

JavaScript (ES6) For Framework

8-2 this의 모든 것

“

- 일반 함수를 호출했을 때 **this**은 어디로 바인딩 되느냐
 - 객체를 호출했을 때 **this**는 어디로 바인딩 되느냐
- 생성자 함수를 호출했을 때 **this**는 어디로 바인딩 되느냐

일반적인 함수 호출의 this 바인딩

함수는 사실 전역 객체의 메서드
전역 변수도 사실 전역 객체의 속성

일반적인 함수 호출의 this 바인딩

```
function myFunc() {  
  console.log(this 값: ", this);  
}
```

`myFunc();` // 내가 정의한 `myFunc` 함수 실행

`window.myFunc();` // 전역 객체 (`window`)의 메서드 `myFunc` 실행

일반적인 함수 호출의 this 바인딩

```
var name = "minchul";  
  
console.log("전역변수 name : ", name);  
console.log("전역객체의 속성 name : ", window.name);  
  
// 결과는 같다
```

“

일반 함수의 호출 과정에서의 *this* 는
전역 객체를 가리킨다

일반적인 함수 호출의 this 바인딩

```
var name = 'KangMinchul';  
console.log(window.name);
```

```
var sayHello = function() {  
    var name = 'Kang Youngsu';  
    console.log(this.name);  
}  
= 전역객체
```

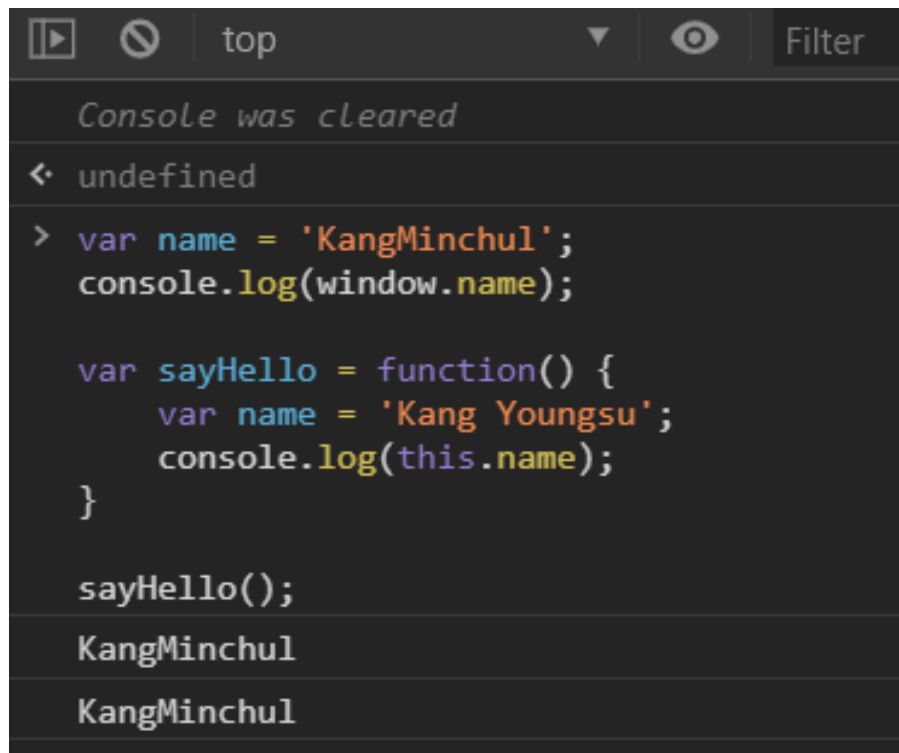
```
sayHello();
```

일반적인 함수 호출의 this 바인딩

```
var name = 'KangMinchul';  
console.log(window.name);
```

```
var sayHello = function() {  
    var name = 'Kang Youngsu';  
    console.log(this.name);  
}  
= 전역객체의 name  
= KangMinchul  
sayHello();
```


일반적인 함수 호출의 this 바인딩



```
top ▼ Filter
Console was cleared
< undefined
> var name = 'KangMinchul';
   console.log(window.name);

   var sayHello = function() {
       var name = 'Kang Youngsu';
       console.log(this.name);
   }

   sayHello();
KangMinchul
KangMinchul
```

객체의 메서드에서의 this 바인딩

객체의 메서드에서 사용된 this는
그 메서드를 호출한 객체로 바인딩 된다.

