JavaScript - Hands On

```
$ node -v
V12.4.0

$ npm -v
6.9.2
```

hello-world.js

```
console.log('Hi there!');
```

```
$ node hello-world.js
Hi there!
```

hello-world.js

```
let name = 'Bardok';
console.log(`Hello ${ name }, welcome!`);
```

```
$ node hello-world.js
Hello Bardok, welcome!
```

process.argv.js

```
console.log(process.argv);
```

```
$ node process.argv.js
[
  'C:\\Program Files\\nodejs\\node.exe',
  'C:\\workspace\\process.argv.js'
]
```

```
$ node process.argv.js a b c “Just a test”
[
  'C:\\Program Files\\nodejs\\node.exe',
  'C:\\workspace\\process.argv.js',
  'a',
  'b',
  'c',
  'Just a test'
]
```

hello-world.js

```
let name = process.argv[2];
console.log(`Hello ${ name }, welcome!`);
```

```
$ node hello-world.js “Luke Skywalker”
Hello Luke Skywalker, welcome!
```

sum.js

```javascript
let numberA = 11;
let numberB = 31;

const total = numberA + numberB;

console.log('The sum is ' + total);
```

```
$ node sum.js
The sum is 42
```

How to sum all numbers passed by arguments?

```
$ node sum.js 1 2 3 4 5
The sum is 15
```

process.argv.js

```
const values = process.argv.slice(2);

for (let index = 0; index < values.length; index++) {
 console.log(`Value at index ${index} is ${values[index]}`);
}
```

```
$ node process.argv.js 1 2 3 4 5
Value at index 0 is 1
Value at index 1 is 2
Value at index 2 is 3
Value at index 3 is 4
Value at index 4 is 5
```

process.argv.js

```
const values = process.argv.slice(2);
values.forEach(value => console.log(value));
```

```
$ node process.argv.js 1 2 3 4 5
1
2
3
4
5
```

process.argv.js

```
function regularPrint(value) {
 console.log(`The value is: ${ value }`);
}

function fancyPrint(value) {
 console.log(`-----> ${ value } <-----`);
}

const values = process.argv.slice(2);
const print = regularPrint;

values.forEach(print)
```

```
$ node process.argv.js 1 2 3 4 5
The value is: 1
The value is: 2
The value is: 3
The value is: 4
The value is: 5
```

What will happen if we change the line below in process.argv.js?

```
const print = fancePrint;
```

javascript-functional.js

```javascript
const names = [
 'Ichigo Kurosaki',
 'Didi Mocó',
 'Bruce Wayne'
];

const print = name => console.log(name);

names
 .sort()
 .forEach(print);
```

```
$ node javascript-functional.js
Bruce Wayne
Didi Mocó
Ichigo Kurosaki
```

javascript-functional.js

```javascript
const names = [
 'Ichigo Kurosaki',
 'Didi Mocó',
 'Bruce Wayne'
]

const lastNameFirst = name => {
 const [ firstName, lastName ] = name.split(' ');
 return `${ lastName }, ${ firstName }`;
}

const print = name => console.log(name);

names
 .map(lastNameFirst)
 .sort()
 .forEach(print)
```

```
$ node javascript-functional.js
Kurosaki, Ichigo
Mocó, Didi
Wayne, Bruce
```

