DANIEL TULL

MAKING SCRIPTABLE APPS

WHY PROVIDE SCRIPTING?

Empower users

Enable features you don’t want to actually build

Allow you to debug the app on device

Trigger actions for testing

SCRIPTING LANGUAGE

Swift

Javascript

Python

AppleScript

JAVASCRIPT

Familiar to many

Very “forgiving”

JavaScriptCore included on iOS and macOS

Available on other platforms

JAVASCRIPTCORE

Objective-C wrapper around WebKit's JavaScript engine.

Insert custom objects into the environment.

JAVASCRIPTCORE

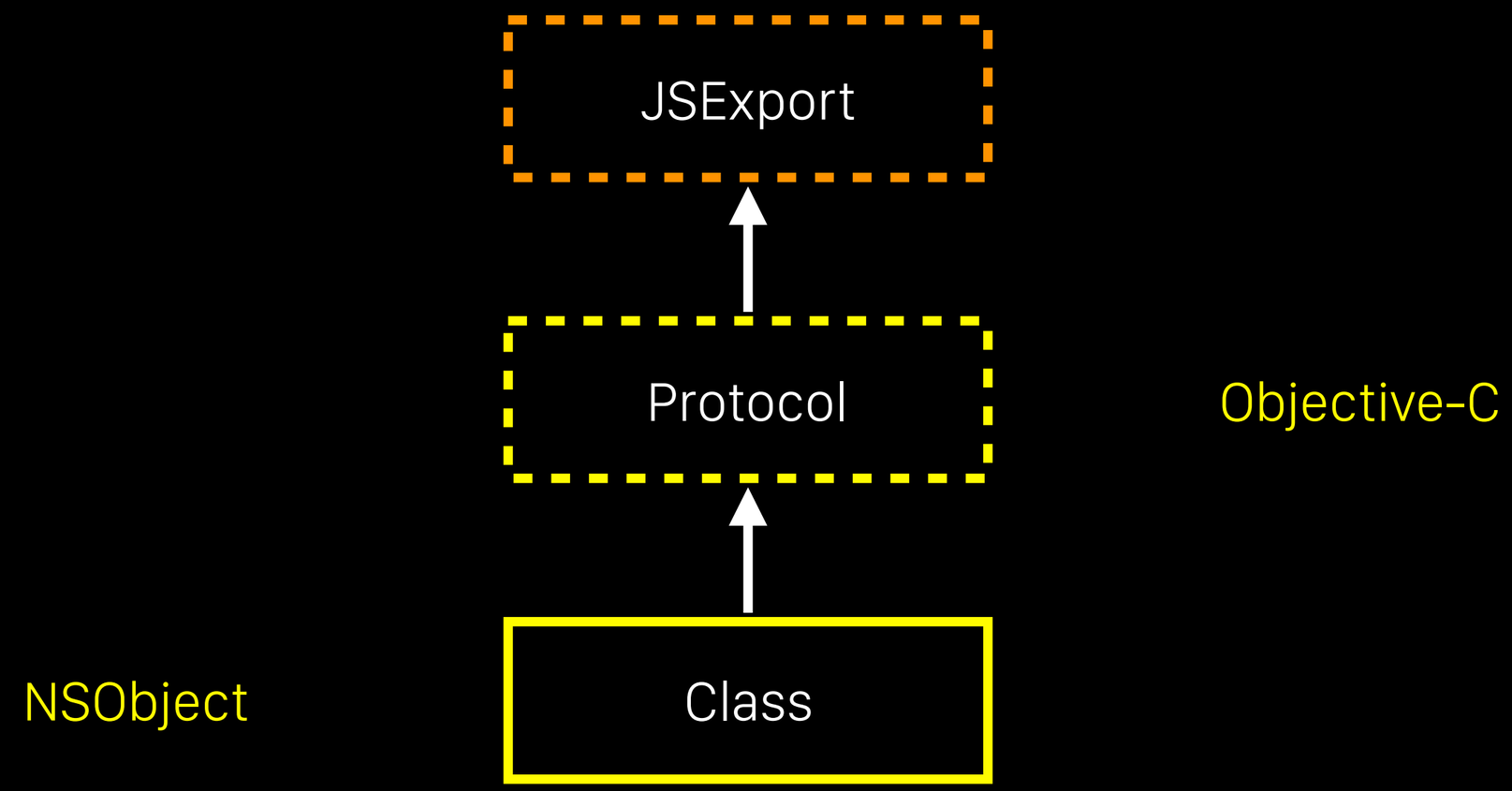
`JSContext` A JS execution environment

`JSValue` Conversion between JS and Objective-C types

`JSExport` A protocol to export Objective-C classes, methods and properties to JS

