Synchronous Vs Asynchronous

Synchronous : Every statement of code get executed one by one

So basically, a statement has to wait for earlier statement to get executed

Eg - console.log ("I");

console.log ("eat");

console.log ("ice-cream");

It will print I first,
then eat,
after that ice-cream

Asynchronous ; It allows program to be executed immediately without blocking the code. Unlike the Synchronous method It doesn't wait for earlier statement to get executed first.

Each task execute completed Independently

Eg- console.log ("I");

```
setTimeout (() => {
    console.log ("eat"); }, 2000)

Console.log ("Ice Cream")
```

It will print

"I"

"Ice Cream" (will execute immediately)

"eat" (will print after 2s)

Asynchronous Functions

→ It contains async keyword.

How to use in Normal Function declaration

```
async function name (arg) {
    }
```

How to use in an arrow function

```
Cons functionName = async (arg) => {
    }
```

Asynchronous functions always return promises

It doesn't matter what you return. The returned value will always be promise.

Eg →

```
const getOne = async () => {
    return 1;
}

const promise = getOne();
console.log(promise)
```

The await keyword

The await keyword lets you wait for promise to resolve. Once promise is resolved it returns the parameter passed into then call.

Eg -

Eg →

```
const getOne = async _ => {
    return 1; }

getOne().then(value => {
                console.log(value) }); // 1
```

Now use of await keyword

```
const test = async _ => {
    const one = await getOne();
    Console.log(one);
}

test()
```

We can only use await when we have async.

Let's implement the fetch API code using async/await :

```
const FetchData = async () => {
const qoutes = await Fetch ("http://--/quotes");
const response = await qoutes.json();

console.log(response);

}

   FetchData();
```

We can also handle errors in async/await by using try and catch.

```
const FetchData = async () => {
try {
   const quotes = await Fetch ("http://...");
   const response = await quotes.json();
     console.log (response);
}
   catch (error) {
       console.log(error);
   }

};
   FetchData();
```

Synchronous Vs Asynchronous

Synchronous : Every statement of code get executed one by one

So basically, a statement has to wait for earlier statement to get executed

Eg - console.log ("I");

console.log ("eat");

console.log ("ice-cream");

It will print I first,
then eat,
after that ice-cream

Asynchronous ; It allows program to be executed immediately without blocking the code. Unlike the Synchronous method It doesn't wait for earlier statement to get executed first.

Each task execute completed Independently

Synchronous Vs Asynchronous

Synchronous : Every statement of code get executed one by one

So basically, a statement has to wait for earlier statement to get executed

Eg - console.log("I");

console.log("eat");

console.log("ice-cream");

It will print I first,
then eat,
after that ice-cream

Asynchronous; It allows program to be executed immediately without blocking the code. Unlike the Synchronous method It doesn't wait for earlier statement to get executed first.

Each task execute completed independently.

Eg- console.log ("I");

setTimeout (() => {
console.log ("eat"); }, 2000)

Console.log ("Ice Cream")

It will print
"I"
"Ice Cream" (will execute immediately)
"eat" (will print after 2s)

Asynchronous Functions

→ It contains async keyword.

How to use in Normal Function declaration

```
async function name ( arg ) {
    }
```

How to use in an arrow function

```
Cons functionName = async ( arg ) => {
    }
```

Asynchronous functions always return promises

It doesn't matter what you return. The returned value will always be promise.

Eg →

```
const getOne = async () => {
    return 1;
}

const promise = getOne();
console.log(promise)
```

The await keyword

The await keyword lets you wait for promise to resolve. Once promise is resolved it returns the parameter passed into then call.

Eg -

Eg →

```
const getOne = async _ => {
    return 1; }

getOne().then(value => {
                console.log(value) }); //1
```

Now use of await keyword

```
const test = async _ => {
    const one = await getOne();
    Console.log(one);
}

test()
```

We can only use await when we have async.

Let's implement the fetch API code using async/await :

Date: __/__/__

```
const FetchData = async () => {
const qoutes = await fetch ("http://--/quotes");
const response = await qoutes.json();

console.log(response);

}
   FetchData();
```

We can also handle errors in async/await by using try and catch.

```
const FetchData = async () => {
try {
   const quotes = await fetch ("http://__");
   const response = await quotes.json();
     console.log (response);
}
   catch (error) {
        console.log(error);
   }

};
   FetchData();
```