Introducing myself

• My research is in Human-Computer Interaction and Ubiquitous Computing

• I use Web technologies in my work to prototype and deploy in the field (evaluate) novel interactive systems
  
  – Especially around the Internet of Things, interaction with autonomous agents & the Electricity Smart Grid

• It's the first time I teach this module (!)
  
  – Feedback is welcome
Resources

• In addition to the COMP6017 module pages on the intranet, please see also:
  
  – https://hci.ecs.soton.ac.uk/wiki/JavascriptReferences
  
  – https://hci.ecs.soton.ac.uk/wiki/NodejsReferences
Building Web services with Node.js

• Node.js is a relatively new (2009) platform that combines
  – Google's V8 Javascript engine
  – An event loop for I/O (e.g. network, DB, ..)
  – Basic infrastructure for internet protocols
• Key Node.js feature: event-driven
  – Javascript is great for that!
    (that's partially why node was written for Javascript)
• Javascript: same language on front-end & back-end
Do You Already Know Javascript?

A. No / not really
B. Yes, I have used it a little
C. Yes, I consider myself an expert
Other Programming Languages?

A. No programming at all
B. Python
C. C++
D. C
E. Matlab (or other math-related specific languages)
F. Java
G. PHP
H. Actionscript
I. Any other?
Quiz

What will the following code print? (assume $.get is an ajax call to GET a URL and someUrl contains a valid URL)

```javascript
$.get(someUrl, function (data) {
    console.log('callback');
});
console.log('javascript');
```

1. "hello world"
2. "callback javascript"
3. "javascript callback"
4. Don't know
What will the following code print? (assume $.get is an ajax call to GET a URL and someUrl contains a valid URL)

```javascript
$.get(someUrl, function (data) {
    console.log('callback');
});
console.log('javascript');
```

1. "hello world"
2. "callback javascript"
3. "javascript callback"
4. Don't know
5. It cannot be predicted
Quiz

What will the following code print? (assume $.get is an ajax call to GET a URL and someUrl contains a valid URL)

```javascript
$.get(someUrl, function (data) {
    console.log('callback');
});
console.log('javascript');
```

1. "hello world"
2. "callback javascript"
3. "javascript callback"
4. Don't know
5. It cannot be predicted
Quiz

Consider the code on the right. What do you think it will print? (ignoring newlines)

1. A A B A
2. A A B B
3. A B B
4. A B A
5. It will not run
6. Don't know

```javascript
var g;

var f = function () {
    x = 'A';
    g = function () {console.log(x);};
};

f();
g();

x = 'B';
console.log(x);
g();
```
Quiz

Consider the code on the right. What do you think it will print? (ignoring newlines)

1. A A B A
2. A A B B
3. A B B
4. A B A
5. It will not run
6. Don't know

```javascript
var g;

var f = function () {
    var x = 'A';
    g = function () {console.log(x);};
};

f();
g();

var x = 'B';
console.log(x);
g();
```
The Name Is Not Helpful

• Javascript is probably the most popular and the most misunderstood language ever

• Javascript: very different from Java

• Javascript: a full and advanced programming language
Good Parts and Bad Parts

• Javascript has good parts and bad parts

• The bad parts mostly seem to come from
  – the fact that the language was designed and implemented in a rush
  – Javascript tries to look like Java, but it is VERY different
  – Companies marketing and politics
Bad Parts: Guessing

• If you do not state things explicitly Javascript "tries to guess" – often it guesses wrong
  – For example if you do not use semi-colon the interpreter will add them for you – this can interact badly with {} 

• The equality operator automatically converts the types of the things you compare [live demo]
  – Always use === (never ==)
Bad Parts: Numbers & "Void Things"

• The numbers are only floating point in IEEE-754 format
  – 0.1 + 0.2 === 0.3 // false!

• There are a lot of ways to say "nothing":
  – false, null, undefined, NaN ..so many it gets confusing

• NaN is not equal to anything ..not even to itself! [live demo]
Bad Parts: Implicit Global

• There is a global object if you do not use the keyword `var` when you declare a variable, the variable gets added to the global object (i.e. it's kind-of implicit global) [see quiz at beginning of lecture]

• Variable declaration can be implicit!

• If you use the keyword `this` without having an object it refers to the global object (no error, no warning)
Bad Parts: No Block Scope

• Scope is defined only by functions, NOT by {} blocks

```javascript
function f() {
    var i;
    // ...
    for (var i=0; i<5; i+=1) {
        // ...
    }
}
```
Bad Parts: Pseudo-classes & More

- There is a new operator that can be used to create objects; this tries to look like Java, but behaves in a very different way
  - Avoid using `new`!

- There are more bad parts, but I hope these examples convinced you to stay away from the bad parts
Good Parts: Objects

• Everything is an object (almost)
• Objects are dynamic and loosely typed
• Prototypal inheritance & we can extend objects retrospectively! [live demo]
Good Parts: Functions

• Functions as objects
  (e.g. passing functions as arguments to other functions)

• Closure: if you define functions inside other functions the inner function inherits the scope of the outer function even after the outer function returns
  – It will hopefully make sense when we look at examples

• Anonymous functions
A Paradigm Shift is Required

• If you programmed using classes (e.g. C++ or Java), when you think of objects and encapsulation you think of classes

• You need to separate those concepts

• In Javascript the same concepts map to other programming patterns, e.g. to functions with closure and prototypal inheritance

• This can be very tricky at first
Good & Bad Parts, One Solution: JSLint

- JSLint is a program that checks Javascript code to verify that none of the bad parts are used.
- It can be annoying at first, but it saves from a lot of troubles.
  - Some programmers don't like it – there is an alternative called JSHint.
- For the coursework we required you use JSLint (*not hint*)
Good Parts and Bad Parts Summary

• I only covered some examples, there is more..

• Short answer: use JSLint!

• Long answer:
  – Watch Douglas Crockford videos
    (my preferred option, links on the wiki)
  – Read Douglas Crockford's book
Node.js

• Everything (almost) in Node.js is done through asynchronous callback functions

• That is where having functions as 1st class objects and closure turn out to be very useful features
  – Anonymous functions too

• Let's get started with Node.js through practical examples! (next lecture)

• Install Node.js as soon as possible
Summary

• Build web services using Node.js, based on Javascript

• Javascript is a programming language with very advanced (and cool!) features, including: functions as objects, closure
  – Good parts and bad parts: only use the good parts!

• Javascript may requires a paradigm shift if you have experience with other prog. languages – JSLint will help