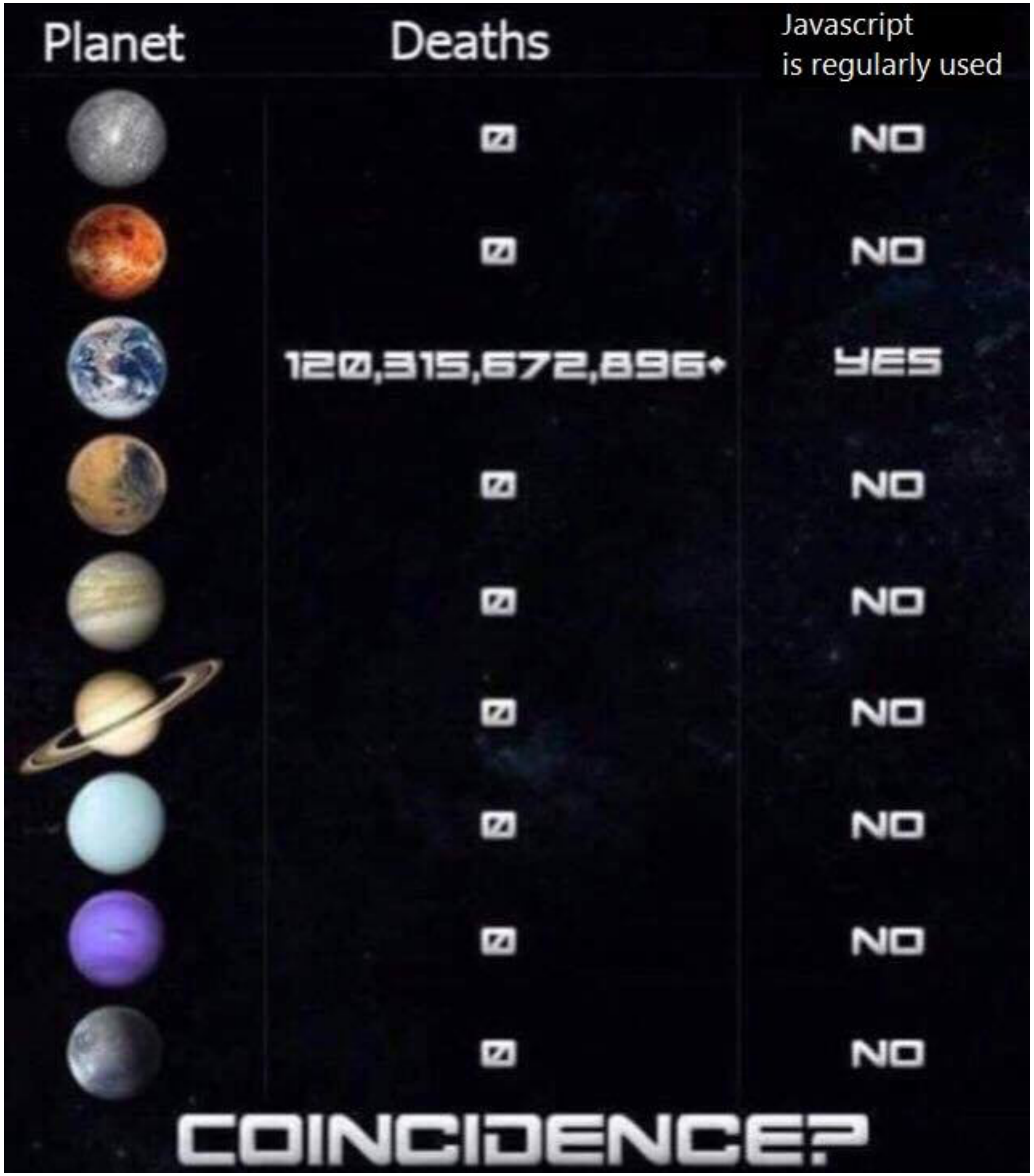
Imperial College
London

Network and Web Security

JavaScript

Dr Sergio Maffeis
Department of Computing
Course web page: https://331.websec.fun

