

함수(function)

- JavaScript 의 정체성
- 콜백함수 (callback function) 사용 가능
 - 함수가 함수의 파라미터로, 또는 리턴값으로 사용됨
 - 다른 언어와는 구별되는 독특한 특징
 - Python, Java, C# 에서도 제공은 하지만, "기본으로" 배우진 않음
 - JavaScript 에서 callback은 "기본"이며, 초보자와 중급자를 가르는 기준
 - JavaScript의 함수는 C의 포인터, Python과 Java의 클래스만큼이나 중요

함수(function) 는 무엇인가?

- 개인적으로, function 이라는 단어의 번역을 "잘못했다고" 생각함
- function 의 기본뜻은 "기능" 이다.
- 수학에서 배운 함수와 프로그래밍에서의 함수는 다른 뜻이다.
 - 수학에서의 함수:
두 집합 x, y 에서 집합 x 의 각 원소에 대하여 집합 y 의 원소가 하나씩만 대응하는것
 - 프로그래밍에서의 함수:
재사용 가능한(reusable) 기능(function)

```
console.log("안녕, 조교행님. 나이는 28세");  
console.log("안녕, 조교행님. 나이는 28세");  
console.log("안녕, 조교행님. 나이는 28세");  
console.log("안녕, 조교행님. 나이는 28세");  
console.log("안녕, 조교행님. 나이는 28세");
```

공통점을 찾아보자.

1. 모두 console.log() 사용
2. '안녕' 이 앞에 들어가고, 그 뒤에 이름, 나이가 온다

만약, '안녕' 대신 '하이' 를 쓰고 싶다면, 또는 '나이' 대신 '연세' 를 써야 한다면?
다섯개면 문제가 안 되겠지만 백 개, 천 개라면, 일일이 '하이' 와 '연세' 로 바꿔줘야.

```
function sayHello(name, age) {  
    return `안녕, ${name}. 나이는 ${age}세`;  
}
```

```
console.log(sayHello('조교행님', 28));  
console.log(sayHello('은지', 16));  
console.log(sayHello('현수', 21));  
console.log(sayHello('지원', 14));  
console.log(sayHello('준수', 23));
```

이제, 이 함수를 분석해보면서 함수가 뭔지 알아보자.

함수 정의부 (function define).

기능을 사용하기 전에 만들자! 항상 쓰기 전에 먼저 만들자!

function으로 시작하고, 이름을 지어준다 (sayHello. 동사(verb)+명사(noun))

중괄호 안에 함수의 내용

```
function sayHello(name, age) {  
    return `안녕, ${name}. 나이는 ${age}세`;  
}
```

```
const result = sayHello('조교행님', 28);
```

함수 호출부 (function call). 기능을 사용하자!

호출부의 소괄호 ()

정의부의 소괄호 ()

함수를 실행한다는 뜻

파라미터 들어갈 자리

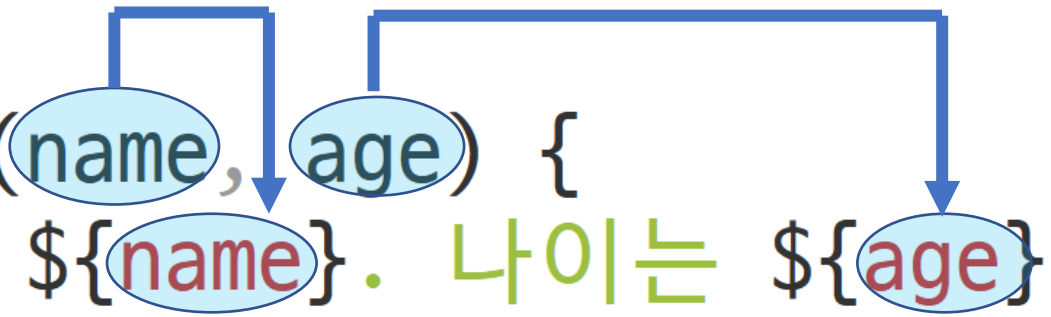
매개변수(parameter)

```
function sayHello(name, age) {  
  return `안녕, ${name}. 나이는 ${age}세`;  
}
```

```
const result = sayHello('조교행님', 28);
```

'조교행님' 과 28 이 sayHello() 함수로 전달됨
이제부터 sayHello 안에서만 '변수' 로 사용가능

```
function sayHello(name, age) {  
  return `안녕, ${name}. 나이는 ${age}세`;  
}
```



```
const result = sayHello('조교행님', 28);
```

sayHello() 함수의 변수이기때문에, sayHello() 안에서 자유롭게 사용 가능

```
function sayHello(name, age) {  
  return `안녕, ${name}. 나이는 ${age}세`;  
}
```

```
const result = sayHello('조교행님', 28);
```

return 은 '반환' 이라고 번역되며, sayHello() 함수가 '내뱉는' 결과다.
그 결과를 result 라는 변수로 받은 것. 당연히, 타입은 string 이다.

