W2d2 Functions HomeWorks

4. When should you choose Function Declaration versus Function Expression and why?

Naha:

1) Function Declaration: function doStuff( ) { };

Function Expression: const doStuff = function ( ) { } or const doStuff = ( ) =>{ };

2) Thinking about Function Declaration for when and where the function is needed:

a) we want a name, otherwise choose Function Expression,since it is “anonymous”.

b) we need to use the function for callback, since function expression can not be use as callback.

c) we want the availability of functions anywhere in our file. Function declarations are hoisted but function expressions are not.

3)

a) Function expressions are invoked to avoid polluting the global scope.

b) When a function is created at the same time it is called, we can use an IIFE. — immediately invoked function expressions — pretty much says it all here. When a function is created at the same time it is called, you can use an IIFE, like (function() => {})().

In short, use function declarations when we want to create a function on the global scope and make it available throughout our code. Use function expressions to limit where the function is available, keep our global scope light, and maintain clean syntax.

5. Functions are values in JavaScript. They can be assigned, copied, or declared in any place of the code. Is this a true or a false statement? Support your answer with examples.

Naha: Yes. It is a true statement.

Because in JS, Functions are values. They can be assigned, copied or declared in any place of the code.

a) If the function is declared as a separate statement in the main code flow, that’s called a “Function Declaration”.

b) If the function is created as a part of an expression, it’s called a “Function Expression”.

c) Function Declarations are processed before the code block is executed. They are visible everywhere in the block.

d) Function Expressions are created when the execution flow reaches them.