EXPORTING TYPES

EXPORTING TYPES



SWIFT TYPE

```
struct Position {

  var x: Double
  var y: Double

  init(x: Double, y: Double) {
    self.x = x
    self.y = y
  }
}
```

EXPORT PROTOCOL

```
@objc protocol PositionExport: JSExport {

  init(x: Double, y: Double)

  var x: Double { get set }
  var y: Double { get set }
}
```

OBJECTIVE-C WRAPPER

```
final class JSPosition: NSObject {

  var position: Position

  init(_ position: Position) {
    self.position = position
  }
}
```

IMPLEMENT PROTOCOL

```
extension JSPosition: PositionExport {

  convenience init(x: Double, y: Double) {
    let position = Position(x: x, y: y)
    self.init(position)
  }

  dynamic var x: Double {
    get { return position.x }
    set { position.x = newValue }
  }
```

EXPORTING TYPES

```
let context = JSContext()
context.setObject(JSPosition.self, for: "Position")
context.evaluateScript(script)
```

JAVASCRIPT

```
var position = new Position(0,0);
position.x = 13;
position.y = 42;
```

SOURCERY

Scans Swift code, generates files

Created by Krzysztof Zabłocki (@merowing)

SOURCERY

```
init?({% for parameter in method.parameters where parameter.name != "identifier" %}
{{ parameter.name }}: {% if parameter.isArray %}{% if
parameter.typeName.array.elementType.implements.JSGenerate %}
[JS{{ parameter.typeName.array.elementTypeName.unwrappedTypeName |
replace:".","" }}]?{% elif
parameter.typeName.array.elementTypeName.unwrappedTypeName == "Positive" or
parameter.typeName.array.elementTypeName.unwrappedTypeName == "Percentage" %}
[Double]?{% elif parameter.typeName.array.elementTypeName.unwrappedTypeName ==
"String" or parameter.typeName.array.elementTypeName.unwrappedTypeName == "Double"
or parameter.typ                                                          ool" %}
[{{ parameter.ty                                                          {% endif %}{%
else %}{% if parameter.type.implements.JSGenerate %}
JS{{ parameter.actualTypeName.unwrappedTypeName | replace:".","" }}?{% elif
parameter.actualTypeName.unwrappedTypeName == "Positive" or
parameter.actualTypeName.unwrappedTypeName == "Percentage" %}Double{% elif
parameter.actualTypeName.unwrappedTypeName == "String" or
parameter.actualTypeName.unwrappedTypeName == "Double" or
parameter.actualTypeName.unwrappedTypeName == "Bool" %}
{{ parameter.actualTypeName.unwrappedTypeName }}{% endif %}{% endif %}{% if not
forloop.last %}, {% endif %}{% endfor %})
```

```
init(x: Double, y: Double)
```

GOTCHAS

NOT A NUMBER

```
var position = new Position();

JSPosition(x: NaN, y: NaN)

Position(x: NaN, y: NaN)
```

Possibly unexpected?

```swift
struct Area {

  var position: Position
  var size: Size

  init(position: Position, size: Size) {
    self.position = position
    self.size = size
  }
}
```

NIL ON NON OPTIONALS!

```
var area = new Area();

JSArea(position: nil, size: nil)

Area(position: nil, size: nil)
```

Crash

SANITISE INPUT

```
init?(jsposition: JSPosition?, jssize: JSSize?) {

  guard
    let jsposition = jsposition,
    let jssize = jssize
  else {
    return nil
  }

  let position = Position(jsposition)
  let size = Size(jssize)
  let area = Area(position: position, size: size)
  self.init(area)
}
```

