1 Callbacks

1.1 Definition

Callback is a function send as parameter to another function. The callback function is executed at the appropriate time.

1.2 Example

1.3 Callbacks in JS

Callbacks are widely used in Javascript, from generic algorithms to async functionality, eg: array sort , nodejs read file .

1.4 Why we use Callbacks in JS ?

Javascript is an event driven language, that means JS will continue executing while listening for events .

1.5 Callbacks in other languages

Java

1.6 Exercises

Implement the following functionalities:

- forEach(array, callback) returns undefined.

  Callback signature (value,index,array) . If the callback returns false, forEach should stop.

- filter(array, callback) returns a new filtered array.

  Callback signature (value,index,array). If the callback returns true, the element should be added to response.

- map(array, callback) returns a new array.

  Callback signature (value,index,array). The return from callback should be added to response.

- sort(array, callback) returns a new sorted array.

  Callback signature (value,index,array). If callback is not specified you will use the $\leq$ operator to compare elements.

2 Promises

2.1 Definition

Promise is an object representing the eventual completion or failure of an asynchronous operation. Essentially, a promise is an object to which you attach callbacks, instead of passing callbacks into a function.

2.2 Example

2.3 Methods

- Promise.all method returns a single Promise that resolves when all of the promises passed as an iterable have resolved or when the iterable contains no promises. It rejects with the reason of the first promise that rejects.
- Promise.race method returns a promise that resolves or rejects as soon as one of the promises in an iterable resolves or rejects, with the value or reason from that promise
- Promise.resolve and Promise.reject

2.4 Exercises

Using es6 classes provide your own implementation of Promise called MyPromise. Your promise should support:

- then(cb), which can be chain.
- catch(cb), wich can be chain as well.
- static method all([Promises]), returns MyPromise with a list of values

3 async / await

3.1 Definition

The async function declaration defines an asynchronous function, which returns an AsyncFunction object. An asynchronous function is a function which operates asynchronously via the event loop, using an implicit Promise to return its result. But the syntax and structure of your code using async functions is much more like using standard synchronous functions.

3.2 Example

3.3 Exercises

Create a wrapper over nodejs readline that returns a list of lines. I want to be able to use it as follows: (await asyncReadLines('a.txt')).forEach(line=¿ console.log(line));

4 try / catch

4.1 Syntax

```
try
{
...
}catch(e)
{
...
}
```

4.2 Example

4.3 Exercises

Extend JS Error class and customize error message. Use es6 class inheritance.
Create a class Validator, that accepts 1 parameter.

- class Validator has the following methods: isNumber, isGreaterThan(value), isLessThan(value), isString, hasLengthGreaterThan(value), hasLengthLessThan(value) all those methods are chainnable.
- If the value send to constructor is null throw NotNullError, is undefined throw NotUndefinedError, is function throw Error with message functions are not supported.
- Create Error classes for all the methods.
- Throw appropriate error if the value is not in range
- eg: new Validator(5).isNumber().isGreaterThan(0).isLessThan(3)

5 Fetch

5.1 Example

5.2 Exercise

Implement your own fetch as a wrapper over node http module