이미 만들어진 함수를 "사용"하려면
어떻게 생겼는지 몰라도 된다.

```
const result = sayHello('조교행님', 28);
```

- 이 함수는 두 개의 파라미터가 필요합니다.
- 첫번째 파라미터는 string 입니다.
- 두번째 파라미터는 number 입니다.
- 리턴값은 "안녕, ~~, 나이는 ~~세" 입니다.

- 즉, 무슨 일을 하는 "기능"인지를 아는 것이 사용의 핵심

재사용 (reuse)

```
function sayHello(name, age) {  
    return `안녕, ${name}. 나이는 ${age}세`;  
}
```

```
console.log(sayHello('조교행님', 28));  
console.log(sayHello('은지', 16));  
console.log(sayHello('현수', 21));  
console.log(sayHello('지원', 14));  
console.log(sayHello('준수', 23));
```

함수는 딱 한번만 만들었고, 재사용했다.

즉, 함수(function)는 무엇인가?

- 재사용가능한(reusable) 기능(function)

“안녕”을 “하이” 로, “나이”를 “연세” 로

```
function sayHello(name, age) {  
  return `안녕, ${name}. 나이는 ${age}세`;  
}
```

```
console.log(sayHello('조교행님', 28));  
console.log(sayHello('은지', 16));  
console.log(sayHello('현수', 21));  
console.log(sayHello('지원', 14));  
console.log(sayHello('준수', 23));
```

백개든 천개든, 내용만 바꾸면 된다

console.log();

console 객체 안에 있는 log() 라는 이름을 가진 함수

사용법

log() 안에 들어갈 파라미터 개수는 1개부터 무한개이다.
들어간 파라미터를 한 칸 띄워서 한 줄에 출력한다.

* console.log() 가 어떻게 생겼는지는 몰라도, 사용법만 알면 쓸 수 있다!

내장객체(Built-in Object)

- 그런데 우린 console 객체도, 그 안에 log() 함수도 만든 적이 없다.
- 왜 쓸 수 있었을까? 기본적으로 제공되는 내장객체이기때문.

지역변수(local variable) vs 전역변수(global variable)

```
const a = 1;  
const b = 2;
```

```
function showResult(a, b) {  
  a = 3;  
  b = 4;  
  return [a, b]; 리턴값이 여러개일때 배열 사용  
}
```

```
const [resultA, resultB] = showResult(a, b);
```

```
console.log(a, b);  
console.log(resultA, resultB);
```

Q. a, b는 바뀌었을까?

A. No.

**함수 안의 a, b 와
함수 바깥의 a, b 는 같지 않다**

배열로 받았지만 사실 배열이 아님

지역변수(local variable) vs 전역변수(global variable)

```
const a = 1;  
const b = 2;
```

```
function showResult(a, b) {  
  a = 3;  
  b = 4;  
  return [a, b];  
}
```

```
const [resultA, resultB] = showResult(a, b);
```

```
console.log(a, b);  
console.log(resultA, resultB);
```

파라미터로 들어가는 순간 값(value)이 복사된 것일 뿐
지역변수를 바꾼다고 원래 전역변수가 바뀌진 않는다.


```
const c = 5;
```

```
function showResult() {  
    console.log(c);  
}
```

파라미터와 리턴값이 없다?
선택사항이기 때문.

참고로 이 경우, 리턴 타입은 void 이다.

```
showResult();
```

전역변수는 파일 안에서 자유롭게 사용 가능하다.

showResult() 에서 c 를 파라미터로 받지 않았지만, 사용 가능하다.

물론, 바꾸려고 하면 const 라서 에러가 뜰 것. 이것은 const를 쓰는 이유 중 하나

그러나, 배열과 객체는 예외다

```
const menu = ["짜장면", "짬뽕", "탕수육"];
```

```
function changeFirstMenu() {  
    const tempMenu = menu;  
    tempMenu[0] = "우동";  
}
```

심지어, tempMenu 라는 지역변수까지 만들어
그걸 우동으로 바꿨을 뿐임.
menu는 건들지도 않았다!

```
changeFirstMenu();  
console.log(menu); [우동, 짬뽕, 탕수육]
```

```
const info = {  
  name: "조교행님",  
  age: 28  
};
```

```
function changeInfoAge() {  
  tempInfo = info;  
  tempInfo.age = 18;  
}
```

