자바스크립트 비동기

2017.08.31
인간중심
윤주형 김성진 정구범

index

- javascript의 비동기
- callback
- promise
- promise + generator / iterator
- async / await

javascript의 비동기

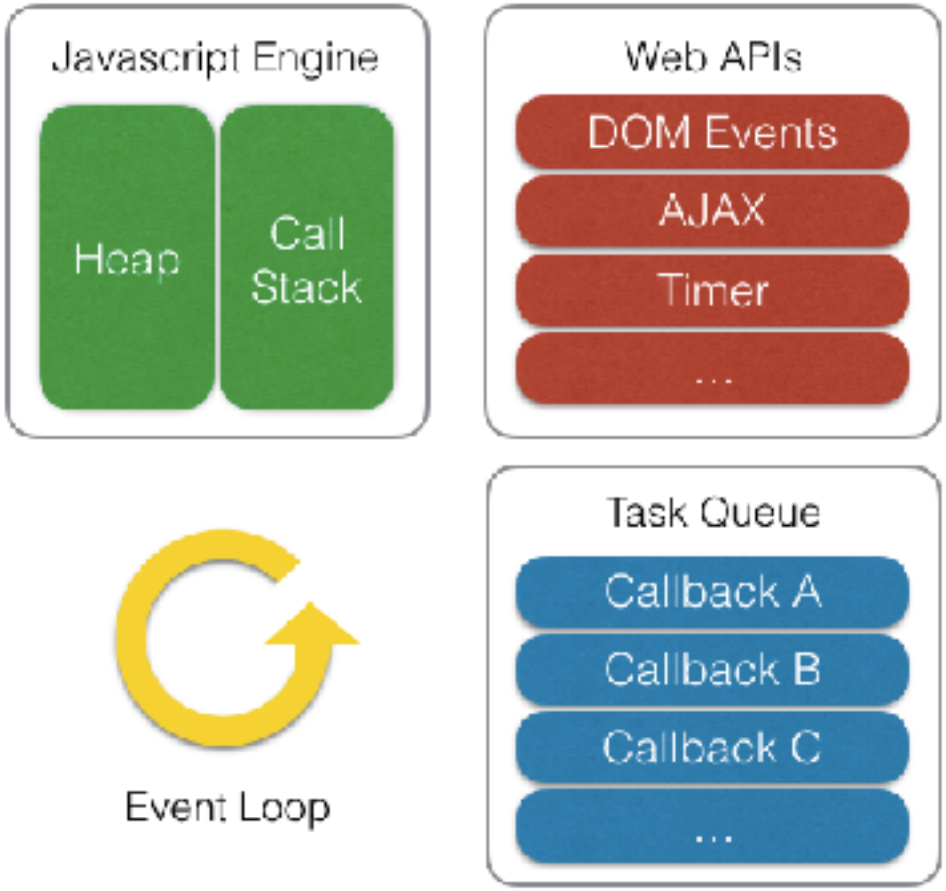
javascript의 비동기

비동기의 필요성

- orderCoffee()
- makeCoffee()

javascript의 비동기

비동기의 처리




```
setTimeout(function(){
    console.log("hello world");
}, 3000);
```

- Call Stack
- Web API
- Callback
- Event Loop

Callback

Callback

일반적인 콜백 사용

```
function setEvent(){
  let text = "click";
  $("button").on("click", function(){
      console.log(text);
  });
}
//es5
$("button").on("click", this.clicked.bind(this));
//es6
$("button").on("click", (e) => clicked(e));
```

- 인자로 함수사용
- 실행 순서 보장

Callback

일반적인 콜백 사용 한계

```
setTimeout(function (ch) {
    var boostCamp = ch;
        setTimeout(function (ch) {
            boostCamp += ch;
                setTimeout(function (ch) {
                    boostCamp += ch;
                        setTimeout(function (ch) {
                            boostCamp += ch;
                                setTimeout(function (ch) {
                                    boostCamp += ch;
                                    console.log(boostCamp);
                                }, 1, 'camp');
                        }, 1, ' ');
                }, 1, 'st');
        }, 1, 'oo');
}, 1, 'B');
```

- 콜백 지옥
- 가독성이 안좋다.

Callback

콜백 개선 방법

1. 동기로 처리

Callback

동기로 처리

```
function(url){
    getName(url, function(result){
        name = result.name;
        getAge(name, function(result){
            age = result.age;
            getEmail(age, function(result){
                email = result.email;
                getPhone(email, function(result){
                    address = result.address;
                    console.log("end");
                })
            });
        });
    });
};
```

```
function(url){
  let name = getName(url);
  let age = getAge(name);
  let email = getEmail(age);
  let phone = getPhone(email);
  let address = getAdress(phone);
  console.log("end");
}
```

2. 콜백 함수 분리

Callback

콜백 함수 분리

```
function(url){
    getName(url, function(result){
        name = result.name;
        getAge(name, function(result){
            age = result.age;
            getEmail(age, function(result){
                email = result.email;
                getPhone(email, function(result){
                    address = result.address;
                    console.log("end");
                })
            });
        });
    });
};
```

```
function getData(url){
    getName(url, getNameCallback);
};
```

```
function getNameCallback(result) {
    getAge(result.name, getAgeCallback);
};
function getAgeCallback(result) {
    getEmail(result.age, getEmailCallback);
};
function getEmail(result) {
    getAge(result.email, getPhoneCallback);
};
function getPhone(result) {
    getEmail(result.phone, getAddressCallback);
};
function getAddress(result) {
    getAge(result.address);
    console.log("end");
};
```

