JavaScript

arrow 함수 – 김근형 강사

Arrow 함수

- arrow 함수
  - 화살표 함수는 function(param) { code } 의 형태를 축약한 것이다.
  - 화살표 함수는 익명(Anonymous)함수이며 함수를 호출하기 위해서는 변수를 통해 할당해야 한다.
  - 아래와 같은 형태로 함수를 작성할 수 있다.

```
(param) => { 코드 };
param => { 코드 };
() => { 코드 }
(param1, param2, … paramN) => { 코드 };
param => ( {key : val} );
(param1, param2, …rest)  => { 코드 };
(param1, param2 = 123, … paramN) => { 코드 };
( [one, two] = [1,2] ) => one + two;
( {key : sum} = { key : 10 + 20 } ) => { 코드 };
```

Arrow 함수

- arrow 함수
  - 만약 로직이 한줄일 경우 return과 {}를 생략하여 사용이 가능하다.

```
var es5 = function(one, two){
  return one + two;
}
var sum = es5(1, 2);
console.log(sum);
```

```
let es6 = (one, two) => {
  return one + two;
};
let result = es6(1, 2);
console.log(result);
```

```
let total = (one, two) => one + two;
let result = total(1, 2);
console.log(result);
```

Arrow 함수

- arrow 함수
  - 파라미터가 하나일 경우 ()를 생략할 수 있으며 만약 파라미터가 없을 경우 ()만 쓰고 함수 선언이 가능하다.

```
let get = value => value + 10;
let result = get(20);
console.log(result);
```

```
let noParam = () => 3 + 4;
let result = noParam();
console.log(result);
```

Arrow 함수

- arrow 함수
  - Arrow 함수에서 {key: value} 형태의 Object를 반환하려면 소괄호 () 안에 {key:value} 를 작성한다.

```
let get = param => ({sports: "축구"});
let result = get();
console.log(result);
```

  - 화살표 함수로는 new 연산자로 인스턴스 생성이 불가능하다.

Arrow 함수

- arrow 함수
  - arrow 함수 내에서는 arguments 객체로의 접근이 불가능하다.

```
let sports = () => {
  try {
    let args = arguments;
  } catch (error) {
    console.log("사용 불가");
  }
}
sports(1, 2);
```

```
사용 불가
```

Arrow 함수

- this
  - arrow 함수에서의 this는 lexical this 를 가진다.
  - 즉, 자신을 둘러싼 환경의 this를 그대로 계승 받는다.
  - arrow 함수는 함수지만 실제 함수의 영역에서 쓰는 this와는 전혀 다른 영역의 this를 사용한다.

```
var a = {
    name : 'foo',
    f1 : function (){
        return () => {
            console.log(this.name);
        }
    }
}
var f = a.f1();
f(); // foo
```

Arrow 함수

- This를 쓸 수 없는 경우
  - 1 – 객체의 바로 밑에 메서드로 활용하기

```
const person1 = {
    name: 'Lee',
    sayHi: () => console.log(`Hi ${this.name}`)
};

person1.sayHi(); // Hi

const person2 = {
    name: 'Lee',
    sayHi: function () {
        console.log(`Hi ${this.name}`)
    }
}

person2.sayHi(); // Hi Lee
```

Arrow 함수

- This를 쓸 수 없는 경우
  - 2 – 프로토타입 함수로 사용 불가

```
const person1 = {
    name: 'Lee',
};

Object.prototype.sayHi1 = () => console.log(`Hi ${this.name}`);

person1.sayHi1(); // Hi

const person2 = {
    name: 'Lee',
};

Object.prototype.sayHi2 = function () {
    console.log(`Hi ${this.name}`);
};

person2.sayHi2(); // Hi Lee
```

Arrow 함수

- This를 쓸 수 없는 경우
  - 3 – 생성자 함수 사용 불가

```
const Foo = () => {
    console.log(this);
}

const foo = new Foo(); // TypeError: Foo is not a constructor
```

  - arrow function에는 prototype 프로퍼티가 없기 때문에 위와 같은 생성자 함수로서 사용은 불가하다.