fun fun function

javascript-async.js

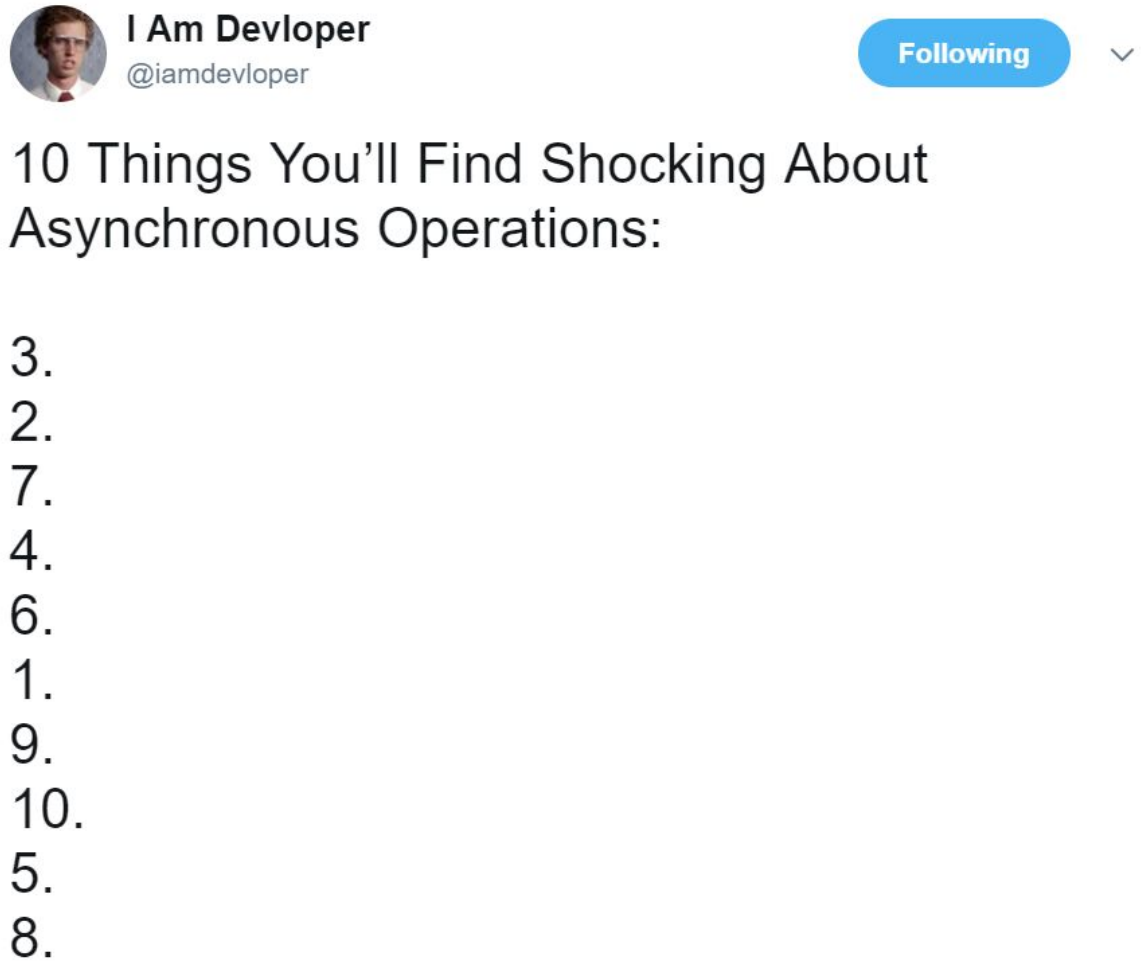
```javascript
const timer = 1000;
const task = () => console.log('An operation just happened!');

setTimeout(task, timer);

console.log('Is it ended?');
```

```
$ node javascript-async.js
Is it ended?
An operation just happened!
```

javascript-async.js

```javascript
function operation1() {
 console.log('Operation 1');
}

function operation2() {
 setTimeout(() => console.log('Operation 2'), 1000);
}

function operation3() {
 console.log('Operation 3');
}

operation1();
operation2();
operation3();
```

```
$ node javascript-async.js
Operation 1
Operation 3
Operation 2
```



I Am Devloper
@iamdevloper

10 Things You’ll Find Shocking About Asynchronous Operations:

3.
2.
7.
4.
6.
1.
9.
10.
5.
8.

