PRESENTS

ASYNCHRONOUS PROGRAMMING

what are we going to learn today....

BASICS OF JS

ASYNCHRONOUS PROGRAMMING

PROJECT:

battle of titan.

A high-level programming language for web development...

We mainly use JavaScript to create

01 Dynamic

02 Dynamically Typed

03 Loosely Typed

04 Interpreted

websites

web applications

server-side applications using Node.js

JAVASCRIPT

JAVASCRIPT
JAVASCRIPT

Basic terms to understand syntax in JS :

01 white space

02 case sensitivity

03 literals

04 identifiers

05 comments

```
// comment
/*
  multi-line
  comments
*/
```

VARIABLES :

Variables in JavaScript do not have any type attached.

```
const a = 0
```

```
let a = 0
```

- **PRIMITIVE**

```
let num = 35                //----> Numbers
let str = "Javascript" //----> Strings
let bool = True          //----> Boolean
```

And two special types: null and undefined.

Expressions:

An expression is a single unit of JavaScript code that the JavaScript engine can evaluate, and return a value.

```
1 / 2
i++
i -= 2
i * 2
```

```
'A ' + 'string'
```

```
a && b
a || b
!a
```

Arithmetic expressions

String expressions

Logical expressions

Conditionals:

An if statement is used to make the program take a route, or another, depending on the result of an expression evaluation.

```
if (true) {
  //do something
}
```

```
if (true) {
  //do something
} else {
  //do something else
}
```

An array is a collection of elements.

Arrays are objects.

```
const a = [1, 2, 3]
const a = Array.of(1, 2, 3)
```

A string is a sequence of characters which is enclosed in quotes or double quotes:

```
'A string'
'Another string'
```

Any value that’s not of a primitive type is an object.

```
const car = {
  color: 'blue',
  brand: 'Ford',
}
```

```
car.brand.name
```

LOOPS;

while

```
const list = ['a', 'b', 'c']
let i = 0
do {
  console.log(list[i]) //value
  console.log(i) //index
  i = i + 1
} while (i < list.length)
```

For

```
const list = ['a', 'b', 'c']

for (let i = 0; i < list.length; i++) {
  console.log(list[i]) //value
  console.log(i) //index
}
```

What is a function?

A function is a block of code, self-contained.

```
function getData() {
  //do something
}

function getData(color) {
  //do something
}

function getData(color, age) {
  //do something
}
```

```
function getData() {
  //...
}
```

