PRE-ONBOARDING

면접 단골 질문
이벤트 루프 다시 돌아보기

FRONTEND CHALLENGE 25차

wanted

면접 단골 질문
이벤트 루프 다시 돌아보기

Week 2-1

이벤트 루프 자세히 알아보기 2

himprover

wanted

강의 진행 미리보기

- 강의 OT
- 이벤트 루프 개념
- 비동기, 동기
- 싱글, 멀티 스레드

- 큐, 스택
- 이벤트루프 큐, 스택

- callback, promise
- async/await
- 이벤트 루프 고려한 코드

- 면접 질문 발표해보기

목차

3시간 미리보기

1. callback, promise 이해하기
2. async, await 이해하기
3. 이벤트 루프를 고려한 코드 작성하기
4. 면접관이 놀랄만한 포인트

wanted

1. Callback, Promise

JavaScript의 Callback

Callback Function

다른 함수가 실행이 끝난 뒤 실행되기를 원하는 함수

```
function 일반함수(callback){
  callback();
}

function 콜백함수(){
  console.log("callback 함수");
}


일반함수(콜백함수);
```

JavaScript의 Callback

```
function getId() {
  setTimeout(() => {
    console.log("ID");
  },500);
}

function getEmail() {
  setTimeout(() => {
    console.log("EMAIL");
  },100);
}

function getPassword(){
  setTimeout(() => {
    console.log("PASSWORD");
  },250);
}

getId();
getEmail();
getPassword();
```

예상되는 실행 결과는?

JavaScript의 Callback

```
function getId() {
  setTimeout(() => {
    console.log("ID");
  },500);
}

function getEmail() {
  setTimeout(() => {
    console.log("EMAIL");
  },100);
}

function getPassword(){
  setTimeout(() => {
    console.log("PASSWORD");
  },250);
}

getId();
getEmail();
getPassword();
```

예상되는 실행 결과는?

EMAIL

PASSWORD

ID

JavaScript의 Callback

```
function getId() {
  setTimeout(() => {
    console.log("ID");
  },500);
}

function getEmail() {
  setTimeout(() => {
    console.log("EMAIL");
  },100);
}

function getPassword(){
  setTimeout(() => {
    console.log("PASSWORD");
  },250);
}

getId();
getEmail();
getPassword();
```

ID
EMAIL
PASSWORD

순으로 console에 보여지게 하려면?

(callback을 사용해서)

JavaScript의 Callback

```
function getId(callback) {
  setTimeout(() => {
    console.log("ID");
    callback();
  },500);
}

function getEmail(callback) {
  setTimeout(() => {
    console.log("EMAIL");
    callback();
  },100);
}

function getPassword(){
  setTimeout(() => {
    console.log("PASSWORD");
  },250);
}

getId(()=>getEmail(()=>getPassword()));
```

JavaScript의 Callback

예상되는 실행 결과는?

```
let id, email, password;

function setId() {
  setTimeout(() => {
    id="1234";
  },500);
}

function setEmail() {
  setTimeout(() => {
    email="test@test.com";
  },100);
}

function setPassword(){
  setTimeout(() => {
    password="1q2w3e4r5t6y";
  },250);
}

function login(id, email, password){
  console.log(id, email, password);
}

setId();
setEmail();
setPassword();

login(id,email,password);
```

JavaScript의 Callback

```
undefined undefined undefined
```

```
let id, email, password;

function setId() {
  setTimeout(() => {
    id="1234";
  },500);
}

function setEmail() {
  setTimeout(() => {
    email="test@test.com";
  },100);
}

function setPassword(){
  setTimeout(() => {
    password="1q2w3e4r5t6y";
  },250);
}

function login(id, email, password){
  console.log(id, email, password);
}

setId();
setEmail();
setPassword();

login(id,email,password);
```

JavaScript의 Callback

id, email, password 순서대로 저장되고

저장이 모두 끝나면 login이 되게 하려면?

```
let id, email, password;

function setId() {
  setTimeout(() => {
    id="1234";
  },500);
}

function setEmail() {
  setTimeout(() => {
    email="test@test.com";
  },100);
}

function setPassword(){
  setTimeout(() => {
    password="1q2w3e4r5t6y";
  },250);
}

function login(id, email, password){
  console.log(id, email, password);
}

setId();
setEmail();
setPassword();

login(id,email,password);
```

JavaScript의 Callback

```
1234 test@test.com 1q2w3e4r5t6y
```

```
let id, email, password;

function setId(callback) {
  setTimeout(() => {
    id="1234";
    callback();
  },500);
}

function setEmail(callback) {
  setTimeout(() => {
    email="test@test.com";
    callback();
  },100);
}

function setPassword(callback){
  setTimeout(() => {
    password="1q2w3e4r5t6y";
    callback();
  },250);
}

function login(id, email, password){
  console.log(id, email, password);
}

setId(()=>setEmail(()=>setPassword(()=>login(id,email,password))));
```

