Non-alphanumeric code

With JavaScript & PHP by Gareth Heyes

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

How did it begin?

- Yosuke Hasegawa posted to the sla.ckers message board

-
  ```
  _=[]|[];$=_++;__=(_<<_);___=(_<<_)+_;____=__+__;_____=__+___;$$=({}+
  "")[_____]+({}+"")[_]+({}[$]+"")[_]+(($!=$)+"")[___]+(($==$)+"")[$]+(($==$)+"")
  [_]+(($==$)+"")[__]+({}+"")[_____]+(($==$)+"")[$]+({}+"")[_]+(($==$)+"")[_];$
  $$=(($!=$)+"")[_]+(($!=$)+"")[__]+(($==$)+"")[___]+(($==$)+"")[_]+(($==$)+"
  ")[$];$_$=({}+"")[_____]+({}+"")[_]+({}+"")[_]+(($!=$)+"")[__]+({}+"")[__+____
  _]+({}+"")[_____]+ ({}+"")[_]+({}[$]+"")[__]+(($==$)+"")[___];
  ($)[$$][$$]($$$+"('"+$_$+"')")();
  ```
- We were all amazed that this executed code without the need of alphanumeric characters

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

How does it work?

- JavaScript is a loosely typed language
- true+true == 2
- The key to non-alpha code is using string indexes to obtain individual letters
- ‘abc’[0] == ‘a’
- Objects can be converted to strings using the concatenation “+” operator and the toString value of that object can be used
- ''+{} == ‘[object Object]’
- We can use each letter of the generated string ‘[object Object][1] == ‘o’
- But we need 1 right? That is alphanumeric

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Getting numbers

- +[] == 0
- [] array returns a blank string when valueOf/toString is called
- + (infix) operator converts an object to a number
- Because the value is a blank string the result is 0
- To get the number 1 we can use the not “!” operator
- ![] == false; !![] == true
- Reverse of false is true then use + infix to convert to one
- +!![] == 1
- But that’s just 0 and 1 right?

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Getting bigger numbers

- @oxotnick from sla.ckers came up with a cool trick
- Using object accessors you can increment values without a variable
- ++[[]][+[]] == 1
- Works with an array inside an array:
  [[]]
- [[]][+[]] then accessing the first element of the array +[] == 0
- Then finally incrementing the value ++[[]][+[]]
- Notice ++[] is illegal but via accessor it works
- The [] is converted to 0 then incremented by 1
- You can increment further by concatting arrays
- ++[[]][+[]]+**[++[[]][+[]]][+[]]**

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Getting Objects

- Firefox 2 and older browsers allowed you to remove the object from function call and returned window instead. This is older ES behaviour.
- (1,[].sort)() == window
- This works on IE9 still ☺
- (1,[].reverse)() == window
- You are removing the “this” value of the array
- [3,1,2].sort() // works as expected
- We need window to call other functions such as alert with non-alpha code

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Generating sort

- We need to generate the string “sort” in order to get our reference to window
- Any ideas?
- fal**s**e
- [**o**bject Object]
- t**r**ue
- **t**rue
- We need the string index for each of the letters
- For false we need the 3rd index starting from 0
- [object Object] requires 1 and so on

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Generating sort cont.

- First we generate false with ![]+[] and convert it to a string
  ![] == false
  + [] // converts to string using a blank array
- Getting the letter is trickier then you first think:
  ![]+[][0] == NaN
- JavaScript is getting the first element of the array instead of concating
- Enclose with another array and access the first element and you get the string “false”
- [false] [0]
  [![]+[]][+[]]
- [‘false’][0][0]
  [![]+[]][+[]][+[]] == ‘f’

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Generating sort cont.

- We need to increment the value to access “s” as shown earlier
- ++[[]][+[]]+[++[[]][+[]]][+[]]+[++[[]][+[]]][+[]] == 3
- Using our string false we access the third element which is “s”
- [**![]+[]][+[]**][++[[]][+[]]+[++[[]][+[]]][+[]]+[++[[]][+[]]][+[]]] == ‘s’
- To generate “o” we only need the number 1 and an object string
- []+{} == [object Object] as string
- [[]+{}][+[]] access first element of array which is the string
- We’ll use a shortcut I mentioned eariler
- [[]+{}][+[]][+!![]]
- [[]+{}][+[]][**+!![]**] accesses 1st element from 0 of our string “o”

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Generating sort cont.