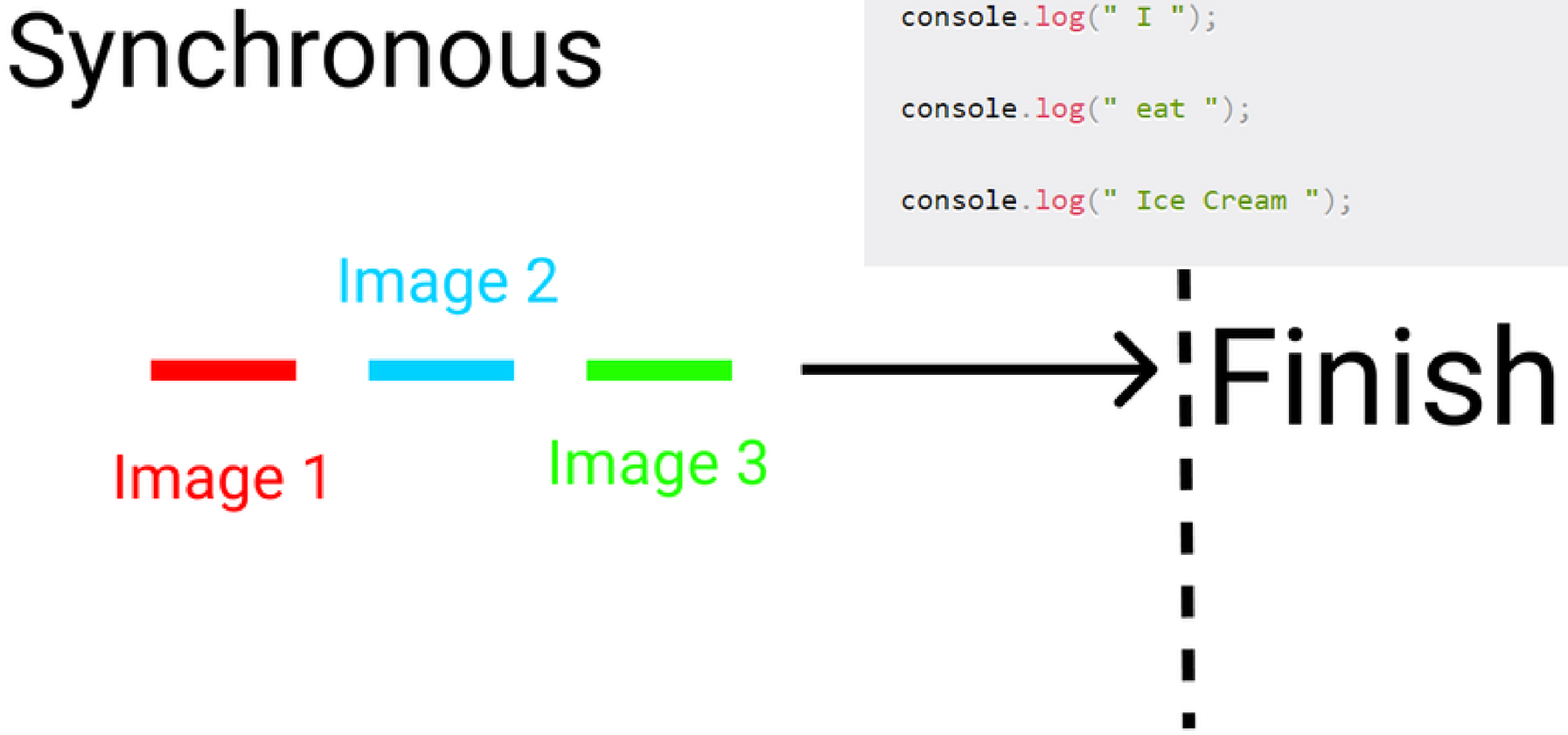
to

```
;() => {
  //...
}
```

Arrow function;

WHAT is Synchronous System?