JavaScript의 Callback

그럴 일이 있나요? 🤣

JavaScript의 Callback

그럴 일이 있나요? 🤣

정말 많아요 🫠

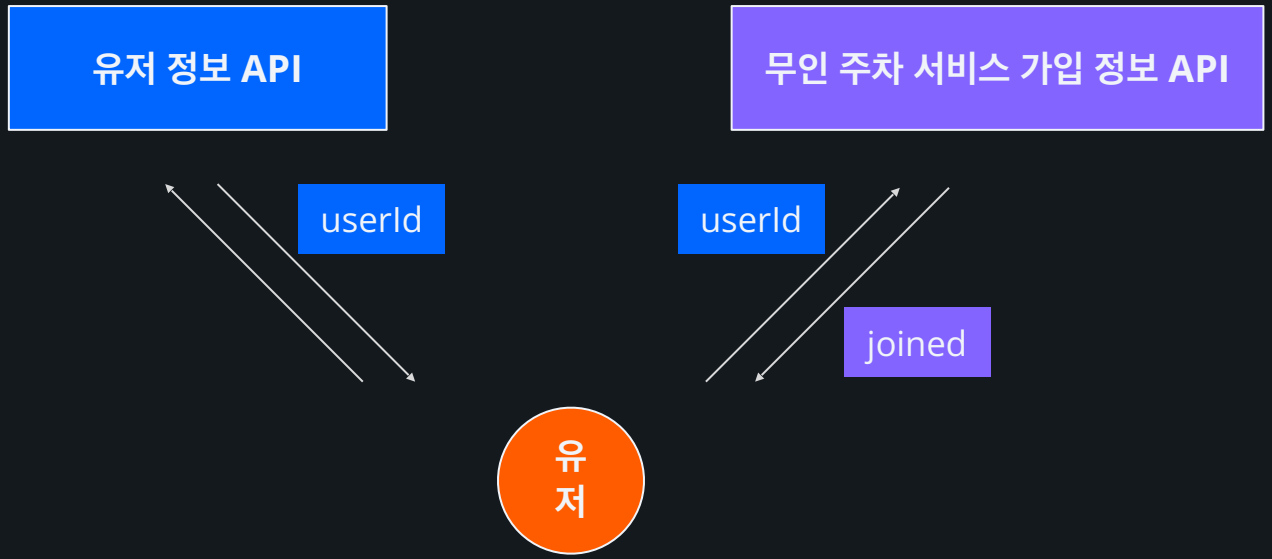
JavaScript의 Callback

유저 정보 API

무인 주차 서비스 가입 정보 API

유저

JavaScript의 Callback



JavaScript의 Callback

유저 정보 API

```
getUserData: (jwtToken:string) => {userId:string};
```

무인 주차 서비스 가입 정보 API

```
getJoinedParkingService: (jwtToken:string, userId:string)
                                                => {joined:boolean}
```

JavaScript의 Callback

유저

/service/parking 진입 시

무인 주차 서비스에 가입되어 있다면 **주차 현황 페이지로**

무인 주차 서비스에 가입되어 있지 않다면 **주차 서비스 소개 페이지로**

JavaScript의 Callback

```
function parkingServicePage () {
  const router = useRouter();

  getUserData(jwtToken, function (userId:string){
    getJoinedParkingService(jwtToken, userId, function (joined:boolean) {
      if(joined) {
        router.replace('/service/parking/active');
      }
      else {
        router.replace('/service/parking/landing');
      }
   });
}
```