심지어, tempInfo 라는 지역변수까지 만들어
그 안에 나이를 바꿨을 뿐임.
info는 건들지도 않았다!

```
changeInfoAge()  
console.log(info); { name: "조교행님", age: 18 }
```

다른건 헛갈려도, 이것만은 기억하자

- 배열과 객체는 함수 안에서 쓸 땐 매우 조심해야한다.
- 같은 이름으로 쓰였더라도, 어디에 있느냐에 따라 완전히 다른 것일 수 있다.
 - ex) result 와 Result 대소문자가 다르므로 당연히 다른 것
 함수 안의 변수 a와 함수 바깥의 변수 a

화살표 함수 (arrow function =>)

- 2015년 ES6 업데이트 이후 추가됨
- JavaScript 의 정체성에 더 가까운 표현방식

```
function sayHello(name, age) {  
    return `안녕, ${name}, 나이는 ${age}세`;  
}
```

```
const sayHello = (name, age) => {  
    return `안녕, ${name}, 나이는 ${age}세`;  
}
```

차이점:

- 1 function 키워드 사라짐
- 2 => 이라는 화살표 생김
3. 변수 선언하듯 *const* 를 썼고, = 를 통해 값을 집어넣음

즉, JavaScript 에서 함수는 변수이며, 하나의 타입이다.
따라서, 함수는 다른 함수의 파라미터나 리턴값으로도 쓰인다.

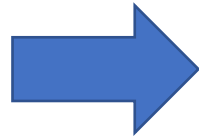
```
const plus = (a, b) => {  
    return a + b;  
}
```

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

화살표 함수를 사용하면, return 도 생략 가능하다.
함수 안에 return 한 줄밖에 없다면, 중괄호 { } 도 생략 가능하다.

객체 안의 함수(메소드): 계산기의 예

```
const myCal = {  
  add: function(a, b) {  
    return a + b;  
  },  
  subtract: function(a, b) {  
    return a - b;  
  },  
  multiply: function(a, b) {  
    return a * b;  
  },  
  devide: function(a, b) {  
    return a / b;  
  }  
};
```



```
const myCal = {  
  add: (a, b) => a + b,  
  subtract: (a, b) => a - b,  
  multiply: (a, b) => a * b,  
  devide: (a, b) => a / b  
};
```

둘 다 따라쳐보고, 함수를 직접 사용해보자!

ex) console.log(myCal.add(5, 2));

같이 해보기

- 로그인 함수를 만들자.
 - 화살표 함수 사용할 것
 - 함수 이름은 login
 - 파라미터는 id와 password 이며, 둘 다 string 타입
 - 아이디와 패스워드가 일치하면 true, 일치하지 않으면 false 리턴
- true, false 를 함수로부터 받아서,
 - true 일 경우, "~~~님 환영합니다." 메시지를 로그 출력. ~~~ 안엔 id가 들어감
 - false 일 경우, "로그인 실패!" 메시지를 로그 출력

```
const myInfo = {  
  id: "assistant0603",  
  password: "ohmypassword",  
};
```

```
const login = (id, password) => {  
  if (id === myInfo.id) {  
    if (password === myInfo.password) {  
      return true;  
    } else {  
      return false;  
    }  
  } else {  
    return false;  
  }  
};
```

함수 안에서 log 를 프린트하는건 최대한 피해야한다.
리턴한 결과를 받아서 log 를 프린트하는 습관을 들이도록 하자

```
const loginSuccess = login("assistant0603", "ohmypassword");  
if (loginSuccess === true) {  
  console.log(`환영합니다, ${myInfo.id}님`);  
} else {  
  console.log("로그인 실패!");  
}
```

직접 해보기

- 술 판매 가능 여부를 리턴하는 함수를 만들라.
 - 화살표 함수 사용할 것
 - 함수 이름은 isOKtoDrink
 - 파라미터는 age 하나이며, number 타입
 - 만약 18세 초과면 true 리턴
 - 18세 이하면 false 리턴

```
console.log(isOKtoDrink(24)); // true
console.log(isOKtoDrink(19)); // true
console.log(isOKtoDrink(18)); // false
```

직접 해보기

- 출근일을 판단하는 함수를 만들라.
 - 화살표 함수 사용할 것
 - 함수 이름은 isGoToWork
 - 파라미터는 월 - 일 중 하나의 string 타입
 - 만약 월 - 금요일이면 true 리턴
 - 만약 토요일, 일요일이면 false 리턴

```
console.log(isGoToWork("월")); // true
console.log(isGoToWork("금")); // true
console.log(isGoToWork("토")); // false
console.log(isGoToWork("일")); // false
```