Real time recompilation of running JavaScript

Peter van der Zee
JSConf.us 2013

Whois

Peter van der Zee
JavaScript developer
Netherlands

JS1K
ZeonJS
ZeParser
HeatFiler

qfox.nl
@kuvos

surfly.com
"Remote desktop in the browser"

Demonstration

Nothing to see here...

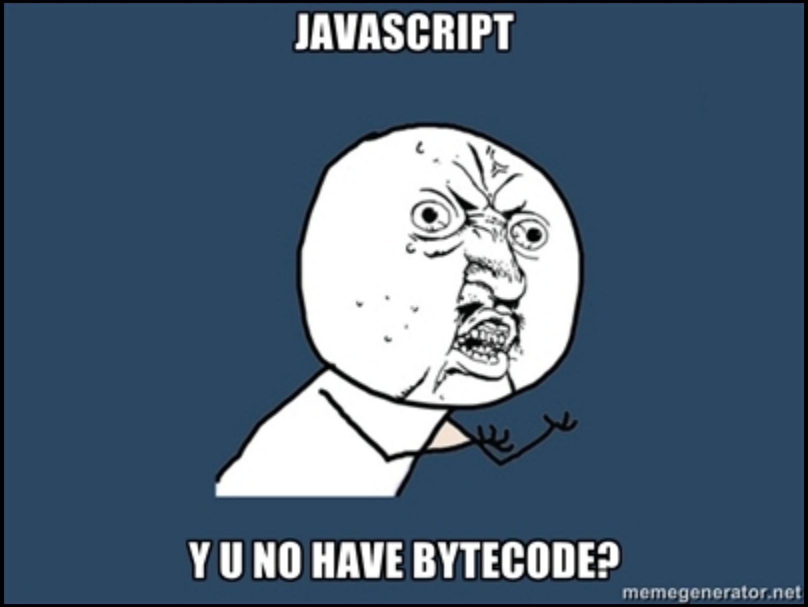
So...

- Compile JS on the fly
- Maintain access to closures
- No restart of app required

