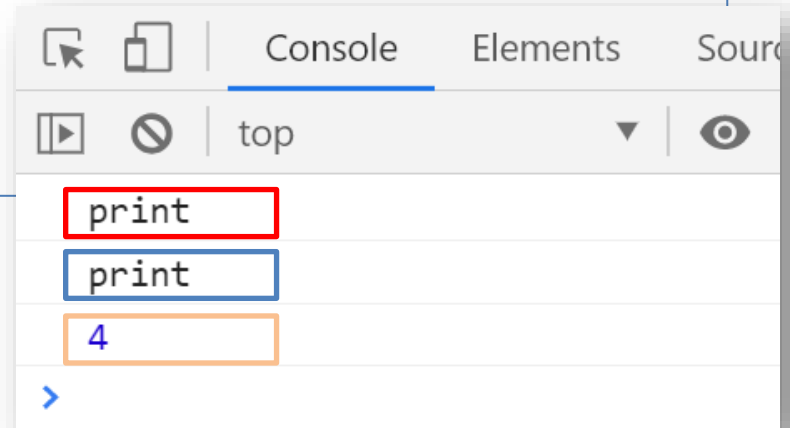
### ● 익명, 변수 값 할당

04-08-function.html

```
const print = function() { // anonymous function
  console.log('print');
}
print();

const printAgain = print;
printAgain();

function sum(a, b) {
  return a + b;
}
const sumAgain = sum;
console.log(sumAgain(1, 3));
```



## ■ First-class Function

### ● 함수를 인자로 사용하여 Callback 처리

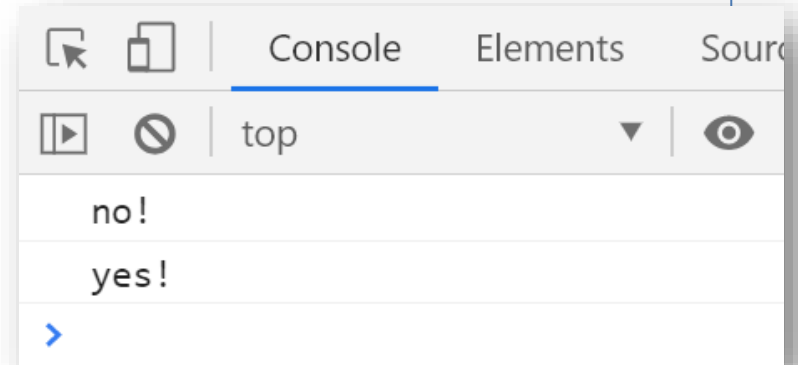
04-09-callback.html

```
function quiz(answer, printYes, printNo) {  
  if(answer === 'love you') {  
    printYes();  
  } else {  
    printNo();  
  }  
}
```

```
const printYes = function() {  
  console.log('yes!');  
}
```

```
const printNo = function () {  
  console.log('no!');  
}
```

```
quiz('wrong', printYes, printNo);  
quiz('love you', printYes, printNo);
```

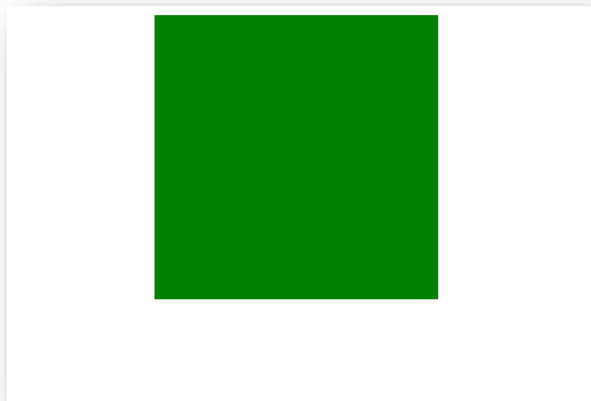


## ■ Callback 예시)

### ● 일정시간 경과 후 동작

04-10-timeout.html

```
function callback() {  
  const label = document.querySelector('.label');  
  label.innerHTML = '<h2>모습 변경</h2>';  
  
  const figure = document.querySelector('.figure');  
  figure.style.backgroundColor = 'red';  
  figure.style.borderRadius = '100px';  
}  
  
setTimeout(callback, 2000);
```



## ■ Callback 예시)

### ● 현재 위도/경도 확인

04-11-geolocation.html

```
function callback(position) {  
  const lat = position.coords.latitude;  
  const lng = position.coords.longitude;  
  
  const label = document.querySelector('.label');  
  label.innerHTML += `<h2>위도: ${lat}</h2>`;   
  label.innerHTML += `<h2>경도: ${lng}</h2>`;   
  
}  
navigator.geolocation.getCurrentPosition(callback);
```