Imperial College London



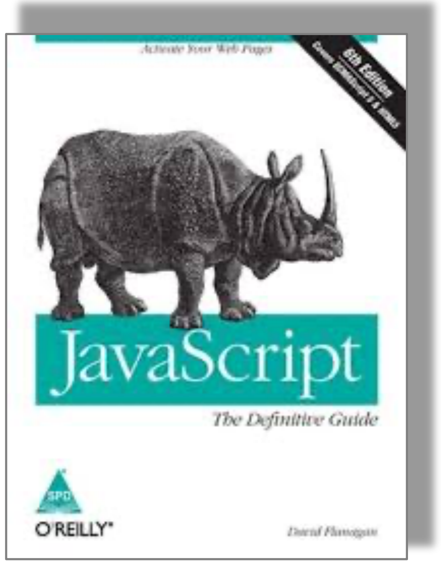
Planet	Deaths	Javascript is regularly used
	0	NO
	0	NO
	120,315,672,896+	YES
	0	NO
	0	NO
	0	NO
	0	NO
	0	NO
	0	NO

COINCIDENCE?

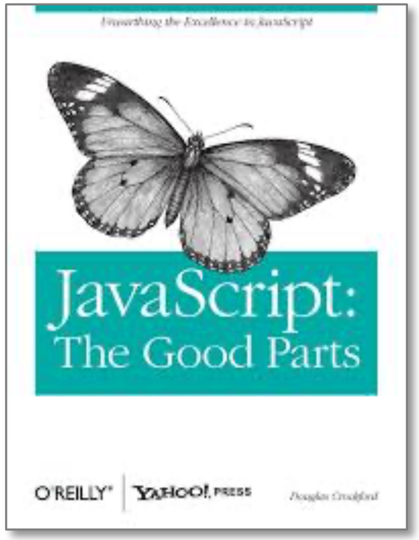
JavaScript

Imperial College London

- 1995: a small language to validate web form inputs in the browser (Brendan Eich)
- 2021: “Language of the web”, and more
  - All major browsers
  - On the server: Node.js
  - Smartphones: React Native
  - Desktop apps: Electron
- Powerful and dangerous
  - Easy to make mistakes: most examples of injection and XSS are in JavaScript
  - Most browser-based malware is JavaScript code, or at least installed by it
- **Goals**
  - **Understand how a web page works, analyse and fix its vulnerabilities**
  - **Analyse JavaScript malware**
- Non-goal
  - Become a proficient JavaScript programmer



1,096 pages



172 pages

JavaScript features

Imperial College London

- Objects as mutable records of functions with implicit `this`:

```
o = {b:function(){return this.a}};
```

- Prototype-based object inheritance:

```
Object.prototype.a = "foo";
```

- Implicit type conversions, that can be redefined.

```
Object.prototype.toString = o.b;
```

- Can convert strings into code:

```
eval("o + o['b']()"); // returns "foofoo"
```

Variables and scope

Imperial College London

- The scope can be manipulated like a language object:

```
window.o === o; // global scope
var s = {x:41};
with (s) {s.x++; console.log(x);} // local scope
```

- Nested scoping of functions (does not happen in PHP!)

```
x=1; y=2;
function a(z){var y=z+x;
              function b(w){return w+x+y}
              return b(z)}
a(20) // returns 42
```

