Promises

1. Code Example of Callbacks and Promises :

```
const cart = ["shoes", "pants", "kurta"];

createOrder(cart, function (orderId) {
  proceedToPayment(orderId);
});

const promise = createOrder(cart);

// data: undefined ?
```

(Cart array)

(Controller function)

(Promise object)

Callbacks

Promises

When this `createOrder()` is called a Promise is created.

Promise is Nothing But an empty object with value undefined.

And once we get the Data from `createOrder` undefined is replaced with that Data.

★ But How do we call `proceedtoPayment()` API ?

```
const cart = ["shoes", "pants", "kurta"];

createOrder(cart, function (orderId) {
  proceedToPayment(orderId);
});

const promise = createOrder(cart);

promise.then(function (orderId) {
  proceedToPayment(orderId);
});
```

Difference ?

We get `.then` in Promise Object and we attach a Callback Function.

```
const cart = ["shoes", "pants", "kurta"];

createOrder(cart, function (orderId) {
  proceedToPayment(orderId);
});

const promise = createOrder(cart);

promise.then(function (orderId) {
  proceedToPayment(orderId);
});
```

In this we are passing a Callback Function.

In this we are attaching a callback Function to a Promise Object.

It may call the callback function twice or thrice or Never.

It will definetly call the callback function (Guarantee Given by JS) and it will call just Once.

★ What is Promise Object ?

```
15  const GITHUB_API = "https://api.github.com/users/akshaymarch7";
16
17  const user = fetch(GITHUB_API);
```

(Promise Object)

We will get this inside user, when we reach (Line : 17).

```
Scope   Watch
Script
  GITHUB_API: "https://api.github.com
  user: Promise
    [[Prototype]]: Promise
    [[PromiseState]]: "pending"
    [[PromiseResult]]: undefined
Global                                Window
```

Immutable → Data Cant be Edited.

★ Promise States :

1. Pending
2. Fullfilled
3. Rejected.

```
13  const GITHUB_API = "https://api.github.com/users
14
15  const user = fetch(GITHUB_API);
16
17  console.log(user);
```

We will get this when we reach (line : 17).

```
▶ Promise {<pending>}                    index.js:17
  ▶ [[Prototype]]: Promise
    [[PromiseState]]: "fulfilled"
    [[PromiseResult]]: Response
```

★ Why are we getting pending at first and then when we expand, we get fullfilled ?

Bcoz JS Line by Line chalta hai, aur Jb (Line : 15) execute hua to (state - pending) that to console mai pending aaya, fir Jb state fullfilled hua to state bhi update hogya.

★ Promise Chaining :

```
const cart = ["shoes", "pants", "kurta"];

createOrder(cart, function (orderId) {
  proceedToPayment(orderId, function (paymentInfo) {
    showOrderSummary(paymentInfo, function (?) {
      updateWalletBalance();
    });
  });
});

// Promises
createOrder(cart)
  .then(function (orderId) {
    return proceedToPayment(orderId);
  })
  .then(function (paymentInfo) {
    return showOrderSummary(paymentInfo);
  })
  .then(function (paymentInfo) {
    return updateWalletBalance(paymentInfo);
  });
```

Code Using Callbacks.

Same above Code using Promises.

Also known as Promise Chaining.

Helps us to prevent Callback Hell.

Do not forget to return the Callback Function. ★