JavaScript의 Callback

유저

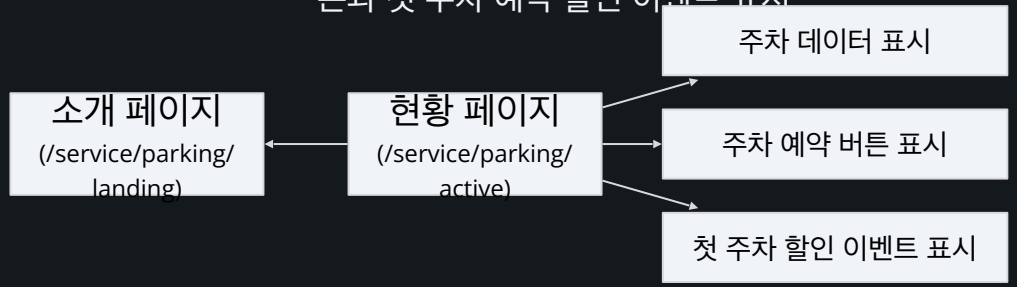
/service/parking/active 진입 시

무인 주차 서비스에 가입되어 있지 않다면 **주차 서비스 소개 페이지로**

무인 주차 서비스에 가입되어 있고, 현재 주차중이면 주차 데이터를 표시

무인 주차 서비스에 가입되어 있고, 현재 주차중이지 않으면 주차 예약 버튼 표시

무인 주차 서비스에 가입되어 있고, 현재 주차중이지 않으며, 한번도 주차한 기록이 없으면 주차 예약 버튼과 첫 주차 예약 할인 이벤트 표시



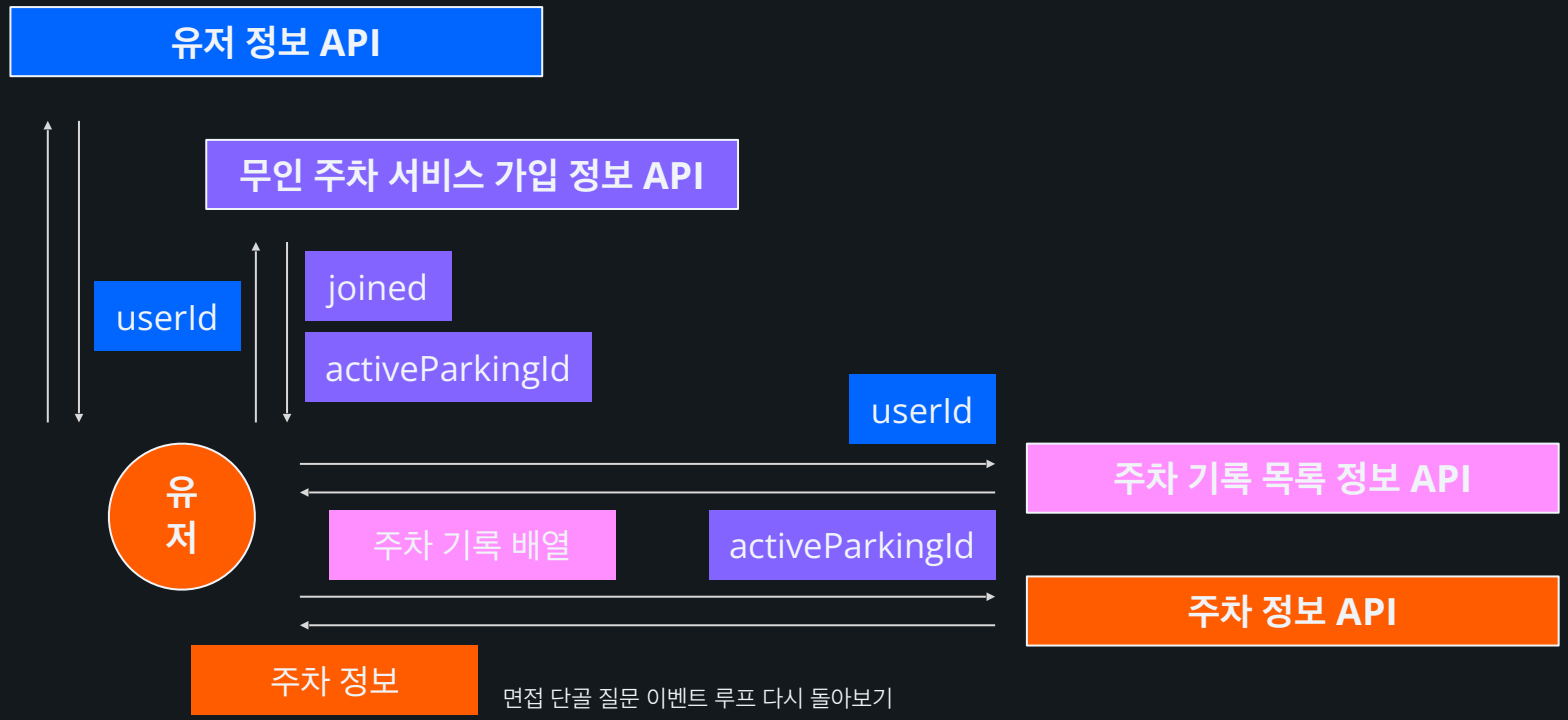
유저 정보 API

무인 주차 서비스 가입 정보 API

주차 기록 목록 정보 API

현재 주차 정보 API

JavaScript의 Callback



면접 단골 질문 이벤트 루프 다시 돌아보기

JavaScript의 Callback

```
getUserData: (jwtToken:string) => {userId:string};

getJoinedParkingService: (jwtToken:string, userId:string)
                => {joined:boolean, activeParkingId: string | null}

getParkingStatus: (jwtToken:string, parkingId:string)
                  => {parkingZoneName:string, parkingStartAt:string,
                                                  currentAmount:string}
getParkingHistory: (jwtToken:string, userId:string)
                => {history: {parkingStartAt:string, parkingEndAt:string}[]
```

1. Call

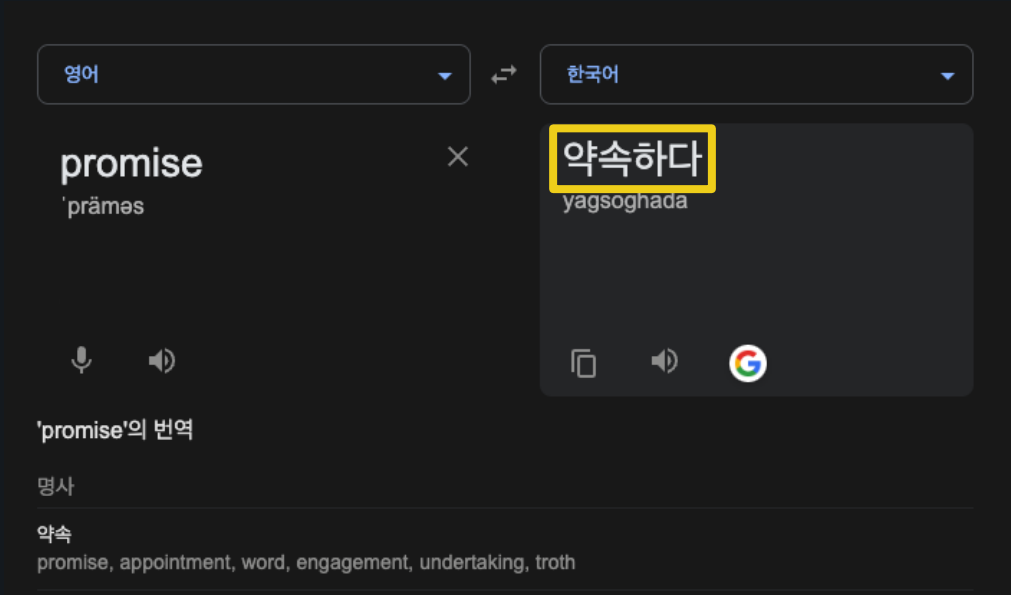
Jav

```
function parkingServicePage ( ) {
  const router = useRouter( );

  getUserData(jwtToken, function (userId:string){
    getJoinedParkingService(jwtToken, userId, function (joined, activeParkingId) {
      if(!joined) {
        router.replace('/service/parking/landing');
        return;
      }

      if(activeParkingId) {
        getParkingStatus(jwtToken, activeParkingId, function (props) {
          showParkingStatus(props.parkingZoneName, props.parkingStartAt,
                                                          props.currentAmount);

          return;
        }
        return;
      }

      getParkingHistory(jwtToken, userId, function ({history}){
        if(history.length === 0){
          showEvent( );
          showReserveButton( );
        }
        else {
          showReserveButton( );
        }
      });
    });
}
```

Callback 지옥

```
함수1(function (결과1){
  함수2(결과1, function (결과2) {
    함수3(결과2, function (결과3) {
      함수4(결과3, function (결과4) {
        함수5(결과4, function (결과5) {
          함수6(결과5, function (결과6) {
            함수7(결과6, function (결과7) {
              console.log(결과7);
            })
          })
        })
      })
    })
  })
});
```

JavaScript의 Promise



JavaScript의 Promise

Promise

`Promise` 객체는 비동기 작업이 맞이할 미래의 완료 또는 실패와 그 결과 값을 나타냅니다.

JavaScript의 Promise

설명

`Promise` 는 프로미스가 생성된 시점에는 알려지지 않았을 수도 있는 값을 위한 대리자로, 비동기 연산이 종료된 이후에 결과 값과 실패 사유를 처리하기 위한 처리기를 연결할 수 있습니다. 프로미스를 사용하면 비동기 메서드에서 마치 동기 메서드처럼 값을 반환할 수 있습니다. 다만 최종 결과를 반환하는 것이 아니고, 미래의 어떤 시점에 결과를 제공하겠다는 '프로미스(promise)'를 반환합니다.

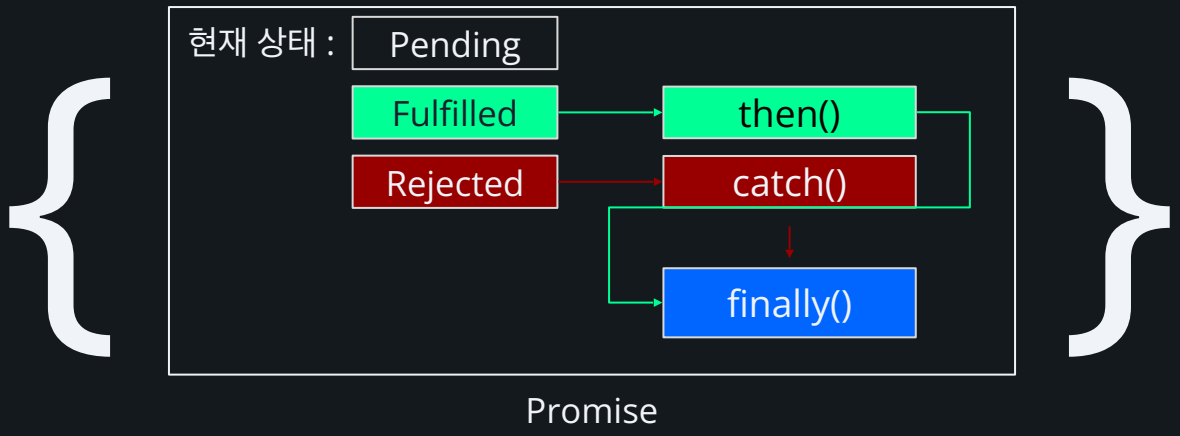
JavaScript의 Promise

Promise

객체

우리가 아는 그 **Object**

JavaScript의 Promise



JavaScript의 Promise

```
const 프로미스객체 = new Promise((resolve, reject) => {
  const 비동기작업결과 = 비동기함수();

  if(비동기작업결과)
    resolve(비동기작업결과);
  else
    reject("비동기 작업 실패");
}
```