객체의 메서드에서의 this 바인딩

```
var myObject = {  
  name: 'minchul',  
  sayName: function () {  
    console.log(this.name);  
  }  
}
```

```
var otherObject = {  
  name: 'youngsu'  
}
```

```
otherObject.sayName = myObject.sayName;
```

```
myObject.sayName(); // minchul  
otherObject.sayName(); // youngsu
```

생성자 함수 호출에서의 this 바인딩


생성자 함수에서의 this는

그 생성자 함수를 통해 생성되어 반환되는 객체에 바인딩

생성자 함수 호출에서의 this 바인딩

```
// Person 생성자 함수
var Person = function(name) {
    this.name = name;
}

// boy 객체 생성
var boy = new Person('민철');
console.log(boy.name); // 민철
```

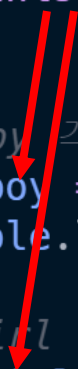


생성자 함수 호출에서의 this 바인딩

```
// Person 생성자 함수
var Person = function(name) {
    this.name = name;
}

// boy 객체 생성
var boy = new Person('민철');
console.log(boy.name); // 민철

// girl 객체 생성
var girl = new Person('영희');
console.log(girl.name); // 영희
```



```
function ComputerClass(name, professor, classno){  
    this.name = name;  
    this.professor = professor;  
    this.classno = classno;  
    this.printInfo = function(){  
        console.log(this.name + '강의'+ this.classno+ '분반입니다. 교수는' + this.professor + '입니다.');    };  
}  
  
var class1 = new ComputerClass('운영체제', '이동희', 2);  
var class2 = new ComputerClass('데이터베이스', '홍의경', 1);
```

생성자 함수 this vs 일반 함수 this

생긴 건 그냥 똑같이 생겼어도

new 로 새로운 객체를 만들면 생성자 함수
(new 없이) 그냥 호출되어 쓰이면 일반 함수


```
var name = "전역변수 name" ;
var classno = "전역변수 classno";
var professor = "전역변수 professor";

function ComputerClass(name, professor, classno){
    this.name = name;
    this.professor = professor;
    this.classno = classno;
    this.printInfo = function(){
        console.log( this.name + '강의'+ this.classno+ '분반입니다. 교수는' + this.professor + '입니다. ');
    };
}

var class1 = ComputerClass('운영체제', '이동희', 2);
var class2 = ComputerClass('데이터베이스', '홍의경', 1);
```

new 없이 ComputerClass 실행하면?

*ComputerClass 는 일반함수로서 실행,
this 는 전역객체에 바인딩됨*

주의 : 내부함수의 `this`

내부함수에서의 `this`는 무조건 전역 객체에 바인딩 된다

주의 : 내부 함수의 this

```
function myFunction() {  
  console.log("myFunction's this: ", this); // window에 바인딩  
  function innerFunction() {  
    console.log("innerFunction's this: ", this); // window에 바인딩  
  }  
  innerFunction();  
}  
myFunction();
```

일반 함수의 내부 함수 **innerFunction**의 **this**는 전역객체에 바인딩

주의 : 내부 함수의 this

```
var value = 1;

var obj = {
  value: 100,
  objmethod: function() {
    console.log("objmethod's this: ", this); // obj에 바인딩됨
    console.log("objmethod's this.value: ", this.value); // obj의 속성 100
    function innermethod() {
      console.log("innermethod's this: ", this); // window에 바인딩됨
      console.log("innermethod's this.value: ", this.value); // 전역변수 value : 1
    }
    innermethod();
  }
};

obj.objmethod();
```

메서드의 내부 함수 **innerMethod**의 **this**도 **전역객체**에 바인딩

주의 : 내부함수의 this

```
function constructor() {  
  console.log("constructor's this: ", this);  
  function innerFunction() {  
    console.log("innerFunction's this: ", this); // window에 바인딩됨  
  }  
  innerFunction();  
}  
constructor();  
  
myobj = new constructor();
```

생성자함수의 내부함수 **innerFunction**의 **this**도 전역객체에 바인딩

apply/call/bind

다음강의

Section 9

DOM 갖고 놀기

Lecturer Github : <https://github.com/kangtegong/>

수업자료링크 : <https://github.com/kangtegong/develop-javascript>

Email : tegongkang@gmail.com