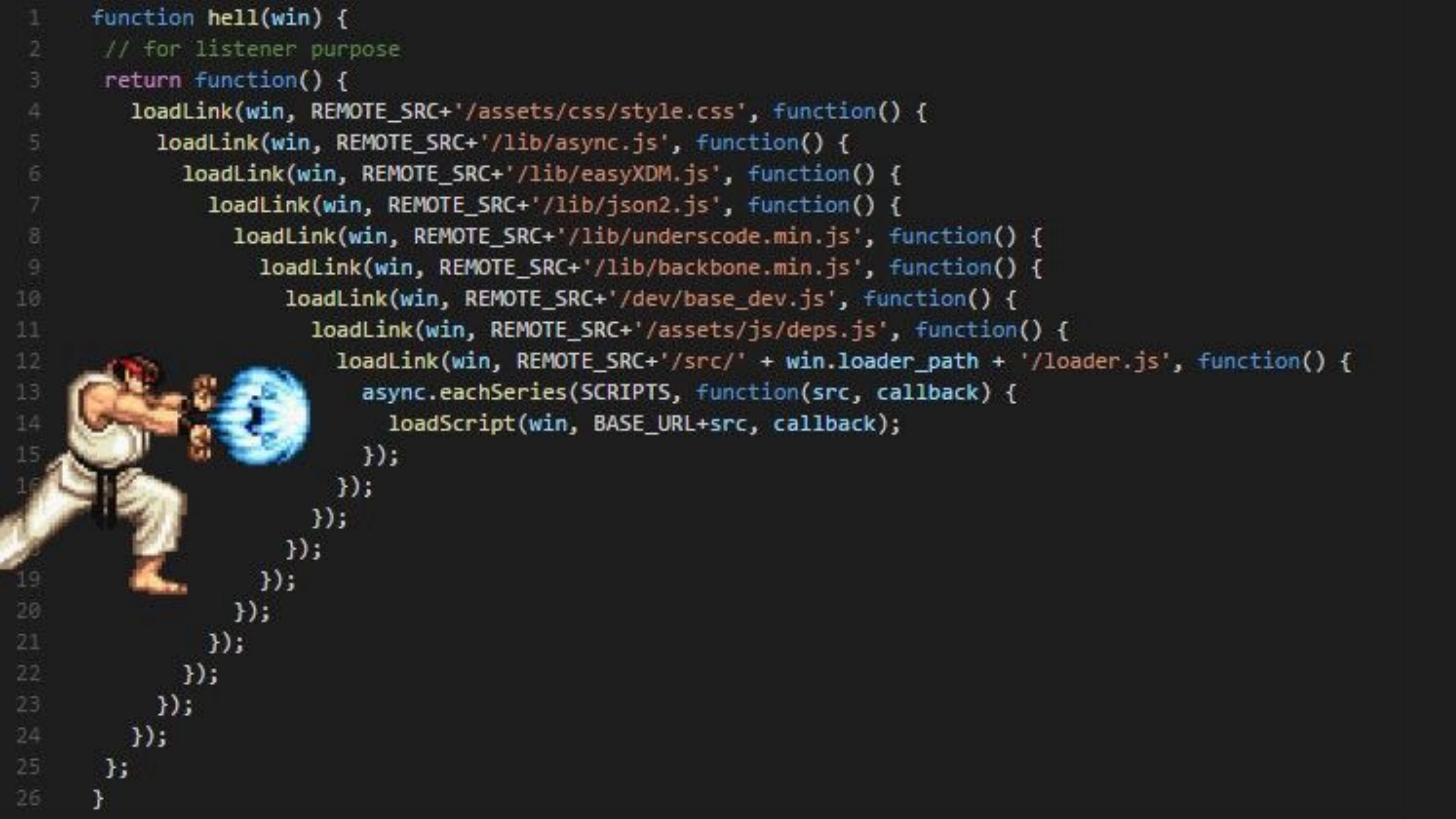
handling-io.js

```
const fs = require('fs');

function printFileContent(error, data) {
   if(error) {
       console.log('File not found!', error);
       return;
   }
   console.log(data);
}

fs.readFile('./students.txt', 'utf8', printFileContent);
console.log('Is it not done yet??');
```

```
$ node index.js
Is it not done yet??
40089   Felipe Godinho de Almeida
40069   Francine de Fátima Braga Amorim
40065   Gerson Lourenço de Carvalho
40068   Henrique Pereira da Conceição
40071   Isabel Francine Mendes
...
```

callback-hell.js

```
const fs = require('fs');

fs.readFile('my-file.txt', 'utf8', (error, data) => {
   if (error) throw error;
   fs.writeFile('my-file.txt', data + ' callback 1', (error) => {
       if (error) throw error;
       fs.writeFile('my-logfile.txt', data + ' callback 2', (error) => {
           if (error) throw error;
           console.log('Process has been finished!');
       });
   });
});
```



```
function hell(win) {
 // for listener purpose
 return function() {
   loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
     loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
       loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
         loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
           loadLink(win, REMOTE_SRC+'/lib/underscode.min.js', function() {
             loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
               loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
                 loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
                   loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
                     async.eachSeries(SCRIPTS, function(src, callback) {
                       loadScript(win, BASE_URL+src, callback);
                     });
                   });
                 });
               });
             });
           });
         });
       });
     });
   });
 };
}
```

promises.js

```
readData('my-file.txt')
    .then(writeData)
    .then(addToLog)
    .then(notifyOnSuccess)
    .catch(notifyOnError);
```

```
$ node promises.js
Process has been finished!
```

async-await.js

```js
async function main() {
   try {
      const file = await readData('my-file.txt');
      const result = await writeData(file);
      await addToLog(result);
      notifyOnSuccess();
   } catch (error) {
      notifyOnError(error);
   }
}

main();
```

```
$ node promises.js
Process has been finished!
```