Promises

Key points to remember

What is a promise ?

“Imagine you are a kid. Your dad promises you that he will buy you a new toy next week” that is a promise.

A promise has 3 states

1. **Pending:** You don’t know if you will get the toy
2. **Fulfilled**: Dad is happy and he will get you a toy
3. **Rejected**: Your dad is not happy, he withholds the toy

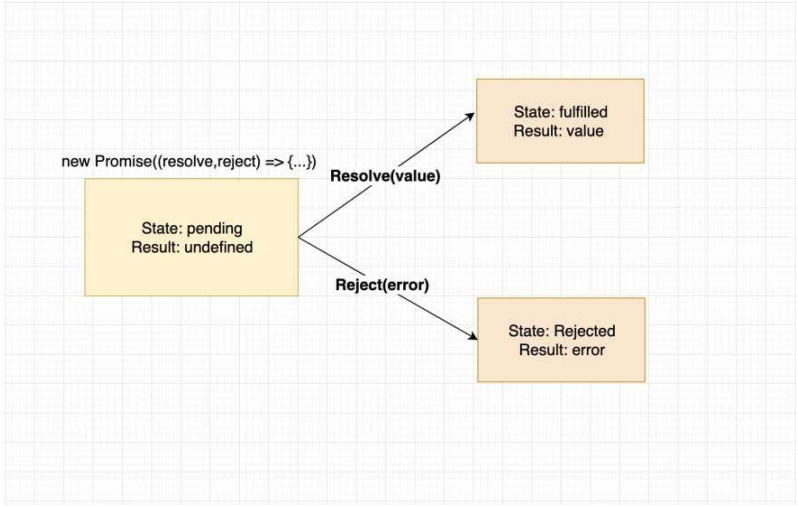
What is a promise in javascript ?

A promise is an object that may produce a single value some time in the future: either a resolved value, or a reason that it’s not resolved.

Here also a Promise can have one of three states

- fulfilled
- rejected
- pending

Syntax of promise and stages of promise




```
new Promise(function(resolve,reject){...})
```

myPromise.state	myPromise.result
"pending"	undefined
"fulfilled"	a result value
"rejected"	an error object

Creation of Promises

Promises are generally used for easier handling of asynchronous operations.

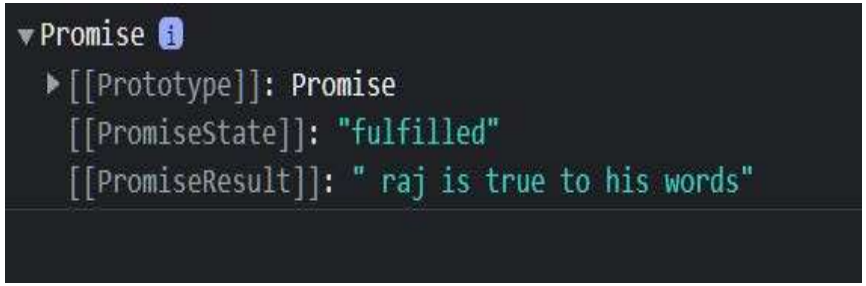
If the asynchronous operations are successful then the expected result is returned by calling the resolve function by the creator of the promise.

Similarly if there was some unexpected error the reasons is passed on by calling the reject function.

Note- every promise has a PromiseState and PromiseResult.

When we create a promise it initializes in its pending state

```
var raj;
raj = true;
promise1 = new Promise(function (resolve, reject) {
  if (raj) {
    resolve(" raj is true to his words");
  } else {
    reject("raj is a not true to his words");
  }
});
console.log(promise1);
```



Working with promises

We need to find what is the result of the promised once the executor runs.
In order to find, we have then and catch.

- The **then** function is attached to promise which executes when the promise is resolved. It sends the value sent through **resolve** method as an argument.
- The **catch** function is attached to promise which executes when the promise is rejected. It takes the error sent through **reject** method as an argument.

```
let passexam = true;

let parentPromise = new Promise(function (resolve, reject) {
  setTimeout(function () {
    if (passexam) {
      resolve("bike");
    } else {
      reject("nothing");
    }
  }, 5000);
});

console.log("pending");

parentPromise.then(function (resolve) {
  console.log("resolve:", resolve);

  console.log("promise fulfilled");
});

parentPromise.catch(function (err) {
  console.log("err:", err);
});
```

output

```
pending
resolve: bike
promise fulfilled
>
```

Promise using async and await

The word **“async”** before a function means one simple thing: a function always returns a promise. Other values are wrapped in a resolved promise automatically.

The keyword **await** makes JavaScript wait until that promise settles and returns its result.

```
async function f() {
  let promise = new Promise((resolve, reject) => {
    setTimeout(() => resolve("done!"), 1000);
  });

  let result = await promise; // wait till the promise resolves (*)

  alert(result); // "done!"
}

f();
```

Note- The function execution “pauses” at the line (*) and resumes when the promise settles, with result becoming its result. So the code above shows “done!” in one second.