"Asyncronous Javascript"

*Performing a simple Xml request by following:→

*new XmlHTTPRequest();
* Open the request.
* Send it.
* wait for the Request to be Loaded.

```
const getSpecificData = function (countryName) {
  const request = new XMLHttpRequest();
  request.open(
    'GET',
    `https://countries-api-836d.onrender.com/countries/name/${
countryName}`
  );
  request.send();
  request.addEventListener('load', function () {
    const [data] = JSON.parse(this.responseText);
    console.log(data);
    const html = `
    <article class="country">
          <img class="country__img" src="${data.flag}" />
          <div class="country__data">
            <h3 class="country__name">${data.name}</h3>
            <h4 class="country__region">${data.region}</h4>
            <p class="country__row"><span>👫</span>${(
              data.population / 1e6
            ).toFixed(1)} ${'M'}</p>
            <p class="country__row"><span>🗣️</span>${data.languages[0].
name}</p>
            <p class="country__row"><span>💰</span>${
              data.currencies[0].name
            }</p>
          </div>
        </article>
`;
    countriesContainer.insertAdjacentHTML('beforeend', html);
    countriesContainer.style.opacity = 1;
  });
};

```

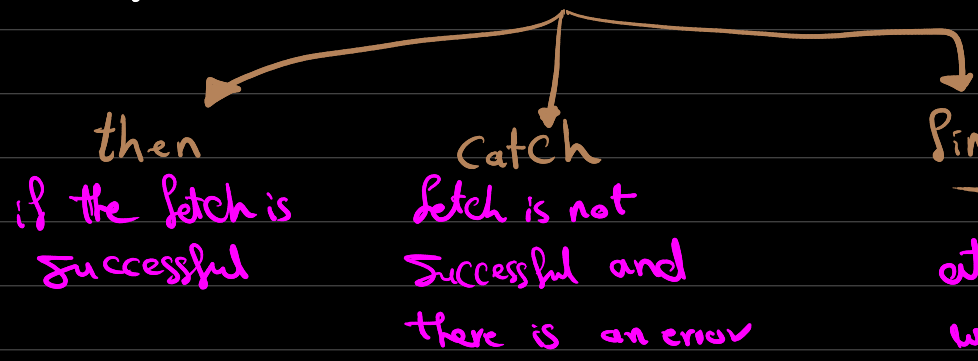
② Promises and fetch the APIs:→

```
const getSpecificCountry = function (countryName) {
  fetch(`https://countries-api-836d.onrender.com/countries/name/${countryName}`)
    .then(Response => Response.json())
    .then(data => renderElmnt(data[0]));
};
```

*This is how we Create a Promises, The first matter is the fetch which will return a Promise Container Then to access the data of this respone we should

Convert the response to Json but this will return a new promise, so we should return this Promise and Chain it by then which after that I'll get the targetted data.

* Catch is used to catch the error happing in the promise at the end of a chain.



* throw the errors manually :→

- As you know if I try to render a country that doesn't exist This will be also accepted promise and it will be handled throught then not Catch.
- the solution For this problem is to catch the err manually this mean I will terminate the process of promise chaining and jump into the catch part where error handling happens.
- In the following snapshot I check for the response status code it it is not ok i will terminate and go to catch.

```js
const renderCountry = function (countryName) {
  fetch(`https://countries-api-836d.onrender.com/countries/name/${countryName}`)
    .then(Response => {
      if (!Response.ok)
        throw new Error(`Country Not Found (${Response.status})`);
      return Response.json();
    })
    .then(data => renderElmnt(data[0]))
    .catch(err => {
      console.log(err);
      renderError(err.message);
    })
    .finally(() => (countriesContainer.style.opacity = 1));
};
renderCountry('Egy3pt');
```

* How JavaScript handles multiple tasks at the same time ?

