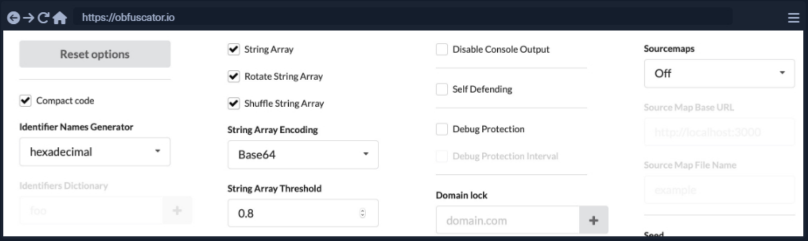
Advanced Obfuscation

So far, we have been able to make our code obfuscated and more difficult to read. However, the code still contains strings in cleartext, which may reveal its original functionality. In this section, we will try a couple of tools that should completely obfuscate the code and hide any remnants of its original functionality.

Obfuscator

Let's visit https://obfuscator.io. Before we click `obfuscate`, we will change `String Array Encoding` to `Base64`, as seen below:



https://obfuscator.io

Reset options

[✓] Compact code

Identifier Names Generator: hexadecimal

Identifiers Dictionary: foo

[✓] String Array

[✓] Rotate String Array

[✓] Shuffle String Array

String Array Encoding: Base64

String Array Threshold: 0.8

[ ] Disable Console Output

[ ] Self Defending

[ ] Debug Protection

[ ] Debug Protection Interval

Domain lock: domain.com

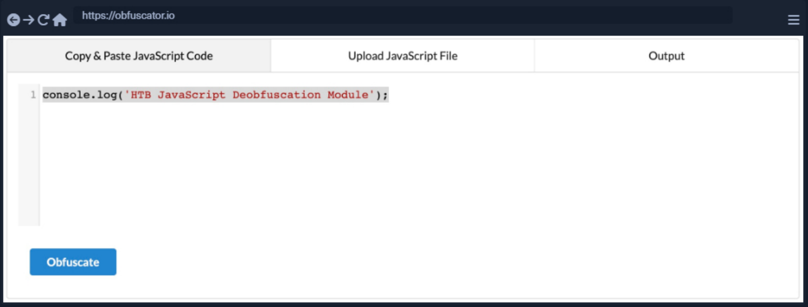
Sourcemaps: Off

Source Map Base URL: http://localhost:3000

Source Map File Name: example

Seed

Now, we can paste our code and click `obfuscate`:



https://obfuscator.io

```javascript
console.log('HTB JavaScript Deobfuscation Module');
```

We get the following code:

Code: javascript

```javascript
var _0x1ec6=['Bg9N','sfrciePHDMfty3jPChqGrgvVyMz1C2nHDgLVBIbnB2r1Bgu='];(function(_0x13249d,_0x1ec6e5){var _0x14f83b=f
```

This code is obviously more obfuscated, and we can't see any remnants of our original code. We can now try running it in https://jsconsole.com to verify that it still performs its original function. Try playing with the obfuscation settings in https://obfuscator.io to generate even more obfuscated code, and then try rerunning it in https://jsconsole.com to verify it still performs its original function.

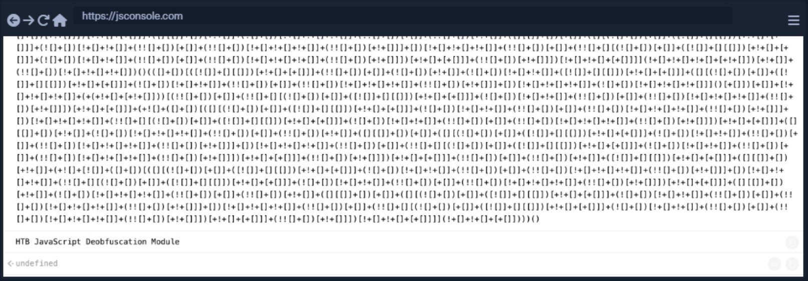
More Obfuscation

Now we should have a clear idea of how code obfuscation works. There are still many variations of code obfuscation tools, each of which obfuscates the code differently. Take the following JavaScript code, for example:

Code: javascript

```javascript
[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!
...SNIP...
[]]+(!![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!
```

We can still run this code, and it would still perform its original function:



https://jsconsole.com

```
[illegible]
HTB JavaScript Deobfuscation Module
<· undefined
```

Note: The above code was snipped as the full code is too long, but the full code should successfully run.

We can try obfuscating code using the same tool in JSF, and then rerunning it. We will notice that the code may take some time to run, which

Cheat Sheet

Table of Contents

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

shows how code obfuscation could affect the performance, as previously mentioned.

There are many other JavaScript obfuscators, like JJ Encode or AA Encode. However, such obfuscators usually make code execution/compilation very slow, so it is not recommended to be used unless for an obvious reason, like bypassing web filters or restrictions.