1. Call

Jav

```
function parkingServicePage ( ) {
  const router = useRouter( );

  getUserData(jwtToken, function (userId:string){
    getJoinedParkingService(jwtToken, userId, function (joined, activeParkingId) {
      if(!joined) {
        router.replace('/service/parking/landing');
        return;
      }

      if(activeParkingId) {
        getParkingStatus(jwtToken, activeParkingId, function (props) {
          showParkingStatus(props.parkingZoneName, props.parkingStartAt,
                                                          props.currentAmount);

          return;
        }
        return;
      }

      getParkingHistory(jwtToken, userId, function ({history}){
        if(history.length === 0){
          showEvent( );
          showReserveButton( );
        }
        else {
          showReserveButton( );
        }
      });
    });
}
```

개선해본다면?

```
function parkingServicePage () {
  const router = useRouter();

  getUserData(jwtToken)
    .then(userId => {
      return getJoinedParkingService(jwtToken, userId).then(({ joined, activeParkingId }) => ({
        userId, joined, activeParkingId
      }));
    })
    .then(({ userId, joined, activeParkingId }) => {
      if (!joined) {
        router.replace('/service/parking/landing');
        return;
      }

      if (activeParkingId) {
        return getParkingStatus(jwtToken, activeParkingId).then(props => {
          showParkingStatus(props);
          return;
        });
      }

      return getParkingHistory(jwtToken, userId).then(({ history }) => {
        if (history.length === 0) {
          showEvent();
        }
        showReserveButton();
      });
    })
}
```

JavaScript의 Promise

Promise 버전의 setTimeout 구현해보기

```
function promiseSetTimeout (ms:number) {
  //
  //
  //
}


function main() {
  const delay500ms = promiseSetTimeout(500);


  delay500ms.then(() => {
    console.log("500ms 딜레이 후 출력");
  });

}
```

JavaScript의 Promise

```
function promiseSetTimeout (ms:number) {
  return new Promise((resolve) => {
    setTimeout(()=>{
      resolve();
    }, ms);
  });
}

function main() {
  const delay500ms = promiseSetTimeout(500);


  delay500ms.then(() => {
    console.log("500ms 딜레이 후 출력");
  });

}
```

Promise Factory Function

JavaScript의 Promise 와 EventLoop

```
const promise = Promise.resolve(1);

promise.then((data)=>console.log(data));

console.log("코드 실행 완료");
```

JavaScript의 Promise 와 EventLoop

```
const promise = Promise.resolve(1);

promise.then((data)=>console.log(data));

console.log("코드 실행 완료");
```

```
코드 실행 완료
1
```

wanted

2. Async, Await

JavaScript의 Promise의 한계

```
getUserData(jwtToken)
    .then(data => getJoinedParkingService(jwtToken, data.userId))
    .then(data => {
        if(!data.joined || !data.activeParkingId)
          return null
        return getParkingStatus(jwtToken, data.userId, data.activeParkingId); })
    .then(data => {
        if(data) console.log(data)
        else console.log("주차 정보 없음"); })
    .catch(error => console.error(error));
```

JavaScript의 Promise의 한계

```
async function main () {
  try {
    const {userId} = await getUserData(jwtToken);
    const {joined, activeParkingId} = await getJoinedParkingService(jwtToken, userId);

    if(!joined || !activeParkingId){
      console.log("주차 정보 없음");
      return;
    }

    const data = await getParkingStatus(jwtToken, userId, activeParkingId);
    console.log(data);

  } catch (error) {
    console.error(error);
  }
}
```

JavaScript의 Async, Await

Async, Await을 도입 한 건 Promise 보다 기능적으로 우수해서?

JavaScript의 Async, Await

Async와 Await은 Promise 객체를 사람이 다루기 쉽게

문법만 다르게 해주는 편의기능

실제로 JavaScript 는 Async, Await을 Promise 객체로 처리함

JavaScript의 Async, Await

문법만 다르게 해주는 편의기능

Syntax Sugar

템플릿 리터럴

속성명 단축

구조분해할당

삼항 조건 연산자

JavaScript의 Async, Await

await 키워드는 async function 내부에서만 사용 가능하다.

async function은 Promise 객체를 반환한다.

async function의 return 1;은 Promise resolve(1) 과 동일하다.

JavaScript의 Async, Await

```
function main() {
  const response = fetch("https://blog.himprover.com/");
  console.log(response);
}
```

```
> main();
  ▼ Promise {<pending>} i
    ▶ [[Prototype]]: Promise
      [[PromiseState]]: "fulfilled"
    ▶ [[PromiseResult]]: Response
```

JavaScript의 Async, Await

```
function main() {
  const response = await fetch("https://blog.himprover.com/");
  console.log(response);
}
```

```
> function main() {
    const response = await fetch("https://blog.himprover.com/");
    console.log(response);
  }
Uncaught SyntaxError: await is only valid in async functions and the top level bodies of modules
```

JavaScript의 Async, Await

```
function main() {
  const response = fetch("https://blog.himprover.com/");
  response.then(data=>console.log(data));
}
```

```
▼ Response {type: 'basic', url: 'https://blog.himprover.com/', redirected: false, status: 200, ok: true, …} i
    body: (...)
    bodyUsed: false
  ▶ headers: Headers {}
    ok: true
    redirected: false
    status: 200
    statusText: ""
    type: "basic"
    url: "https://blog.himprover.com/"
  ▶ [[Prototype]]: Response
```

