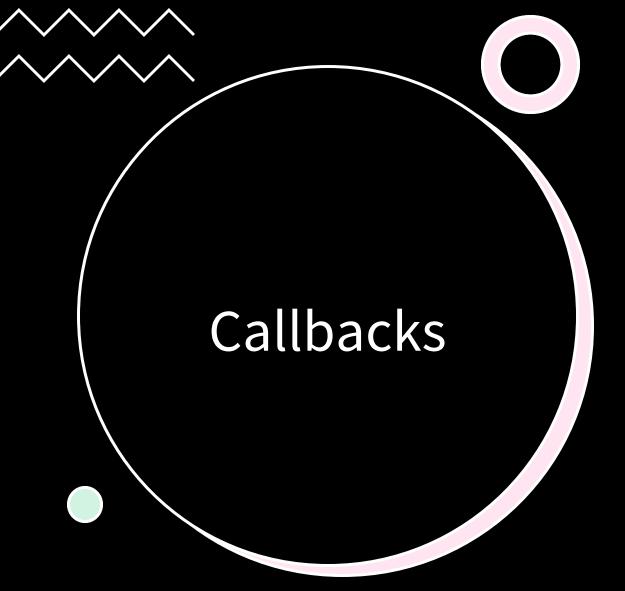


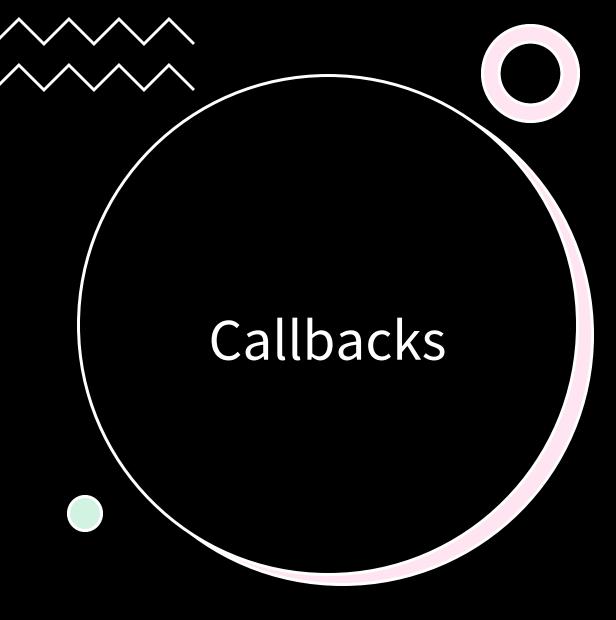
# A S Y N C H R O N O U S J A V A S C R I P T



### index.js

```
console.log('Before');
getUser(1, (user) => {
    console.log('User', user);
});
console.log('After');

function getUser(id, callback){
    setTimeout(() => {
        console.log('Reading a user from a DB');
        callback({id: id, gitHubUsername: 'marizza'});
    }, 2000);
}
```



#### !!! Exercise:

Convert this function 'getRepositories' to asynchronous function using callback

### index.js

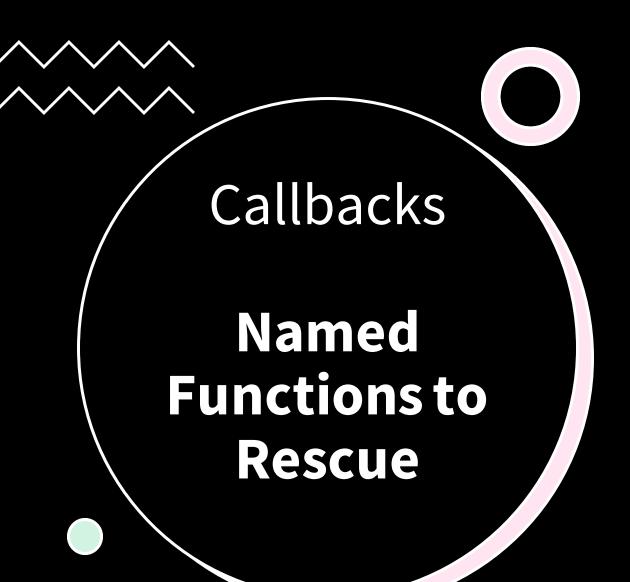
```
console.log('Before');
getUser(1, (user) => {
    console.log('User', user);
    // Get the repositories
});
console.log('After');
function getUser(id, callback){
    setTimeout(() => {
        console.log('Reading a user from a DB');
        callback({id: id, gitHubUsername: 'marizza'});
    }, 2000);
function getRepositories(username){
    return ['repo1', 'repo2', 'repo3'];
```