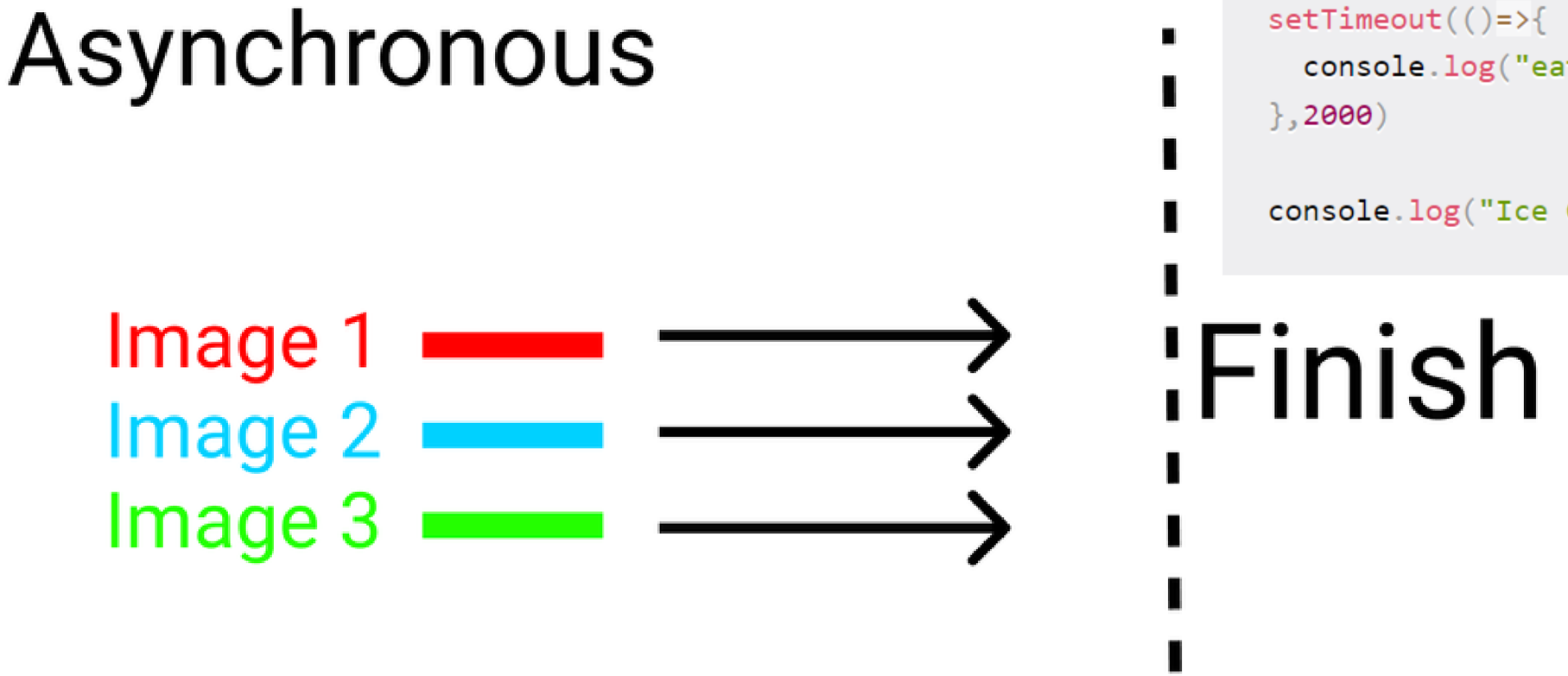
In a synchronous system, tasks are completed one after another.



WHAT is an Asynchronous System?