JavaScript의 Async, Await

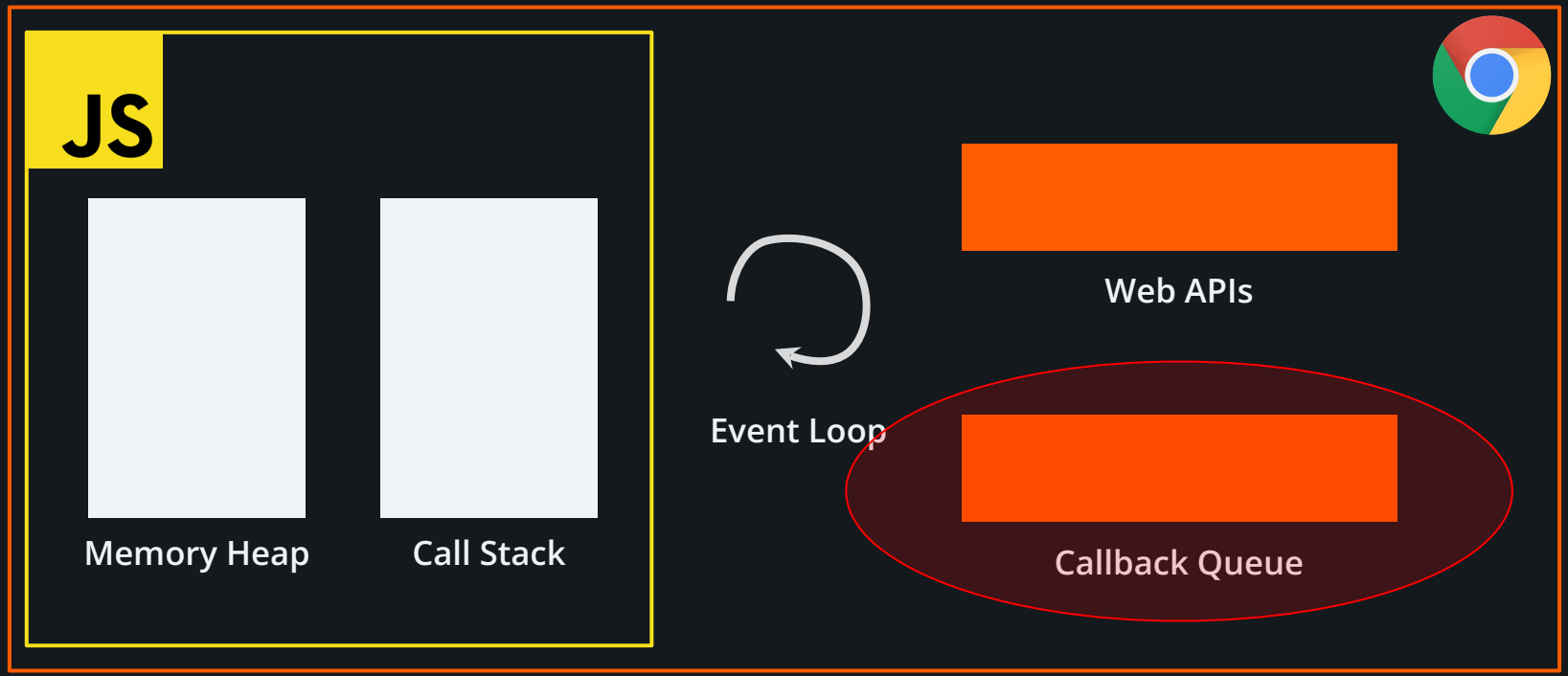
```
async function asyncFn (){
  return 1;
}

function main( ) {
  const response = asyncFn( );
  console.log(response);
}
```

```
main();
▼ Promise {<fulfilled>: 1} i
  ▶ [[Prototype]]: Promise
    [[PromiseState]]: "fulfilled"
    [[PromiseResult]]: 1
```