### **Solution**

index.js

```
Callbacks
```

```
console.log('Before');
getUser(1, (user) => {
    console.log('User', user);
    getRepositories(user.gitHubUsername, (repos) => {
        console.log('Repose', repos);
    });
});
console.log('After');
function getUser(id, callback){
    setTimeout(() => {
        console.log('Reading a user from a DB');
        callback({id: id, gitHubUsername: 'marizza'});
    }, 2000);
function getRepositories(username, callback){
    setTimeout(() => {
        console.log('Calling GitHub API ...');
        callback(['repo1', 'repo2', 'repo3']);
    }, 2000);
```



```
index.js
console.log('Before');
getUser(1, getRepositoriesByUser);
console.log('After');
function getRepositoriesByUser(user) {
    getRepositories(user.gitHubUsername, getCommitsByRepo);
function getRepositories(user){
    getRepositories(user.gitHubUsername, getCommits);
function getCommitsByRepo(repo){
    getCommits(repo, displayCommits);
function displayCommits(commits){
    console.log(commits);
function getUser(id, callback){
    setTimeout(() => {
       console.log('Reading a user from a DB');
       callback({id: id, gitHubUsername: 'marizza'});
    }, 2000);
function getRepositories(username, callback){
    setTimeout(() => {
        console.log('Calling GitHub API ...');
       callback(['repo1', 'repo2', 'repo3']);
    }, 2000);
function getCommits(repo, callback) {
    setTimeout(() => {
        console.log('calling GitHub API...');
       callback(['commit']);
    },2000);
```



### **Promise.js**

```
const p= new Promise ((resolve, reject) => {
    //Kick off some async work
    //...

setTimeout(() => {
        resolve(1); // pending => resolved, fulfilled
        reject(new Error('message')); // pending => rejected
    }, 2000);

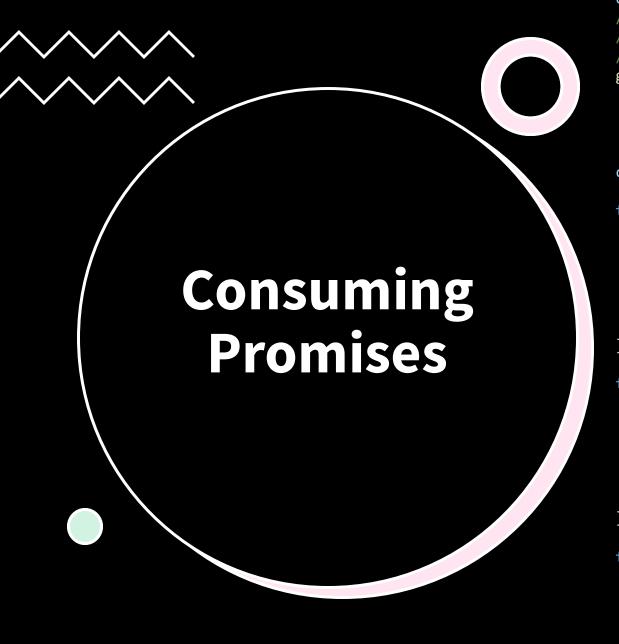
//reject(new Error('message'));

p
    .then(result => console.log('Result', result))
    .catch(err => console.log('Errore', err.message));
```

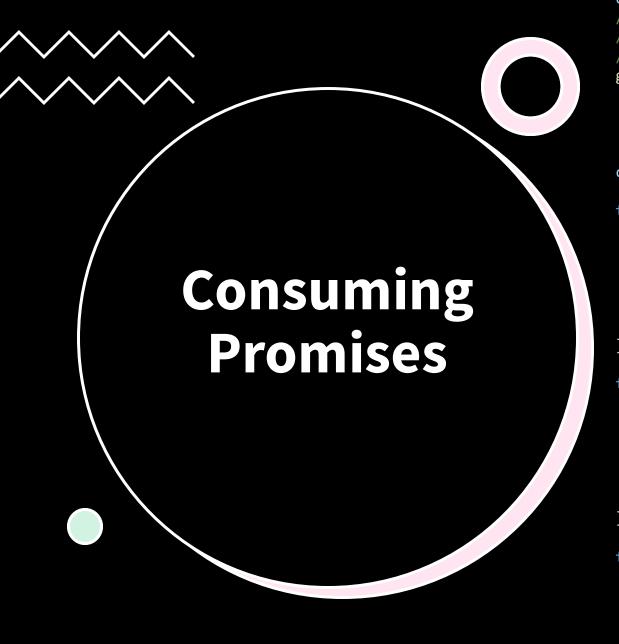
### index.js console.log('Before'); getUser(1, (user) => { getRepositories(user.gitHubUsername, (repos) => { getCommits(repos[0], (commits) => { console.log(commits); }); }); }); console.log('After'); function getUser(id){ return new Promise ((resolve, reject) => { setTimeout(() => { console.log('Reading a user from a DB'); resolve({id: id, gitHubUsername: 'marizza'}); }, 2000); }); function getRepositories(username){ return new Promise((resolve, reject) => { setTimeout(() => { console.log('Calling GitHub API ...'); resolve(['repo1', 'repo2', 'repo3']); }, 2000); }); function getCommits(repo) { return new Promise(() => { setTimeout((resolve, reject) => { console.log('calling GitHub API...'); resolve(['commit']); },2000);

});

