PROMISES

A promise is a proxy for a value which is not necessarily known when the promise is created. Promises lets the asynchronous methods returns value like synchronous methods but instead of final values, the asynchronous methods returns a promise for a value in some time in future.

In simple terms “A promise is a word taken for some action, the other party who gave the promise can fulfil or deny it”. In case of fulfilling the promise gets resolved or it gets rejected.

- Any promise that performs async operations should call any one of the two methods **resolve** or **reject**.
- The code which uses a promise should call then function on that promise. It takes 2 functions as parameters. The first function executes if the promise is resolved and the second function executes if the promise is rejected.
- The promise will be in **pending** state if we try to access the value from promise before it is resolved or rejected.

Creating a Promise:

We can create a promise in Node JS program using a new constructor.

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

})
```

Code Example:

We will be using the Github REST api to fetch details about users, repositories.

https://api.github.com/users/Gaurav-Walia

If you make a HTTP GET request with this URL, you will receive a JSON with all the information about my github account like repos, followers etc.

For making HTTP GET request, we are installing a small package **request**

npm install request --save

- options object is used to set URL and Headers for request
- request.get makes a GET request to the Github API

- body consists of the JSON response from the server

We are calling resolve method to pass data back to the handler which implements **then** on the promise.

promises.js

```
var request = require('request');
var userDetails;

function initialize() {
    var options = {
        url: 'https://api.github.com/users/Gaurav-Walia',
        headers: {
            'User-Agent': 'request'
        }
    }

    return new Promise(function(resolve, reject) {
        request.get(options, function(err, resp, body) {
            if(err) {
                reject(err);
            } else {
                resolve(JSON.parse(body));
            }
        })
    })
}

function main() {
    var initializePromise = initialize();
    initializePromise.then(function(result) {
        userDetails = result;
        console.log('Initialized user details');
        console.log(userDetails);
    }, function(err) {
        console.log(err);
    })
}
main();
```

Result:


```
id: 2232191,
node_id: 'MDQ6VXNlcjIyMzIxOTE=',
avatar_url: 'https://avatars3.githubusercontent.com/u/2232191?v=4',
gravatar_id: '',
url: 'https://api.github.com/users/Gaurav-Walia',
html_url: 'https://github.com/Gaurav-Walia',
followers_url: 'https://api.github.com/users/Gaurav-Walia/followers',
following_url:
 'https://api.github.com/users/Gaurav-Walia/following{/other_user}',
gists_url: 'https://api.github.com/users/Gaurav-Walia/gists{/gist_id}',
starred_url:
 'https://api.github.com/users/Gaurav-Walia/starred{/owner}{/repo}',
subscriptions_url: 'https://api.github.com/users/Gaurav-Walia/subscriptions',
organizations_url: 'https://api.github.com/users/Gaurav-Walia/orgs',
repos_url: 'https://api.github.com/users/Gaurav-Walia/repos',
events_url: 'https://api.github.com/users/Gaurav-Walia/events{/privacy}',
received_events_url: 'https://api.github.com/users/Gaurav-Walia/received_events',
type: 'User',
site_admin: false,
name: 'Gaurav Walia',
company: 'Tata Consultancy Services',
blog: '',
location: 'Noida',
email: null,
hireable: true,
bio: null,
public_repos: 17,
public_gists: 0,
followers: 1,
following: 0,
created_at: '2012-08-28T05:32:46Z',
updated_at: '2019-05-20T17:37:55Z' }
```


Suppose you want to perform an operation after a promise is fulfilled use another **then** method to - transform the data you obtained from the promise.

```
function main() {
    var initializePromise = initialize();
    initializePromise.then(function(result) {
        userDetails = result;
        console.log('Initialized user details');
        return userDetails;
    }, function(err) {
        console.log(err);
    }).then(function(result) {
        console.log('Nested then block');
        console.log(result.public_gists + result.public_repos);
    })
}
```

PROMISE.ALL

- **Promise.all** function which takes a list of promises in the given order and returns another promise which we can use a **then** method to conclude the logic.
- We should use **Promise.all** when we don't care about the order of execution.
- **Promise.all** fails if any one of the Promise got rejected.

```
var message = '';

promise_1 = new Promise((resolve, reject) => {
    setTimeout(() => {
        message += "promise ";
        resolve(message);
    }, 2000);
})

promise_2 = new Promise((resolve, reject) => {
    setTimeout(() => {
        message += "all ";
        resolve(message);
    }, 2000);
})

promise_3 = new Promise((resolve, reject) => {
    setTimeout(() => {
        message += "description";
        resolve(message);
    }, 2000);
})

var printResult = (results) => {console.log("Results = ", results, "message=", message)};

function main() {
    Promise.all([promise_1, promise_2, promise_3]).then(printResult);
    Promise.all([promise_2, promise_1, promise_3]).then(printResult);
    Promise.all([promise_3, promise_2, promise_1]).then(printResult);
}

main();
```

Result:

```
PS C:\Users\dell\Documents\Tutorials\demo examples> node .\promise-all.js
Results =  [ 'promise ', 'promise all ', 'promise all description' ] message= promise all description
Results =  [ 'promise all ', 'promise ', 'promise all description' ] message= promise all description
Results =  [ 'promise all description', 'promise all ', 'promise ' ] message= promise all description
```