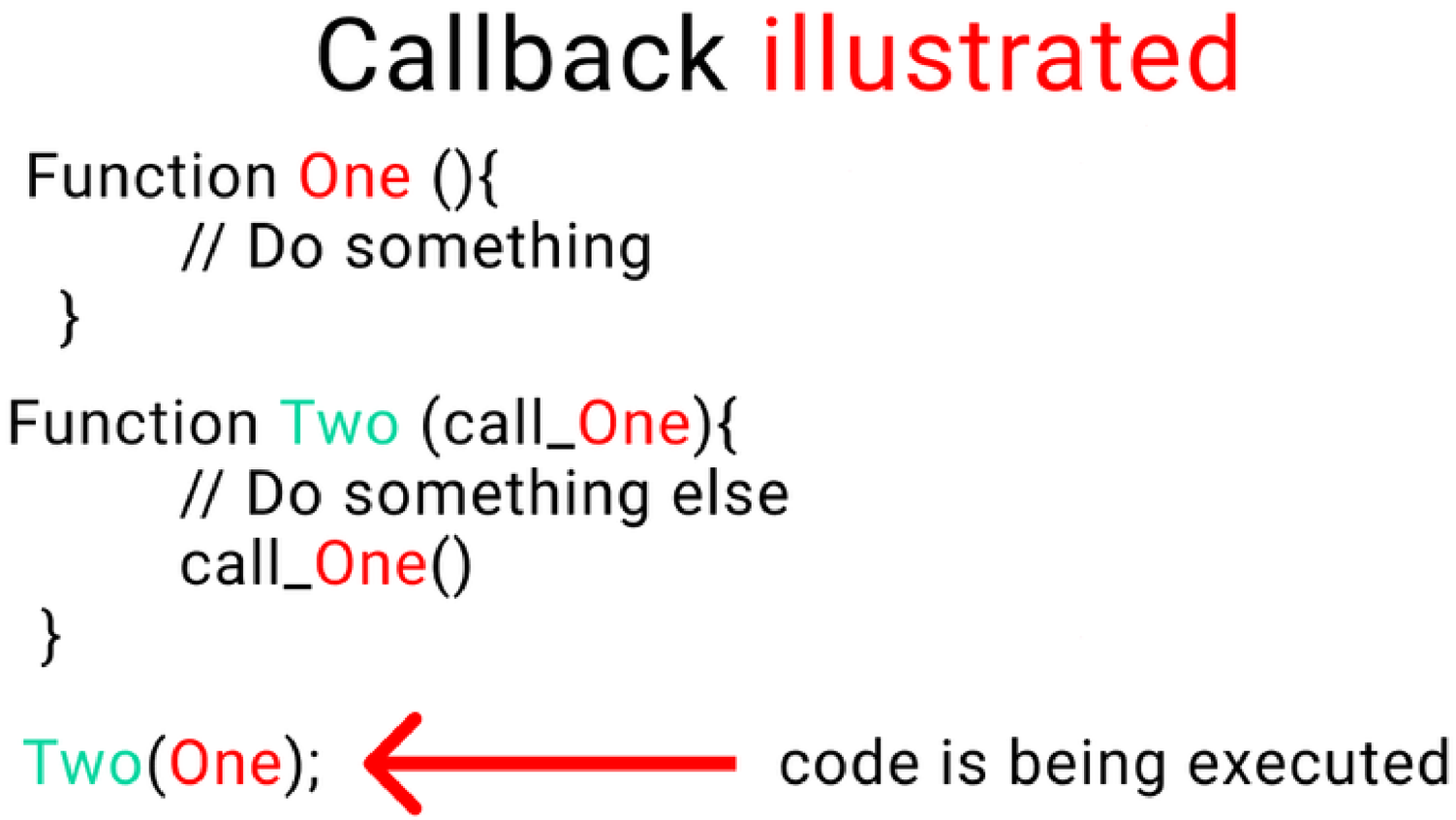
In this system, tasks are completed independently.

```
console.log("I");

// This will be shown after 2 seconds

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

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



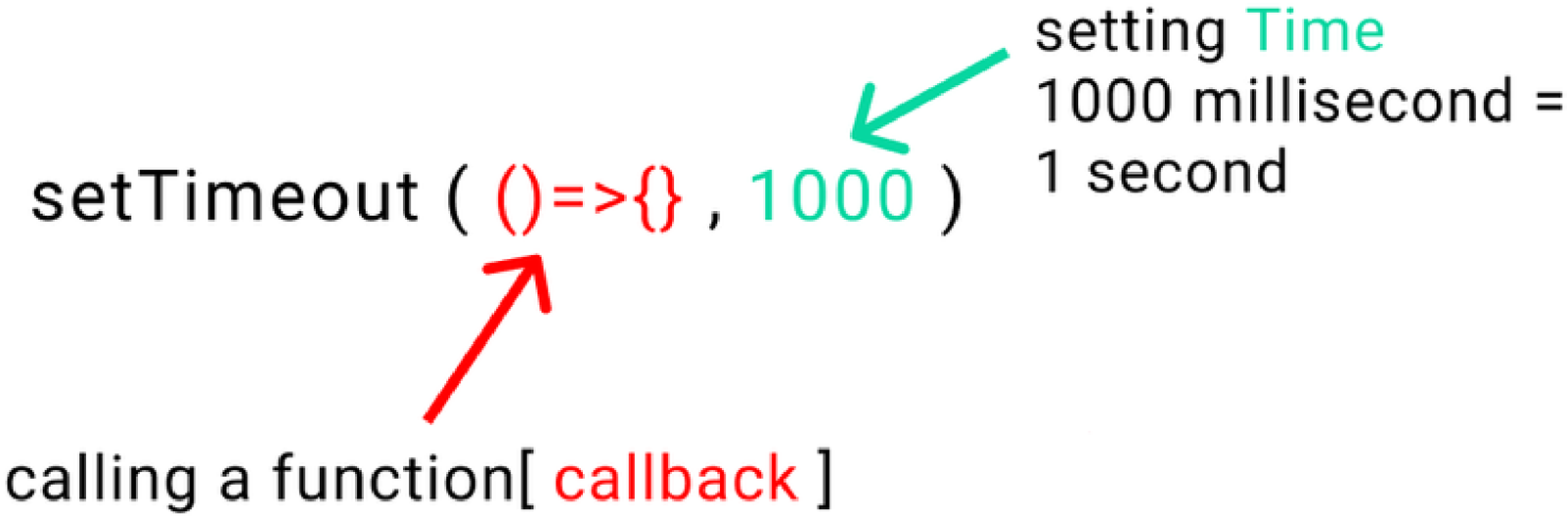
What are Callbacks in JavaScript?

When you nest a function inside another function as an argument, that's called a callback.