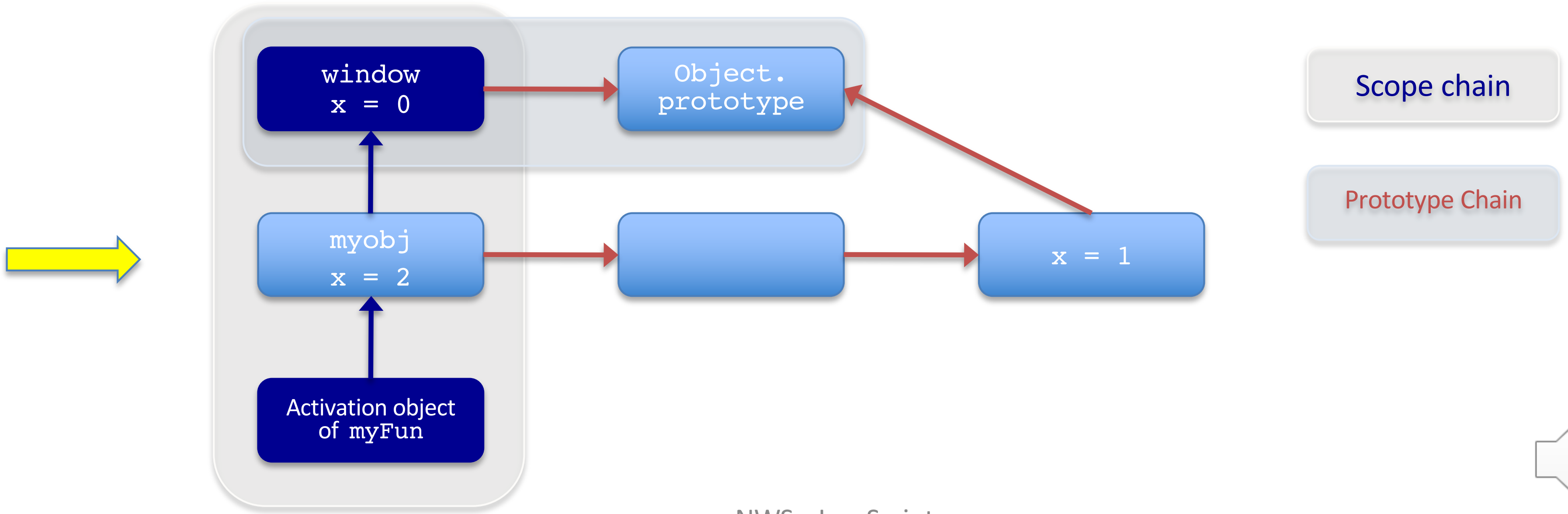
- Can encapsulate scope via function closures

```
var API = (function(){
     var x=13;
     return [function(y){return x+y;},
             function(z){x=z; return x;}]
          )();
API[1](API[0](29)); //returns 42
```

Scopes and prototypes

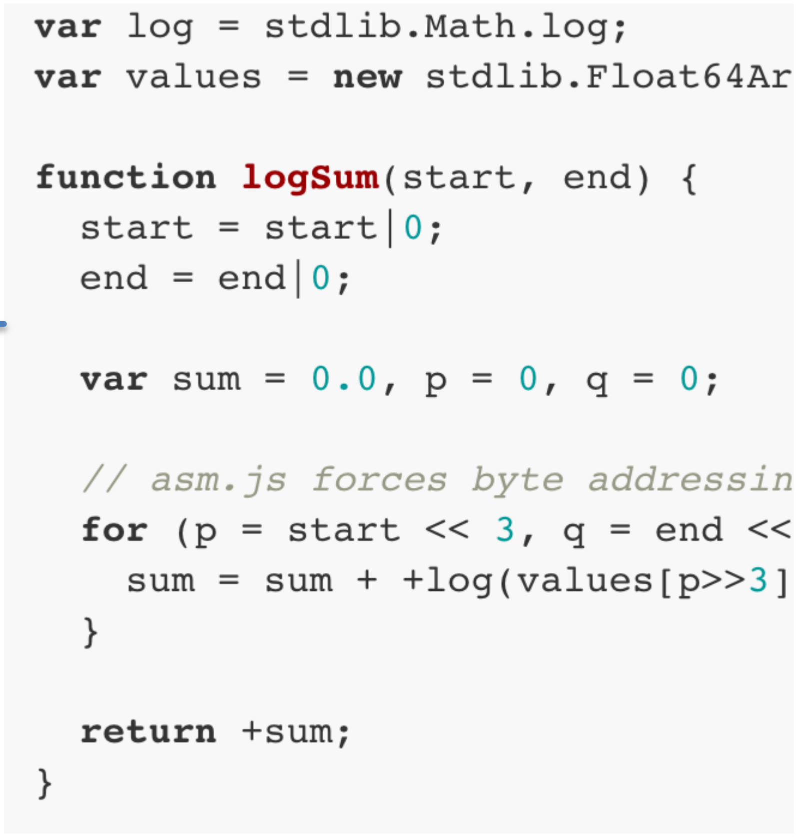
- Variable `x` is resolved as property `x` of the current scope object.
  - If `x` is not present, look in the parent scope object.
- Expression `myObj.x` evaluates to the property `x` of object `myObj`.
  - If `x` is not present, look in the prototype of `myObj`.
- Example:

```
with(myObj){myFun = function(){return x+=1};};
myFun();
```