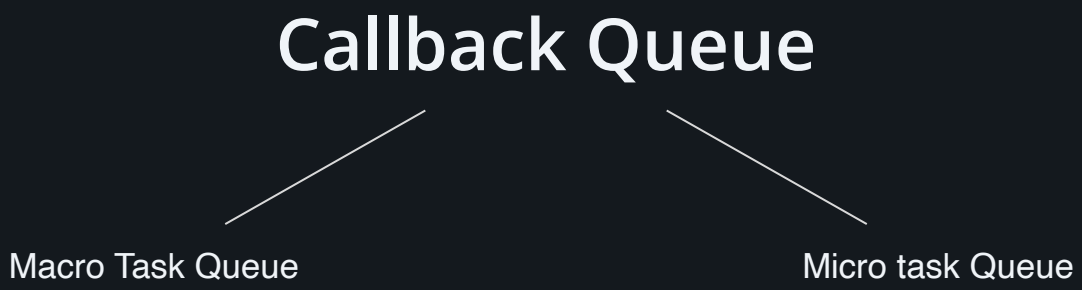
wanted

3. 이벤트 루프를 고려한 코드

JavaScript의 Event Loop 신경쓰기



JavaScript의 Event Loop 신경쓰기



JavaScript의 Event Loop 신경쓰기

우선순위

Macro Task Queue < Micro task Queue

JavaScript의 Event Loop 신경쓰기

Macro Task Queue

setTimeout, setInterval, DOM, I/O, Network request, Event handlers

Micro task Queue

Promise callback, mutation observer API, await in async func

JavaScript의 Event Loop 신경쓰기

Micro Task Queue

1번 수행할 때 큐를 **모두** 비운다

Macro Task Queue

1번 수행할 때 1개의 Task만 비운다

JavaScript의 Event Loop 신경쓰기

Micro Task Queue

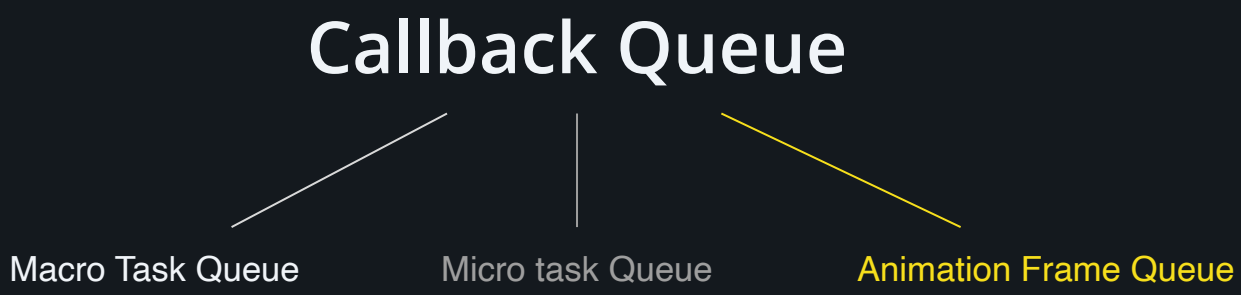
1번 수행할 때 큐를 **모두** 비운다

Macro Task Queue

1번 수행할 때 1개의 Task만 비운다

Micro Task Queue가 모두 비워질 때 까지 다른 것을 할 수 없다

JavaScript의 Event Loop 신경쓰기



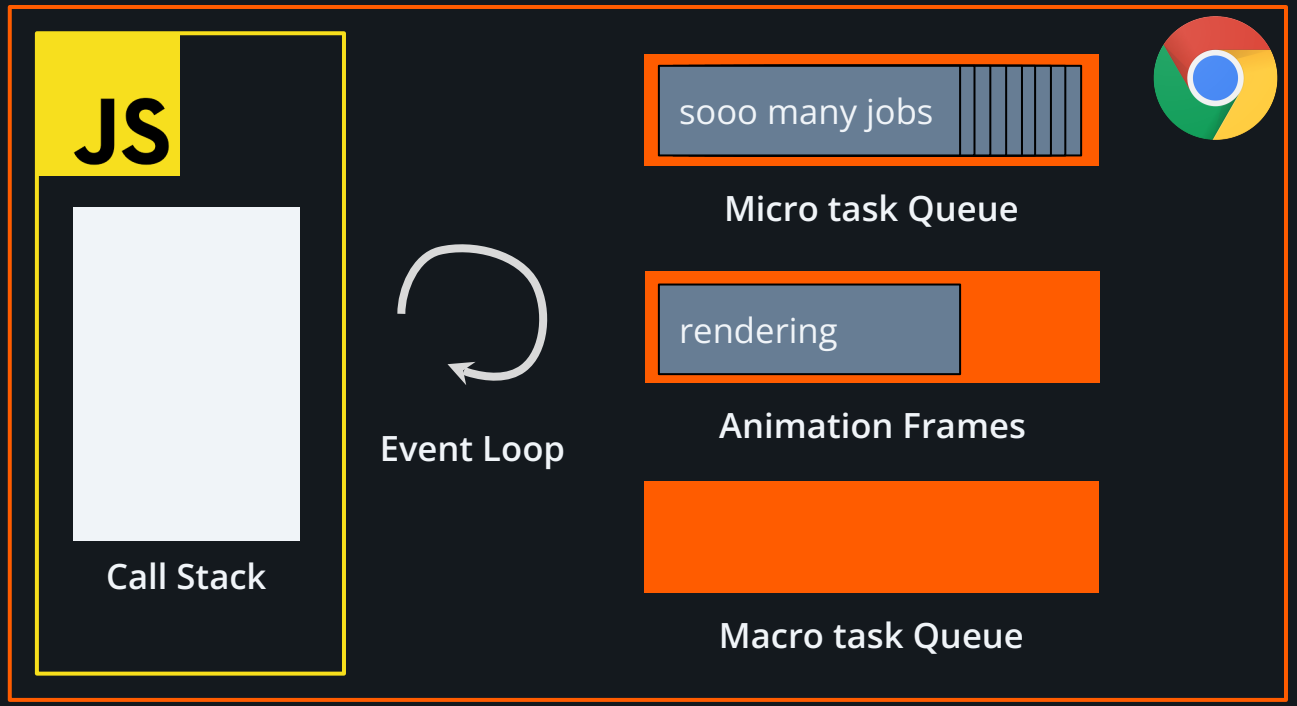
JavaScript의 Event Loop 신경쓰기

우선순위

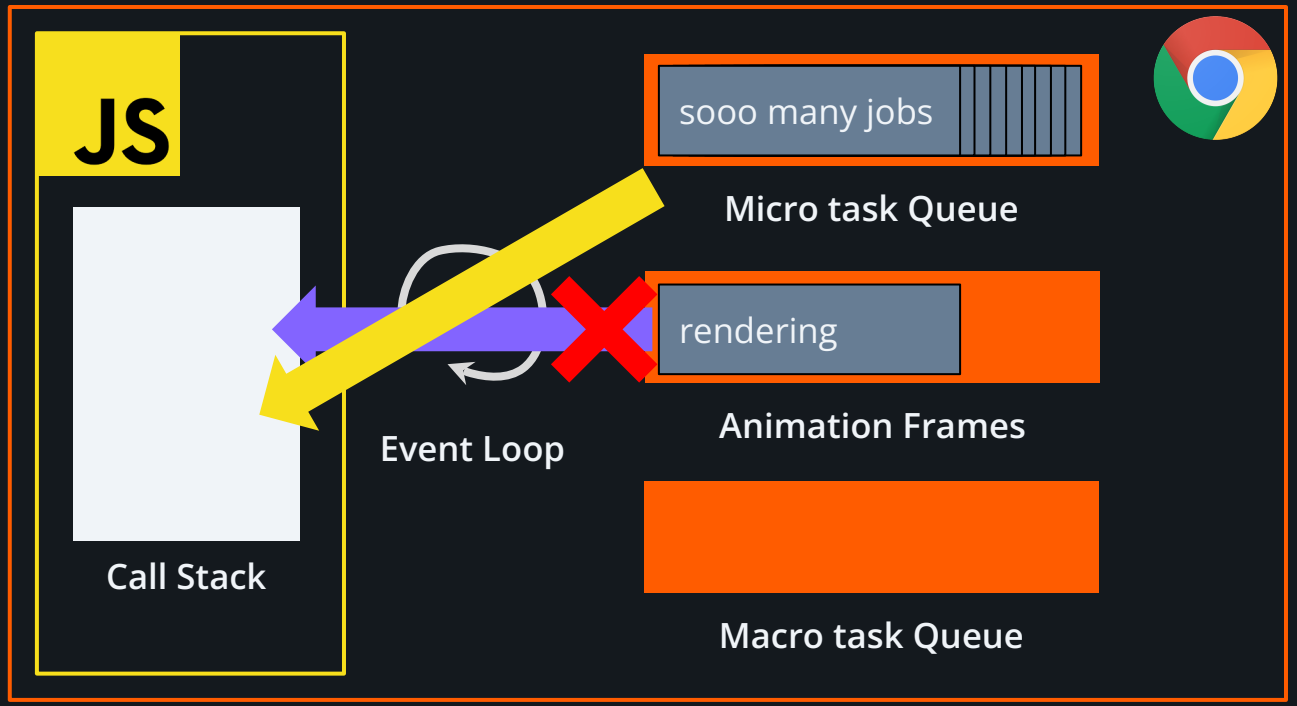
Macro Task Queue < Animation Frames < Micro task Queue

