JavaScript – alternatives to eval()

The idea of eval

We use eval to work out the value of some code, in such a way as to be able to change the code programmatically, and work it out again.

In other words we want to be able to say

```
var exp = "x*y+5";
var x = 2;
var y = 3;
```

then get the code to work out 2*3+5. But we cannot just say

```
var z=x*y+5;
```

because at other times we want exp to be different – maybe "x+y" or whatever.

Eval does this – we just say

var z = eval(exp);

then whatever exp is, eval evaluates it.

This treats data (like "x*y+5") as if it were code. Treating data as code is one of the fundamentals of computer science.

The two problems with eval

The first is security. The string being eval'd is often inputted from the user. If they type in some malicious code, our JavaScript executes it.

The second is performance. Eval is supposed to be very slow.

This note looks at an alternative to eval, and tests the performance issue.

Using Function

JavaScript has a built-in object named Function, which provides features relevant to functions.

If we use it as a constructor (by saying .. =new Function..) it makes a new function. We usually write functions, as code. Here our code creates a new function as it executes. IOW data is code.

Function takes a variable number of arguments. The last one is the function body, and the first ones are arguments to that function. So, if we simply want a function that returns the value of some expression:

```
var exp = "x*y+5";
var x = 2;
var y = 3;

const func = new Function('x', 'y', 'return ' + exp + ';');

const result = func(x,y,exp);
```

We could change exp, and func would work out something else.

Performance of eval

Is eval really that slow? We can find out:

```
var exp = "x*y+5";
var x = 2;
var y = 3;
var start = performance.now();
```

```
for (var i = 0; i < 1000; i++) {
  const result = eval(exp);
}
var end = performance.now();
console.log("time taken in ms =", end - start); // 83.16499995999038

const func = new Function('x', 'y', 'return ' + exp + ';');
start = performance.now();

for (var i = 0; i < 1000; i++) {
  const result = func(x,y,exp);
}
end = performance.now();
console.log("time taken in ms =", end - start); // 0.1550000160932541
```

So, 83 ms for 1000 executions, against 0.155 for a Function. The times vary, depending on what else the machine is multi-tasking. But *eval is roughly 100 times or more slower than the use of a Function*.