Scripting Languages

Scripting Languages (prehistory)

- Scripting languages have ***always*** been important in computer systems
  - They are the glue that ties the different elements of the system together
  - Their origins go back to the days of card-based operating systems
    - JCL (OS360 JCL)
    - GEORGE II, GEORGE III
  - And they were much used in minicomputer operating systems
    - Data General's AOS
    - Unix

Scripting Languages (history)

- Scripting languages originate in systems which were used to join together programs (or tasks)
- Unix and other 1980's operating systems introduced powerful commands
  - And scripting languages could put these together to produce quite powerful tools quickly and easily.
  - Shell scripting is still much used particularly by system administrators.
- Later, the concept of scripting began to be used to describe inputs to programs that interpreted their input to produce desired results
  - So one might consider scripting to be a command language for an interpreter program: application-specific languages
- But there's no definition of scripting that really distinguishes them from mainstream programming languages.
- See http://en.wikipedia.org/wiki/Scripting_language

An alternative view

- Scripting is about producing simple very-high-level-languages that are friendly to the programmer who has has a life.
- Modern sophisticated HLLS
  - Java, C++, C#, etc. are extremely complex
  - (they have a nasty tendency to get bigger and bigger as designers add more and more useful facilities, and interface components, and bells and whistles, …)
  - Take a long time to learn to use (but are wonderful when you really understand them).
- Scripting languages are relatively simple, and often allow users
  - Who are not necessarily C.Eng programmers
- … to do complex things
- Hence: very high level programming systems
- See http://www.softpanorama.org/People/Scripting_giants/scripting_languages_as_vhll.shtml

Scripting now

- There are many scripting languages:
  - Python, perl, R, javascript, PHP, …
- Some scripting languages are used primarily in web systems.
  - Some are self-standing languages. However, others are embedded in HTML and used to enhance web pages.

- Here, we shall mainly look at how they can be used with the web.
  - N.B. This is not their only area of use.
    - See Matlab, for example. Or Octave, or R, …

Overview

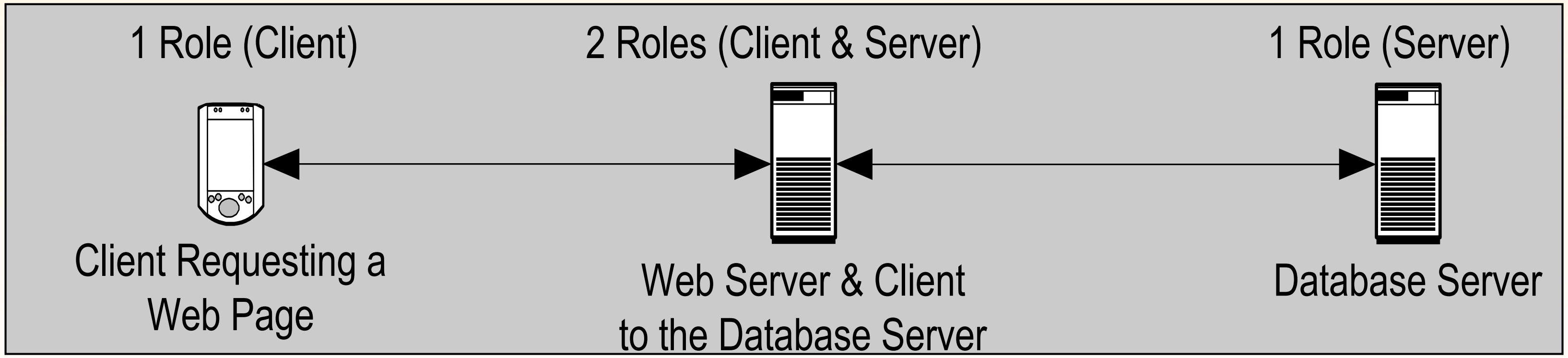
- Some core features of programming languages:
    1. Basic (built-in) types.
    2. Strings (and string handling including pattern matching)
    3. Data structures (associative arrays, but also abstract data types).
    4. Control structures.
    5. Modular programming.
    6. Type regime.
    7. Operating environment.