브라우저 렌더링 역할

JavaScript의 Event Loop 신경쓰기



JavaScript의 Event Loop 신경쓰기



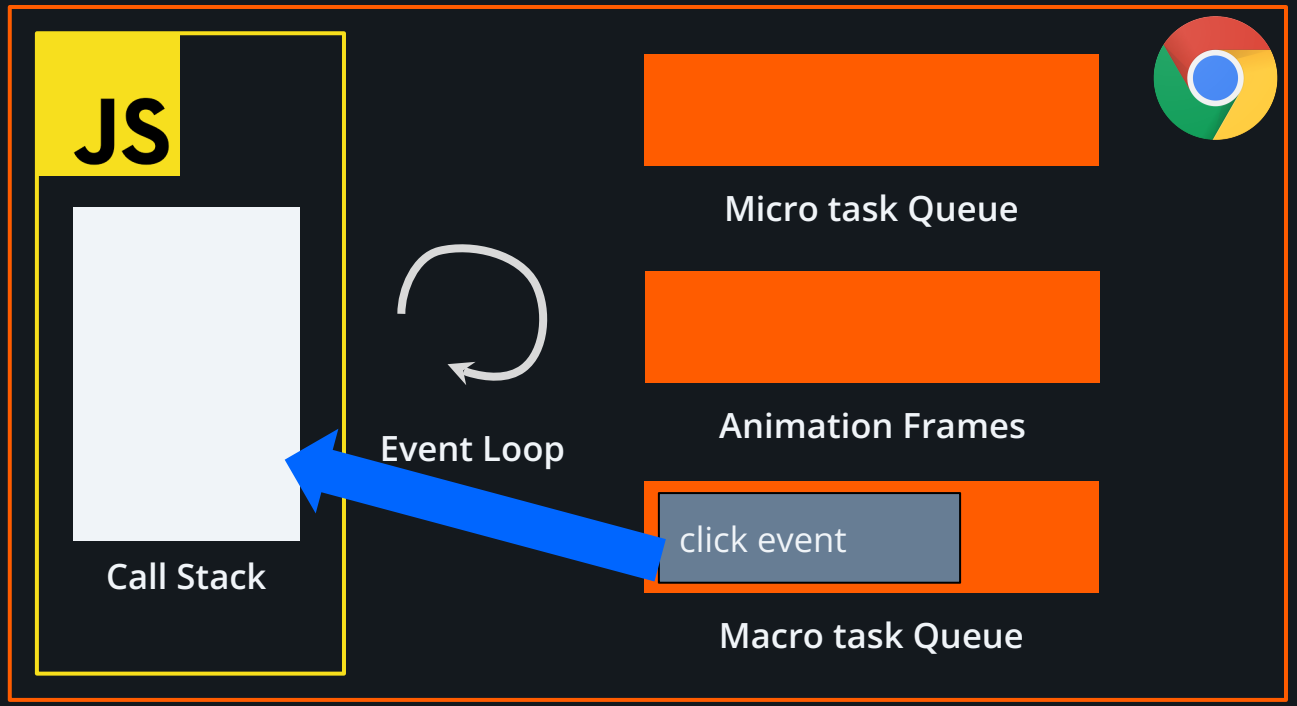
JavaScript의 Event Loop 신경쓰기

```
for(let i = 0; i < 10**7; i+=1) {
  queueMicrotask(()=>{});
}
```

JavaScript의 Event Loop 신경쓰기

```
for(let i = 0; i < 10**15; i+=1) {
  queueMicrotask(()=>{});
}
```

JavaScript의 Event Loop 신경쓰기



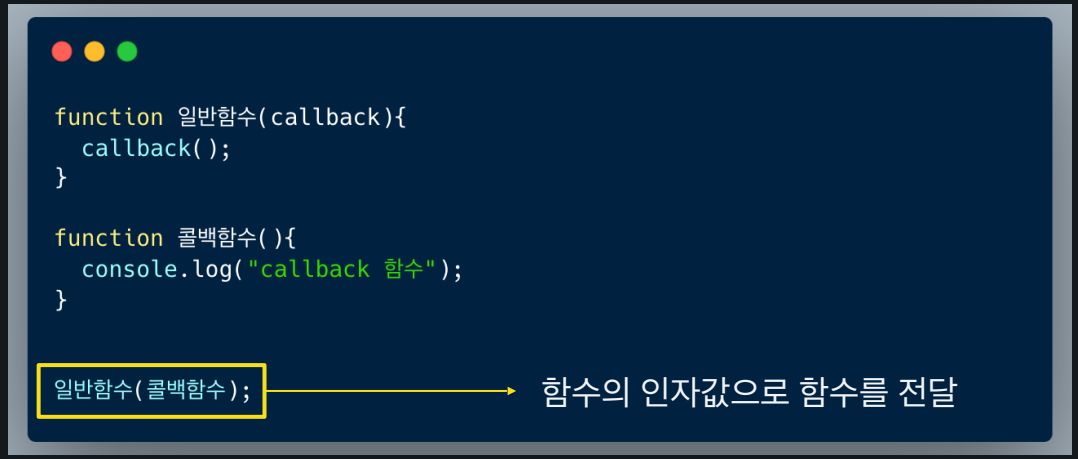
wanted

4. 면접관이 놀랄만한 포인트

JavaScript의 일급객체

Callback Function

다른 함수가 실행이 끝난 뒤 실행되기를 원하는 함수



JavaScript의 일급객체

```
const 함수가들어간변수 = function () {
  console.log("Hello");
}

함수가들어간변수();
```

JavaScript의 일급객체

당연한 거 아닌가요? 🤷‍♂️

당연한 건 아니에요 💁‍♂️

JavaScript의 일급객체

일급객체

First Class Object

기술 X **개념 O**

JavaScript의 일급객체

JavaScript의 함수는 일급객체이다

JavaScript는 일급함수를 가진다

JavaScript의 일급객체

다른 객체들에 일반적으로 적용 가능한 연산을 모두 지원하는 객체

JavaScript의 일급객체

함수의 매개변수가 될 수 있어야 한다.

