JS-중급 11

태그	
날짜	@2023년 6월 5일

setTimeout/setInterval

```
setTimeout : 일정시간이 지난 후 함수 실행
setInterval : 일정시간 간격으로 함수를 반복


function fn(){
  console.log(3)
}

setTimeout(fn, 3000)

// 3초 후에 fn을 실행
// 두개의 매개변수 1. 일정시간이 지난 후 실행하는 함수, 2. 시간
// 아래 코드와 동일하다

setTimeout(function(){
    console.log(3)
},3000);

--------------------------------------------------------------------------

인수가 필요하다면 시간 뒤에 작성

const tId = function showName(name){
    console.log(name);
}

setTimeout(showName, 3000, 'Mike');

clearTimeout(tId); //스케줄을 취소 3초가 지나기전 실행되기때문에 아무일도 일어나지 않음


-------------------------------------------------------------------------------------

setInterval

function showName(name){
```

```
    console.log(name);
}

const tId = setInterval(showName, 3000, 'Mike');
//3초 간격으로 showName실행

clearInterval(tId);// interval 중

--------------------------------------------------------------------------------

delay타임을 0 으로 적으면 바로 실행되는게 아니라 4ms이후에 실행 되기때문에 정상작동하지않는다.
```

call, apply, bind

```
call, apply, bind :
함수 호출 방식과 관계없이 this를 지정할 수 있음

const mike = {
  name : "Mike"
};

const tom = {
  name : "Tom"
};

function showThisName(){
  console.log(this.name);
}

showThisName(); // window
showThisName.call(mike);
showThisName.call(tom);

"CodePen"
"Mike"
"Tom"

// call 메소드는 모든 함수에서 사용 할 수 있으며 this를 특정값으로 지정할 수 있다.
// 함수는 this.name을 보여준다.
// 빈 문자열로 하면 window를 가리킨다.
// .call로 mike를 가리키면 mike의 name을 본다.
```

```
// 함수로 호출하면서 콜을 사용하고 디스로 사용할 객체를 넘기면 해당 함수가 주어진
// 객체 메소드인 것처럼 사용할 수 있다.
// 콜의 첫번째 매개 변수는 디스로 사용할 값이고 매개변수가 더 있으면 매개변수로 호출하는
// 함수로 전달된다.


--------------------------------------------------------------------------------

const mike = {
  name : "Mike"
};

const tom = {
  name : "Tom"
};

function showThisName(){
  console.log(this.name);
}

function update(birthYear, occupation) {
  this.birthYear = birthYear;
  this.occupation = occupation;
};

update.call(mike, 1999, 'singer')

console.log(mike);

// 업데이트 함수를 만들었음
// 업데이트 함수는 생년과 직업을 받아서 이 객체의 정보를 새로운 데이터로 업데이트 해준다.

// 업데이트에 콜을 하고 this로 사용할 마이크 전달, 첫번째 매개변수인 생년 1999 전달,
// 두번째 매개변수인 직업 singer전달
// 그 후 마이크 확인

// [object Object]
{
  "name": "Mike",
  "birthYear": 1999,
  "occupation": "singer"
}

//tom도 해보면
update.call(tom, 1994, 'homeprotector');
console.log(tom);
// [object Object]
{
  "name": "Tom",
  "birthYear": 1994,
  "occupation": "homeprotector"
}

// 첫번째 매개변수는 this로 사용된 값, 두번째 매개변수부터는 함수에 사용될 매개변수를
```

```
// 순서대로 사용

--------------------------------------------------------------------------------

apply

//콜과 비슷한 어플라이도 알아보자
// 어플라이는 함수 매개변수를 처리하는 방법을 제외하면 콜과 같다.
// 콜은 일반적인 함수와 마찬가지로 매개변수를 직접받지만
// 어플라이는 매개변수를 배열로 받는다.

const mike = {
  name : "Mike"
};

const tom = {
  name : "Tom"
};

function showThisName(){
  console.log(this.name);
}

function update(birthYear, occupation) {
  this.birthYear = birthYear;
  this.occupation = occupation;
};

update.apply(mike, [1999, 'singer'])

console.log(mike);

update.apply(tom, [1994, 'homeprotector']);
console.log(tom);

// [object Object]
{
  "name": "Mike",
  "birthYear": 1999,
  "occupation": "singer"
}
// [object Object]
{
  "name": "Tom",
  "birthYear": 1994,
  "occupation": "homeprotector"
}

// apply는 배열 요소를 함수 매개변수로 사용할 때 유용하다.

const nums = [3,10,1,6,4];
const minNum = Math.min(...nums);
const maxNum = Math.max(...nums);
```

```
console.log(minNum);
console.log(maxNum);
1
10

apply사용

const nums = [3,10,1,6,4];
// const minNum = Math.min(...nums);
// const maxNum = Math.max(...nums);

const minNum = Math.min.apply(null, nums);
// =Math.min.apply(null, [3,10,1,6,4])
// apply

const maxNum = Math.max.call(null, ...nums);
// =Math.max.apply(null, [3,10,1,6,4])
// call

console.log(minNum);
console.log(maxNum);

1
10

// apply는 두번째 매개변수로 배열을 사용하면 그 요소들을 차례대로 인수로 사용한다.
// null은 this로 사용될 값인데 위 코드는 this가 사용업성서 null을 사용
// call 과 apply는 동작방식은 같다.
// 매개 변수를 받는 방법만 다른것
```

```
this 값을 바꿀 수 있는것은 바인드이다.

const mike = {
  name : "Mike",
}

function update(birthYear, occupation){
  this.birthYear = birthYear;
  this.occupation = occupation;
}

const updateMike = update.bind(mike);

updateMike(1980, "police");
console.log(mike)

// [object Object]
{
  "name": "Mike",
```

```
  "birthYear": 1980,
  "occupation": "police"
}

// 바인드를 사용하면 함수의 this값을 영구히 바꿀 수 있다.
// update를 이리저리 호출할때 this값이 항상 mike로 고정할때 bind를 사용


const user = {
  name : "Mike",
  showName: function(){
    console.log(`hello, ${this.name}`);
  },
};

user.showName();

let fn = user.showName;
fn.call(user);
fn.apply(user);

let boundFn = fn.bind(user);

boundFn();

"hello, Mike"
"hello, Mike"
"hello, Mike"
```