Arrow 함수

- This를 쓸 수 없는 경우
  - 4 – addEventListener 안에서 사용 시 this 불가.

```
<button id="box1">Arrow</button>
<button id="box2">Function</button>
<script>
    const box1 = document.getElementById('box1');

    box1.addEventListener('click', () => {
        console.log(this); //window
    });
    const box2 = document.getElementById('box2');

    box2.addEventListener('click', function () {
        console.log(this); //button
    });
</script>
```

Arrow 함수

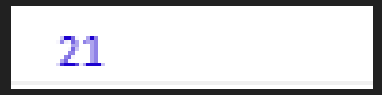
- This를 쓰기 올바른 예
  - 화살표 함수가 간단하게 코드를 작성할 수 있어 편리하지만 this의 참조를 고려해야 한다.
  - This라는 함수가 자칫 잘못하면 window 객체를 참조할 수 있는데 그렇게 되면 생각치 못한 동작이 일어날 수 있다.

```
true
undefined
```

```
let Sports = function(){
  this.count = 20;
};
Sports.prototype = {
  plus: function(){
    this.count += 1;
  },
  get: function(){
    setTimeout(function(){
      console.log(this === window);
      console.log(this.plus);
    }, 1000);
  }
};
let newSports = new Sports();
newSports.get();
```

Arrow 함수

- This를 쓰기 올바른 예
  - Arrow 함수를 쓰게 되면 this가 상위 함수는 window를 참조하는 것을 방지할 수 있다.




```
let Sports = function(){
  this.count = 20;
};
Sports.prototype = {
  plus: function(){
    this.count += 1;
  },
  get: function() {
    setTimeout(() => {
      this.plus();
      console.log(this.count);
    }, 1000);
  }
};
let newSports = new Sports();
newSports.get();
```

Arrow 함수

- This를 쓰기 올바른 예
  - 프로토 타입에 화살표 함수를 연결할 경우 this가 인스턴스를 참조하지 못한다.
  - 따라서 Prototype으로 함수를 선언하기 위해서는 화살표 함수가 아닌 일반 function으로 연결해야 한다.

```
20
NaN
```

```
let Sports = function(){
  this.count = 20;
};
Sports.prototype = {
  add: () => {
    this.count += 1;
  }
};
let newSports = new Sports();

newSports.add();
console.log(newSports.count);
console.log(window.count);
```

Arrow 함수

- Default parameter
  - Arrow 함수의 파라미터에도 디폴트 값을 작성할 수 있다.
  - 호출하는 함수에 파라미터 값을 넘겨주지 않거나 undefined를 넘겨주면 적용된다.

```
let plus = (one, two = 2) => one + two;
console.log(plus(1));

console.log(plus(1, undefined));

console.log(plus(1, 70));
```

```
3
3
71
```

Arrow 함수

- Default parameter
  - 파라미터 디스트럭쳐링 적용

```
let getTotal = ([one, two] = [10, 20]) => one + two;
console.log(getTotal());

let getValue = ({two: value} = {two: 20}) => value;
console.log(getValue());
```

```
30
20
```

Arrow 함수

- Arrow 함수 Rest 파라미터
  - Arrow 함수에 적용되는 Rest 파라미터 예제

```
let get = (one) => {
  console.log(one);
}
get(...[1, 2, 3]);
```

```
1
```

```
let get = (...rest) => {
  console.log(rest);
  console.log(Array.isArray(rest));
}
get(...[1, 2, 3]);
```

```
▼Array(3) i
    0: 1
    1: 2
    2: 3
    length: 3
  ▶ __proto__: Array(0)
true
```

```
let get = (one, ...rest) => {
  console.log(one);
  console.log(rest);
}
get(...[1, 2, 3]);
```

```
1
▼Array(2) i
    0: 2
    1: 3
    length: 2
  ▶ __proto__: Array(0)
```