What is client-side and server-side?

- Any machine can play the role of either a client or a server
  - You could even have a machine being both

- Some languages, e.g. Javascript, are said to be ***client-side***.
  - Run on the user’s browser/web client

- Other languages, e.g. PHP, are said to be ***server-side***.
  - Run on the server that is delivering content to the user


1 Role (Client)
2 Roles (Client & Server)
1 Role (Server)
Client Requesting a Web Page
Web Server & Client to the Database Server
Database Server

Servers

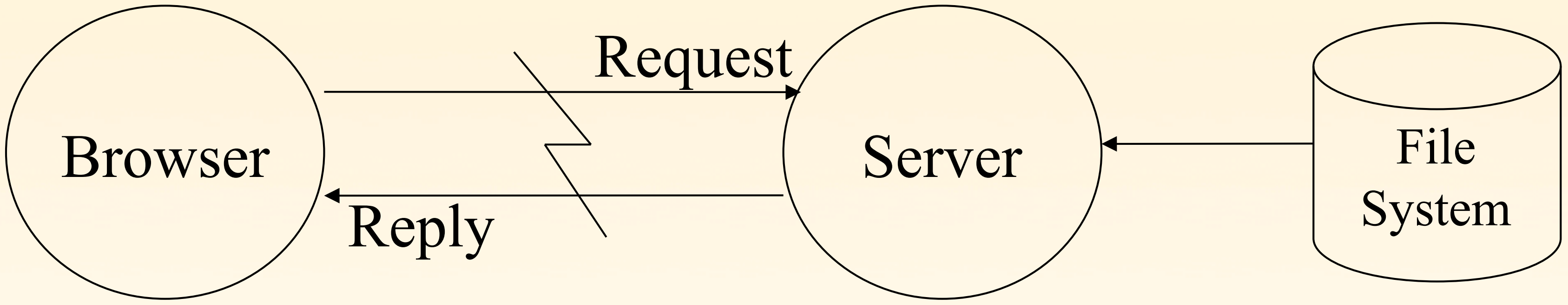
- Two important examples are:
  - Web Servers
  - Database Servers

- Some of these machines may be powerful computers dedicated to the task, i.e Database server

- A web server is a machine holding and delivering HTML pages that can be accessed by remote (client) machines over the Internet.