Promise

Promise

ES6의 Promise 사용

```
let _promise = new Promise(function (resolve, reject) {
        window.setTimeout(function () {
            if (true) {
                resolve("complete");
            }
            else {
                reject(Error("error"));
            }
        }, 3000);
    });
```

Promise

ES6의 Promise 사용

```
let _promise = new Promise(function (resolve, reject) {
        window.setTimeout(function () {
            if (true) {
                resolve("complete");
            }
            else {
                reject(Error("error"));
            }
        }, 3000);
    });
```

```
> _promise
< ▶ Promise {[[PromiseStatus]]: "pending", [[PromiseValue]]: undefined}
```

Promise

ES6의 Promise 사용

```
let _promise = new Promise(function (resolve, reject) {
        window.setTimeout(function () {
            if (true) {
                resolve("complete");
            }
            else {
                reject(Error("error"));
            }
        }, 3000);
    });
```

```
> _promise
< ▶ Promise {[[PromiseStatus]]: "pending", [[PromiseValue]]: undefined}
```

3초 후

```
> _promise
< ▶ Promise {[[PromiseStatus]]: "resolved", [[PromiseValue]]: "complete"}
```

Promise

ES6의 Promise 사용

```
let _promise = new Promise(function (resolve, reject) {
        window.setTimeout(function () {
            if (true) {
                resolve("complete");
            }
            else {
                reject(Error("error"));
            }
        }, 3000);
    });
```

```
> _promise
< ▶ Promise {[[PromiseStatus]]: "pending", [[PromiseValue]]: undefined}
```

3초 후

```
> _promise
< ▶ Promise {[[PromiseStatus]]: "resolved", [[PromiseValue]]: "complete"}
```

```
_promise.then(function(result){
    console.log(result);
});
```

```
complete
< ▶ Promise {[[PromiseStatus]]: "resolved", [[PromiseValue]]: undefined}
```

Promise

ES6의 Promise 사용

```
var _promise = function (param) {
    return new Promise(function (resolve, reject) {
        window.setTimeout(function () {
            if (param) {
                resolve("complete");
            }
            else {
                reject(Error("error"));
            }
        }, 3000);
    });
};
```

```
_promise(true)
.then(function (text) {
    console.log(text);
}, function (error) {
    console.error(error);
});
```

Promise

promise chain

```
function(url){
    getName(url, function(result){
        name = result.name;
        getAge(name, function(result){
            age = result.age;
            getEmail(age, function(result){
                email = result.email;
                getPhone(email, function(result){
                    address = result.address;
                    console.log("end");
                })
            });
        });
    });
};
```

```
function(url){
  getName(url).then(function(result) {
    name = result.name;
    return getAge(name);
  })
  .then(function(result){
      age = result.age;
      return getEmail(age);
  })
  .then(function(result){
      email = result.email;
      return getPhone(email);
  })
  .then(function(result) {
      phone = result.phone;
      return getAddress(phone);
  })
  .then(function(result) {
      console.log("end");
  })
}
```

Promise

동기인듯 동기아닌 동기같은 비동기

```
function(url){
  let name = getName(url);
  let age = getAge(name);
  let email = getEmail(age);
  let phone = getPhone(email);
  let address = getAdress(phone);
  console.log("end");
}
```

```
function(url){
  getName(url).then(function(result) {
    name = result.name;
    return getAge(name);
  })
  .then(function(result){
      age = result.age;
      return getEmail(age);
  })
  .then(function(result){
      email = result.email;
      return getPhone(email);
  })
  .then(function(result) {
      phone = result.phone;
      return getAddress(phone);
  })
  .then(function(result) {
      console.log("end");
  })
}
```

Generator / Iterator

Generator / Iterator

Generator / Iterator 기본 사용법

```
function* generator(){
  let a = yield 10;
  let b = yield a+20;
  let c = yield b+30;
  return a + b + c; //1 + 2 + 3
}

let iterator = generator();
console.log(iterator.next()); //done : false, value : 10;
console.log(iterator.next(1)); //done : false, value : 21;
console.log(iterator.next(2)); //done : false, value : 32;
console.log(iterator.next(3)); //done : true, value: : 6;
```

Generator / Iterator

Generator / Iterator + Promise

```
function* generator(url){
  const name = yield getName(url);
  const age = yield getAge(name);
  const email = yield getEmail(age);
  const phone = yield getPhone(email);
  const address = yield getAdress(phone);
  console.log("end");
}
```

```
function runner(generator, onEnd) {
    const iter = generator();
    let result = iter.next();

    function loop(result) {
        if (result.done) {
            if(onEnd !== undefined) {
                onEnd(result.value);
            }
        } else {
            if (!(result.value instanceof Promise)) {
                //this is not promise
                loop(iter.next(result.value));
            } else {
                //this is promise
                result.value.then(value => {
                    loop(iter.next(value));
                });
            }
        }
    }
    loop(result);
}
```

async / await

async / await

동기같은 코드

```
async function generator(url){
  const name = await getName(url);
  const age = await getAge(name);
  const email = await getEmail(age);
  const phone = await getPhone(email);
  const address = await getAdress(phone);
  console.log("end");
}
```

실제 코드에 적용

Callback

1.

2.

실제 코드에 적용

Promise

1.

2.

실제 코드에 적용

Generator / iterator

1.

2.

실제 코드에 적용

async / await

1.

2.

Thank you