Scripting Languages

Sed

Example: Sed. A very simple scripting language, capable of just one task (“do one thing, but do it well”)

Scripting Languages

Sed

Example: Sed. A very simple scripting language, capable of just one task (“do one thing, but do it well”)

String manipulation: sed is a *stream editor*

Scripting Languages

Sed

Example: Sed. A very simple scripting language, capable of just one task (“do one thing, but do it well”)

String manipulation: sed is a *stream editor*

Reads files one line at a time (as a *stream*) and edits them, one line at a time, according to given rules

Scripting Languages

Sed

Example: Sed. A very simple scripting language, capable of just one task (“do one thing, but do it well”)

String manipulation: sed is a *stream editor*

Reads files one line at a time (as a *stream*) and edits them, one line at a time, according to given rules

`sed -e 's/hello/goodbye/'`
Substitutes `goodbye` for occurrences of `hello` in its input

Scripting Languages

Sed

Uses *regular expressions* for patterns: e.g., `hello.*world` matches all strings of characters starting with `hello` and ending with `world`

Scripting Languages

Sed

Uses *regular expressions* for patterns: e.g., `hello.*world` matches all strings of characters starting with `hello` and ending with `world`

Why not just use a regular editor?

Scripting Languages

Sed

Uses *regular expressions* for patterns: e.g., `hello.*world` matches all strings of characters starting with `hello` and ending with `world`

Why not just use a regular editor?

Firstly, it doesn't use a GUI, and so can be automated

Scripting Languages

Sed

Uses *regular expressions* for patterns: e.g., `hello.*world` matches all strings of characters starting with `hello` and ending with `world`

Why not just use a regular editor?

Firstly, it doesn't use a GUI, and so can be automated

Secondly, it can edit files too large to fit into memory all at once

Scripting Languages

Sed

Uses *regular expressions* for patterns: e.g., `hello.*world` matches all strings of characters starting with `hello` and ending with `world`

Why not just use a regular editor?

Firstly, it doesn't use a GUI, and so can be automated

Secondly, it can edit files too large to fit into memory all at once

Or streamed data, so you don't have it in a file

Scripting Languages

Sed

Sed is often used as a component in shell scripts: you don't have to sit down and drive a text editor by hand

Scripting Languages

Sed

Sed is often used as a component in shell scripts: you don't have to sit down and drive a text editor by hand

But it is limited to line-by-line editing

Scripting Languages

Sed

Sed is often used as a component in shell scripts: you don't have to sit down and drive a text editor by hand

But it is limited to line-by-line editing

More general is *awk*, but better is *Perl*

Scripting Languages

Perl

Perl was used a lot in Web page creation and manipulation, amongst many other things

Scripting Languages

Perl

Perl was used a lot in Web page creation and manipulation, amongst many other things

The basic datatype is the string

Scripting Languages

Perl

Perl was used a lot in Web page creation and manipulation, amongst many other things

The basic datatype is the string

The basic operation is pattern matching in text

Scripting Languages

Perl

Perl was used a lot in Web page creation and manipulation, amongst many other things

The basic datatype is the string

The basic operation is pattern matching in text

It is also procedural, has first class functions, has objects and so on

Scripting Languages

Perl

Perl was used a lot in Web page creation and manipulation, amongst many other things

The basic datatype is the string

The basic operation is pattern matching in text

It is also procedural, has first class functions, has objects and so on

The syntax is based on the usual C/Java/whatever, but with a few features

Scripting Languages

Perl

Perl was used a lot in Web page creation and manipulation, amongst many other things

The basic datatype is the string

The basic operation is pattern matching in text

It is also procedural, has first class functions, has objects and so on

The syntax is based on the usual C/Java/whatever, but with a few features

We'll have a quick look at some Perl code as an exercise in looking at an unfamiliar language

Scripting Languages

Perl

```
open IN, '<', 'infile';
open OUT, '>outfile';
$count = 0;
while (<IN>) {
  s/world/everybody/ if (/hello/);
  print OUT;
  $count++;
}
close IN;
close OUT;
print "$count lines\n";
```

Scripting Languages

Perl

- `open`: takes strings `'<'` to indicate input and `'>'` to indicate output; then a file name

Scripting Languages

Perl

- `open`: takes strings `'<'` to indicate input and `'>'` to indicate output; then a file name
- or joined onto the filename

