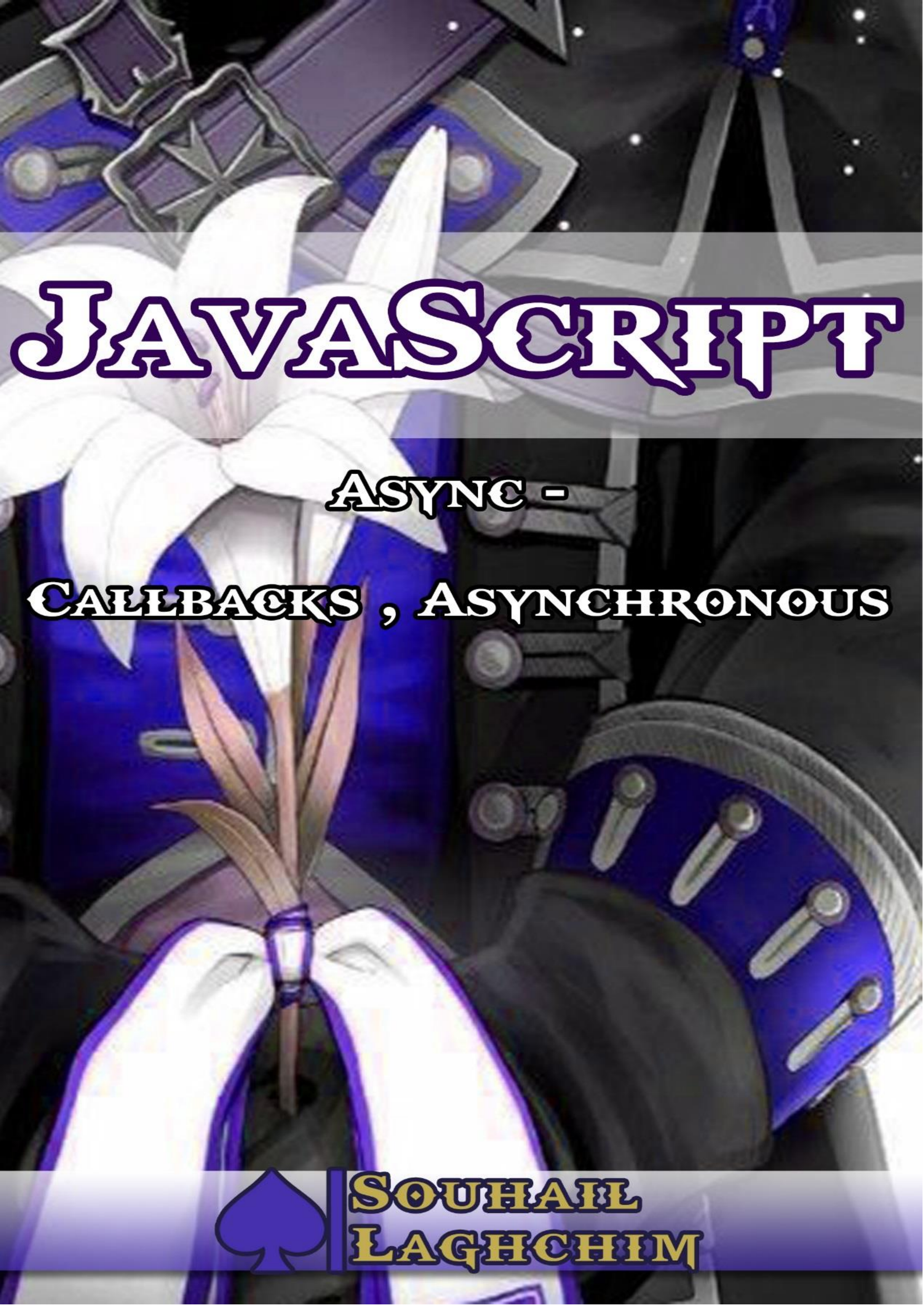
JavaScript

Async -

Callbacks , Asynchronous

Souhail Laghchim



JavaScript Callbacks

"I will call back later!"

A callback is a function passed as an argument to another function

This technique allows a function to call another function

A callback function can run after another function has finished

Function Sequence

JavaScript functions are executed in the sequence they are called. Not in the sequence they are defined.

This example will end up displaying "Goodbye":

Example

```
function myFirst() {
  myDisplayer("Hello");
}

function mySecond() {
  myDisplayer("Goodbye");
}

myFirst();
```

This example will end up displaying "Hello":

Example

```
function myFirst() {
  myDisplayer("Hello");
}

function mySecond() {
  myDisplayer("Goodbye");
}

mySecond();
myFirst();
```

Sequence Control

Sometimes you would like to have better control over when to execute a function.

Suppose you want to do a calculation, and then display the result.