To establish the timing, the function setTimeout() is excellent as it is also uses a callback by taking a function as an argument.

setTimeout() Syntax



TIME FOR THE PROJECT

BATTLE ON TITAN

01

ORDER FROM THE CAPTAIN

02

LAUNCH THE ATTACK

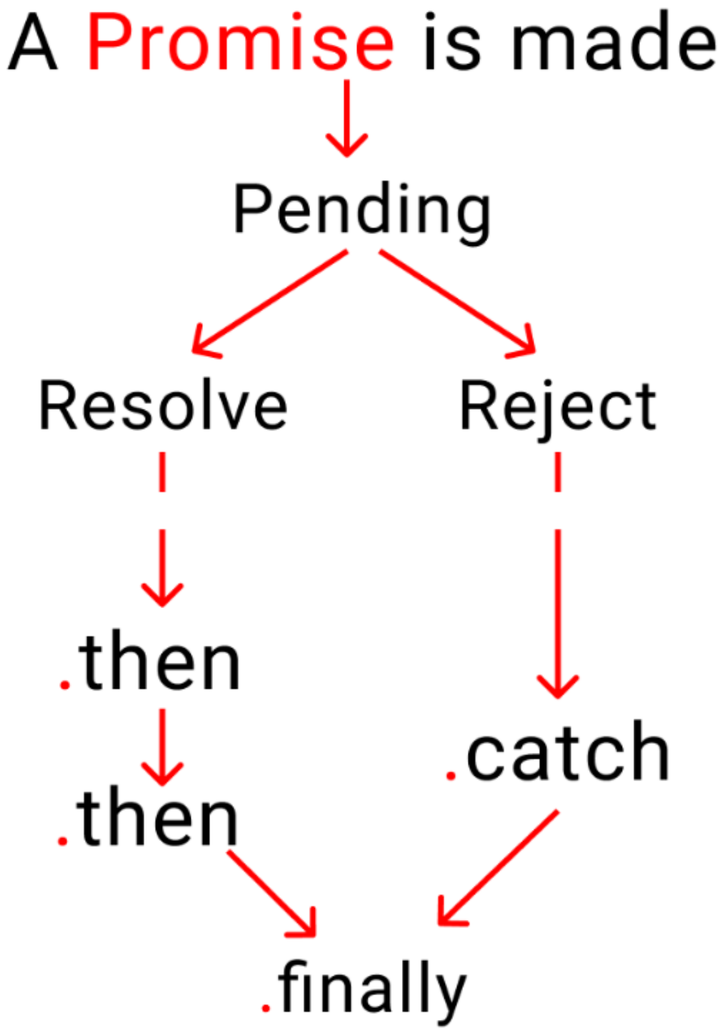
```
let order = (call_launch)=>{
setTimeout(()=>{
    console.log("Order received from Iron Man")
    call_launch();
    },3000)
}

