**Promises –** Something that gives you a value sometime in the future

Use this url - <https://jsonplaceholder.typicode.com/users/1>

console.**log**(**fetch**("https://jsonplaceholder.typicode.com/users/1"))

doing this command will show the promises are pending

to show the actual data we fetched we must:

1. Use then
2. Use Async/await (best practice but both extremely important to understand

**Then method**

console.**log**(response.**json**())

converting something we got from the backend that is compatible with the frontnend, something we can read using frontend code

**fetch**("https://jsonplaceholder.typicode.com/users/1").**then**(response => {

    response.**json**().**then**(data => {

        console.**log**(data)

    })

})

To get the full data

Two promises we need to use – first the fetch and then the response.json

To unlock a promise we use the then feature.

Why is this printing null? Because the javascript is running before html so we must use DEFER on script

const emailRef = document.**querySelector**(".email");

console.**log**(emailRef)

Selecting data from a specific element

const emailRef = document.**querySelector**(".email");

console.**log**(emailRef)

**fetch**("https://jsonplaceholder.typicode.com/users/1").**then**(response => {

    response.**json**().**then**(data => {

        console.**log**(data)

        emailRef.innerHTML = data.email;

    })

})

When we return inside of a then the entire thing becomes the response.json()

**fetch**("https://jsonplaceholder.typicode.com/users/1").**then**(response => {

    return response.**json**()

    })

And a .then at the end of this with the data and console log it

**fetch**("https://jsonplaceholder.typicode.com/users/1")

  .**then**((response) => {

    return response.**json**();

  })

  .**then**((data) => {

    console.**log**(data);

  });

Much cleaner

To then simply get certain data

**fetch**("https://jsonplaceholder.typicode.com/users/1")

  .**then**((response) => {

    return response.**json**();

  })

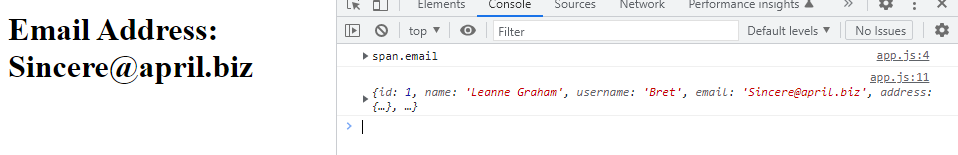
  .**then**((data) => {

    console.**log**(data);

    emailRef.innerHTML = data.email

  });

Results below



**Async/await method**

You need to await in an async function

async function **main**() {

    await **fetch**("https://jsonplaceholder.typicode.com/users/1")

}

**main**()

whenever you have a promise and you await it that is how you unlock what is inside of it

async function **main**() {

    const response = await **fetch**("https://jsonplaceholder.typicode.com/users/1")

    const data = await response.**json**()

    console.**log**(data)

    emailRef.innerHTML = data.email

}

**main**()

so if we console log response.json without the data or second await we don’t unlock it

async function **main**() {

    const response = await **fetch**("https://jsonplaceholder.typicode.com/users/1")

    const data = await response.**json**()

    console.**log**(data)

    emailRef.innerHTML = data.email

}

**main**()

**How to create a promise** (will go more in detail in interview section of FES – this is to further understanding of promise as a whole)

const subRef = document.**querySelector**(".sub")

function **getSubscriptionStatus**() {

    return new Promise((**resolve**, **reject**) => {

**setTimeout**(() => {

**resolve**("VIP")

        }, 2000)

    })

}

async function **main**() {

    const sub = (await **getSubscriptionStatus**())

    subRef.innerHTML = sub

}

**main**();

always put const at top of page



Inside of the try you have the promise and then you catch it – so if it returns an error it automatically runs whats below

    try {

        console.**log**(await **getVideo**(status))

    }

    catch (e) {

        console.**log**(e)

        videoRef.innerHTML = e;

    }

Challenge completed

function **getSubscriptionStatus**() {

    return new Promise((**resolve**, **reject**) => {

**setTimeout**(() => {

**resolve**(undefined)

        }, 2000)

    })

}

//challenge 23:33 includes subscription status from above

function **getVideo**(subscriptionStatus) {

    return new Promise((**resolve**, **reject**) => {

        if(subscriptionStatus === "VIP") {

**resolve**("show video")

        }

        else if(subscriptionStatus === "FREE") {

**resolve**("show trailer")

        }

        else{

**reject**("no video")

        }

    })

}

async function **main**() {

    const status = await **getSubscriptionStatus**();

    statusRef.innerHTML = status

    try {

        console.**log**(await **getVideo**(status))

    }

    catch (e) {

        console.**log**(e)

        videoRef.innerHTML = e;

    }

}

**main**();

If the subscription status says that there is a video then the error message will simply not appear. However as it is undefined we catch the error message being the else reject (“no video”)