# Replacing Callbacks with Promises



```
console.log('Before');
                                                            index.js
// getUser(1, (user) => {
// });
getUser(1)
    .then(user => getRepositories(user.gitHubUsername))
    .then(repos => getCommits(repos[0]))
    .then(commits => console.log('Commits', commits));
console.log('After');
function getUser(id){
    return new Promise ((resolve, reject) => {
        setTimeout(() => {
            console.log('Reading a user from a DB');
            resolve({id: id, gitHubUsername: 'marizza'});
        }, 2000);
    });
function getRepositories(username){
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            console.log('Calling GitHub API ...');
            resolve(['repo1', 'repo2', 'repo3']);
        }, 2000);
   });
function getCommits(repo) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            console.log('calling GitHub API...');
            resolve(['commit']);
        },2000);
    });
```



```
console.log('Before');
                                                            index.js
// getUser(1, (user) => {
// });
getUser(1)
    .then(user => getRepositories(user.gitHubUsername))
    .then(repos => getCommits(repos[0]))
    .then(commits => console.log('Commits', commits));
console.log('After');
function getUser(id){
    return new Promise ((resolve, reject) => {
        setTimeout(() => {
            console.log('Reading a user from a DB');
            resolve({id: id, gitHubUsername: 'marizza'});
        }, 2000);
    });
function getRepositories(username){
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            console.log('Calling GitHub API ...');
            resolve(['repo1', 'repo2', 'repo3']);
        }, 2000);
   });
function getCommits(repo) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            console.log('calling GitHub API...');
            resolve(['commit']);
        },2000);
    });
```

```
Creating
Settled
Promises
```

```
const p = Promise.resolve({ id: 1});
p.then(result => console.log(result));
```

promise-api.js

```
const p = Promise.reject('reason for rejection...');
p.catch(err => console.log(err));
```

## Running Promises in Parallel

```
const p1 = new Promise((resolve) => {
   setTimeout(() => {
        console.log('Async operation 1 ...');
        resolve(1);
    }, 2000);
});
const p2 = new Promise((resolve) => {
    setTimeout(() => {
        console.log('Async operation 2 ...');
        resolve(2);
   }, 2000);
});
Promise.all([p1, p2])
    .then(result => console.log(result));
```

### promise-api.js

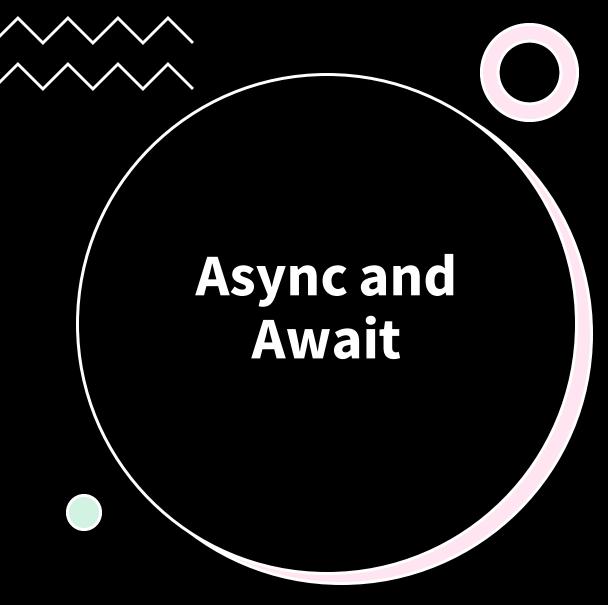
## Running Promises in Parallel

```
const p1 = new Promise((resolve, reject) => {
    setTimeout(() => {
        console.log('Async operation 1 ...');
        reject(new Error('becouse comething failed...'))
    }, 2000);
});
const p2 = new Promise((resolve) => {
    setTimeout(() => {
        console.log('Async operation 2 ...');
        resolve(2);
    }, 2000);
});
Promise.all([p1, p2])
    .then(result => console.log(result))
    .catch(err => console.log('Errore', err.message));
```

### promise-api.js

## Running Promises in Parallel

```
const p1 = new Promise((resolve) => {
    setTimeout(() => {
        console.log('Async operation 1 ...');
        resolve(1);
   }, 2000);
});
const p2 = new Promise((resolve) => {
    setTimeout(() => {
        console.log('Async operation 2 ...');
        resolve(2);
    }, 2000);
});
Promise.race([p1, p2])
    .then(result => console.log(result))
```



```
index.js
console.log('Before');
//Promise-based approach
// getUser(1)
       .then(user => getRepositories(user.gitHubUsername))
       .then(repos => getCommits(repos[0]))
       .then(commits => console.log('Commits', commits))
       .catch(err => console.log('Error', err.message));
//Async And Await approach
async function displayCommits(){
    try{
        const user = await getUser(1);
        const repos = await getRepositories(user.gitHubUsername);
        const commits = await getCommits(repos[0]);
        console.log(commits);
    catch{
        console.log('Error', err.message);
displayCommits();
console.log('After');
function getUser(id){
function getRepositories(username){
function getCommits(repo) {
```