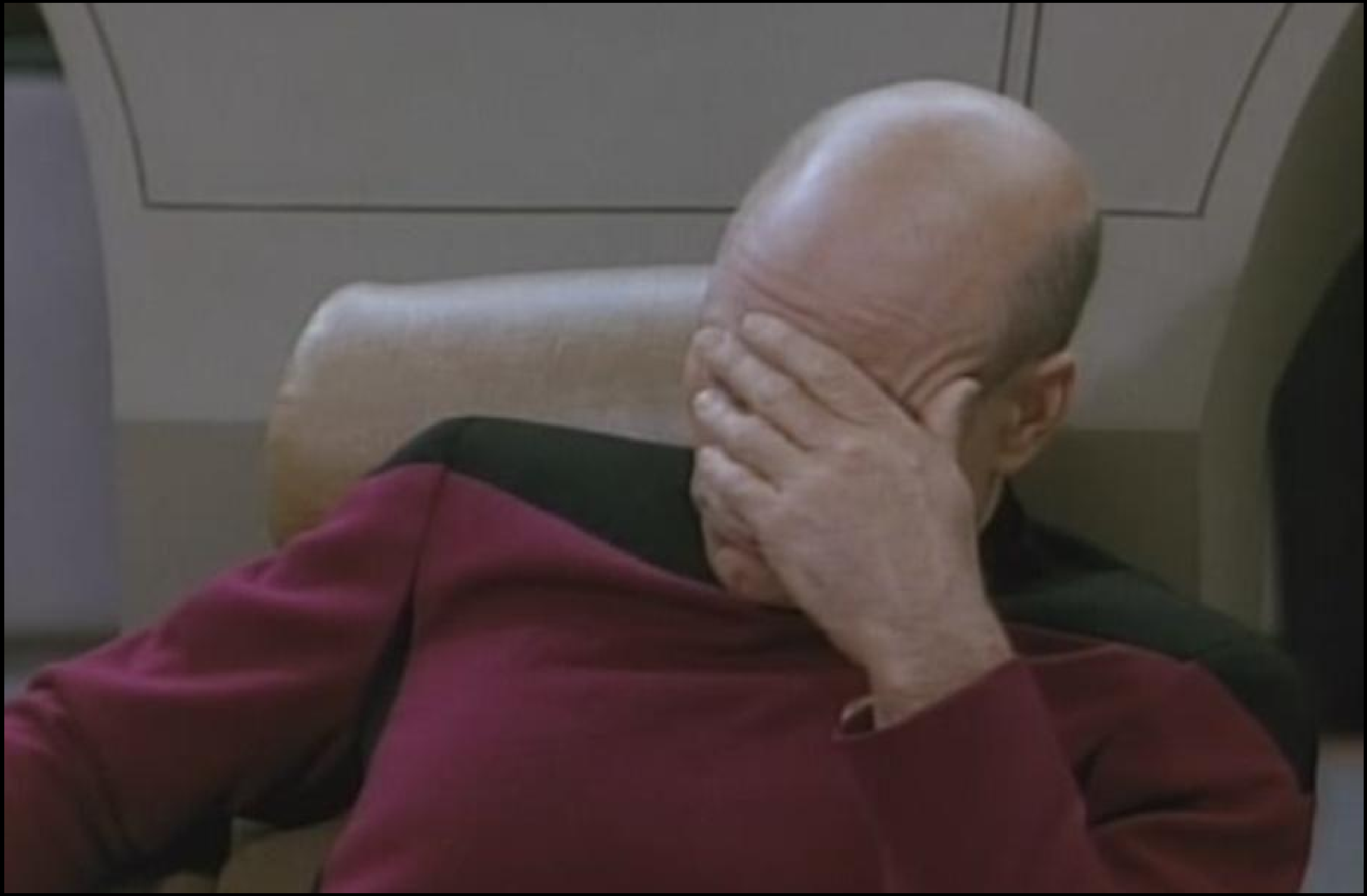
Features

- Lazy evaluation
- Access to variables in a closure
- Almost anything works

Compile in JS?



Live coding woohoo



Issues

- closures
- local variables
- function declarations
- named function expressions
- performance

Direct vs indirect eval

```
function f(evaller){
  var foo = 15;
  eval('foo'); // yep
  evaller('foo'); // nope
}
f(eval);
eval('foo'); // nope

Function === indirect eval
```

Basically

Indirect direct eval

```
function f(){
  var foo = 5;
  return function(){ log(foo); };
}
var g = f();
g(); // logs 5
```

Indirect *direct* eval

```
var $body = '(function(){ log(foo); })';
function f(){
  var foo = 5;
  return eval($body);
}
var g = f();
g(); // logs 5
```

Indirect direct eval

```
var $body = '(function(){ log(foo); })';
function f(){
  var foo = 5;
  return $get(1, eval);
}
function $get(id, evaller){
  return evaller($body);
}
var g = f();
g(); // error: foo is undefined
```

Indirect ***direct*** eval

```
var $body = '(function(){ log(foo); })';
function f(){
  var foo = 5;
  return $get(1);
}
function $get(id){
  return eval($body);
}
var g = f();
g(); // error: foo is undefined
```

Indirect direct eval

- Direct eval has access to scope
- Indirect access only access to global
- Functions can access parent scopes

Indirect direct eval

- Direct eval has access to scope
- Indirect access only access to global
- Functions can access parent scopes

=> Direct eval wrapped in function

Indirect direct eval

```
var $body = '(function(){ log(foo); })';
function f(){
  var foo = 5;
  return $get(
    1,
    function(s){ return eval(s); })
}
function $get(id, evaller){
  return evaller($body);
}
var g = f();
g(); // logs 5
```

Named function expressions

```
var foo = function(){
  console.log("hello "+world);
};
```

Named function expressions

```
var foo = function(){
  var $compiler = function(){
    return eval(arguments[0]);
  };
  console.log("hello "+world);
};
```

Named function expressions

```
var foo = function name(){
  console.log("hello "+world);
};
```

Named function expressions

```
var foo = (function(){
  var name = function(){
    var $compiler = function(){
      return eval(arguments[0]);
    };
    console.log("hello "+world);
  };
  return name;
})();
```

Named function expressions

```
setTimeout(function repeat(){
  ...
  setTimeout(repeat, n);
}, n);
```

Open issues

- Inserting new functions
- Variable clashes
- Hard to explain

Code

http://github.com/qfox/recompiler

Other magic

- Parameter tracking
- Source-to-output tracking
- Output-to-source tracking
- Modify source with UI

Screencast: http://vimeo.com/53017149

Parameter tracking

- For DSL

- Wrap symbols
`` `new Rect(..) -> sym(new Rect(..))` ``

- Wrap params
`` `-> new Rect(param(...),..)` ``

Source-to-output tracking

Symbols

- have a unique source id
- have (unique) source range
- can generate multiple instances

Output-to-source tracking

- Special mouse events
- Find source range for element

Modify source with UI

1. Lookup source range for object
2. Carefully (!) modify existing expression
3. Recompile result