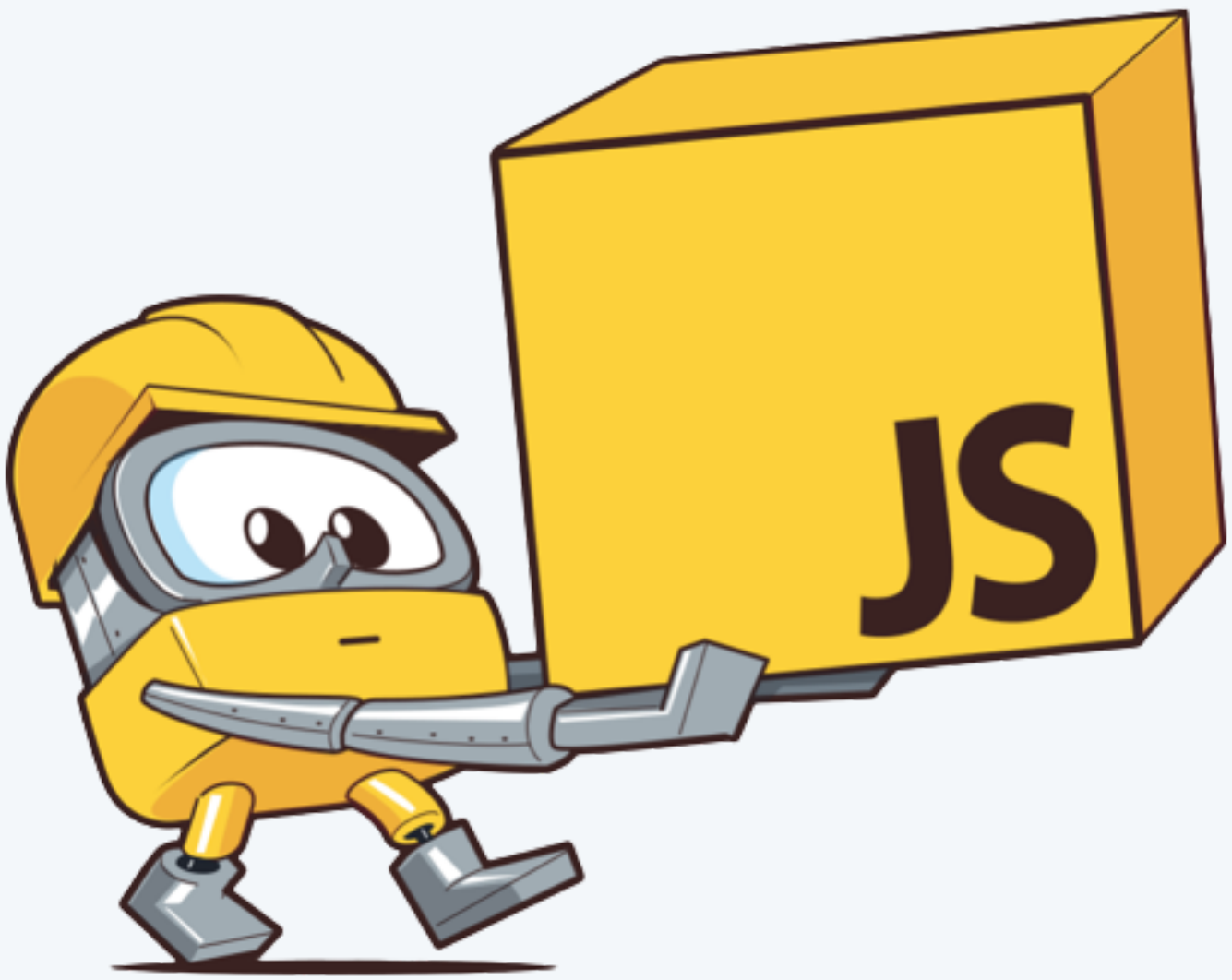
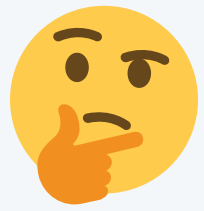
JavaScript

Callback Vs Promises



But why do we need this ??



- We need this to achieve asynchronous code.

- Asynchronous code allows multiple things to happen same time.

- We can achieve this by using two methods:

  - Callbacks
  - Promises

Callbacks

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

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

// callback function
function callMe() {
    console.log('I am callback function');
}

// passing function as an argument
greet('Peter', callMe);
```

Promises

- Promises are Javascript objects that represent an eventual completion or failure of an asynchronous operation.

- A promise is a returned object where you attach callbacks, instead of passing callbacks into a function.

```
const count = true;

let countValue = new Promise(function (resolve, reject)
{   if (count) {
        resolve("There is a count value.");
    } else {
        reject("There is no count value");
    }
});

console.log(countValue);

//Output
Promise {<resolved>: "There is a count value."}
```

Callbacks Vs Promises

- **A key difference between the two:**

  - 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 any error that may
    //occur before this point
});
```

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
            }
        });
    });
});
```