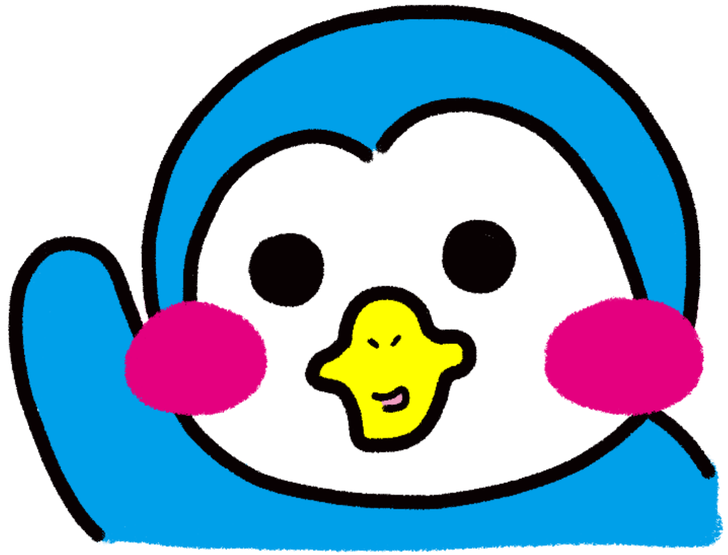
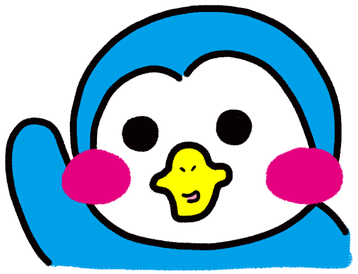
이런 함수도 있습니다



화살표 함수와 디폴트 파라미터

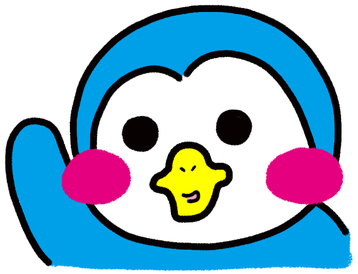
화살표 함수



화살표 함수는 함수 표현식보다 단순하고 간결한 문법으로
함수를 만들 수 있는 방법이다. 여기에는 다음과 같은 특징이 있다.

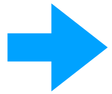
- function 표현식에 비해 구문이 짧다.
- this나 super등의 객체를 바인딩하지 않는다.
- 기본적으로 익명 함수이다. 따라서 메소드가 아닌 개별 함수로 더 적합하다.
- return 키워드의 생략이 가능하다(제한적).

화살표 함수가 만들어지는 과정

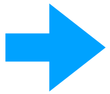


화살표 함수는 다음과 같은 형태를 지닌다.

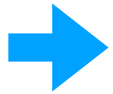
```
const funcBasic = function(){
  // 이것은 함수 표현식
}
```



```
const funcBasic = (){
  // function 키워드를 지워준다
}
```

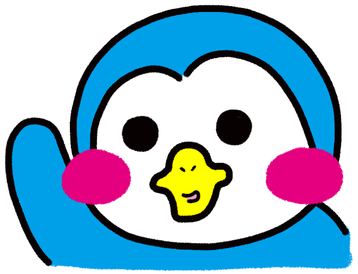


```
const funcBasic = () => {
  // 매개변수 자리 옆에 화살표 추가
}
```



```
const funcBasic = () => {
  console.log("화살표 함수 호출!")
}

funcBasic();
```

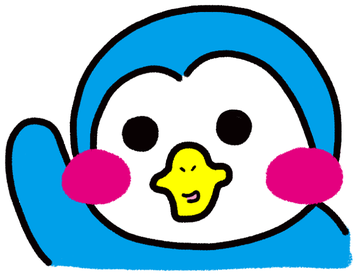
화살표 함수의 특징 1



호이스팅(함수 선언보다 먼저 호출)이 불가하다.

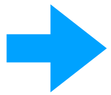
```
funcBasic(); // 여기선 안되고

const funcBasic = () => {
  console.log("화살표 함수 호출!")
}

funcBasic(); // 여기선 됩니다
```

화살표 함수의 특징 2



반환값만 존재하는 경우, 중괄호와 return 키워드를 생략할 수 있다.

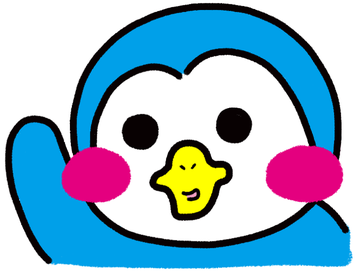
```
const getTen1 = () => {
  return 10
}
const getTen2 = () => 10


console.log(getTen1())
console.log(getTen2())
```



```
10
10
```

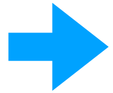
=> 반환값 외에 구문이 존재하는 경우에는 다시 중괄호와 return 필요해져요!

화살표 함수의 특징 3



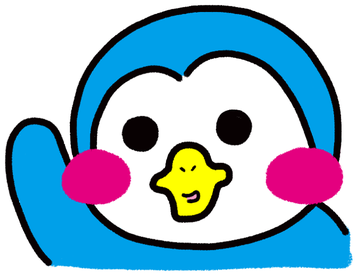
매개변수가 하나인 경우에는 소괄호 생략 가능하다. 없거나 다수면 생략 불가!

```
const printName = name => {
  console.log(name)
}
const printNames = (name1, name2) => {
  console.log(`${name1}와 ${name2}`)
}

printName("축구왕 슛돌이")
printNames("흥부", "놀부")
```



```
축구왕 슛돌이
흥부와 놀부
```

디폴트 파라미터

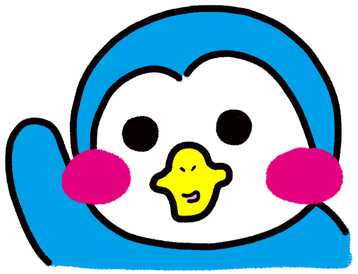


디폴트 파라미터(default parameter)는 매개변수에 인자가 전달되지 않았을 때에 사용할 수 있는 값을 미리 정의해 두는 것을 뜻한다.

```
const printNames = (name1="흥부", name2="놀부") => {
  console.log(`${name1}와 ${name2}`)
}

printNames()
```

=> 위 함수에 값을 전달하면 매개변수의 값이 변경된다!

내용 정리



- 화살표 함수는 함수 표현식의 한 형태이다.
- 구문이 짧고 간결하여 개별 함수를 선언해 재사용하기 용이하다.
- 화살표 함수는 호이스팅 불가하다.
- 경우에 따라 매개변수 소괄호, 몸체의 중괄호, return을 생략할 수 있다.
- 디폴트 파라미터는 매개변수에 인자가 전달되지 않았을 때에 사용할 수 있는 값을 미리 정의해 두는 것을 뜻한다.