INCORRECT TYPES

```
var size = new Size(10, 20);

var area = new Area(size, size);

JSArea(position: size, size: size)

Area(position: size, size: size)
```

Crash

SANITISE INPUT

```
init?(jsposition: Any?, jssize: Any?) {

    guard
        let jsposition = jsposition as? JSPosition,
        let jssize = jssize as? JSSize
    else {
        return nil
    }

    let position = Position(jsposition)
    let size = Size(jssize)
    let area = Area(position: position, size: size)
    self.init(area)
}
```

```swift
extension Area {
  var center: Position {
    let x = position.x + size.width/2
    let y = position.y + size.height/2
    return Position(x: x, y: y)
  }
}
```

```
extension JSArea {
  var center: JSPosition {
    return JSPosition(area.center)
  }
}
```

SETTING READ-ONLY PROPERTIES

```
var position = new Position(0, 0);

var size = new Size(100, 100)

var area = new Area(position, size);

area.center = new Position(5, 5);
```

Nothing…

JAVASCRIPT EXCEPTIONS

```
extension JSArea {
  var center: Any? {
    get {
      return Position(area.center)
    }
    set {
      let context = JSContext.current()
      let message = “center is not settable”
      context.exception =
JSValue(newErrorFromMessage: message, in: context)
    }
}
```

THREADING

Calls to your code made on the thread of the JSContext.

That thread is whatever one you instantiate the JSContext on.

Maybe best not to block the main thread.

