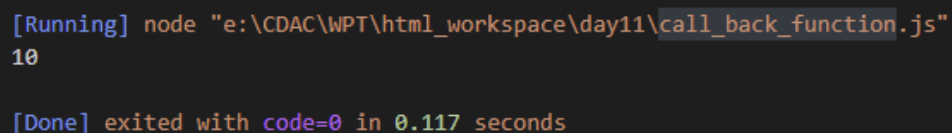


1. Assignment on JavaScript call back functions.

//A call-back is a function passed as an argument to another function.

```
function myCalculator(num1, num2, myCallback) {
  let sum = num1 + num2;
  myCallback(sum);
}
function display(sum) {
  console.log(sum);
}
function main() {
  myCalculator(5, 5, display); // display function is passed to myCalculator() as an argument.
  //When we pass a function as an argument, remember not to use parenthesis().
}
main();
```

Output:



```
[Running] node "e:\CDAC\WPT\html_workspace\day11\call_back_function.js"
10
[Done] exited with code=0 in 0.117 seconds
```

2. Assignment on Timers, Promises, and Async & Await

Timers

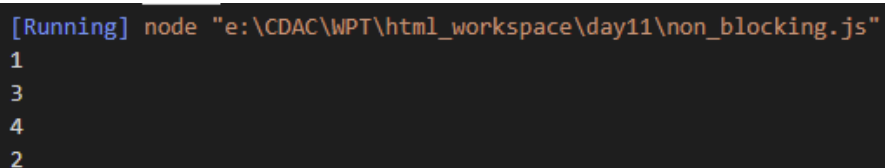
```
function helloAsync() {
  console.log("1");

  setTimeout(() => console.log("2"), 3000);
  //This is a non-blocking statement it will execute after timeout 3000ms/3seconds.

  console.log("3");
  console.log("4");
}

helloAsync();
```

Output:



```
[Running] node "e:\CDAC\WPT\html_workspace\day11\non_blocking.js"
1
3
4
2
```

Promise: While a Promise object is "pending" (working), the result is undefined.

```
async function getNumber() {  
  return;  
}  
function main() {  
  let num = getNumber();  
  console.log(num);  
}
```

Output:

```
[Running] node "e:\CDAC\WPT\html_workspace\day11\asynchronous_function.js"  
Promise { undefined }
```

Promise: When a Promise object is "fulfilled", the result is a value.

```
async function getNumber() {  
  return 100;  
}  
  
function main() {  
  let num = getNumber();  
  console.log(num); // Promise {100}  
}
```

Output:

```
[Running] node "e:\CDAC\WPT\html_workspace\day11\promise_and_await.js"  
Promise { 100 }
```

Async:

- **async** makes a function return a **Promise**
- To make a function asynchronous use keyword -> **async**
- Return type of async function -> **Promise{}**

```
function sumSync(n1, n2) { } //Synchronous function
```

```
async function sumAsync(n1, n2) { } //Asynchronous function
```

```
let output1 = sumSync(1, 1);  
let output2 = sumAsync(1, 1);
```

```
console.log(output1);  
console.log(output2);
```

Output:

```
[Running] node "e:\CDAC\WPT\html_workspace\day11\asynchronous_function.js"  
undefined  
Promise { undefined }
```

Await: await makes a function wait for a Promise

```
async function getNumber() {  
  return 100;  
}
```

```
// To use await we must make a function async  
async function main2() {  
  let num = await getNumber();  
  console.log("Using Await", num); // 100  
}
```

Output:

```
[Running] node "e:\CDAC\WPT\html_workspace\day11\promise_and_await.js"  
Using Await 100
```