# JavaScript: The Good, The Bad, The Future

By Tay Yang Shun

## What this workshop is about

- JavaScript and what it can do
- Kickstart your learning of JavaScript
- **NOT** a crash course on JavaScript

## Introduction

- Also known as Mocha, LiveScript, JScript, ECMAScript
- Developed in an extremely short amount of time
- Scripting language
  - Not compiled, interpreted and executed on-the-fly
- Dynamic typing
- Supports object-oriented, imperative and functional programming styles
- Despite its name, it has nothing to do with Java

## What can JavaScript do?

- Provides interactivity to web applications
- Mainly executed in the browser to:
  - Manipulate DOM elements
  - Load content into the document without reloading
- Can be used in an application back end via NodeJS

## Getting Started

- No installation needed!
  - Unless you're using Node
- JavaScript comes with all browsers. Yes, even IE!
- Simply fire up your browser's console

## Language and Syntax

- Similar to C
- Learn it yourself ☺
- But beware of wats... (from 1:22 onwards)
- Read more about JS quirks here

## Points to Note

- Case-sensitive
  - `getElementById !== getElementByid`
- Semicolons are optional
  - But please add them
- Variable scoping
  - Blocks do not have scope
  - Only functions do
  - Beware of global variables

## Use Your Semicolons

- JS parser does Automatic Semicolon Insertion (ASI)

```
// Before ASI
a = b + c
foo()
// After ASI
a = b + c; foo() // All is good
```

- However, ASI is only applied if the parser needs to do so in order to make sense of the code in question

## Use Your Semicolons

- However, if your code looks like this...

```
// Before ASI
a = b + c
[1].push(a)
// After ASI
a = b + c[1].push(a) // KABOOM!
```

- ASI is not applied because without the semicolon the code still makes sense
- Just use your semicolons. Thanks

We noticed you've been missing a lot of semicolons lately, Peter...
I wouldn't say I've been missing them Bob.
But seriously, just put in the fucking semicolons.

## Variable Scoping

- Declaring variables without the `var` keyword:

```
> var foo = function () { bar = 1; }
> foo();
> console.log(bar); // bar is now a global var
1
```

- You almost never want to use globals

## Closures

- Closures are functions that refer to independent variables.
- The function defined in the closure 'remembers' the environment in which it was created.

```
function makeFunc () {
  function displayName() {
    alert(name);
  }
  var name = 'CS3216 Rocks!';
  return displayName;
}
var myFunc = makeFunc();
myFunc();
```