Scripting Languages

Perl

- `open`: takes strings `'<'` to indicate input and `'>'` to indicate output; then a file name
- or joined onto the filename
- `IN`: filestream variables are syntactically different from normal variables

Scripting Languages

Perl

- `open`: takes strings `'<'` to indicate input and `'>'` to indicate output; then a file name
- or joined onto the filename
- `IN`: filestream variables are syntactically different from normal variables
- `$count`: scalar (single value) variable names are prefixed by `$`

Scripting Languages

Perl

- `open`: takes strings `'<'` to indicate input and `'>'` to indicate output; then a file name
- or joined onto the filename
- `IN`: filestream variables are syntactically different from normal variables
- `$count`: scalar (single value) variable names are prefixed by `$`
- array variable names are prefixed by `@`

Scripting Languages

Perl

- `open`: takes strings `'<'` to indicate input and `'>'` to indicate output; then a file name
- or joined onto the filename
- `IN`: filestream variables are syntactically different from normal variables
- `$count`: scalar (single value) variable names are prefixed by `$`
- array variable names are prefixed by `@`
- function variable names are prefixed by `&`

Scripting Languages

Perl

- `open`: takes strings `'<'` to indicate input and `'>'` to indicate output; then a file name
- or joined onto the filename
- `IN`: filestream variables are syntactically different from normal variables
- `$count`: scalar (single value) variable names are prefixed by `$`
- array variable names are prefixed by `@`
- function variable names are prefixed by `&`
- `$f` is separate from `@f` and `&f`

Scripting Languages

Perl

- `open`: takes strings `'<'` to indicate input and `'>'` to indicate output; then a file name
- or joined onto the filename
- `IN`: filestream variables are syntactically different from normal variables
- `$count`: scalar (single value) variable names are prefixed by `$`
- array variable names are prefixed by `@`
- function variable names are prefixed by `&`
- `$f` is separate from `@f` and `&f`
- `<>`: operator returns a single line of the file each time it is called

Scripting Languages

Perl

- `open`: takes strings `'<'` to indicate input and `'>'` to indicate output; then a file name
- or joined onto the filename
- `IN`: filestream variables are syntactically different from normal variables
- `$count`: scalar (single value) variable names are prefixed by `$`
- array variable names are prefixed by `@`
- function variable names are prefixed by `&`
- `$f` is separate from `@f` and `&f`
- `<>`: operator returns a single line of the file each time it is called
- and false at end of file

Scripting Languages

Perl

- `open`: takes strings `'<'` to indicate input and `'>'` to indicate output; then a file name
- or joined onto the filename
- `IN`: filestream variables are syntactically different from normal variables
- `$count`: scalar (single value) variable names are prefixed by `$`
- array variable names are prefixed by `@`
- function variable names are prefixed by `&`
- `$f` is separate from `@f` and `&f`
- `<>`: operator returns a single line of the file each time it is called
- and false at end of file
- it assigns to the variable `$_`

Scripting Languages

Perl

- `$_` is a *default* argument and can be left out of many places, e.g., `print;` is equivalent to `print $_;`

Scripting Languages

Perl

- `$_` is a *default* argument and can be left out of many places, e.g., `print;` is equivalent to `print $_;`
- `statement if (test);` as well as the usual `if (test) { statements; }` (and with `else`)

Scripting Languages

Perl

- `$_` is a *default* argument and can be left out of many places, e.g., `print;` is equivalent to `print $_;`
- `statement if (test);` as well as the usual `if (test) { statements; }` (and with `else`)
- `//` for pattern matching; in this case it looks to see if the default `$_` contains the string `hello`

Scripting Languages

Perl

- `$_` is a *default* argument and can be left out of many places, e.g., `print;` is equivalent to `print $_;`
- `statement if (test);` as well as the usual `if (test) { statements; }` (and with `else`)
- `//` for pattern matching; in this case it looks to see if the default `$_` contains the string `hello`
- `s///`: if the string contains `world`, replace it by the string `everybody` (again, using the default `$_`)

Scripting Languages

Perl

- `$_` is a *default* argument and can be left out of many places, e.g., `print;` is equivalent to `print $_;`
- `statement if (test);` as well as the usual `if (test) { statements; }` (and with `else`)
- `//` for pattern matching; in this case it looks to see if the default `$_` contains the string `hello`
- `s///`: if the string contains `world`, replace it by the string `everybody` (again, using the default `$_`)
- `$count++;` lots of C-like features (as are the `{}` and `;` and `while`, etc.)

