Promsie

RWTH Aachen University

1st Papop Lekhapanyaporn 2nd Tobias Piatek 3rd Tobias Waerder

Abstract

In Javascript there are two ways to handle asynchronous code. The first way is to use callbacks. The second way is to use promises. This paper will explain the concept of promises and how to use them. The paper will also compare the two ways to handle asynchronous code.

I. EVENTLOOP

In Javascript, a single threaded programming language, It is possible to handle thousands of operation symaltaniously by using eventloop. Eventloop is created to fix the single threaded problem that program can only run one function at a time. In practice, the program will be unusable without event loop.

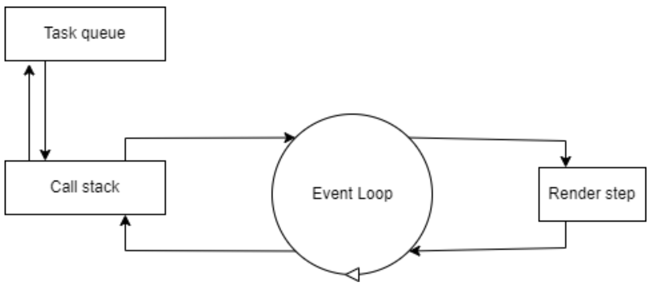

Figure 1. eventloop

in figure 1 the eventloop will constantly check the task queue. If there is a task in the task queue, the first task in the queue will be removed and put into the call stack. The eventloop will then take a detour to the call stack and execute the task that was in the callstack. After a certain amount of time has passed the event loop will then go through the render step which contain styles generation, layout positioning , and pixel data generation that will be displayed on the browser.

II. CALLBACKS

III. PROMISES

Promise is a object use to represent a task that will eventually be either be fullfilled or rejected. if the Task is fullfilled, then the promise will be resolved. If the task is rejected, then the promise will be rejected.

```
const promise = new Promise((resolve,reject)=>{
  setTimeout(()=>{
    resolve('done');
  },1000);
});
```

A. *Callback Handling*

Promise will eventually either be fullfilled or rejected.

1) then(): then can be use to handle the fullfilled promise.

```
promise.then()
```

2) catch(): catch can be use to handle the rejected promise.

```
promise.catch()
```

B. *Chaining*

Promise can be chained together to handle multiple asynchronous funtion. often use when fetching data from HTTP server and convert to json.

```
promise.then().then().catch()
```

C. *Catching Errors*

D. *fetch API*

fetch API is a promise based API that can be use to make HTTP request.

```
const res = fetch('...');
```

IV. Asynchronous methods

A. *async*

B. *await*

V. Conclusion

References

[1] Author,Title *Bookname*, Band and publication number,Page,Year.
[2] Author, TItle *Conference Name*, Page, Year
[3] Author, *Title*, Publisher, Year