Retake Quiz

IIFE Quiz

IIFEs are one way to prevent the pollution of the global namespace by creating functions and variables that will disappear after the IIFE has been invoked.

- [x] True
- [ ] False

EXPLANATION

Variables and functions written within an IIFE cannot be accessed outside that function!

What does IIFE stand for?

- [ ] Invoked Immediately Function Enunciation
- [x] Immediately-Invoked Function Expression
- [ ] Involuntarily Invoked Function Expression
- [ ] Immediately-Invoked Function Embellishment

EXPLANATION

IIFE stands for Immediately-Invoked Function Expression.

A single IIFE can be invoked multiple times throughout an application.

- [x] False
- [ ] True

EXPLANATION

The exact opposite is true! An IIFE is invoked once then never again.

```
(function() {
  const test = "Hello world!";
})();

console.log(test); // ???
```

What will be printed when the above code snippet is run?

- [ ] `[Function]`
- [ ] `"Hello world!"`
- [x] An error is thrown.

EXPLANATION

The variables defined within an IIFE are not available in an outer scope.

```
function() {
  console.log("hello world!");
}(); // => 'hello world!'
```

True or False: The above IIFE syntax is correct.

- [ ] `true`

- [x] `false`

EXPLANATION

False! When we define and IIFE we need to wrap our anonymous function in the grouping operator before we invoke it. The above will give us a syntax error.

```
const result = (function() {
  return "food";
})();

console.log(result); // ???
```

What will be printed when the above code snippet is run?

- [ ] `[Function]`
- [x] `"food"`
- [ ] An error is thrown.

EXPLANATION

When an IIFE is assigned to a variable the function will be invoked and then the `return` value of that function will be assigned to the variable name. So in the above example the `result` variable would have the value returned by the IIFE (which in this case is `food`).