JavaScript compilation

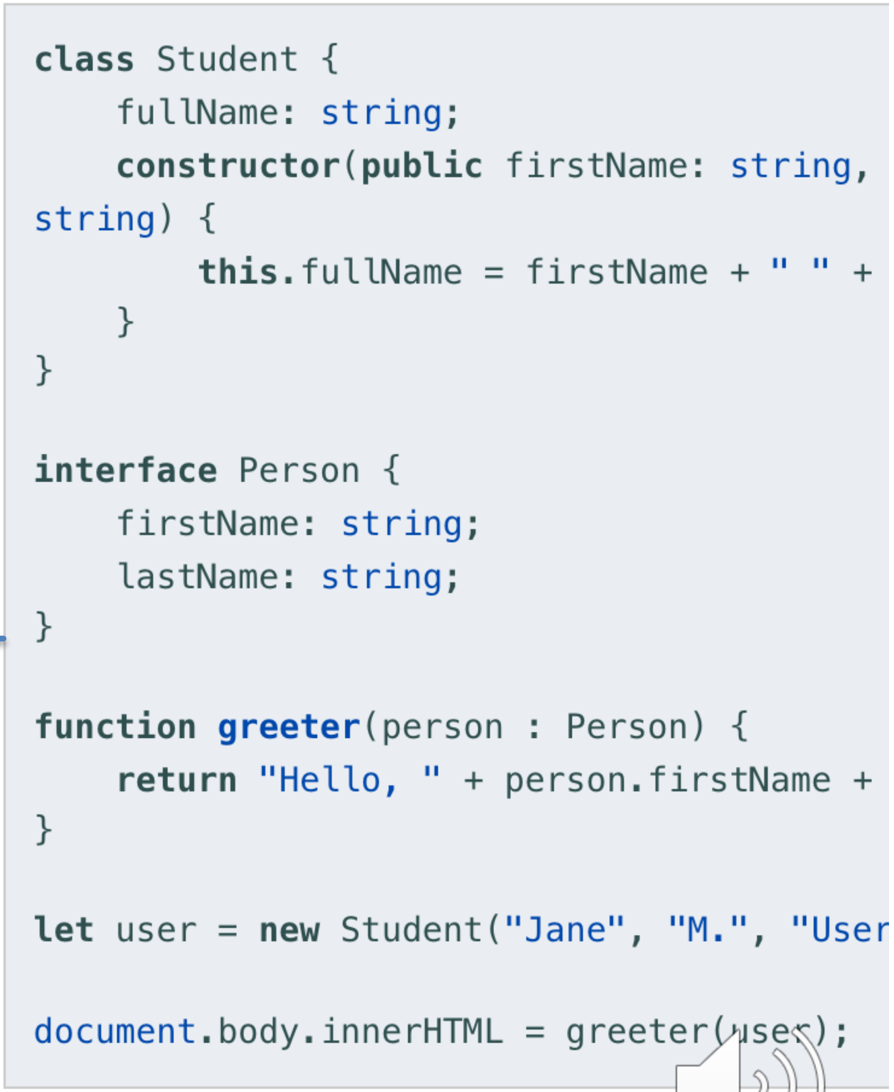
- JavaScript compilers
  - In principle JavaScript is an interpreted language
  - Main engines use bytecode and *just-in-time* (JIT) compilation to machine code
  - Optimising compilers: IonMonkey (Mozilla), Crankshaft (Google)
- asm.js
  - Fast subset of JavaScript, close to machine code
    - No nested functions, no objects
    - Main data structure are typed-arrays
    - All values are Int, Double, Float (signed/unsigned)
    - Roll-your-won memory management!
- WebAssembly (wasm)
  - Portable size- and load-time-efficient binary format suitable for compilation to the web
  - Aims for native speed
  - No longer JavaScript: think C/C++ for the web, interoperable with JavaScript
- Emscripten
  - Compiles any LLVM bitcode to asm.js, wasm



```
var log = stdlib.Math.log;
var values = new stdlib.Float64Ar

function logSum(start, end) {
  start = start|0;
  end = end|0;

  var sum = 0.0, p = 0, q = 0;

  // asm.js forces byte addressin
  for (p = start << 3, q = end <<
    sum = sum + +log(values[p>>3]
  }

  return +sum;
}
```

Frameworks and types

- JavaScript frameworks
  - jQuery, Angular, EmberJs, Mocha, React…
  - Wrap DOM and other common interfaces (AJAX)
  - Provide convenient syntactic sugar and programming patterns
  - Facilitate unit-testing, portability
- TypeScript
  - Statically typed, class-based superset of JavaScript
    - Best effort typing, no general soundness guarantee
  - Compiled down to JavaScript, hence fully compatible
  - Originated by Microsoft
- Flow
  - Facebook’s answer to TypeScript
  - Static type checking and type inference for JavaScript



```
class Student {
    fullName: string;
    constructor(public firstName: string,
string) {
        this.fullName = firstName + " " +
    }
}

interface Person {
    firstName: string;
    lastName: string;
}

function greeter(person : Person) {
    return "Hello, " + person.firstName +
}

let user = new Student("Jane", "M.", "User

document.body.innerHTML = greeter(user);
```

