Handling the Fulfilled or Rejected States

'then', 'onFulfilled' and 'onRejected' methods

Promises can be passed around as values, as function arguments, and as return values. Values and reasons for rejection can be handled by handlers inside the `then` method of the promise.

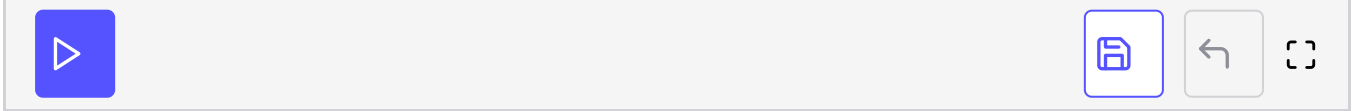
```
promise.then( onFulfilled, onRejected );
```

- `then` may be called more than once to register multiple callbacks. The callbacks have a fixed order of execution.
- `then` is chainable, it returns a promise.
- if any of the arguments are not functions, they have to be ignored.
- `onFulfilled` is called once with the argument of the fulfilled value if it exists.
- `onRejected` is called once with the argument of the reason why the promise was rejected.

Examples:

```
let promisePaymentAmount = Promise.resolve( 50 );

promisePaymentAmount
    .then( amount => {
        amount *= 1.25;
        console.log('amount: '+amount)
        console.log('amount * 1.25: ', amount );
        return amount;
    } ).then( amount => {
        console.log('amount: ', amount );
        return amount;
    } );
```

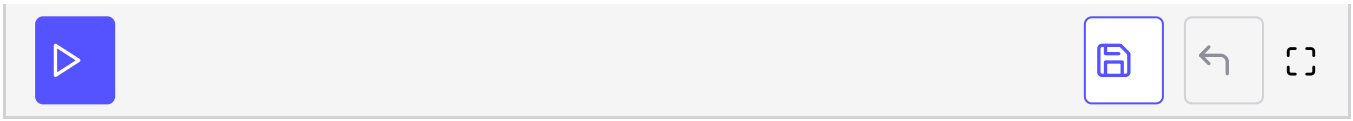


Notice the return value of the callback function of the first `then` call. This

Notice the return value of the callback function of the first `then` call. This value is passed as `amount` in the second `then` clause.

```
let promiseIntro = new Promise( function( resolve, reject ) {
    setTimeout( () => reject( 'Error demo' ), 2000 );
});

promiseIntro.then( null, error => console.log( error ) );
```



Instead of

```
promise.then( null, errorHandler );
```

You can also write:

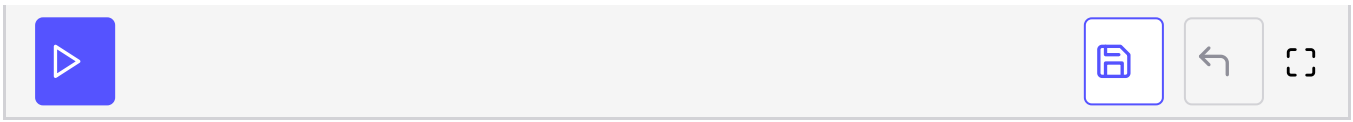
```
promise.catch( errorHandler );
```

to make error handling more semantic.

The best practice is to always use catch for handling errors, and place it at the end of the promise handler chain. Reason: catch also catches errors thrown inside the resolved handlers. Example:

```
var p = Promise.resolve( 5 );

p.then( ( value ) => console.log( 'Value:', value ) )
 .then( () => { throw new Error('Error in second handler' ) } )
 .catch( ( error ) => console.log(error.toString() ) );
```



As `p` is resolved, the first handler logs its value, and the second handler throws an error. The error is caught by the `catch` method, displaying the error message.

It is also possible to handle multiple promises at the same time. We will study this in detail in the next lesson.