- The same techniques can be applied to get “r” from true
- [!![]+[]][+[]][+!![]] == “r”
- And [!![]+[]][+[]][+[]] == “t”
- Put it all together and what have you got?
- [![]+[]][+[]][++[[]][+[]]+[++[[]][+[]]][+[]]+[++[[]][+[]]][+[]]]+[[]+{}][+[]][+!![]]+[!![]+[]][+[]][+!![]]+[!![]+[]][+[]][+[]] == ‘sort’
- Using the trick from before we can use an array to get window
- ([],[][[![]+[]][+[]][++[[]][+[]]+[++[[]][+[]]][+[]]+[++[[]][+[]]][+[]]]+[[]+{}][+[]][+!![]]+[!![]+[]][+[]][+!![]]+[!![]+[]][+[]][+[]]])() // kaboom! Window on IE9!

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Calling alert(1)

- Making “a”, “l”, “e” follows the same process, we already have “r” and “t”
- `[![]+[]][+[]][++[[]][+[]]]` // “a” 1st position from false
- `[![]+[]][+[]][++[[]][+[]]+[++[[]][+[]]][+[]]]` // “l” 2nd position from false
- `[![]+[]][+[]][++[[]][+[]]+[++[[]][+[]]][+[]]+[++[[]][+[]]][+[]]+[++[[]][+[]]][+[]]]`
  // “e” 4th position from false
- Lets combine it all together
-
```
([],[][[![]+[]][+[]][++[[]][+[]]+[++[[]][+[]]][+[]]+[++[[]][+[]]][+[]]]+[[]+{}][+[]][+!![]]+[!![]
+[]][+[]][+!![]]+[!![]+[]][+[]][+[]]])()[[![]+[]][+[]][++[[]][+[]]]+[![]+[]][+[]][++[[]][+[]]+[++
[[]][+[]]][+[]]]+[![]+[]][+[]][++[[]][+[]]+[++[[]][+[]]][+[]]+[++[[]][+[]]][+[]]+[++[[]][+[]]][+
[]]]+[!![]+[]][+[]][+!![]]+[!![]+[]][+[]][+[]]](+!![])
// alert(1)
```

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Executing arbitrary JavaScript

- Generating every single character is hard work
- Using Function you can generate a character from a octal escape
- “a” looks like \141 in octal
- Function('return"\\141"'); // returns “a”
- We need the letters in “Function”, “return” and generate the required octal number
- The conversion function converts the input into octal
- A reference to window is added, along with the a range of numbers 0-9
- Any code can be executed using this method

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

The great char wall

- How many characters are required to execute non-alphanumeric code?
- 6 characters. ()[]!+
- The great char wall was named by me and sirdarckcat as a impossible barrier to break
- We tried many many times
- You need “(“ and “)” to call functions “[“ and “]” to access string indexes and access properties and + to concat and convert into a number. “!” is important because you can generate true or false
- Can you break the char wall?

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Attempts to break the wall

- Many JavaScript warriors have tried and failed to break the wall
- Without “!” you can’t get true or false with “[]+()”
- Without “(“ and “)” you can’t get window
- Without “p” and “_” you can’t get properties such as __parent__ on older browsers
- The wall is an impossible problem that cannot be solved without some sort of JavaScript quirk
- I tired and failed to break it again

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

A non-alpha demo

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Decoding non-alpha code

- Properties are unknown and are calculated at run time
- E.g. Obj[x] but we don’t know “x” until the script runs
- You could convert basic patterns such as +[] to zero etc but subtle variations in the code could bypass this
- We don’t know the property and we don’t know the object either
  e.g. !obj true or false? We don’t know

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

How can you decode then?

- Sandbox the JavaScript code
- Proxy any calls to native functions and observe the result
- Remember Function?
- If we can change “Function” then we can have the decoded result
- The sandboxed code runs as normal but in a fake environment that we control

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Decode demo

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

How decoding works

- Uses free JavaScript sandbox that I developed called JSReg

```
parser.extendWindow("$sandbox$", function(code){
var js = JSReg.create(), result;
js.setDebugObjects({doNotFunctionEval:true,functionCode:
   function(code) {
code = code.replace("J.F();var $arguments$=J.A(arguments);",'');
result = code;
}});
try {
js.eval(code);
} catch(e){
return e;
}
return result;
});
```

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

How decoding works cont.

- Extends the environment with a custom sandbox function
- Changes JSReg behaviour to only run Function but not execute the code
- Removes any unneeded sandboxing code that remains
- JSReg does the rest ☺

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Fooling the decoder

- If you can detect you’re in a sandboxed environment you could alter the code’s behaviour
- You could break the sandbox
- You could find a syntax quirk inside the sandbox to prevent execution
- You could use eval instead of Function
- You could use a browser DOM object

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Defending the sandbox

- You could change eval and related Functions behaviour and log the result
- Provide a fake DOM object that an attacker thinks is real
- I challenge awesome security researchers on sla.ckers to break JSReg to prevent sandbox escapes and syntax problems
- Make the environment seem real by overwriting toString/valueOf of every native object/function to return the expected result

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Is non-alpha code evil?

