後端網絡開發人員證書
(一) 後端網絡開發

2. Node.js 函式
Node.js function

Presented by Krystal Institute

Learning Outcome

1. Study what is callback function in JavaScript
2. Study the reasons and requirements for using callback function

2.3 Callback Function

Review of rest and default parameter

Rest parameter

- Allows a function to accept an indefinite number of arguments
- For example, you want to input some comments on website to a function
- Required to be the last parameter of the function

Review of rest and default parameter

Default parameter

- Allows parameters to be initialized with a default value that to be use when no value is input
- For example, we can use default parameters for date parameter
- Default parameters should be the last parameters

Review of rest and default parameter

Default parameter and rest parameter

- When both default parameter and rest parameter exist, rest parameter should go last
- For example:

```
function function1(a, b=1, ...c){
        command...}
```

What is callback function

- In the past few topic about, we are focusing on function and its parameters
- Those parameters can be number, string, Boolean, etc., and as many parameters as it can be

What is callback function

- As we can see, all of the parameters that we use are objects
- Function, which is also an object in JavaScript
- Can we use it as a parameter for another function?

What is callback function

- The answer is yes, and this is what we call callback function
- A callback is a function passed as a parameter to another function
- Let's look at its syntax

What is callback function

- Syntax of callback function:

```
function f1(f2){

        f2(parameter);

}
```

Callback function – learning activity

- Now let's take a look at a simple example
- Suppose we have a function to display a inputted value on the console:

```
function print(x){
    console.log(x);
}
```

Callback function – learning activity

- Then we will have another function that with three parameters, two for calculation and one for callback
- To use the callback, we need not to do anything when we assign the parameter:

```
function cf(a,b,f1){



}
```

Callback function – learning activity

- Instead, we will use the parameter like a function inside the function:

```
function cf(a,b,f1){
    var c = a+b;
    f1(c);
}
```

Callback function – learning activity

- Finally, let's run the second function with the first function as the argument
- Note that parenthesis is not needed for callback function:

```
cf(1,2,print);
```

Callback function – learning activity

- Full script of cf.js

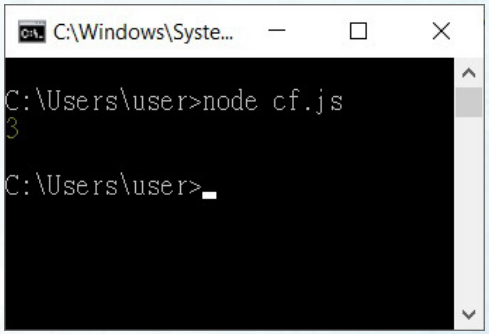
```
function print(x){
        console.log(x);
}
function cf(a,b,f1){
        var c = a+b;
        f1(c);
}
cf(1,2,print);
```

Callback function – learning activity

- Now let's take a look at the result of the script:

  node cf.js

- It returns 3 after running the script file



Callback function – learning activity

- This is the way to let a function to call another function
- In the last example, we can see that the second function will wait until the second function compute the sum of a and b

Why using callback function

- In the previous example, we have try to use callback function
- But it seems like it is just like the normal practice
- So why we have to use callback function?

Why using callback function

- The reason is that we can run the main function with a callback
- While letting the main function run after the wanted process is finished

Why using callback function

- Let's look back to the previous example
- Suppose we split the function and do the same job
- To achieve the goal we have to call two functions instead of one

- Full script of cf.js

```
function print(x){
        console.log(x);
}
function cf(a,b){
        return a+b;}
var c = cf(1,2);
print(c);
```

Why using callback function

- Another approach is to put the first function to the second function
- To achieve the goal we have to just need to call one function
- We must run print() if we call cf() to calculate a+b

- Full script of cf.js

```
function print(x){
        console.log(x);
}
function cf(a,b){
        var c = a+b;
        print(c);}
cf(1,2);
```

Using callback – learning activity

- Using the previous example may not easy to understand the usefulness of callback function
- Let's look at an example that we have used before
- Which is the method for adding all array entries

Using callback – learning activity

- Recall the last lesson:
- We will calculate the mean mark of assignments, just like calculating mean for statistics parameters

```
var aaverage =
assignment.reduce(function(a,b){return
a+b},0) / assignment.length;
```

Using callback – learning activity

- Let's break the statement to two parts
- Part one is the reduce method of array

ARRAY.reduce()

Using callback – learning activity

ARRAY.reduce()

- Now let's search for the arguments of this method
- Link: https://www.w3schools.com/jsref/jsref_reduce.asp

Using callback – learning activity

ARRAY.reduce(function(total, currentValue, currentIndex, arr), initialValue)

- The first argument is a callback
- It controls the way to handle array items when we target to reduce the array to a number

Using callback – learning activity

```
function(a,b){return a+b}
```

- Then let's look at the callback that we used before
- It is just a function that return the sum of parameter a and b
- Note that when we put function as an argument, name is not necessary, just like putting number or string to a function

Using callback – learning activity

```
[1,2,3,4,5].reduce(function(a,b){return a+b},0);
```

- Now let's study a quick example to see how the callback function works
- The aim of this statement is to calculate the sum of all array items

Using callback – learning activity

```
[1,2,3,4,5].reduce(function(a,b){return a+b},0);
```

- Firstly, we have a initial value 0
- And then the initial value will put to the callback as the first argument
- And then each array items will be the second argument of the callback, the return will be the next first argument

Using callback – learning activity

0 1 1

[1,2,3,4,5].reduce(function(a,b){return a+b},0);

↓

1 2 3

[1,2,3,4,5].reduce(function(a,b){return a+b},0);

↓

3 3 6

[1,2,3,4,5].reduce(function(a,b){return a+b},0);

↓

6 4 10

[1,2,3,4,5].reduce(function(a,b){return a+b},0);

↓

10 5 15

[1,2,3,4,5].reduce(function(a,b){return a+b},0);

Using callback – learning activity

- After calculating all array items, the callback function will return the value 15
- The callback function will only return the value after the inputted function complete

Using callback – learning activity

- Since the callback is only use for the callback function, it is not suggested to define it like other function
- And using callback in callback function may make the statement too long

Using callback – learning activity

- So we will use arrow function to shorten the callback
- For example:

```
[1,2,3,4,5].reduce(function(a,b){return a+b},0);
                          ↓
        [1,2,3,4,5].reduce((a,b)=>a+b,0)
```

- Arrow function will be introduced later on

Using callback – learning activity

- Since the callback is only use for the callback function, it is not suggested to define it like other function
- And using callback in callback function may make the statement too long

Using callback – learning activity

- Callback function is extremely useful when we want to control the executing sequence of the code
- But this the usefulness of it is not that obvious on synchronous JavaScript

Using callback – learning activity

- Synchronous is that the code within will execute straight line on a single thread
- At this point, almost all JavaScript code that we face is synchronous
- Where really need callback function is asynchronous JavaScript

Using callback – learning activity

- Asynchronous JavaScript means that other code may need to wait until part of the code complete
- This is very common when we have to access to resources from other device such as database

Using callback – learning activity

- And with callback function, we can easily create an asynchronous function
- Which is one function has to wait for another function to complete it work

Learning Outcome

1. Callback function in JavaScript

Syntax:

```
function f1(f2){
        f2(parameter);
}
```

Learning Outcome

2. The reasons and requirements for of using callback function

- Asynchronous JavaScript
- Synchronous JavaScript
- Arrow function

References

- Node.js – https://nodejs.org/
- JavaScript Callbacks – https://www.w3schools.com/js/js_callback.asp