JavaScript transformations

- Minification
  - The source code of scripts is sent over the Internet
  - The length of a script affects latency of page loading and consumes network bandwidth
  - Scripts can be minified by removing comments, spaces, newlines, shortening identifiers, sharing constants, etc
- JavaScript implements server-mandated behaviour on the client
  - The source code of scripts is available for inspection
  - The algorithm implemented by a script may constitute intellectual property
  - The script may be malicious and may contain pointers to attacker-controlled assets (domains, IPs, keys)
  - Scripts can be obfuscated to hinder analysis, reverse engineering and detection
- Study from 2019: 37% of scripts from Alexa top 100k are minified, 1% obfuscated

Obfuscation examples

- String array

```
const strArr = [’sc’, ’ty’, ’chec’, ’te’, ’ your car’, ’a’, ’xt/jav’, ’lert("Got’,
    ’ls")’, ’pe’ , ’ript’, ’d detai’, ’kout’ ]
var malicious = document.createElement(strArr[0] + strArr[10]);
var attr = strArr[1] + strArr[9];
var attr2 = strArr[3] + strArr[6] + strArr[5] + strArr[0] + strArr[10]
malicious.setAttribute(attr, attr2);
var node = strArr[5] + strArr[7] + strArr[4] + strArr[11] + strArr[8];
var maliciousCode = document.createTextNode(node);
malicious.appendChild(maliciousCode);
document.getElementById(strArr[2] + strArr[12]).appendChild(malicious);
```

- String manipulation

```
var cmd = "var m" + "alicio" + "us = d" + "ocumen" + "t[’cre" + "ateEle" + "ment’]"
     + "(’scri" + "pt’);m" + "alicio" + "us.set" + "Attrib" + "ute(’t" + "ype’, " +
     "’text/" + "javasc" + "ript’)" + ";var m" + "alicio" + "usCode" + " = doc" + "
    ument." + "create" + "TextNo" + "de(’al" + "ert(\"" + "Got yo" + "ur car" + "d
    deta" + "ils\")" + "’);mal" + "icious" + ".appen" + "dChild" + "(malic" + "
    iousCo" + "de);do" + "cument" + ".getEl" + "ementB" + "yId(’c" + "heckou" + "t
    ’).ap" + "pendCh" + "ild(ma" + "liciou" + "s);"
eval(cmd)
```

Obfuscation examples

- String encoding

```javascript
var malicious = document['\x63\x72\x65\x61\x74\x65\x45\x6c\x65\x6d\x65\x6e\x74']('\x73\x63\x72\x69\x70\x74');
malicious['\x73\x65\x74\x41\x74\x74\x72\x69\x62\x75\x74\x65']('\x74\x79\x70\x65', '\x74\x65\x78\x74\x2f\x6a\x61\x76\x61\x73\x63\x72\x69\x70\x74');
var maliciousCode = document['\x63\x72\x65\x61\x74\x65\x54\x65\x78\x74\x4e\x6f\x64\x65']('\x61\x6c\x65\x72\x74\x28\x22\x47\x6f\x74\x20\x79\x6f\x75\x72\x20\x63\x61\x72\x64\x20\x64\x65\x74\x61\x69\x6c\x73\x22\x29');
malicious['\x61\x70\x70\x65\x6e\x64\x43\x68\x69\x6c\x64'](maliciousCode);
document['\x67\x65\x74\x45\x6c\x65\x6d\x65\x6e\x74\x42\x79\x49\x64']('\x63\x68\x65\x63\x6b\x6f\x75\x74')['\x61\x70\x70\x65\x6e\x64\x43\x68\x69\x6c\x64'](malicious);
```

- Identifier mangling

```javascript
var _0x2179ac = document['createElement']('script');
_0x2179ac['setAttribute']('type', 'text/javascript');
var _0x475631 = document['createTextNode']('alert(\x22Got\x20your\x20card\x20details\x22)');
_0x2179ac['appendChild'](_0x475631);
document['getElementById']('checkout')['appendChild'](_0x2179ac);
```

