Arrow Function

☑ arrow function 기본

Function Expression보다 단순하고 간결한 문법으로 함수를 만들 수 있는 방법이 있다.
바로 arrow function(arrow function)를 사용하는 것이다. arrow function라는 이름은 문법의 생김새를 차용해 지어졌다.

```
let func = (arg1, arg2, ...argN) => expression;
```

이렇게 코드를 작성하면 인자 arg1..argN를 받는 함수 func이 만들어진다. 함수 func는 화살표(=>) 우측의 표현식(expression)을 평가하고, 평가 결과를 반환한다.

아래 함수의 축약 버전이라고 할 수 있다.

```
let func = function(arg1, arg2, ...argN) {
  return expression;
};
```

좀 더 구체적인 예시를 살펴보자.

```
let sum = (a, b) => a + b;

/* 위 arrow function는 아래 함수의 축약 버전이다.
let sum = function(a, b) {
  return a + b;
};
*/

console.log( sum(1, 2) );         // 3
```

보시는 바와 같이 (a, b) => a + b는 인수 a와 b를 받는 함수이다. (a, b) => a + b는 실행되는 순간 expression(표현식) a + b를 평가하고 그 결과를 반환한다.

인수가 하나밖에 없다면 인수를 감싸는 괄호를 생략할 수 있다. 괄호를 생략하면 코드 길이를 더 줄일 수 있다.

[예시]

```
let double = n => n * 2;
// let double = function(n) { return n * 2 }과 거의 동일하다.

console.log( double(3) ); // 6
```

인수가 하나도 없을 땐 괄호를 비워놓으면 된다. 다만, 이 때 괄호는 생략할 수 없다.

```
let sayHi = () => console.log("안녕하세요!");

sayHi();
```

arrow function는 Function Expression과 같은 방법으로 사용할 수 있다.

아래 예시와 같이 함수를 동적으로 만들 수 있다.

```
let age = prompt("나이를 알려주세요.", 18);

let welcome = (age < 18) ?
  () => alert('안녕') :
  () => alert("안녕하세요!");

welcome();
```

arrow function를 처음 접하면 가독성이 떨어진다. 익숙지 않기 때문이다. 그러나 문법이 눈에 익기 시작하면 적응은 쉽게 된다.

함수 본문이 한 줄인 간단한 함수는 arrow function를 사용해서 만드는 게 편리한다. 타이핑을 적게 해도 함수를 만들 수 있다는 장점이 있다.

❖ 본문이 여러 줄인 arrow function

위에서 소개한 arrow function들은 => 왼쪽에 있는 인수를 이용해 => 오른쪽에 있는 expression을 평가하는 함수들이었다.

그런데 평가해야 할 expression이나 구문이 여러 개인 함수가 있을 수도 있다. 이 경우 역시 arrow function 문법을 사용해 함수를 만들 수 있다. 다만, 이때는 curly brace(중괄호) 안에 평가해야 할 code를 넣어주어야 한다. 그리고 return 지시자를 사용해 명시적으로 결과값을 반환해 주어야 한다.

아래와 같이 한다.

```
let sum = (a, b) => {  // 중괄호는 본문 여러 줄로 구성되어 있음을 알려준다.
  let result = a + b;
```

```
  return result; // 중괄호를 사용했다면, return 지시자로 결과값을 반환해주어야 한다.
};

console.log( sum(1, 2) ); // 3
```

지금까진 간결함이라는 특징을 중심으로 arrow function에 대해 알아보았다. 그러나 이게 다가 아니다. arrow function는 여기서 소개한 기능 이외에도 다른 흥미로운 기능을 지원한다.

지금까진 본문이 한 줄인 arrow function, arrow function가 callback으로 쓰인 경우에 대해서 알아보았다.

요약하면 arrow function는 본문이 한 줄인 함수를 작성할 때 유용하다. 본문이 한 줄이 아니라면 다른 방법으로 arrow function를 작성해야 한다.

① 중괄호 없이 작성: (...args) => expression - 화살표 오른쪽에 expression을 둔다. 함수는 이 expression을 평가하고, 평가 결과를 반환한다.
② 중괄호와 함께 작성: (...args) => { body } - 본문이 여러 줄로 구성되었다면 중괄호를 사용해야 한다. 다만, 이 경우는 반드시 return 지시자를 사용해 반환 값을 명기해 주어야 한다.

[과 제]

☞ **arrow function으로 변경하기**

Function Expression을 사용해 만든 아래 함수를 arrow function로 바꿔보자.

```
function ask(question, yes, no) {
  if (confirm(question)) yes()
  else no();
}

ask(
  "동의하십니까?",
  function() { alert("동의하셨습니다."); },
  function() { alert("취소 버튼을 누르셨습니다."); }
);
```

[해답]

```
function ask(question, yes, no) {
  if (confirm(question)) yes()
  else no();
}
```

```
ask(
  "동의하십니까?",
  () => alert("동의하셨습니다."),
  () => alert("취소 버튼을 누르셨습니다.")
);
```

좀더 간결하고 명확해졌다.