위도: 36.609992

경도: 127.2842323

## ■ 함수 작성 형태

### ● 선언식(Declarations) / 표현식(Expressions)

04-12-declarations-expressions.html

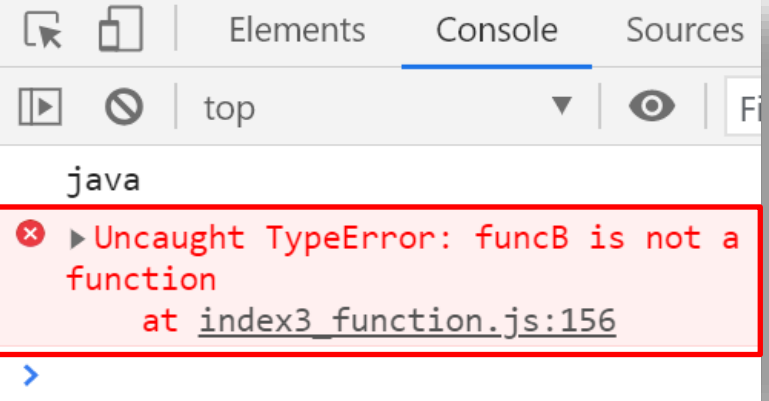
```
funcA();
```

```
funcB();
```

```
function funcA() {  
  console.log('java');  
};
```

```
const funcB = function() {  
  console.log('script');  
}
```

error



## ■ 함수 작성 형태

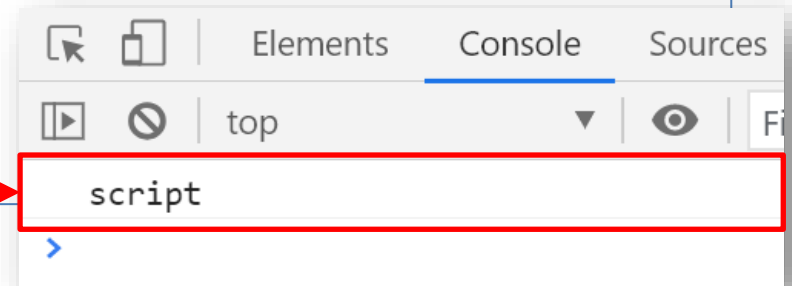
- 선언식(Declarations) / 표현식(Expressions)

04-12-declarations-expressions.html

```
const funcA = function() {  
  console.log('script');  
};
```

```
function funcA() {  
  console.log('java');  
}
```

```
funcA();
```



## ■ Arrow Function

- => 표현식을 사용하여 함수를 작성하는 방법

04-13-arrow-function.html

```
const simplePrint = function() {  
  console.log('simplePrint!');  
}
```

```
const simplePrint = () =>  
  console.log('simplePrint!');
```

```
const simplePrint = () => {  
  console.log('simplePrint!');  
}
```

- 항상 익명
- 파라미터가 1개인 경우 () 생략 가능

```
const minus = a => console.log(a);  
minus(1);
```

## ■ Arrow Function

- => 표현식을 사용하여 함수를 작성하는 방법

04-14-arrow-function.html

```
const add = function(a, b) {  
  return a + b;  
}
```



```
const add = (a, b) => a + b;
```

- 함수 내의 코드가 한줄이면 return 생략

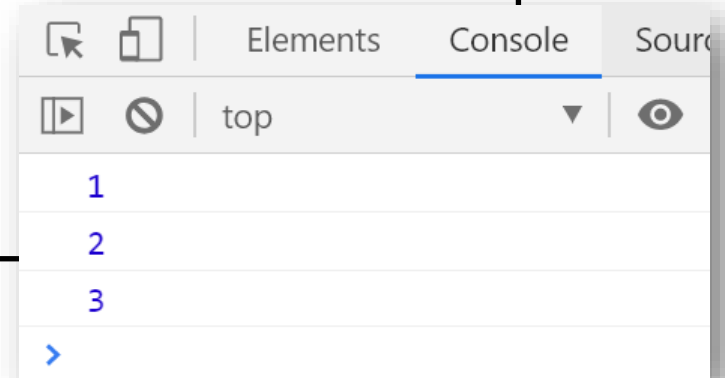


## ■ Closure

- 지역 변수가 사라지지 않고 계속 기억되는 현상
  - 호출 할 때마다 숫자를 증가시키는 함수

04-15-closure.html

```
function sequence() {  
  let seq = 0;  
  return function () {  
    return ++seq;  
  };  
}  
const seq = sequence();  
console.log( seq() );  
console.log( seq() );  
console.log( seq() );
```