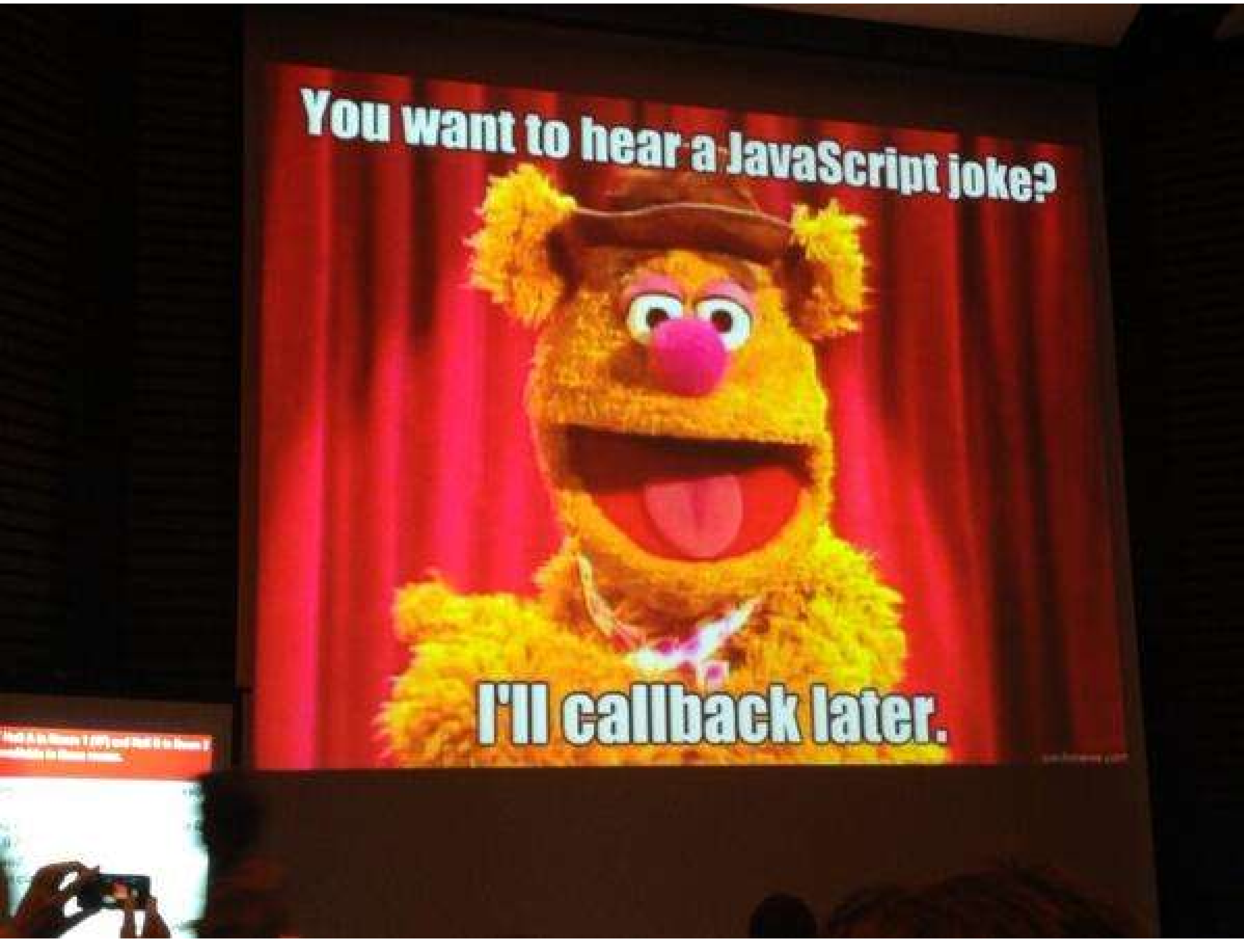
## Closures

- A function factory can create closures with different environments

```
function makeAdder (x) {
  return function (y) {
    return x + y;
  };
}
var add5 = makeAdder(5);
var add10 = makeAdder(10);
console.log(add5(2));  // 7
console.log(add10(2)); // 12
```

## Callbacks

- Callback functions are derived from functional programming
- A function is passed into another function as a parameter
- Callback functions are used in:
  - Asynchronous executions
  - In Event Listeners/Handlers
  - In `setTimeout` and `setInterval` methods
- Read more on JavaScript callbacks here


You want to hear a JavaScript joke?
I'll callback later.

## Callbacks

Mistakes that many beginners make

```
for (var i = 0; i < 5; i++) {
    setTimeout(function () {
      console.log(i);
    }, i * 1000);
}
// vs
for (var i = 0; i < 5; i++) {
    function print (x) {
      setTimeout(function () {
        console.log(x);
      }, x * 1000);
    }
    print(i);
}
```

## JS in the Browser

To run javascript in your HTML file, simply do:

```
<script src="myscript.js" type="text/template"></script>
// or
<script type="text/template">
  console.log('Hello World!');
</script>
```

# Manipulating DOM

Retrieving DOM elements using JavaScript

```
document.getElementById('some-id');
document.getElementsByClassName('some-class');
document.getElementsByTagName('some-tag');
```

Try this on your IVLE class roster!

```
var rows = document.querySelectorAll('tr[class*="dataGridCtrl-Alter"],
  tr[class*="dataGridCtrl-Item"]');
for (var i = 0; i < rows.length; i++) {
  rows[i].childNodes[3].innerHTML = 'Nala Cat';
  var img = rows[i].querySelectorAll('img')[0];
  img.src = 'http://nalacat.com/wp-content/uploads/2013/01/photo-2.jpg';
}
```

## Event Handling

- Attach functions to events
  - Examples of events: `click`, `focus`, `blur`, `hover`, `change`, `keydown`, etc

```
<div class="clickable" onclick="handleClick();"></div>
<div class="focusable" onfocus="handleFocus();"></div>
<div class="keyable" onkeyup="handleKeyup();"></div>
function handleClick() { ... }
function handleFocus() { ... }
function handleKeyup() { ... }
```

## Vanilla JS

- Vanilla JS is a fast, lightweight, cross-platform framework for building incredible, powerful JavaScript applications
- Used by Facebook, Google, Twitter, YouTube, Yahoo, Wikipedia, etc
- In fact, Vanilla JS is already used on more websites than jQuery, Prototype JS, MooTools, YUI, and Google Web Toolkit - combined
- Download the source **here**

## jQuery

- jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML
- Write less, do more
- Use either the `jQuery` or `$` object
- Get the source **here**

## jQuery Basics

```
$(document).ready(function () {
  // Do your stuff
});
```

- Have to wait for DOM to load before you start manipulating it

## jQuery - Getting DOM

| **Vanilla** | **jQuery** |
|---|---|
| `document.getElementById('some-id')` | `$('#some-id');` |
| `document.getElementByClassName('some-class')` | `$('.some-class');` |
| `document.getElementByTagName('some-tag')` | `$('some-tag');` |

# jQuery - Event Binding

**Vanilla**

```
<div class="clickable"
onclick="handleClick();"></div>

<div class="focusable"
onfocus="handleFocus();"></div>

<div class="keyable"
onkeyup="handleKeyup();"></div>
```