Scripting Languages

Perl

- undeclared variables: just using a variable is enough to tell Perl a variable exists; though you can declare them if you want

Scripting Languages

Perl

- undeclared variables: just using a variable is enough to tell Perl a variable exists; though you can declare them if you want
- untyped variables: a variable can hold numbers and strings and other types; so ++ has first to check if the value is a number before adding 1 (or a string containing a number, which it then converts to a number)

Scripting Languages

Perl

- undeclared variables: just using a variable is enough to tell Perl a variable exists; though you can declare them if you want
- untyped variables: a variable can hold numbers and strings and other types; so ++ has first to check if the value is a number before adding 1 (or a string containing a number, which it then converts to a number)
- flexibility over `()` around function arguments

Scripting Languages

Perl

- strings are both single and double quoted

Scripting Languages

Perl

- strings are both single and double quoted
- single quote is unevaluated: `'hello\n'` prints as `hello\n`

Scripting Languages

Perl

- strings are both single and double quoted
- single quote is unevaluated: `'hello\n'` prints as `hello\n`
- double quote is interpolated: `"hello\n"` prints as `hello` with a newline

Scripting Languages

Perl

- strings are both single and double quoted
- single quote is unevaluated: `'hello\n'` prints as `hello\n`
- double quote is interpolated: `"hello\n"` prints as `hello` with a newline
- `"The count is $count"` sticks the current value of `$count` into the string at that point

Scripting Languages

Perl

- strings are both single and double quoted
- single quote is unevaluated: `'hello\n'` prints as `hello\n`
- double quote is interpolated: `"hello\n"` prints as `hello` with a newline
- `"The count is $count"` sticks the current value of `$count` into the string at that point
- and a *lot* more

Scripting Languages

Perl

Exercise Read about Perl 6, which was such a large step forward(?) from Perl 5 they changed the name of the language to *Raku*

Scripting Languages

Perl

Exercise Read about Perl 6, which was such a large step forward(?) from Perl 5 they changed the name of the language to *Raku*

But then came Perl 7, which reverted (mostly) back to the old ways

Scripting Languages

Perl

Exercise Read about Perl 6, which was such a large step forward(?) from Perl 5 they changed the name of the language to *Raku*

But then came Perl 7, which reverted (mostly) back to the old ways

Exercise Compare

```
$count = "99"; $count++;
```

and

```
$count = "cat"; $count++;
```

Scripting Languages

Perl

There are several very fat books on Perl

Scripting Languages

Perl

There are several very fat books on Perl

Its flexibility means it is used in a lot of places

Scripting Languages

Perl

There are several very fat books on Perl

Its flexibility means it is used in a lot of places

And it is easy to write unreadable code in Perl

Scripting Languages

Perl

There are several very fat books on Perl

Its flexibility means it is used in a lot of places

And it is easy to write unreadable code in Perl

Exercise Read about *PHP*, a language derived from Perl for specifically generating Web pages

Scripting Languages

Python

Python is an interesting case as it often isn't thought of as a scripting language any more

Scripting Languages

Python

Python is an interesting case as it often isn't thought of as a scripting language any more

Like many languages, it has grown and adapted over the years

Scripting Languages

Python

Python is an interesting case as it often isn't thought of as a scripting language any more

Like many languages, it has grown and adapted over the years

But, still, it not terribly well suited for pure computation, e.g., numerics

Scripting Languages

Python

But what about the thousands of people who write Python code to do numerical analysis and machine learning?

Scripting Languages

Python

But what about the thousands of people who write Python code to do numerical analysis and machine learning?

They are actually using Python as a front end to use library code written in C — very little of the execution time is spent running the Python code wrapper

Scripting Languages

Python

But what about the thousands of people who write Python code to do numerical analysis and machine learning?

They are actually using Python as a front end to use library code written in C — very little of the execution time is spent running the Python code wrapper

So they are using Python primarily as a way of controlling C code!

Scripting Languages

Python

But what about the thousands of people who write Python code to do numerical analysis and machine learning?

They are actually using Python as a front end to use library code written in C — very little of the execution time is spent running the Python code wrapper

So they are using Python primarily as a way of controlling C code!