Refresher on Web Dynamics

- Static Web Model



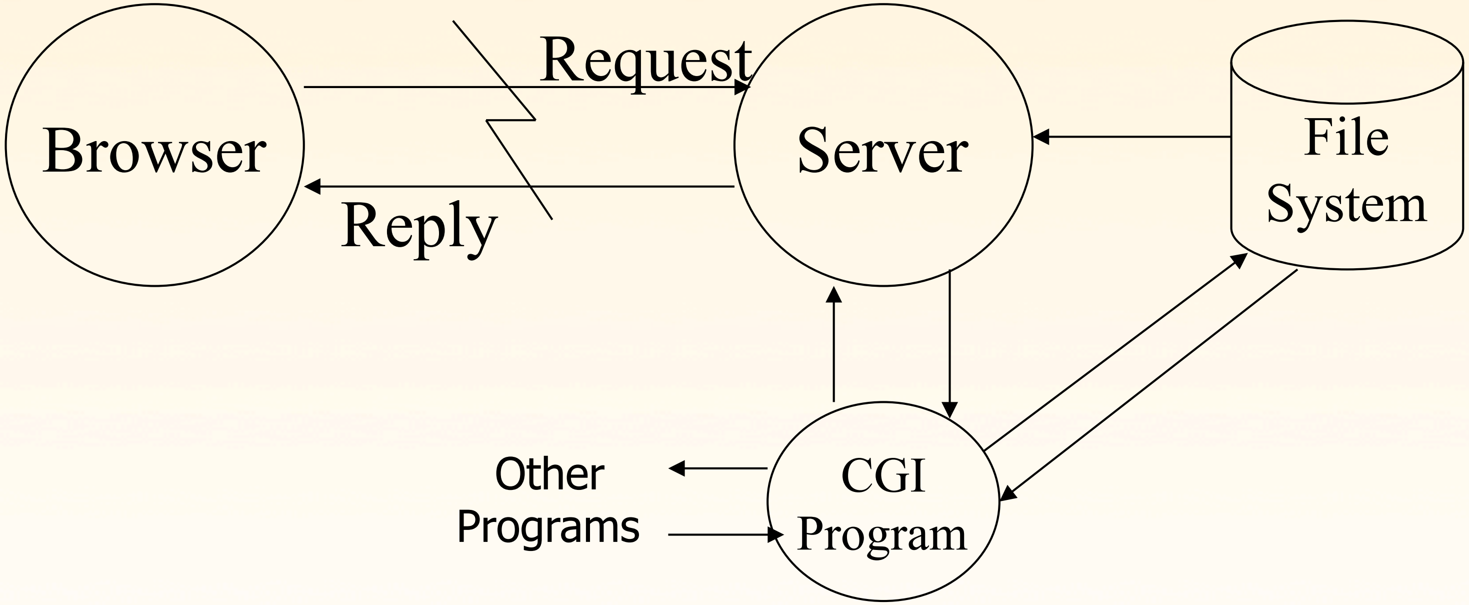
- You (the client) send a request to the server for a web page. The server looks up the web page using part of the URL you have sent it, then returns the HTML page which your browser subsequently displays on your machine.

More on Web Dynamics

- Let us now consider a more dynamic model.
  - You (the client) send a request to the server and it dynamically determines the HTML that is to be returned.
  - The dynamics of the reply is achieved through extending the ***web server*** with a program (script) that does some data processing and creates HTML output based on the data you sent (e.g. contents of a form).
  - The process of generating the HTML response is performed ***server-side***.

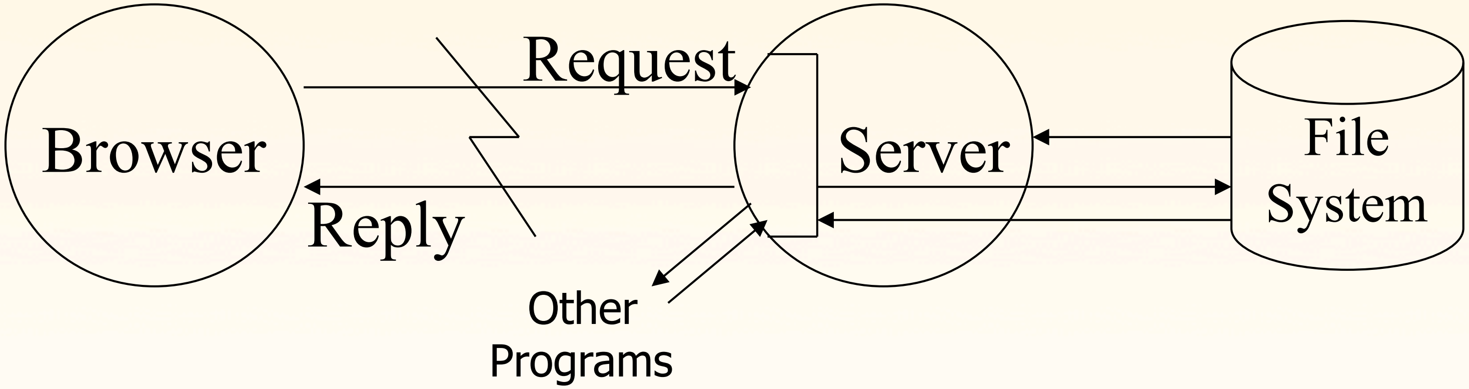
Server-side scripting

- Dynamic Web Model
  - One approach is the Common Gateway Interface (CGI) where we have a separate program that can be executed.



