삼육대학교 SW융합교육원 김진호

자바스크립트 기초교육

<class>
</body>
<head>
<title>

02. arrow function (화살표 함수)

2-1. arrow function basic syntax

2-2. arrow function feature

2. arrow function

2-1. arrow function basic syntax (화살표 함수 기본 문법)

2-1-1. arrow function basic syntax (화살표 함수 기본 문법)

- ES6에서 도입 된 화살표 함수는 function 키워드 대신 화살표를 사용해 좀 더 간략한 방법으로 함수를 선언할 수 있다.
- 화살표 함수는 항상 익명 함수로 정의한다. 본문이 한 줄인 함수를 작성할 때 유용하다.

2. arrow function

2-1. arrow function basic syntax (화살표 함수 기본 문법)

2-1-1. arrow function basic syntax (화살표 함수 기본 문법)

```
let message;

// 기존 function 정의
message = function () {
    return "Hello World!";
};

console.log(message());

// function 키워드 생략 가능
message = () => {
    return "Arrow Function!";
};

console.log(message());

// 명령문이 하나만 있는 경우 중괄호 생략 가능
// 이 때 함수 몸체 내부의 문이 값으로 평가 될 수 있는 표현식인 문이라면 암묵적으로 반환 된다.
// return 키워드 생략 가능
message = () => "Arrow Functions are Simple!";

console.log(message());
```

```
// 매개변수가 있을 경우
message = (val1, val2) => "Arrow " + val1 + val2;

console.log(message('Function', '!'));

// 매개변수가 하나면 소괄호 생략 가능
// 매개변수가 없거나 여러 개일 경우 생략 불가하다
message = val=> "Arrow " + val;

console.log(message('Functions are GOOD!!!'));

// 객체 리터럴을 반환하는 경우 소괄호로 감싼다.
const createUser= (id, name) => ({ id, name });
console.log(createUser(1, '유관순'));
// 소괄호로 감싸지 않을 경우 함수 몸체 중괄호로 잘못 해석한다.
const createUser2 = (id, name) => { id, name };
console.log(createUser2(2, '홍길동'));

// 고차 함수에 인수로 전달 가능하며 일반적인 함수 표현식 보다 표현이 간결해진다.
console.log([1,2,3,4,5].map(function(val){ return val* 10}));
console.log([1,2,3,4,5].map(val=> val* 10));
```

2. arrow function

2-2. arrow function feature (화살표 함수 특징)

2-2-1. arrow function feature (화살표 함수 특징)

- 화살표 함수는 기존의 함수보다 표현만 간략한 것이 아니라 내부 동작 또한 간략화 되어있다.
- 일반 함수와의 차이점을 알아보자.

2. arrow function

2-2. arrow function feature (화살표 함수 특징)

2-2-1. arrow function feature (화살표 함수 특징)

화살표 함수는 this를 가지지 않는다.

- 이러한 특징은 객체의 메서드 안에서 동일한 객체의 프로퍼티를 대상으로 순회하는데 사용할 수 있다.

```
let theater = {
    store: "건대점",
    titles: ["어벤져스", "겨울왕국", "스파이더맨", "라이온킹", "알라딘"],

    showMovieList() {
        this.titles.forEach(
            // 화살표 함수 본문에서 this에 접근하면 외부에서 값을 가져오므로
this.store는 thearter.store "건대점"을 의미한다.
            title => console.log(this.store + ': ' + title)
        );
    }

    // showMovieList() {
    //     this.titles.forEach(function(title) {
    //        // 화살표 함수가 아닌 일반 함수를 사용하면 this.store는 undefined이다.
    //       console.log(this.store + ': ' + title);
    //     });
    //   }
};

theater.showMovieList();
```

2. arrow function

2-2. arrow function feature (화살표 함수 특징)

2-2-1. arrow function feature (화살표 함수 특징)

화살표 함수는 new와 함께 호출 할 수 없다.

- this가 없기 때문에 생성자 함수로 사용할 수 없다.

```
const arrowFunc = () => {};
// new arrowFunc();    // 에러 발생 : TypeError: arrowFunc is not a constructor
// 화살표 함수는 인스턴스를 생성할 수 없으므로 prototype 프로퍼티가 없고 프로토타입도 생성하지 않는다.
console.log(arrowFunc.hasOwnProperty('prototype')); // false
```

2. arrow function

2-2. arrow function feature (화살표 함수 특징)

2-2-1. arrow function feature (화살표 함수 특징)

화살표 함수는 super를 가지지 않는다.

```
class Animal {

    constructor(name, weight) {
        this.name = name;
        this.weight = weight;
    }

    move(lostWeight) {
        if (this.weight > lostWeight)
            this.weight -= lostWeight;
        console.log(`${this.name}(은)는 움직임으로 인해 ${lostWeight}kg 감량되어 ${this.weight}kg이 되었습니다.`)
    }

}
```

```
class Tiger extends Animal {

    move(lostWeight) {
        // 화살표 함수는 super를 지원하지 않아 super를 외부 함수에서 가져오기 때문에 Animal의 move() 호출이 가능하다.
        setTimeout(() => super.move(lostWeight), 1000);
        // 일반 함수를 사용할 경우 에러가 발생한다 - SyntaxError: 'super' keyword unexpected here
        // setTimeout(function() { super.move(lostWeight) }, 1000);
        console.log('먹이를 찾아 산기슭을 어슬렁');
    }
}

let tiger = new Tiger("백두산 호랭이", 90);
tiger.move(1);
```

2. arrow function

2-2. arrow function feature (화살표 함수 특징)

2-2-1. arrow function feature (화살표 함수 특징)

화살표 함수는 arguments를 지원하지 않는다.

```
(function(){
    // 화살표 함수는 본인의 arguments 3,4가 아닌 상위 스코프인 즉시 실행 함수의 arguments 1,2를 참조한다.
    const arrowFunc = () => console.log(arguments);
    arrowFunc(3,4);
}(1,2));
```

- 화살표 함수는 다른 함수의 인수로 전달 되어 콜백함수로 사용되는 경우가 많다.
- 위와 같은 특징들은 콜백 함수 내부의 this가 외부 함수의 this와 다르기 때문에 발생하는 문제를 해결하기 위해 의도적으로 설계 된 것이라 할 수 있다.