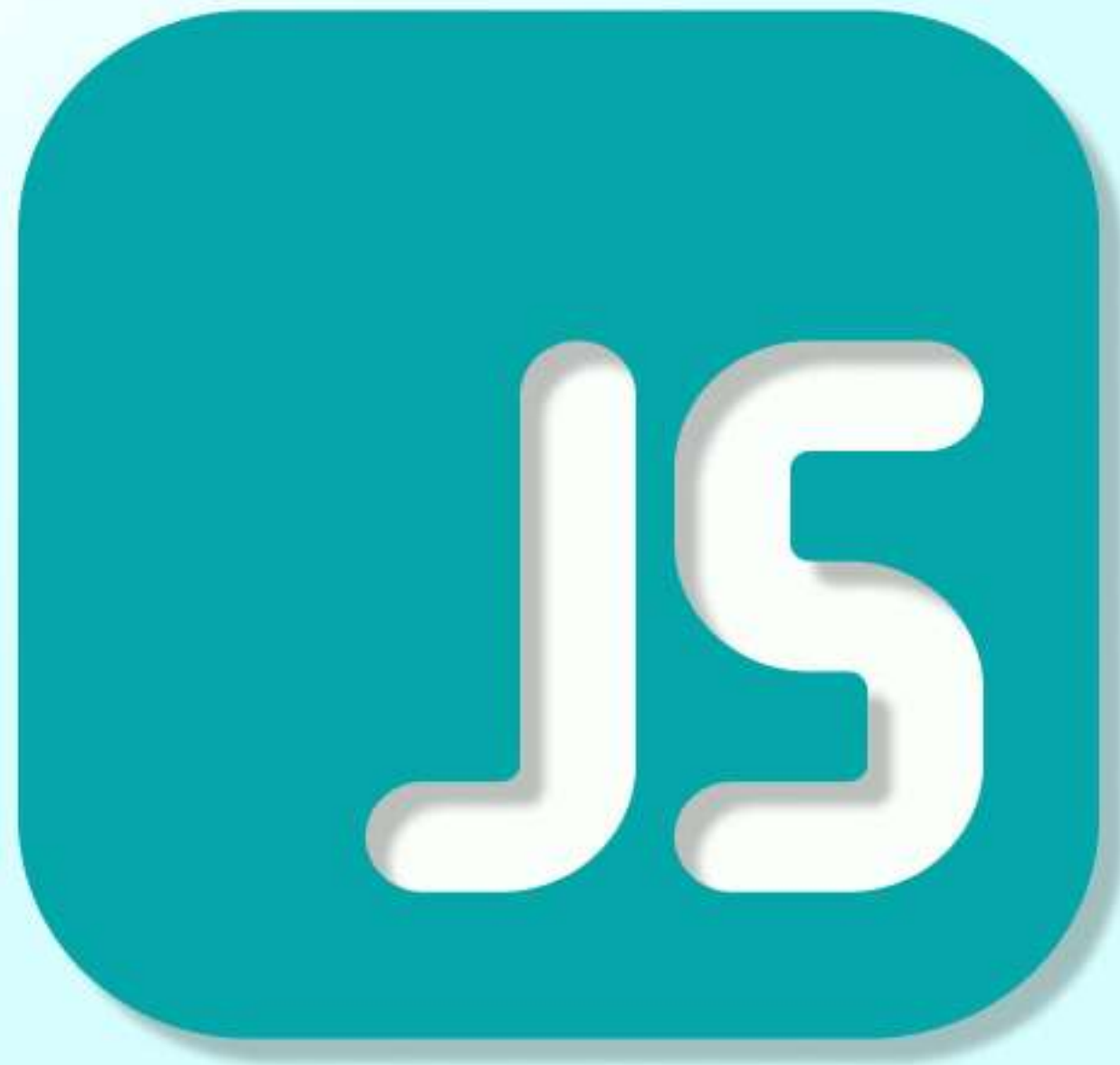




@CODE.CLASH

JavaScript

Callback *vs* Promises



NEXT →

1

@CODE.CLASH

Hey Everyone 🙌

If you're **new to JavaScript** and have a hard time **trying to understand** how promises work.

In this **Post**, we will see difference between Javascript **Callback vs Promises**.

Do **Like, save and Share** This Post If You Found This Helpful.

NEXT →

The Goal behind Using It

- The Goal is to achieve **asynchronous code**.
- Async code allows multiple things to happen at the same time.
- We can achieve async code using two methods:
 - Callbacks
 - Promises.

Callbacks

- A callback function is a function passed into another function as an argument, which is called inside the otherFunction.

```
//callback function
const greet=(name) =>{ console.log('Hi '+name) }

// function
const callMe = ( callback ) =>{
  // Take input and save in name
  let name = prompt('Enter your name');
  callback(name);
}

// passing funtion as parameter
callMe(greet);
```


Promise

- Promises are JavaScript objects that represent an eventual completion or failure of an asynchronous operation.
- A promise has two possible outcomes: it will either be kept when the time comes, or it won't.
- A promise is a returned object where you attach callbacks, instead of passing callbacks into a function.

```
//callback function
const greet = (name) => {
  console.log('Hi ' + name);
};

// function
const callMe = () => {
  return new Promise((resolve, reject) => {
    let name = prompt('Enter your name');
    if (name) resolve(name);
    reject(false);
  });
};

// start call
callMe().then((result) => greet(result));
```


Callbacks Vs Promises

- A key difference between the two is
- when using the callback approach, we'd normally just pass a callback into a function.
- In promises, however, you attach callbacks on the returned promise object.
- Making callbacks async can cause issues such as callback hell, so to avoid this we can use promises.

Promises

- The syntax is user-friendly and easy to read.
- Error Handling is easier to manage.

```
api()  
  .then(function (result) {  
    return api2();  
  })  
  .then(function (result2) {  
    return api3();  
  })  
  .then(function (result3) {  
    // do work  
  })  
  .catch(function (error) {  
    // handle error  
  });
```

Callbacks

- The syntax is difficult to understand.
- Error handling may be hard to manage.

```
api(function (result) {  
  api2(function (result2) {  
    api3(function (result3) {  
      // do work  
      if (error) {  
        // do something  
      } else {  
        // do something  
      }  
    });  
  });  
});
```


Did you find it Useful?

Leave a **comment!**



Alamin CodePapa

@CodePapa360

FOLLOW FOR MORE

Like



Comment



Repost