## ■ Closure

### ● 지역 변수가 사라지지 않고 계속 기억되는 현상

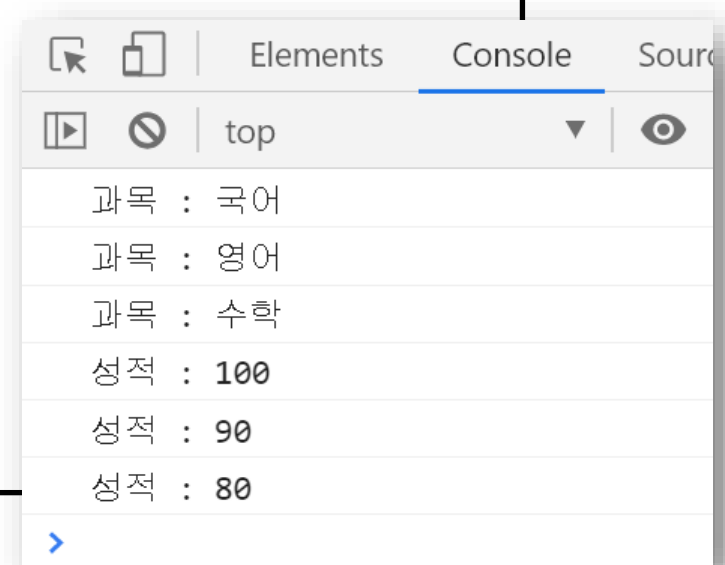
04-16-closure.html

- 분류 항목과 데이터를 저장하는 템플릿 형태의 함수

```
function foo(x) {  
  return function (y) {  
    console.log(`${x} : ${y}`);  
  }  
}
```

```
const bar1 = foo("과목");  
bar1("국어");  
bar1("영어");  
bar1("수학");
```

```
const bar2 = foo("성적");  
bar2(100);  
bar2(90);  
bar2(80);
```



## ■ Closure 활용 예)

### ● 온라인 시험 시 부정행위 확인

04-17-closure.html

```
<body>
  <script>
    function sequence() {
      let seq = 0;
      return function () {
        seq++;
        return seq;
      };
    }
    const check = sequence(); * 브라우저가 포커스를 잃는 경우
    window.addEventListener('blur', () => {
      const count = check();
      if(count > 5) {
        alert('시험 종료!');
        window.close();
      }
    }); * 브라우저(탭) 종료
  </script>
</body>
```

## ■ Immediately Invoked Function Expression

### ● 함수 정의 및 호출을 한번에 표현하는 방법

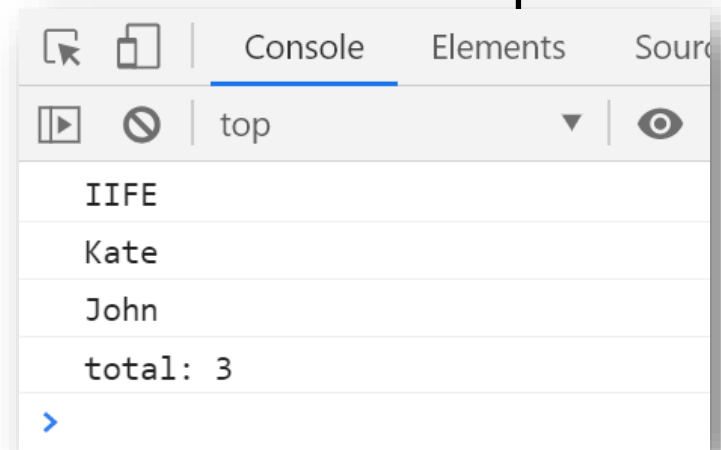
04-18-IIFE.html

```
(function hello() {  
  console.log('IIFE');  
})();
```

```
(function () {  
  // 내부에서 정의된 변수는 외부에서 접근 불가  
  var aName = "Kate";  
  console.log(aName);  
})();
```

```
// 결과만 저장  
const result = (function () {  
  let name = "John";  
  return name;  
})();
```

```
const total = ((x, y) => x + y); // 파라미터 사용  
console.log(`total: ${total(1, 2)}`);
```



## ■ IIFE를 활용한 closure 오류 수정

### ● 잘못 적용된 closure 예)

04-19-error-closure.html

```
<button type="button" id="btn-1">버튼1</button>
<button type="button" id="btn-2">버튼2</button>
<button type="button" id="btn-3">버튼3</button>

<script>
  function addEvent() {
    for (var i = 1; i <= 3; i++) {
      document.querySelector(`#btn-${i}`).addEventListener("click", () => {
        alert(i);
      });
    }
  }
  addEvent();
</script>
```

## ■ IIFE를 활용한 closure 오류 수정

### ● IIFE를 활용한 오류 수정 예)

04-20-fix-closure.html

```
<button type="button" id="btn-1">버튼1</button>
<button type="button" id="btn-2">버튼2</button>
<button type="button" id="btn-3">버튼3</button>

<script>
  function addEvent() {
    for (var i = 1; i <= 3; i++) {
      (function(j) {
        document.querySelector(`#btn-${j}`).addEventListener("click", () => {
          alert(j);
        });
      })(i);
    }
  }
  addEvent();
</script>
```