**jQuery**

```
$('div').click(function() { ...
});

$('div').focus(function() { ...
});

$('div').on('keyup', function()
{ ... });
```

## jQuery Animations

- jQuery comes with some handy animations including: `fadeIn`, `fadeOut`, `hide`, `slideUp`
- However, jQuery animations are **SLOW**
- Use CSS3 animations or other JavaScript animation libraries instead

## DOM Injection

- Beware of DOM injection when rendering user-submitted content on your webpage!
- Use jQuery `.text()` method for encoding of special characters such as `<` and `>`

## JavaScript Tools

- UnderscoreJS
  - A library of functional programming helpers, such as `map`, `filter`, `reduce`, etc
  - A must use for ~~functional~~ all programmers
- RequireJS
  - Forces you to write modular javascript
  - Handles nested dependencies
- Bower
  - A package manager, not a library
  - Keeps your libraries structured

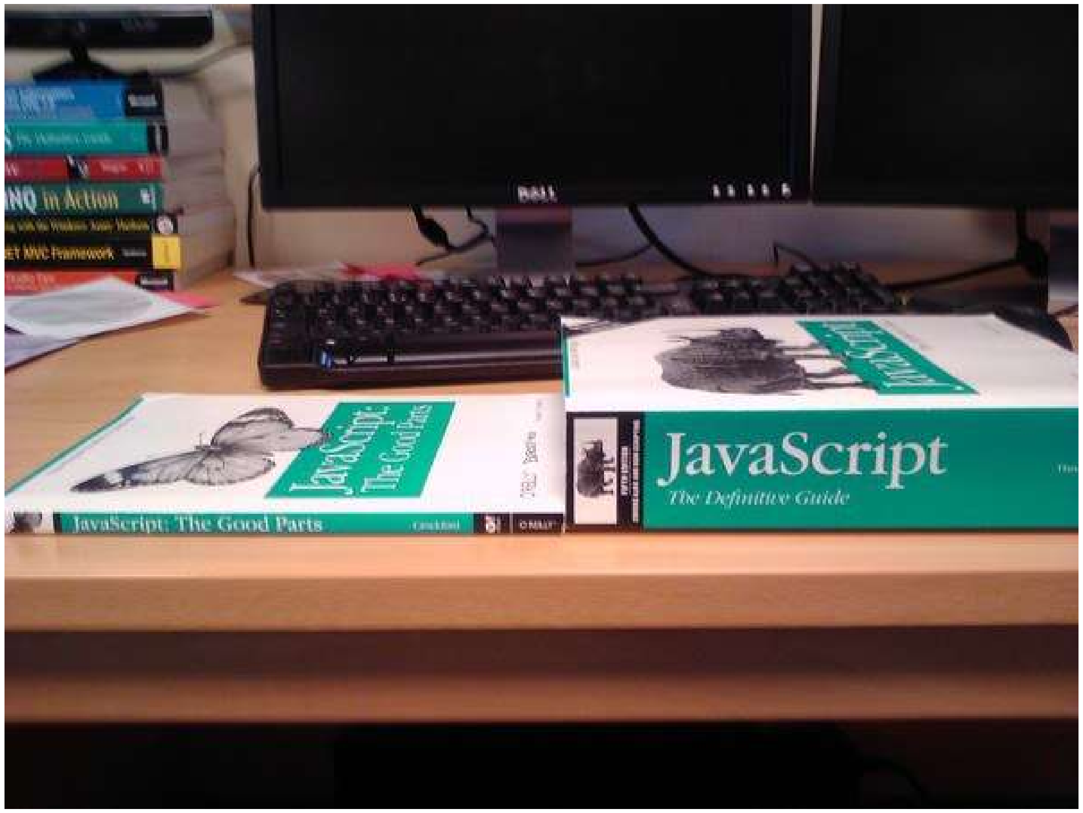
## JavaScript Frameworks

- Why use front end MVC frameworks?
  - Prevent DOM manipulation logic from being intermingled with application logic and network requests
  - Better organziation of front end code
  - When does it make sense to use an MVC framework for JavaScript
- Examples: AngularJS, EmberJS, BackboneJS.
- Bonus: Full stack JS framework - MeteorJS
- JavaScript Framework Comparisons

## JavaScript Resources

- DailyJS
- SuperheroJS
- Best Resources to Learn JavaScript
- Must Watch Videos of JavaScript

## Readings

- Eloquent JavaScript
- Secrets of the JavaScript Ninja
- JavaScript: The Definitive Guide
- JavaScript: The Good Parts
- 

## References

- JavaScript:The World Most Misunderstood Programming Language
- Douglas Crockford: The JavaScript Programming Language
- Semicolons
- The Truth About Semicolons in JavaScript
- JavaScript Function Closures

Thank You!