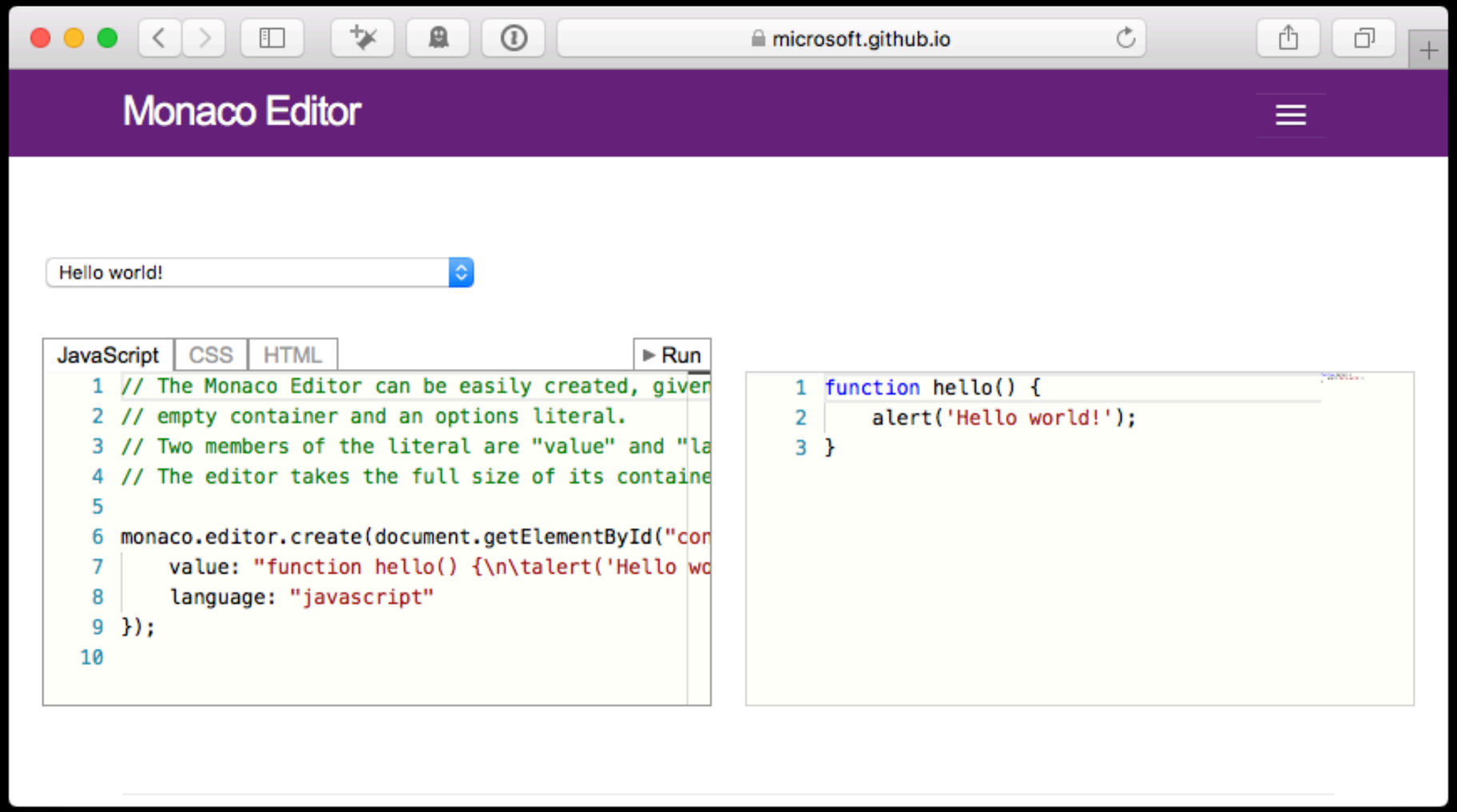
CODE EDITOR

UITEXTVIEW

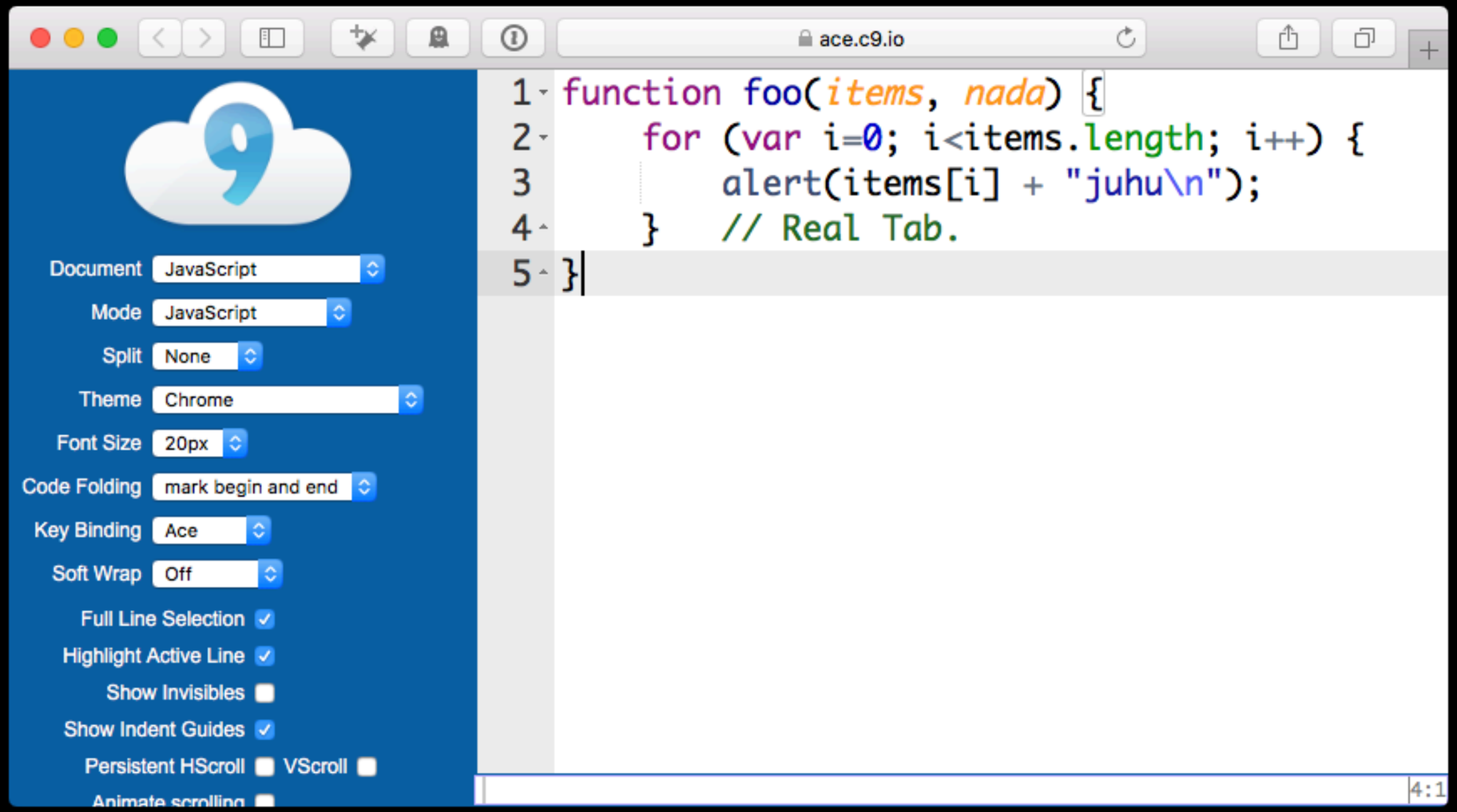
UITextView — Edited

```
function hello() {
    alert('Hello World!');
}
```