let launch_battle= ()=>{ ···
}

order(launch_battle);
```

Once we receive an order from captain , we can act according to the strategy. So, we'll create two functions ->

- order
- launch_battle

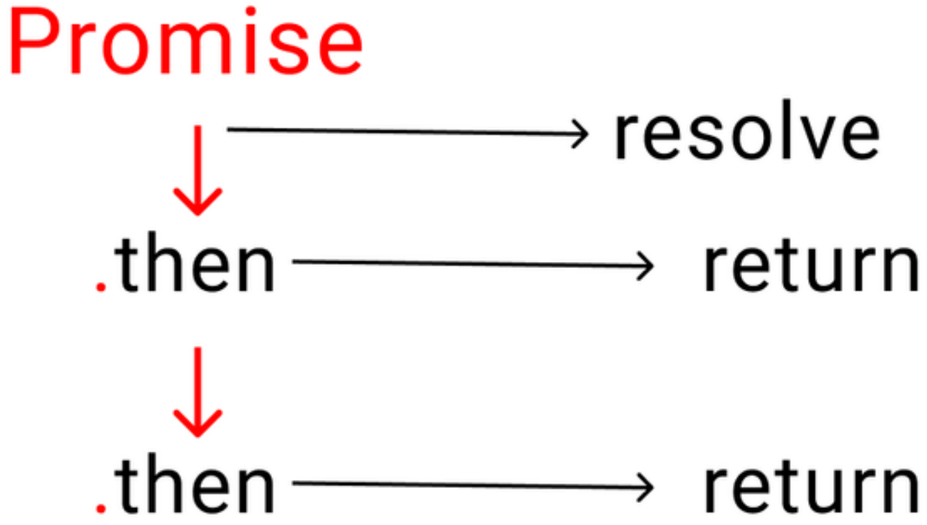
```
let launch_battle= ()=>{
setTimeout(()=>{
    console.log("Launch the operation....")
        setTimeout(()=>{
            console.log("Spiderman attacks Thanos");
                setTimeout(()=>{
                    console.log("Mr.Strange !! It's your turn....");
                        setTimeout(()=>{
                            console.log("Drax !! Get hold of Thanos's hand..")
                                setTimeout(()=>{
                                    console.log("Mantis...Let's give Thanos some nap!!")
                                },2000)
                        },2000)

                },2000)

        },1000)

},5000);

}
```

Callback Hell




```
let Thanos_is_alone = true;
```

```
let order = ( time , action )=>{

    return new Promise((resolve , reject)=>{
        if(Thanos_is_alone){

            setTimeout(()=>{
                resolve(action())

            },time)

        }
        else{
            reject(console.log("we lost"))
        }
    })
}
```




```
order(2000,()=>console.log("Order received from Iron Man"))

.then(()=>{
    return order(5000, ()=>console.log("Launch the operation...."))
})

.then(()=>{
    return order(1000, ()=>console.log("Spiderman attacks Thanos"))
})

.then(()=>{
    return order(2000, ()=>console.log("Mr.Strange !! It's your turn...."))
})

.then(()=>{
    return order(2000, ()=>console.log("Drax !! Get hold of Thanos's hand.."))
})

.then(()=>{
    return order(3000, ()=>console.log("Mantis...Let's give Thanos some nap!!"))
})
```

