**Function Declaration vs Function Expression**

Over time, we change. We grow up. We make mistakes. We learn and adapt. We identify a goal (e.g. buy a new car, a new house, take a holiday, etc.) and then work to achieve this goal.

**JavaScript is no different.**It is ‘growing up’ all the time, adapting to the needs and wants of the web experience.

TAKE A STEP BACK ...

In JavaScript, a function is not a “magical language structure”. A function is just a special kind of value.

One way to create a function is known as a **Function Declaration**.

A declared function is “saved for later use”. You run the function later, when it is invoked (called).

Just as a JavaScript variable declaration must start with let, Function Declarations must begin with the word function.

Example: function message() {return "secret message"};

The function is declared here, but to use it, it must be invoked using function name. e.g. message();

But a JavaScript function can also be defined using an expression.

A **Function Expression** can be stored in a variable like this:

let message = function() { return "secret message"};

Functions stored in variables do not need function names (to be pedantic, they can have names, but not required). They are always run / invoked / called using the variable name. e.g. message();

This can be really handy if you want to start using the variable (which is actually a function ) as arguments to other functions.

**Bottom line: a Function Declaration and a Function Expression are 2 ways of creating a function in JavaScript. In both examples above, its just a value stored in a variable called message. In other words, the meaning of the above code samples is the same: "create a function and put it into the variable called message".**

SO WHY USE ONE OVER THE OTHER ?

There are a few reasons why you would want to use one over the other.

**#1.** Function Declarations load before any code is executed while Function Expressions load only when the interpreter reaches that line of code.

In other words, you can invoke Function Declarations before you define the actual function, and your code will work just fine, but with Function Expressions you can’t do this and the JavaScript interpreter will throw an error at you.

**Bottom line: A Function Declaration can be called before its defined. A Function Expression is created only when the execution engine reaches it and is usable only from that moment.**

Huh?

Examples speak louder than words so let me show you.

Let’s use a Function Expression, and try call it before we define it in our code (you’ll notice we get an error).

Text

Description automatically generated with medium confidence

Now, lets do the same thing with a function declaration (you’ll see it executes).

Graphical user interface, application

Description automatically generated

The reason this is happening is that function expressions aren’t hoisted (well, to be really geeky they are hoisted but not initialized), which allows them to retain a copy of the local variables from the scope where they were defined. Don’t worry too much about what this all means when you’re first starting out. Just know that hoisting is a JavaScript behavior which moves all declarations to the top of your code before execution.

**#2.**A benefit to Function Expressions is that you don’t have to store the function in a variable. What you can do is invoke the function immediately. This is known as an 'immediately invoked function expression' or IIFE.

For example we could write something like this:

Graphical user interface

Description automatically generated

I don’t want to talk too much about IIFEs now, because they were often used with the module pattern to limit what a module would expose to the caller. But now have better options for scoping, modules and intelligent bundling of assets, so its use case is diminishing.

**#3.** Finally, in strict mode, when a Function Declaration is within a code block, it’s visible everywhere inside that block. But not outside of it.

Huh?

Okay, let’s code up a simple example where we want to only show a welcome messaged when a user is logged in. Let’s define our message function as a Function Declaration. You'll see if we try and access our message function outside if the IF statement, we get an error.

Text

Description automatically generated

What can we do to make message visible outside of the IF statement?

The solution is to use a Function Expression and assign message to the variable that is declared outside of if and has the proper visibility.

This code works as intended:

Graphical user interface, text

Description automatically generated with medium confidence

WHICH TO USE?

Personal preference, I guess.

I mostly use function expressions. But some developers are different to me, and prefer function declarations. They say that it gives them more flexibility in how to organize their code because they can call these functions before they are declared.

Hope this article has helped to wrap your head around the differences.

Keep going, and see you in the next lectures.

Clyde