MONACO EDITOR



ACE


ace.c9.io
Document JavaScript
Mode JavaScript
Split None
Theme Chrome
Font Size 20px
Code Folding mark begin and end
Key Binding Ace
Soft Wrap Off
Full Line Selection
Highlight Active Line
Show Invisibles
Show Indent Guides
Persistent HScroll
VScroll
Animate scrolling
1 function foo(items, nada) {
2 for (var i=0; i<items.length; i++) {
3 alert(items[i] + "juhu\n");
4 } // Real Tab.
5 }
4:1

CODEMIRROR

codemirror.net

CodeMirror

Home
Manual
Code

Language modes
JavaScript

JavaScript mode

```
// Demo code (the actual new parser character stream implementation)

function StringStream(string) {
  this.pos = 0;
  this.string = string;
}

StringStream.prototype = {
  done: function() {return this.pos >= this.string.length;},
  peek: function() {return this.string.charAt(this.pos);},
  next: function() {
    if (this.pos < this.string.length)
      return this.string.charAt(this.pos++);
  },
  eat: function(match) {
    var ch = this.string.charAt(this.pos);
```

JavaScript mode supports several configuration options:

- `json` which will set the mode to expect JSON data rather than a JavaScript

CODEEDITOR.HTML

```
<!DOCTYPE html>
<head>
<meta name="viewport" content="user-scalable=no,
width=device-width”>
<script src="CodeMirror/lib/codemirror.js"></script>
</head>
<body>

<textarea id="CodeEditor"></textarea>

<script src="CodeEditor.js"></script>

</body>
</html>
```

CODEEDITOR.JS

```
var editor =
CodeMirror.fromTextArea(document.getElementById(
"CodeEditor"), {
   lineNumbers: true,
   lineWrapping: true,
   mode: "javascript",
   matchBrackets: true,
   autoCloseBrackets: true
});
```

CODEEDITOR.SWIFT

```swift
webView.evaluateJavaScript("editor.getValue();")
{ (result, error) in

  guard let script = (result as? String) else {
    // Handle the error
    return
  }

  // Do stuff with the script
}
```

JUST ADD MORE

Language	files	blank	comment	code
JavaScript	84	5628	11323	43066
JSON	1	0	0	1929
CSS	11	107	24	747
HTML	4	34	0	257
Swift	2	44	21	121
Bourne Shell	5	27	12	66
C/C++ Header	1	2	0	3
SUM:	108	5842	11380	46189