```
.catch(()=>{
    console.log("We lost")
})
```

Async, Await.

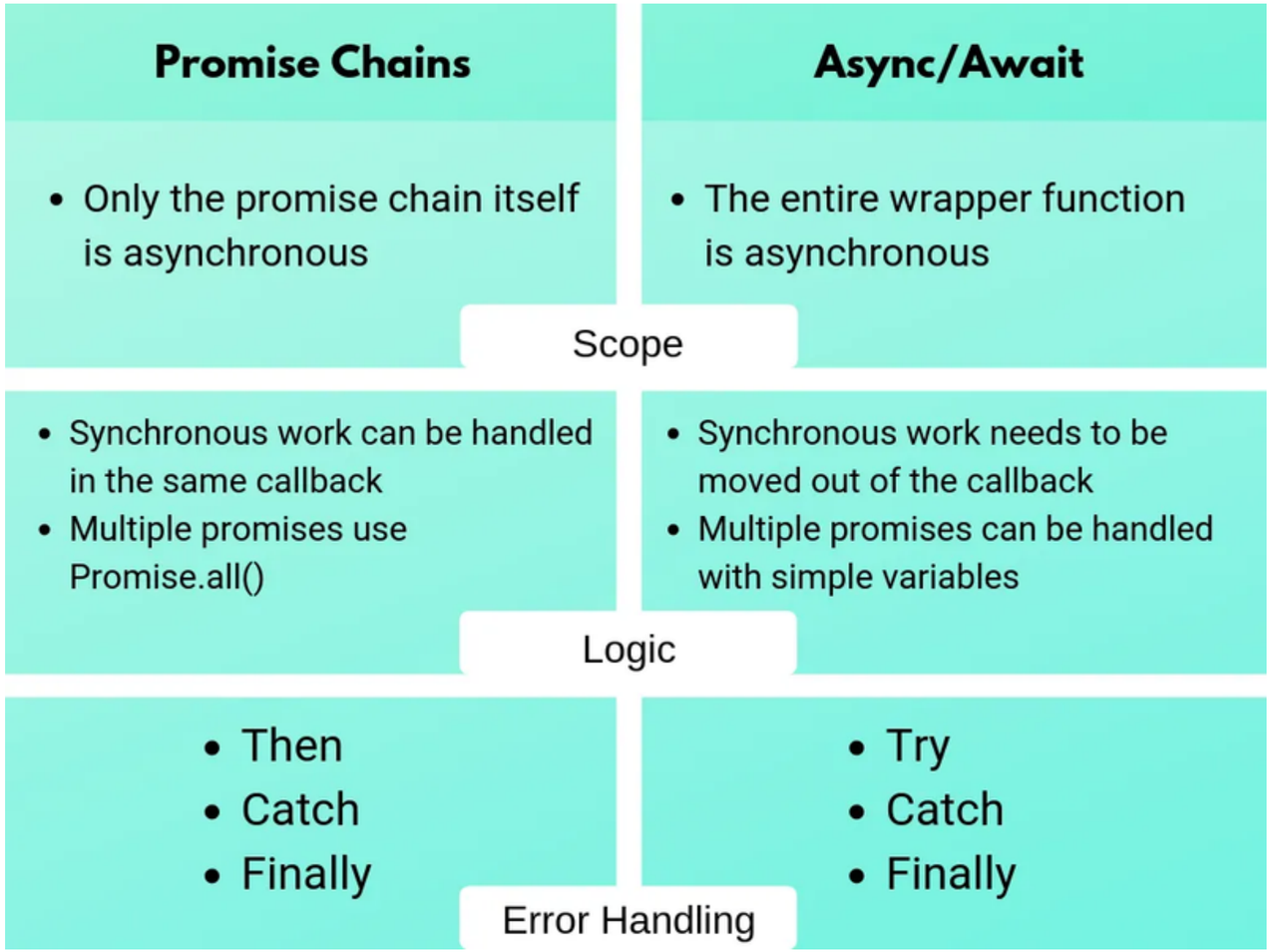
An async function returns a promise

Any code that wants to use this function will use the await keyword right before the function

```
let Thanos_is_alone = true;

function time(ms){
    return new Promise((resolve, reject)=>{
        if(Thanos_is_alone){

            setTimeout(resolve, ms)
        }
        else{
            reject(console.log("we lost"))
        }
    }
    )
}
```

```
async function attack(){
    try{
        await time(2000)
        console.log("Order received from Iron Man")
        await time(5000)
        console.log("Launch the operation....")
        await time(1000)
        console.log("Spiderman attacks Thanos")
        await time(2000)
        console.log("Mr.Strange !! It's your turn....")
        await time(2000)
        console.log("Drax !! Get hold of Thanos's hand..")
        await time(3000)
        console.log("Mantis...Let's give Thanos some nap!!")
        }
        catch(error){
            console.log("we lost")
        }
}
attack();
```

Promise chains are probably still a good option if you're trying to quickly and concisely grab the results from a promise,



Async/await is an excellent option if you find yourself writing long, complicated waterfalls of .then statements.