- Without testing the boundaries of what is possible we cannot hope to provide adequate defences
- The attacker could figure it out anyway
- Anyone researching malicious non-alpha code will be forced to use tools that can decode the data
- Improves tools

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

PHP Non-alpha

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

You can do this in PHP?

- I wanted to emulate this stuff in PHP
- Nobody thought it was possible
- PHP lacks the toString/valueOf properties of JavaScript
- How can you generate characters from nothing?

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Thinking about non-alpha PHP

- PHP does a string conversion for arrays and results in “Array”
- Maybe we can use those characters?
- Hmmm what can we call with that? Not much
- How can we generate other characters from Array to enable us to execute code

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Bitwise operators on strings

- I figured out that PHP allows bitwise operators on strings
- A | B == C
- Generating “_” is difficult though
- Using two or more operations can result in different characters. E.g. Generate “C” use “C” to generate “D” etc

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Process for gen underscore

- Create array, concat with self to string, for loop to find char to xor, second loop to gen first char with A
- $§[]=$§; //creates an array
- $§=$§.$§; // converts to string ArrayArray
- $§§=+$§; // converts to zero
- $§[+$§§++]// gets “A” from the string
- $§[$§§+$§§+$§§]// gets “A”
- “A” | (XOR “a” with 0x7f)
- Result: $§[]=$§;$§=$§.$§;$§§=+$§;$§[+$§§++]|($§[$§§+$§§+$§§]^0x7f);
- 0x7f is the literal character

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Clever but...

- Stefan Esser (security god of PHP) said nice but why not use ++ or --
- Increment/decrement works on strings!
- $x='a';$x++;echo $x; // “b”
- Generating underscore was fun but it is smarter to use ++ or --

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Calling print “hello”

```
<?php
    $§[]=$§;$§=$§.$§;$Ƨ=+$§;$Ɣ=$Ƨ;$Ɣ++;$ɣ=$Ɣ+$Ɣ;$Ϭ=$ɣ+$Ɣ;$ϭ=$Ϭ+$Ɣ;$
    Ϯ=$ϭ+$Ɣ;$ϯ=$Ϯ+$Ɣ;$ϰ=$ϯ+$Ɣ;$ϱ=$ϰ+$Ɣ;$ï=$ϱ+$Ɣ;$Ұ=$§[$Ƨ]|($§[$Ϭ]^□);$ұ
    =$§[$Ɣ];$Ӥ=$§[$Ƨ]|($§[$Ɣ]&â);$ӥ=$§[$ï+$Ɣ];$Ö=$Ұ^($ϰ.ӥ);$ö=$Ӥ.$Ö.$ұ;$
    ϴ=$ö($ï.$ϰ).$ö($Ɣ.$Ɣ.$Ϯ).$ö($Ɣ.$Ɣ.$Ϯ).$ö($Ɣ.$Ƨ.$Ɣ).$ұ.$ö($Ɣ.$Ɣ.$ϯ);$ϴ($
    ö($Ɣ.$Ɣ.$ɣ).$ö($Ɣ.$Ɣ.$ϭ).$ö($Ɣ.$Ƨ.$Ϯ).$ö($Ɣ.$Ɣ.$Ƨ).$ö($Ɣ.$Ɣ.$ϯ).$ö($Ϭ.$
    ɣ).$ö($Ϭ.$ϭ).$ö($Ɣ.$Ƨ.$ϭ).$ö($Ɣ.$Ƨ.$Ɣ).$ö($Ɣ.$Ƨ.$ϱ).$ö($Ɣ.$Ƨ.$ϱ).$ö($Ɣ.
    $Ɣ.$Ɣ).$ö($Ϭ.$ϭ).$ö($Ϯ.$ï));
?>
```

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Generating non-alpha PHP

- Generate letters and numbers required
- Assert == eval in PHP
- Use chr to generate require letters
- Convert every character into their charcode then use chr to generate and assert to call

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

PHP Demo

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Thanks

- Yosuke Hasegawa ,Stefan Esser, Mario Heiderich, @Sirdarckcat, @Lever_one, @thornmaker, @rvdh, @oxotnick, @SW, @theharmonyguy and all my sla.ckers friends

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```

Questions?

```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[
$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$
=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.
$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\
"+$.__$+$.$$_+$._$_+$.__+"("+$.__$+")"+"\"")())();
```