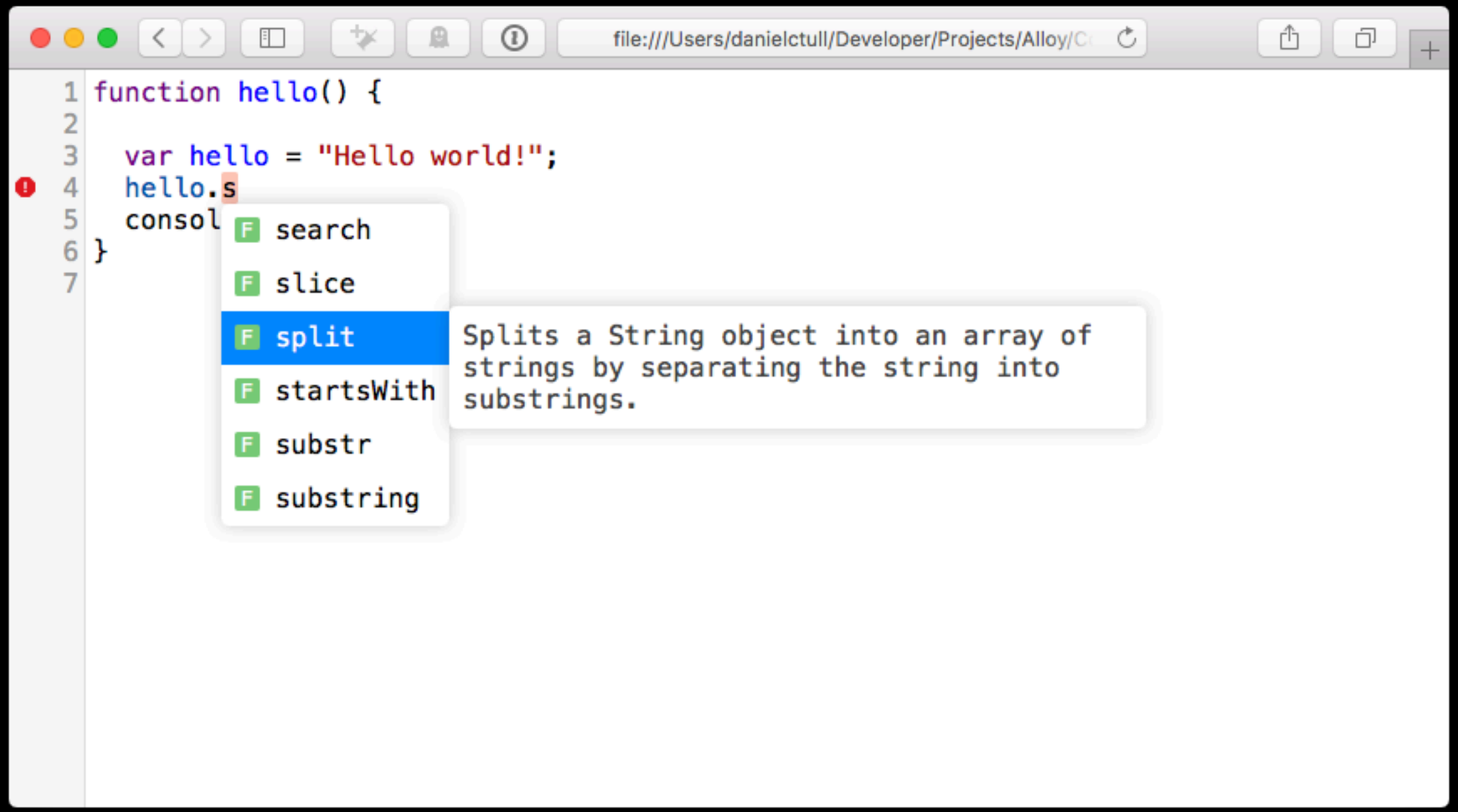
EDITOR


file:///Users/danielctull/Developer/Projects/Alloy/C
function hello() {
var hello = "Hello world!";
console.log(hello);
}

LINTING


file:///Users/danielctull/Developer/Projects/Alloy/C
function hello() {
Missing semicolon.
var hello = "Hello world!"
console.log(hello);
}

AUTOCOMPLETE



JSON DEFINITION

```
"Position": {
  "!type": "fn(x: number, y: number) -> Position",
  "prototype": {
    "x": "number",
    "y": "number"
  }
},
```

@DANIELCTULL

DANIELTULL.CO.UK