That is, as a scripting language

Scripting Languages

Python

But what about the thousands of people who write Python code to do numerical analysis and machine learning?

They are actually using Python as a front end to use library code written in C — very little of the execution time is spent running the Python code wrapper

So they are using Python primarily as a way of controlling C code!

That is, as a scripting language

Exercise Compare Python features with Perl features

Scripting Languages

JavaScript

JavaScript is another case that isn't usually thought of as a scripting language any more

Scripting Languages

JavaScript

JavaScript is another case that isn't usually thought of as a scripting language any more

Originally intended to be a scripting language to manage Web page content, it is now more likely to be thought of as a special-purpose language to enable dynamic Web content

Scripting Languages

JavaScript

JavaScript is another case that isn't usually thought of as a scripting language any more

Originally intended to be a scripting language to manage Web page content, it is now more likely to be thought of as a special-purpose language to enable dynamic Web content

Web applications are coded using JavaScript

Scripting Languages

JavaScript

Standard warning:

> Java and JavaScript are **very different** languages

Scripting Languages

JavaScript

Standard warning:

Java and JavaScript are **very different** languages

The name is unfortunate: their syntax is broadly similar but their semantics are wildly different

Scripting Languages

JavaScript

Standard warning:

Java and JavaScript are **very different** languages

The name is unfortunate: their syntax is broadly similar but their semantics are wildly different

The name was originally chosen as JavaScript was intended to be “reminiscent” of Java, but now it’s just a source of confusion

Scripting Languages

JavaScript

Standard warning:

Java and JavaScript are **very different** languages

The name is unfortunate: their syntax is broadly similar but their semantics are wildly different

The name was originally chosen as JavaScript was intended to be “reminiscent” of Java, but now it’s just a source of confusion

It can be argued that JavaScript is more like Scheme/Lisp than Java in the way it behaves

Scripting Languages

JavaScript

Standard warning:

Java and JavaScript are **very different** languages

The name is unfortunate: their syntax is broadly similar but their semantics are wildly different

The name was originally chosen as JavaScript was intended to be “reminiscent” of Java, but now it’s just a source of confusion

It can be argued that JavaScript is more like Scheme/Lisp than Java in the way it behaves

But you shouldn’t take this analogy too far

Scripting Languages

Aside

Java and JavaScript is a good example of a pair of languages that look fairly similar, but behave very differently

Scripting Languages

Aside

Java and JavaScript is a good example of a pair of languages that look fairly similar, but behave very differently

While Scheme and JavaScript is a good example of a pair of languages that look very different, but behave alike(ish)

Scripting Languages

Aside

Java and JavaScript is a good example of a pair of languages that look fairly similar, but behave very differently

While Scheme and JavaScript is a good example of a pair of languages that look very different, but behave alike(ish)

```
(define (inc n) (+ n 1))
function inc(n) { return n+1; }
```

Scripting Languages

Aside

Java and JavaScript is a good example of a pair of languages that look fairly similar, but behave very differently

While Scheme and JavaScript is a good example of a pair of languages that look very different, but behave alike(ish)

```
(define (inc n) (+ n 1))
function inc(n) { return n+1; }
```

We need to be aware of both syntax and semantics when thinking about programming languages

Scripting Languages

JavaScript

JavaScript is more properly called *ECMAScript* as the name “JavaScript” is trademarked by Oracle

Scripting Languages

JavaScript

JavaScript is more properly called *ECMAScript* as the name “JavaScript” is trademarked by Oracle

This is a standards document (*not* a programming language in itself!) with several close-but-not-wholly-compatible implementations

Scripting Languages

JavaScript

JavaScript is more properly called *ECMAScript* as the name "JavaScript" is trademarked by Oracle

This is a standards document (*not* a programming language in itself!) with several close-but-not-wholly-compatible implementations

- JavaScript: Mozilla, Google Chrome, Safari, etc.
- JScript: Microsoft
- ActionScript: Adobe

Scripting Languages

JavaScript

JavaScript is more properly called *ECMAScript* as the name "JavaScript" is trademarked by Oracle

This is a standards document (*not* a programming language in itself!) with several close-but-not-wholly-compatible implementations

- JavaScript: Mozilla, Google Chrome, Safari, etc.
- JScript: Microsoft
- ActionScript: Adobe

Ecma: formerly *European Computer Manufacturers Association*, now just "Ecma International", is a standards body