1. normal executable sentences.
2. the ones which run in Background of web Apis such as set time out and Promise
3. Promise will go to Micro que which has higher Priority and executes first.
4. The other setTime out will go to the ordinary que which has less Priority

```js
console.log('Test Started');                                  → 1
setTimeout(() => {
  console.log('0 sec passed');                                → 4
}, 0);

Promise.resolve('First Promise Has Resolved').then(res => {   → 3
  for (let i = 0; i <= 10000000000; i++) {}
  console.log('Promise 1 Done');
});
console.log('Test Ended');                                    → 2
```

*Create your own Promise :→

```
const lotteryPromise = new Promise(function (resolve, reject) {
  console.log('Lottey Started');
  setTimeout(() => {
    if (Math.random() >= 0.5) {
      resolve('You Win');
    } else {
      reject(new Error('OPS! You Lost'));
    }
  }, 2000);
});

lotteryPromise.then(res => console.log(res)).catch(err => console.error(err));
```

This is well accept and reject a promise Based upon Some data.

*async - await functions :→

```
whereAmI()
  .then(msg => console.log(msg))
  .catch(err => console.log(`${err.message} 🤯🤯`))
  .finally(() => console.log('Done!'));
```

=

```
const rept = async function () {
  try {
    const res1 = await whereAmI();
    console.log(res1);
  } catch (err) {
    console.log(`${err.message} 🤯🤯`);
  }
  console.log('Done!');
};
```

The success which inside the Try will be excated in then and error will be caught in Catch and finally will be out of Both.

* Run multiple Promises at the Same time :→

↳ Promise.All

receive an array and returns array.

```
const data = await Promise.all([
  getJSON(`https://restcountries.eu/rest/v2/name/${c1}`),
  getJSON(`https://restcountries.eu/rest/v2/name/${c2}`),
  getJSON(`https://restcountries.eu/rest/v2/name/${c3}`),
]);
```

* If one Promise rejects, it short Circuts and return immediately.

* Promise.race :→

if I wanna make a get request but if it took too long it will be rejected.

. many fetch functions and return the first fullfilled one.

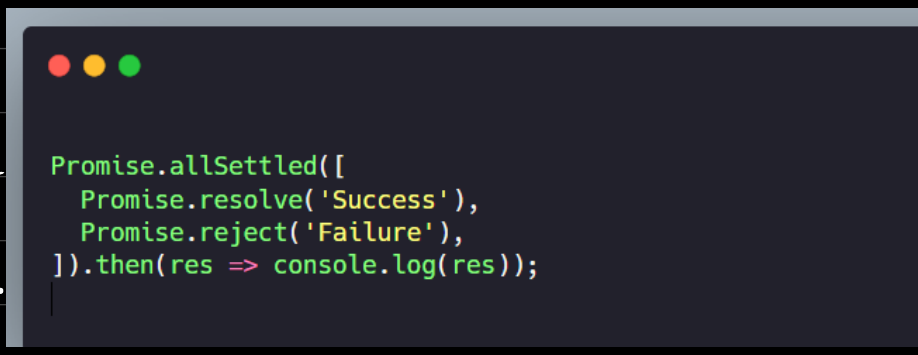
if it took more than 0.2 sec the await Func get to res as error

```
const awaitFunc = function (sec) {
  return new Promise(function (_, reject) {
    setTimeout(() => reject(new Error('Took Too Long!')), sec * 1000);
  });
};

const promiseRaceImplement = async function (countryName) {
  const res = await Promise.race([
    getJSON(`https://countries-api-836d.onrender.com/countries/name/egypt`),
    awaitFunc(0.2),
  ]);
  return res;
};

promiseRaceImplement()
  .then(res => console.log(res))
  .catch(err => console.error(err));
```

* Promise.allSettled

It takes an array of Promises and out an array The difference that the output contains the fullfilled and rejected Promises.

it will return an array of res even there is any rejected Promises, not like all.



```
Promise.allSettled([
  Promise.resolve('Success'),
  Promise.reject('Failure'),
]).then(res => console.log(res));
```