Javascript Promises

Alok Guha

JS Functions

- Functions are first class citizens in JavaScript
- Functions can take functions as arguments and call them when they are done.

First-class functions

```
function sqr(x) {
 return x * x; }

// a function may be assigned to a variable
var func = sqr;

// a function may have properties or be a property
func.doubleMe = function (x) {
return x + x;
};

// a function may be used as a parameter
[1, 2, 3].map(func.doubleMe); OR
[1, 2, 3].map(func);
```

First-class functions

```
function calculate(config){
  //choose a function based on ref data
  return config.flag ? sqr : cube;
}


function sqr(x){ return x*x; }
function cube(x){ return sqr(x)*x; }
```

Callbacks

- Callback functions are derived from a programming paradigm called functional programming.(At a simple and fundamental level, functional programming is the use of functions as arguments).

```
fileSys.readFile("file1.txt", function (file1Content) {
return file1Content;
});
```

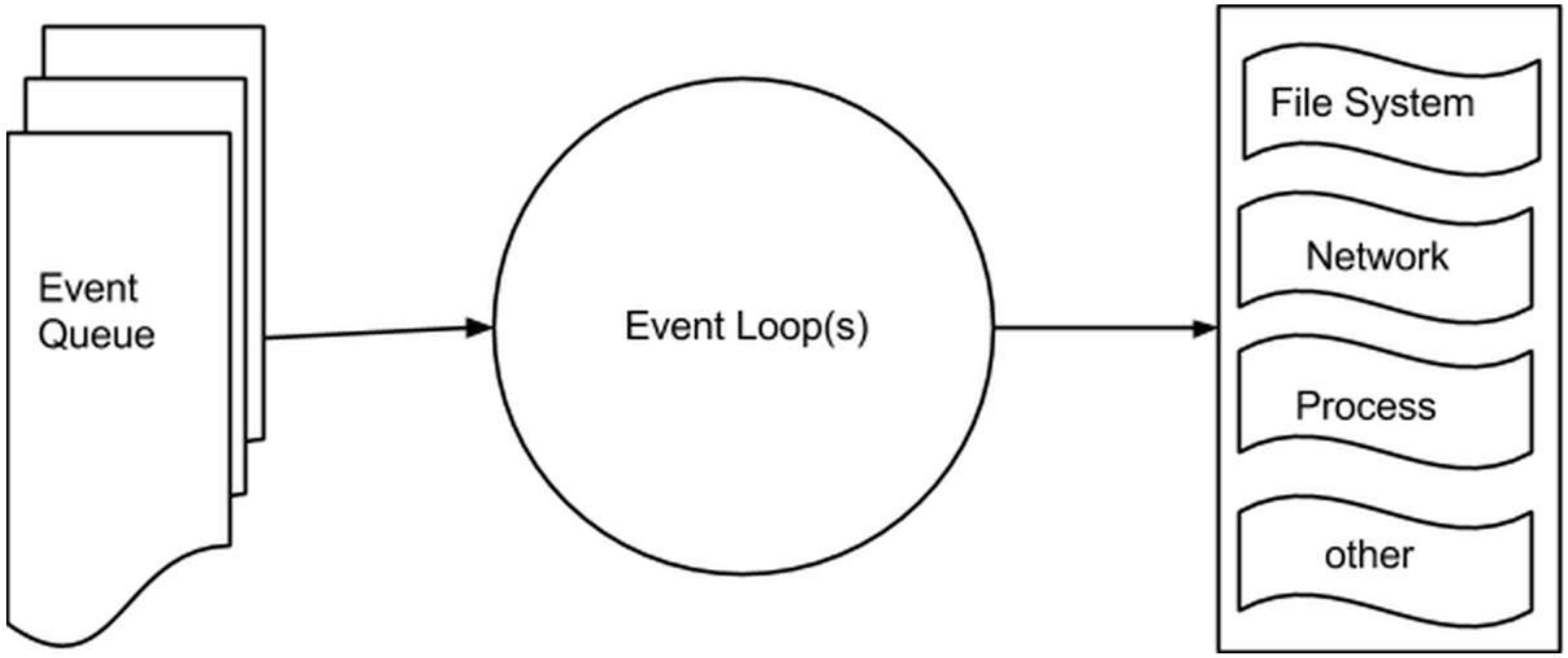
Is Javascript really Async ?

- All of the JavaScript engines including V8 Engine are single-threaded.
- But all I/O is evented and asynchronous It is possible via the EventLoop.

Event Loops

- An entity that handles and processes external events and converts them into callback invocations
- Every asynchronous operation adds itself to the EventLoop Event Loops

How it works ?



Promise

"A declaration or assurance that one will do something or that a particular thing will happen"

- A Promise can be
  - Pending
  - Fulfilled
  - Rejected

JS Promise

- A Promise is an object that represents a one-time event, typically the outcome of an async task like an AJAX call.
- At first, a Promise is in a pending state. Eventually, it's either resolved or rejected.
- Once a Promise is resolved or rejected, it'll remain in that state forever, and its callbacks will never fire again

How to promise ?

```
var promise = new Promise(function(resolve, reject){

  // do a thing, possibly async, then…

  if (/* everything turned out fine */) {
    resolve("Stuff worked!");
  }
  else {
    reject(Error("It broke"));
  }
});
```

Lets promise..

```
var num = 4;
var simpleSqrPromise = new Promise(function(resolve, reject){
      resolve(num*num);
      reject();
});

//execute it
simpleSqrPromise.then(function(data){
    console.log("worked "+data);
}).catch(function(err){ console.log(err);});
```

Simple Functional Transform

```
var user = getUser();
var userName = user.name;

// becomes

var userNamePromise = getUser().then(function (user) { return
user.name; });
```

Simple coding path

- Can we code a Promise which parses JSON ?
  And ensure its value for next operation..

Do you know them ?

Consider this ;/

```
step1(function (value1) {
    step2(value1, function (value2) {
        step3(value2, function (value3) {
            step4(value3, function (value4) {
                step5(value4, function (value5) {
                    //do something with value5
                })
            })
        })
    })
});
```

Callbacks are hell :D

- They are literally the simplest thing that could work.
- But as a replacement for synchronous control flow, they s**k.
- There’s no consistency in callback APIs.
- There’s no guarantees.
- We lose the flow of our code writing callbacks that tie together other callbacks.
- We lose the stack-unwinding semantics of exceptions, forcing us to handle errors explicitly at every step.

Now see this..

```
someCall(promisedStep1)
.then(promisedStep2)
.then(promisedStep3)
.then(promisedStep4)
.then(function(value4){
        // do something
    })
.done();
```

Promise Guarantees

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

- Only one of onFulfilled or onRejected will be called.
- onFulfilled will be called with a single fulfillment value (⇔ return value).
- onRejected will be called with a single rejection reason (⇔ thrown exception).
- If the promise is already settled, the handlers will still be called once you attach them.
- The handlers will always be called asynchronously.

JS Promise frameworks

- Jquery Promise
- When
- Q
- A+
- JSPromise

Etc..

References

- Google.com ☺
- ..
- ..
- ..
- ..
- ..

Thanks

```
var speakerMap = {
    name: 'Alok Guha',
    email: 'Alok.Guha@synerzip.com',
    skype: 'aloksguha',
    twitter: 'aalokism'

}
```