Scripting Languages

JavaScript

```
function getdoc()
{
   var input = document.getElementById("num");
   var num = input.value;

   if (parseInt(num) == num && num > 0 && num < 5000) {
     document.location = "http://www.rfc-editor.org/rfc/rfc"
                         + num + ".txt";
   }
   else {
     alert("Not a valid RFC number!");
     input.value = "";
   }
   return false;
}
```

Scripting Languages

JavaScript

- functions declared by `function`, no type annotations

Scripting Languages

JavaScript

- functions declared by `function`, no type annotations
- variables declared by `var` or `let`, no type annotations

Scripting Languages

JavaScript

- functions declared by `function`, no type annotations
- variables declared by `var` or `let`, no type annotations
- any variable can hold items of any type

Scripting Languages

JavaScript

- functions declared by `function`, no type annotations
- variables declared by `var` or `let`, no type annotations
- any variable can hold items of any type
- syntax reminiscent of Java (and C and so on)

Scripting Languages

JavaScript

JavaScript is easy to start learning and use, and is good for prototyping code

Scripting Languages

JavaScript

JavaScript is easy to start learning and use, and is good for prototyping code

Though as you learn more, you discover that it has some frighteningly complex features

Scripting Languages

JavaScript

JavaScript is easy to start learning and use, and is good for prototyping code

Though as you learn more, you discover that it has some frighteningly complex features

Projects using JavaScript often prescribe a set of allowed features to manage the excess of dynamic behaviour

Scripting Languages

JavaScript

JavaScript is easy to start learning and use, and is good for prototyping code

Though as you learn more, you discover that it has some frighteningly complex features

Projects using JavaScript often prescribe a set of allowed features to manage the excess of dynamic behaviour

Exercise Have a look at *asm.js*, a subset of JavaScript that limited the worst excesses of JavaScript (now deprecated)

Scripting Languages

JavaScript

JavaScript is OO, but in a very different way from other languages, particularly Java