Obfuscation examples

Imperial College London

- Encryption obfuscation

```
var hash ='
    dmFyIG1hbGljaW91cyA9IGRvY3VtZW50WydjcmVhdGVFbGVtZW50J10oJ3NjcmlwdCcpO21hbGlja-
    W91cy5zZXRBdHRyaWJ1dGUoJ3R5cGUnLCAndGV4dC9qYXZhc2NyaXB0Jyk7dmFyIG1hbGljaW91c0N-
    vZGUgPSBkb2N1bWVudC5jcmVhdGVUZXh0Tm9kZSgnYWxlcnQoIkdvdCB5b3VyIGNhcmQgZGV0YWlsc-
    yIpJyk7bWFsaWNpb3VzLmFwcGVuZENoaWxkKG1hbGljaW91c0NvZGUpO2RvY3VtZW50LmdldEVsZW1-
    lbnRCeUlkKCdjaGVja291dCcpLmFwcGVuZENoaWxkKG1hbGljaW91cyk7'
eval(atob(hash))
```

- Combined example

```
var _0x2bd1=['\x20\x64\x65\x74\x61','\x61\x70\x70\x65\x6e','\x64\x43\x68\x69\x6c','
    \x67\x65\x74\x45\x6c','\x65\x6d\x65\x6e\x74','\x42\x79\x49\x64','\x63\x68\x65\
    x63\x6b','\x65\x45\x6c\x65\x6d','\x65\x6e\x74','\x73\x63\x72\x69\x70','\x74\x72
    \x69\x62\x75','\x74\x79\x70\x65','\x63\x72\x69\x70\x74','\x63\x72\x65\x61\x74',
    '\x65\x54\x65\x78\x74','\x4e\x6f\x64\x65','\x28\x22\x47\x6f\x74','\x20\x79\x6f\
    x75\x72','\x20\x63\x61\x72\x64'];(function(_0x5854a7,_0x40bdb0){var _0xe76549=
    function(_0x5f0118){while(--_0x5f0118){_0x5854a7['push'](_0x5854a7['shift']())
    ;}};_0xe76549(++_0x40bdb0);}(_0x2bd1 ,0x66));var _0x2ade=function(_0x5854a7,
    _0x40bdb0){_0x5854a7=_0x5854a7 -0x0;var _0xe76549=_0x2bd1[_0x5854a7];return
    _0xe76549;};var _0x40bdb0=document['\x63\x72\x65\x61\x74'+_0x2ade('0x0')+
    _0x2ade('0x1')](_0x2ade('0x2')+'\x74');_0x40bdb0['\x73\x65\x74\x41\x74'+_0x2ade
    ('0x3')+'\x74\x65'](_0x2ade('0x4'),'\x74\x65\x78\x74\x2f'+'\x6a\x61\x76\x61\x73
    '+_0x2ade('0x5'));var _0x38eee4=document[_0x2ade('0x6')+_0x2ade('0x7')+_0x2ade(
    '0x8')]('\x61\x6c\x65\x72\x74'+_0x2ade('0x9')+_0x2ade('0xa')+_0x2ade('0xb')+
    _0x2ade('0xc')+'\x69\x6c\x73\x21\x22'+'\x29');_0x40bdb0[_0x2ade('0xd')+_0x2ade(
    '0xe')+'\x64'](_0x38eee4);document[_0x2ade('0xf')+_0x2ade('0x10')+_0x2ade('0x11
    ')](_0x2ade('0x12')+'\x6f\x75\x74')['\x61\x70\x70\x65\x6e'+_0x2ade('0xe')+'\x64
    '](_0x40bdb0);
```

JavaScript obfuscation

- Obfuscation quality
  - Should be hard or impossible to deobfuscate
  - Should preserve the behavior of the script
    - And in particular not crash
  - Should not make the script too slow
- Many online tools available, but pros roll their own
  - Obfuscation: daftlogic, javascriptobfuscator, jfogs, jsfuck (?!), ...
  - Deobfuscation: jsnice, illuminate.js, jstillery, de4js, ...
- Active area of research
  - Detection: tends to be the easier part, but risk of false positives is high
  - Analysis: extract interesting information from malicious script (URLs, etc)
  - Deobfuscation: make the obfuscated file readable to a human
    - JSnice (from ETH) uses DL techniques to come up with sensible names for variables
  - Anti-reversing techniques