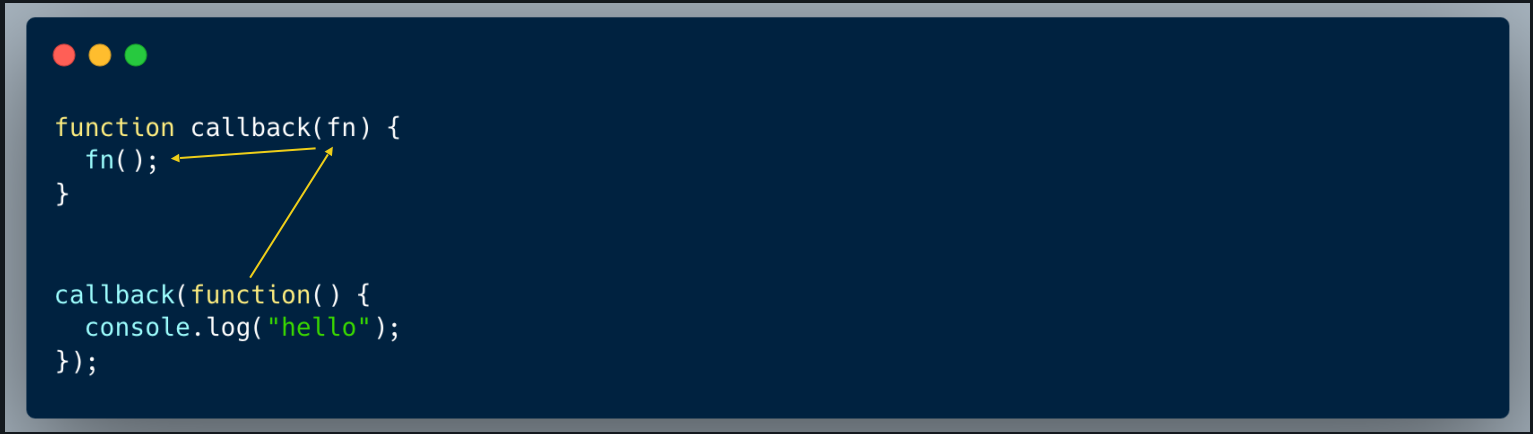
함수의 반환 값이 될 수 있어야 한다.

명령문으로 할당할 수 있어야 한다.

동일하게 비교할 수 있어야 한다.

JavaScript의 일급객체

함수의 매개변수가 될 수 있어야 한다.



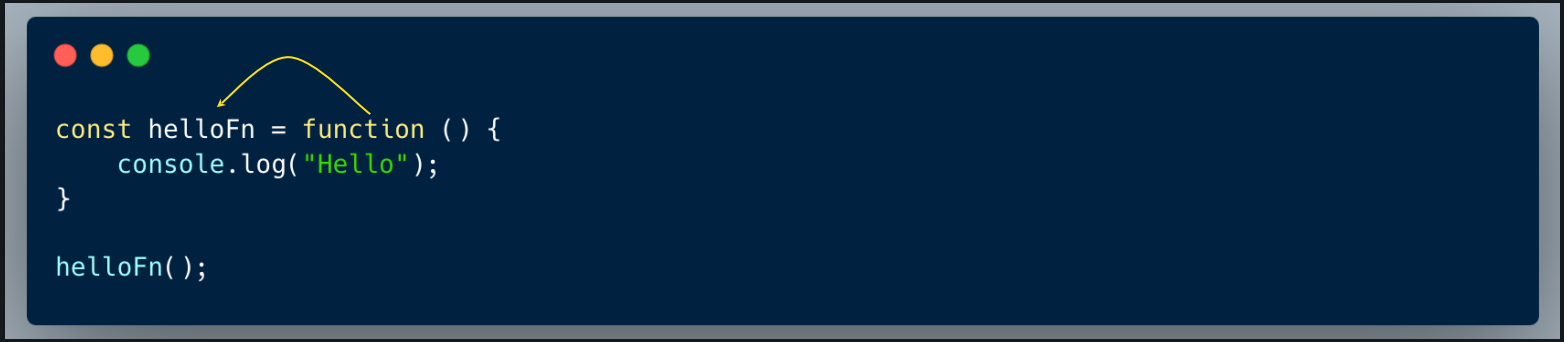
JavaScript의 일급객체

함수의 반환 값이 될 수 있어야 한다.

```
function getCreateDataHandler() {

  return function (userId) {
    mutateCreate(userId);
  }
}


const createDataHandler = getCreateDataHandler();

createDataHandler('1234');
```

JavaScript의 일급객체

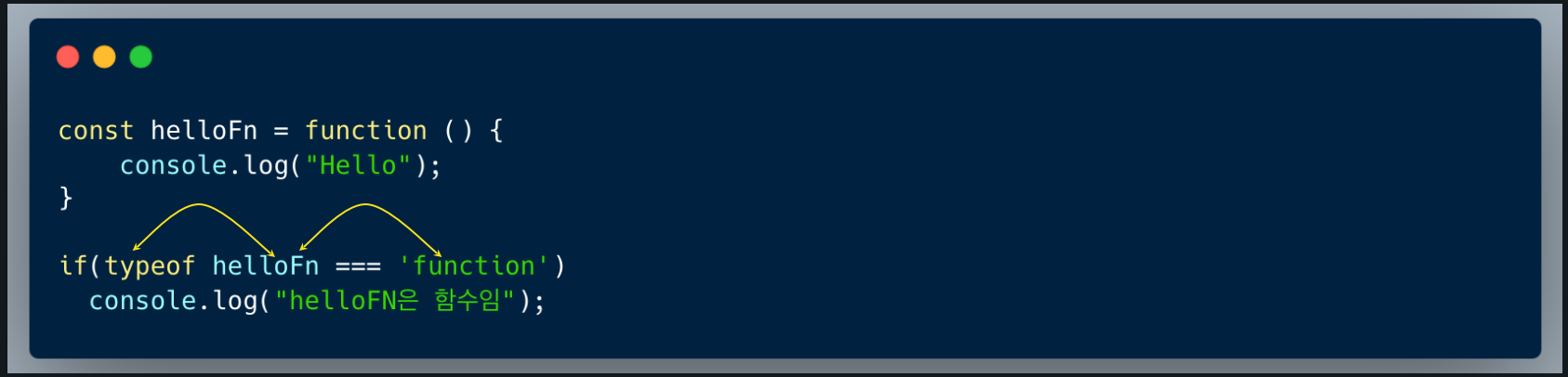
명령문으로 할당할 수 있어야 한다



```
const helloFn = function () {
    console.log("Hello");
}

helloFn();
```

JavaScript의 일급객체

동일하게 비교할 수 있어야 한다.



JavaScript의 일급객체

```
const getInformation = (name) => (phone) => (address) => `${name}, ${phone}, ${address}`;
```

JavaScript의 일급객체

```
const getInformation = (name) => (phone) => (address) => `${name}, ${phone}, ${address}`;

getInformation('황주현')('01012345678')('서울');
```

JavaScript의 일급객체

```
const getInformation = (grade) => (classNum) => (name) => `${grade}학년, ${classNum}반, ${name}`;


const get1Grade2Class = getInformation(2)(1);


get1Grade2Class('황주현');
get1Grade2Class('홍길동');
```

wanted

고생하셨습니다.