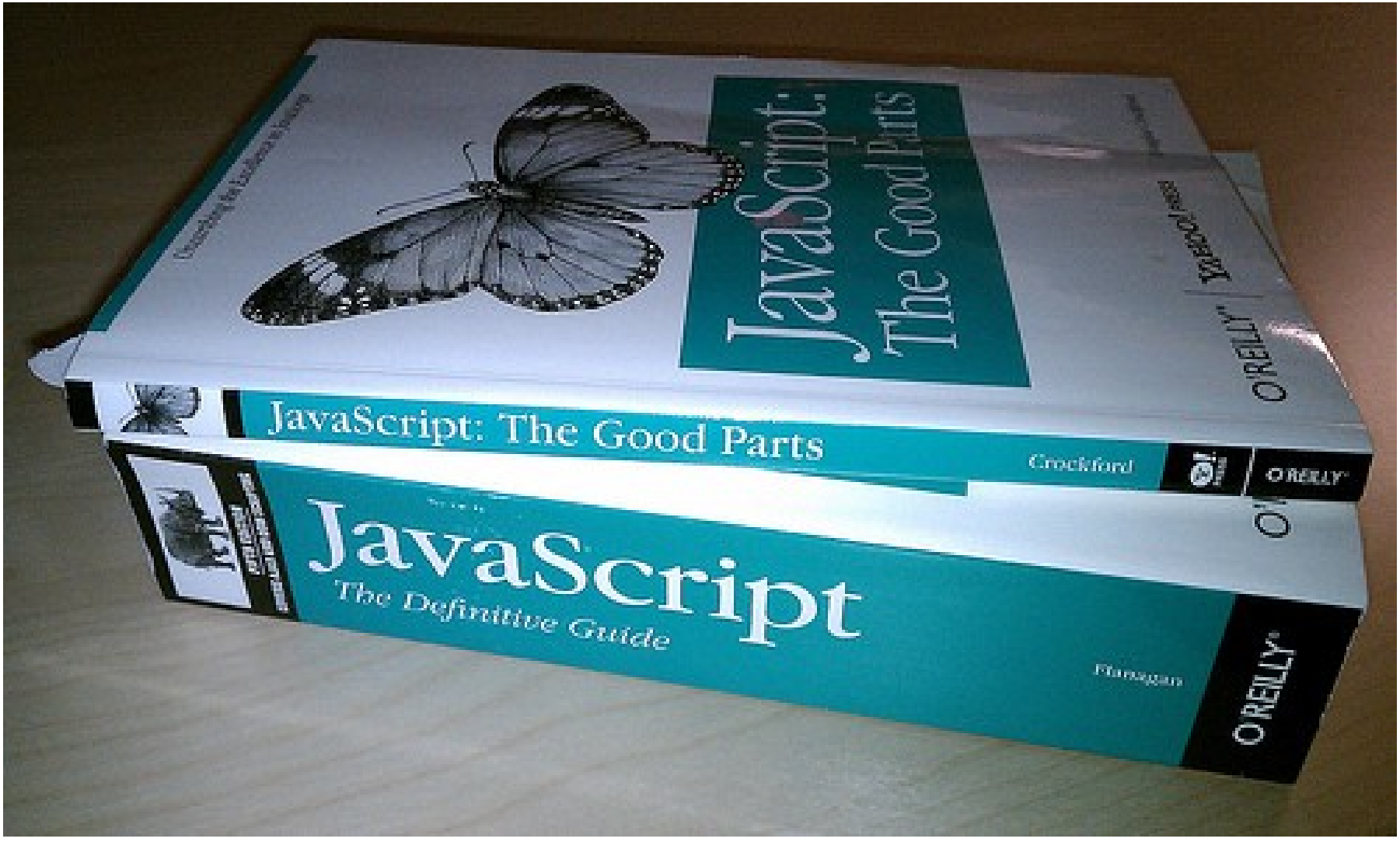
Scripting Languages

JavaScript

JavaScript is OO, but in a very different way from other languages, particularly Java

We shall be talking more about JavaScript when we examine OO in detail, but, for example, JavaScript does not have classes

Scripting Languages



JavaScript books; source: Filip Sufitchi

Scripting Languages

Aside

If you don't like JavaScript, an emerging approach to programming the Web is *Web Assembly* (WASM), a low level machine-code-like language that can be executed at "near native" speeds in the browser

Scripting Languages

Aside

If you don't like JavaScript, an emerging approach to programming the Web is *Web Assembly* (WASM), a low level machine-code-like language that can be executed at "near native" speeds in the browser

It is precompiled to a kind of machine-independent assembly code before delivery to the browser and this much faster to parse and then execute than JavaScript

Scripting Languages

Aside

If you don't like JavaScript, an emerging approach to programming the Web is *Web Assembly* (WASM), a low level machine-code-like language that can be executed at "near native" speeds in the browser

It is precompiled to a kind of machine-independent assembly code before delivery to the browser and this much faster to parse and then execute than JavaScript

It can be further compiled down to the destination machine code as fast as you can download it

Scripting Languages

Aside

If you don't like JavaScript, an emerging approach to programming the Web is *Web Assembly* (WASM), a low level machine-code-like language that can be executed at "near native" speeds in the browser

It is precompiled to a kind of machine-independent assembly code before delivery to the browser and this much faster to parse and then execute than JavaScript

It can be further compiled down to the destination machine code as fast as you can download it

source	$\longrightarrow$	WASM	$\longrightarrow$	machine code
	compiler	delivery	compiler	executed

Scripting Languages

Aside

Initial experiments with JavaScript to WASM compilers experiments are very promising

Scripting Languages

Aside

Initial experiments with JavaScript to WASM compilers experiments are very promising

But, importantly, other languages (such as Rust and C++) can compile to WASM, too

Scripting Languages

Aside

Initial experiments with JavaScript to WASM compilers experiments are very promising

But, importantly, other languages (such as Rust and C++) can compile to WASM, too

Large C++ programs, such as ray tracers and games, have been successfully compiled to WASM and can now run within a browser

Scripting Languages

Aside

Initial experiments with JavaScript to WASM compilers experiments are very promising

But, importantly, other languages (such as Rust and C++) can compile to WASM, too

Large C++ programs, such as ray tracers and games, have been successfully compiled to WASM and can now run within a browser

Exercise This that a good idea?

Scripting Languages

Aside

Initial experiments with JavaScript to WASM compilers experiments are very promising

But, importantly, other languages (such as Rust and C++) can compile to WASM, too

Large C++ programs, such as ray tracers and games, have been successfully compiled to WASM and can now run within a browser

Exercise This that a good idea?

The future of the Web may not be with JavaScript!

