**The Pitfall Of JavaScript Functions**

Talking about javascript, everyone knows how to programmed it. Some framework even uses as for it’s language framework, such as React, React Native or Vue...etc to name a few. This article will discuss about the pitfall of Javascript that people seldom notice.

What is fuction call, acording to Microsoft copilot:A function call is a way to invoke or execute a function in programming. Functions are reusable blocks of code designed to perform specific tasks. When you "call" a function, you are essentially instructing the program to run the code inside that function.

Here’s the general idea:

1. **Function Definition**: A function is defined with a specific name, optional parameters, and some code.
2. **Function Call**: You use the function name followed by parentheses (sometimes including arguments) to execute the function.

For example:

function addition() {

const a = 2;

const b = 3;

console.log(a +b)

}

To successfully call this function you just need to type in:

addition() // It will print out 5 to the console

This is a kind of typical way of most programmers execute a function, but one may ask is there another way of executing it rather that specifically typing “additon()” to call it. The answer is yes! Please take a look below:

(function addition() {

const a = 2;

const b = 3;

console.log(a + b);

})()

// It will print out 5 on the console

This kind of executing the fuction immediately, does not require you to call “additon()” and will directly prints out 5 to the console. We have a term for this kind of function call, we call it “Immediate Invoke Fuction” (For short IIF). Below the diagram illustrate why this works.

function addition() {

const a = 2;

const b = 3;

console.log(a +b)

}

addition (); //prints out 5 to the console

As you can see, the square is the whole function that is corresponding to the square of your function call “**addition**”, in order to execute the function, all you need is to add a parathensis after it.

Let’s split the function call to 2 parts.

addition () ;

The first part of the square is the whole function, the second part of the triangle is the executing part. Now lets look at the IFF.

(function addition() {

const a = 2;

const b = 3;

console.log(a + b);

}) ()

To call the function without specifically type in “addition()”, all you need is just add a parathensis right after the funciton and it will be execute.

You can also pass arguments into the IIF functions:

(function addition(a, b) {

console.log(a + b);

})(a, b)

What is the purpose and usage of this type of function call? The answer is initializtion. As for programmers, we sometimes need to initialize variable to a default state or even sometimes creating database and network connections. Typically, we do this kind of initializtion throught the development life cycle of the program. For instance, for React we have “willMount” or “didMount”, but sometimes it is not feasible when we initialize an object or component which is not ready for use. Then this type of IIF comes into play.

After discussing fuction call, we willl jump into another topic. For the first part we discuss how to immediately invoke a function without specifically calling it. This part is the reverse, how to avoid the function to be call immediately without ones interaction.

Let’s take a look when a user clicks a button.

function addition() {

const a = 2;

const b = 3;

console.log(a +b)

}

A button onClick event:

<button onclick="additon()">Click Me</button>

The above is a typical onClick event, which calls the “additon()” function when the user clicks on the button. By looking at the syntax above, it seems there is no problem to this kind of method call. Ah...not quite! Recall back your memory, to successfully call and excute a function, we need to provide 2 things; one is the whole function (the square), the other part is the execute part (the triangle). What happens to the above statement is the function will be execute without user clicks on it (for every first time the program refresh). How to make it to execute the function only when the user clicks the button. The answer is “Arrow Function”. Below demostrate how arrow function works when incorportate with the onClick event.

<button onclick="() -> addition()">Click Me</button>

As you can see, for the button onClick event, the addtion() will not be execute only when the users clicks on it. This type of method call will avoid function being called during the refresh or initialize state.