You could call a calculator function (`myCalculator`), save the result, and then call another function (`myDisplayer`) to display the result:

Example

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2) {
  let sum = num1 + num2;
  return sum;
}

let result = myCalculator(5, 5);
myDisplayer(result);
```

Or, you could call a calculator function (`myCalculator`), and let the calculator function call the display function (`myDisplayer`):

Example

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2) {
  let sum = num1 + num2;
  myDisplayer(sum);
}

myCalculator(5, 5);
```

The problem with the first example above, is that you have to call two functions to display the result.

The problem with the second example, is that you cannot prevent the calculator function from displaying the result.

Now it is time to bring in a callback.

JavaScript Callbacks

A callback is a function passed as an argument to another function.

Using a callback, you could call the calculator function (`myCalculator`) with a callback, and let the calculator function run the callback after the calculation is finished:

Example

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2, myCallback) {
  let sum = num1 + num2;
  myCallback(sum);
}

myCalculator(5, 5, myDisplayer);
```

In the example above, `myDisplayer` is the name of a function.

It is passed to `myCalculator()` as an argument.

When you pass a function as an argument, remember not to use parenthesis.

Right: myCalculator(5, 5, myDisplayer);

Wrong: ~~myCalculator(5, 5, myDisplayer())~~;

Asynchronous JavaScript

"I will finish later!"

Functions running in parallel with other functions are called asynchronous

A good example is JavaScript setTimeout()

Asynchronous JavaScript

The examples used in the previous chapter, was very simplified.

The purpose of the examples was to demonstrate the syntax of callback functions:

Example

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2, myCallback) {
  let sum = num1 + num2;
  myCallback(sum);
}

myCalculator(5, 5, myDisplayer);
```

In the example above, `myDisplayer` is the name of a function.

It is passed to `myCalculator()` as an argument.

In the real world, callbacks are most often used with asynchronous functions.

A typical example is JavaScript `setTimeout()`.

Waiting for a Timeout

When using the JavaScript function `setTimeout()`, you can specify a callback function to be executed on time-out:

Example

```
setTimeout(myFunction, 3000);

function myFunction() {
  document.getElementById("demo").innerHTML = "I love You !!";
}
```

In the example above, `myFunction` is used as a callback.

The function (the function name) is passed to `setTimeout()` as an argument.

3000 is the number of milliseconds before time-out, so `myFunction()` will be called after 3 seconds.

When you pass a function as an argument, remember not to use parenthesis.

Right: setTimeout(myFunction, 3000);

Wrong: ~~setTimeout(myFunction(), 3000)~~;

Instead of passing the name of a function as an argument to another function, you can always pass a whole function instead:

Example

```
setTimeout(function() { myFunction("I love You !!!"); }, 3000);

function myFunction(value) {
  document.getElementById("demo").innerHTML = value;
}
```

In the example above, `function(){ myFunction("I love You !!!"); }` is used as a callback. It is a complete function. The complete function is passed to setTimeout() as an argument.

3000 is the number of milliseconds before time-out, so `myFunction()` will be called after 3 seconds.

Waiting for Intervals:

When using the JavaScript function `setInterval()`, you can specify a callback function to be executed for each interval:

Example

```
setInterval(myFunction, 1000);

function myFunction() {
  let d = new Date();
  document.getElementById("demo").innerHTML=
  d.getHours() + ":" +
  d.getMinutes() + ":" +
  d.getSeconds();
}
```

In the example above, `myFunction` is used as a callback.

The function (the function name) is passed to `setInterval()` as an argument.

1000 is the number of milliseconds between intervals, so `myFunction()` will be called every second.

Waiting for Files

If you create a function to load an external resource (like a script or a file), you cannot use the content before it is fully loaded.

This is the perfect time to use a callback.

This example loads a HTML file (`mycar.html`), and displays the HTML file in a web page, after the file is fully loaded:

Waiting for a File:

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function getFile(myCallback) {
  let req = new XMLHttpRequest();
  req.open('GET', "mycar.html");
  req.onload = function() {
    if (req.status == 200) {
      myCallback(this.responseText);
    } else {
      myCallback("Error: " + req.status);
    }
  }
  req.send();
}

getFile(myDisplayer);
```

In the example above, `myDisplayer` is used as a callback.

The function (the function name) is passed to `getFile()` as an argument.

Below is a copy of `mycar.html`:

mycar.html

```
<img src="img_car.jpg" alt="Nice car" style="width:100%">

<p>A car is a wheeled, self-powered motor vehicle used for
transportation.
Most definitions of the term specify that cars are designed to
run primarily on roads, to have seating for one to eight people,
to typically have four wheels.</p>

<p>(Wikipedia)</p>
```

Email : souhail.developer@gmail.com

SOUHAIL DEVELOPER

Souhail Laghchim | Developer & Designer

Email : souhail.developer@gmail.com
WebSite : https://souhail-laghchim-pro.blogspot.com/