Server-side scripting

- Dynamic Web Model
  - An alternative is to have extra code in the HTML that can be executed on the ***server*** to determine the HTML that is to be returned.
  - That is how PHP works.



Client-Side Scripting

- The other (complementary) approach is to do the work on the client machine.
  - Again we have extra code in the HTML, but now it is executed by the user's browser (i.e. ***client-side***).
    - Most common client side script is Javascript.
  - An example of its use is when a web page has a form. We can use Javascript to validate the input data client-side before it is sent to a server.
- If we do the validation on the client, this reduces the work that the server has to do and reduces the time taken to respond to the user.
- HTML5 essentially includes Javascript elements to enhance its power.

Client-Side Scripting

- Javascript can also be used to create dynamic web page content. For example:
    - We could change the content based on the fact that you visited the web page before.
    - Time of day.
    - JavaScript popup menus.

Javascript

- Both Javascript and PHP are embedded within HTML code. Here is some Javascript, available at:

  http://www.cs.stir.ac.uk/courses/CSC9Y4/examples/y1.html
- (see also http://www.cs.stir.ac.uk/~lss/CSC941/javascript/main.html)

```
<html>
<head>
  <title>A First Program in JavaScript</title>
</head>
<body>
  <h1>Dynamic generation</h1>
  <script language = "JavaScript">
      document.writeln("<p>Welcome to
  JavaScript                    Programming!
  </p>");
  </script>
</body>
</html>
```

Javascript and PHP

- And here is some PHP held in:

  http://www.cs.stir.ac.uk/courses/CSC9Y4/examples/y1.php

```
<html>
<head>
  <title>A First Program in PHP</title>
</head>
<body>
  <h1>Dynamic generation</h1>
  <?php
  echo "<p>Welcome to PHP Programming</p>”;
   ?>
</body>
</html>
```

Client and server

- Note that we have HTML in which we have an embedded script that is to be executed.

- There are clear similarities both in terms of the syntax and in how they are used.

- However, there is one ***important difference***:
  - the PHP script is executed on the ***web server***
  - the Javascript is executed by the ***browser*** on the client's machine.

- The page with Javascript goes in an ordinary `xx.html` file while the page with PHP goes in an `xx.php` file.

Viewing source

- When we view the source of the Javascript example, we see the code written on the slide.
  - That is because that is the HTML that is handled by the browser.
  - With the PHP example, on the other hand, the PHP is executed on the server and the result of that execution is sent to the browser.
  - Hence, when we view the source for the server-side executed web page, we do not see any of the PHP code.