Event Driven Languages

Purpose: interactive systems

Examples: Visual Basic, Simula, SPICE, Java Swing, Tcl/Tk, Qt, GTK, . . .

NB: most of these are event-driven libraries used by existing languages

Notable features: based on the idea of having code executed as a consequence of something (an event) happening, rather than in some pre-specified order

Event Driven Languages

Feet

- Visual Basic: You do a Google search on how to shoot yourself in the foot. You find seventeen completely different ways to do it, none of which are properly structured. You paste the first example into the IDE and compile. It brushes your teeth

Event Driven Languages

Feet

- Visual Basic: You do a Google search on how to shoot yourself in the foot. You find seventeen completely different ways to do it, none of which are properly structured. You paste the first example into the IDE and compile. It brushes your teeth
- Visual Basic (2): You'll really only *appear* to have shot yourself in the foot, but you'll have so much fun doing it that you won't care.

Event Driven Languages

Perhaps more used as an approach to programming, rather than a family of languages

Event Driven Languages

Perhaps more used as an approach to programming, rather than a family of languages

Lots of general purpose languages can be used in the event driven style, though there are a few languages specifically designed for this, e.g., Simula

Event Driven Languages

Perhaps more used as an approach to programming, rather than a family of languages

Lots of general purpose languages can be used in the event driven style, though there are a few languages specifically designed for this, e.g., Simula

Widely used to support GUIs and other interfaces and control systems, e.g., embedded controllers

Event Driven Languages

Perhaps more used as an approach to programming, rather than a family of languages

Lots of general purpose languages can be used in the event driven style, though there are a few languages specifically designed for this, e.g., Simula

Widely used to support GUIs and other interfaces and control systems, e.g., embedded controllers

E.g., Facebook's React library

Event Driven Languages

Typically the code contains a *main loop* that waits for events (key presses, mouse clicks, temperature limits reached, data packets arriving, clock timeouts, etc.) which then chooses which chunk of code (*event handler*) to run in response

Event Driven Languages

Typically the code contains a *main loop* that waits for events (key presses, mouse clicks, temperature limits reached, data packets arriving, clock timeouts, etc.) which then chooses which chunk of code (*event handler*) to run in response

```
while (FAMNextEvent(&fc, &fe)) {
    if (fe.code == FAMExists || fe.code == FAMEndExist)
      continue;
    t = time(NULL);
    tm = localtime(&t);
    strftime(buf, 32, "%H:%M:%S", tm);
    printf("%s %s: %s\n", buf, trim(fe.filename),
                          event[fe.code]);
}
```

Event Driven Languages

The event driven style is very much like interrupt processing

Event Driven Languages

The event driven style is very much like interrupt processing

Code has to be written with the understanding that you don't know in what order the parts of the code will be executed

Event Driven Languages

The event driven style is very much like interrupt processing

Code has to be written with the understanding that you don't know in what order the parts of the code will be executed

Widely used in interactive applications

Event Driven Languages

The event driven style is very much like interrupt processing

Code has to be written with the understanding that you don't know in what order the parts of the code will be executed

Widely used in interactive applications

Also very important in *simulation*

Simulation

This is where you simulate some situation, e.g., molecules in a gas, tanks on a battlefield, to discover its properties

Simulation

This is where you simulate some situation, e.g., molecules in a gas, tanks on a battlefield, to discover its properties

Objects interact by events, e.g., a molecule has hit another, a tank has fired a missile

Simulation

This is where you simulate some situation, e.g., molecules in a gas, tanks on a battlefield, to discover its properties

Objects interact by events, e.g., a molecule has hit another, a tank has fired a missile

These events trigger some behaviour, e.g., molecules change direction of travel, tanks explode

Simulation

This is where you simulate some situation, e.g., molecules in a gas, tanks on a battlefield, to discover its properties

Objects interact by events, e.g., a molecule has hit another, a tank has fired a missile

These events trigger some behaviour, e.g., molecules change direction of travel, tanks explode

Widely used in a huge variety of situations

Simulation

This is where you simulate some situation, e.g., molecules in a gas, tanks on a battlefield, to discover its properties

Objects interact by events, e.g., a molecule has hit another, a tank has fired a missile

These events trigger some behaviour, e.g., molecules change direction of travel, tanks explode

Widely used in a huge variety of situations

So there are lots of simulation specific languages, e.g., Simula, SPICE, and so on