Another Javascript Example

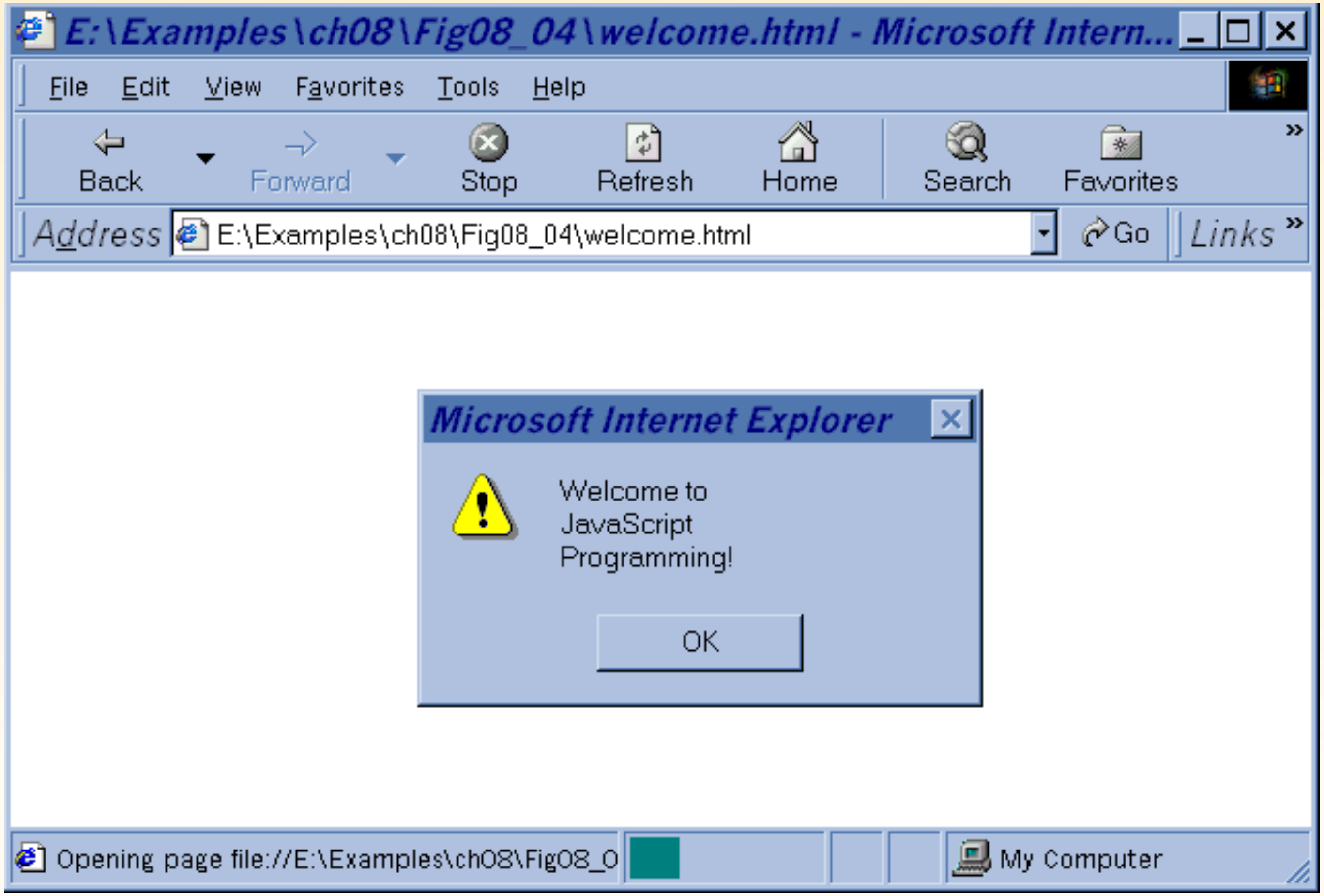
Javascript is often used to create pop ups as in the following example held in:

www.cs.stir.ac.uk/courses/CSC9Y4/examples/y2.html

```
<html>
<head><title>An alert</title></head>
<body>
  <h1>Pop up example</h1>
  <script language = “JavaScript”>
    window.alert(“Welcome
          to\nJavaScript\nProgramming!”);
  </script>
</body>
</html>
```

Alert box

-



E:\Examples\ch08\Fig08_04\welcome.html - Microsoft Intern...
File Edit View Favorites Tools Help
Back
Forward
Stop
Refresh
Home
Search
Favorites
Address
E:\Examples\ch08\Fig08_04\welcome.html
Go
Links
Microsoft Internet Explorer
OK
My Computer

Welcome to JavaScript Programming!

Opening page file://E:\Examples\ch08\Fig08_0

Server-side processing

- The following is the standard use of server-side processing:
  - Our browser is sent a static HTML page that contains a form. We type information into a textbox and press the submit button to send the information off to the server.

  - The server dynamically creates a new HTML page whose content depends on our input and returns this to us.

  - Search engines, for example, work in this way.

The form

- Let us look at how this would be done in PHP. Suppose that the original web page contains a form as in:

- `www.cs.stir.ac.uk/courses/CSC9Y4/examples/y3.html`
    - We have a textfield and a submit button.
    - An *action* field tells us where the PHP file is held that should receive the contents of the form.

The HTML code for a simple form

```
<html>
<head>
<title>Processing a form</title>
</head>
<body>
<h1>Processing a form</h1>
<form
action="http://www.cs.stir.ac.uk/courses/CSC9Y4/examples/form.php”
method="post”
>

What is your name? <input type="text" name="myname"><br>
<input value="Submit name" type="submit">

</form>
</body>
</html>
```

Execution

- If we type the name **`Jimmy`** into the text box and press the submit button then the following URL is sent:

`http://www.cs.stir.ac.uk/courses/CSC9Y4/examples/form.php?myname=Jimmy`

  - This causes the PHP program held in file **`form.php`** to be executed with the **`myname`** parameter having the value **`Jimmy`**.
  - Let us now look at the contents of the PHP file.

PHP program

```
<html>
<head>
  <title>Hello</title>
</head>
<body>
  <h1>
  <?php
    $name = $_POST["myname"];
    echo "Hello ";
    echo $name;
  ?>
  </h1>
</body>
</html>
```

PHP

- The value of the `myname` parameter is extracted from an array of Strings called `$_POST` and is saved in `$name`.
  - Note that we can access this array using a string label rather than an index number

- We then output a message that depends on the string that was sent. Hence a new web page is sent back that displays the string:

Hello Jimmy

The form

Let us now look at how this would be done using CGI and Perl.

http://www.cs.stir.ac.uk/courses/CSC9Y4/examples/y4.html

The original HTML is very similar although now the action field tells us where the CGI program is held.

```
<form action =
"http://www.cs.stir.ac.uk/cgi-bin/CSC9Y4/simple">
  What is your name? <input Name = "myname">
  <p>
  <input Type = submit Value = "Submit name">
  </p>
</form>
```

Execution

Again we type the name `Jimmy` into the text box and the following URL is sent:

http://www.cs.stir.ac.uk/cgi-bin/CSC9Y4/simple?myname=Jimmy

- This causes the CGI program held in file `simple` to be called with the `myname` parameter having the value `Jimmy`.
  - The CGI program is held in a special folder.
- Here is a possible Perl program to deal with this. It creates a new web page that displays the string:

Hello Jimmy

Perl

```
#!/usr/bin/perl
use CGI ":standard";
$name = param("myname");
print <<FirstPart;
Content-type: text/html
  <html>
    <head>
    <title>Hello</title>
    </head>
    <body>
    <h1>Hello
FirstPart
print $name;
print <<Remaining;
    </h1>
    </body>
  </html>
Remaining
```

Perl

- The first line tells the system the file is in Perl and the second brings in the Perl CGI library.
  - The `param` operation extracts the value of the `myname` parameter which is saved in `$name`.
- We could now do lots of Perl processing.
- The rest of the program consists of three print statements.
  - Note the strange bracketing convention (here-document):

```
print <<SomeLabel;
…
SomeLabel
```

  - treats the parts between the brackets as a doubly-quoted string.

CGI and Perl

- The big difference is that with PHP, the code fragments are embedded in the HTML.

- With Perl, we have a stand-alone program and it uses print statements to generate the HTML.
  - As Perl is an ordinary programming language, it can be used for lots of tasks other than generating web pages.

Accessing databases

- These simple examples have shown how information can be sent from the client to the server and how the server can dynamically create a web page that is to be returned.
  - In practice, the server will do a lot more processing.
  - Typically, the server will access a database and will either update the database or retrieve information from it. That information is then used in the creation of the dynamic web page.