Murdoch
UNIVERSITY

Topic 2: Client-Server Architecture and the World-Wide-Web

ICT373: Software Architectures

Recap: Week 1

- Basics of software architectures
- Software development life cycles
- Various models e.g., pipe and filter

Murdoch UNIVERSITY

Overview

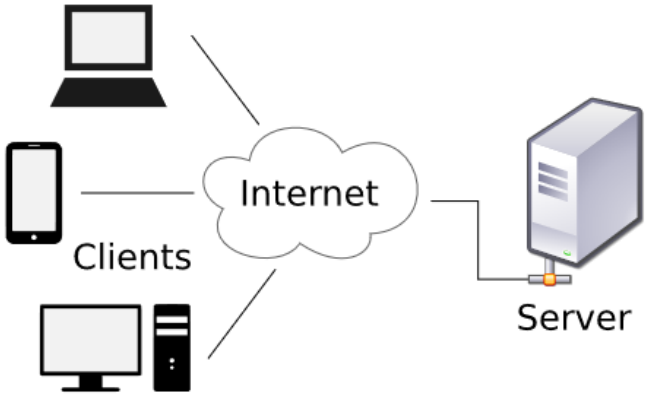
- Overview
- HTML
- JavaScript

Murdoch UNIVERSITY

Learning objectives

- Introduce the Client-Server architecture using WWW..
- Describe the use of the Web to support a client-server system.
- Describe the roles and operations of browsers, HTML, links, URLs, forms and MIME.
- Give reasons for using client-side programming and explain how it can be done.
- Explain the concepts of objects and events on an HTML page (as used by JavaScript).
- Describe the use of batch and run-time validation.
- Be able to implement simple batch and run-time validation using JavaScript.

Client-Server Architecture



Server:

- central repository of information
- centrally located so it can contain the latest info and can be changed easily
- consists of
  - the information,
  - the software to manage the information,
  - the software to manage the distribution of information and
  - the central machine(s)

Clients: each consists of:

- software that communicates with the server, fetches the information, processes this information and displays it for the user of the remote machine,
- and the remote (consumer) machine.

Murdoch UNIVERSITY

Client-Server Architecture

Examples:

- distribution of data (e.g., stock market, scientific, government),
- taking online orders and credit-card transactions.

Challenges:

- Overloaded servers; congestion
  - We want error-free, fair processing.
  - Simultaneous transaction processing (two transactions in a bank account).
- Supporting multiple types of client machines and multiple types of client operating systems.
- Supporting software changes especially if the clients also have to change.

Client-Server Architecture

Performance issues:

- Eg, if the clients or connections are unreliable then the server cannot delay responding to client A while it waits for some answer from client B.
- Eg, if there are many clients involved then the server processing has to be very efficient.

Server-client processing is very important and becoming increasing popular.

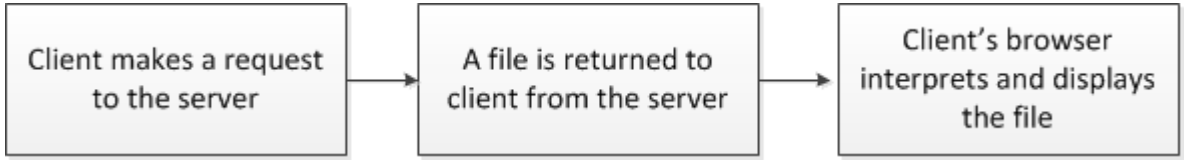
Murdoch UNIVERSITY

The World-Wide-Web as a Client-Server System

- The Internet is the network formed by millions of computers
- In the early 1990's the WWW was added.
  - It uses a client-server model (defined by HTTP) to allow a number of pages of information (containing text, pictures, sound, video, etc) with links to other pages to be set up on a server.
  - The information can be read by a client (anywhere else on the Internet) using a commonly available piece of software - the web browser.
- Despite many more recent exciting developments in this set up, the basic WWW client-server arrangement is still used as the underlying architecture for many applications.
- We will discuss a client-server relationship using a WWW set up when designing a piece of software.

Murdoch UNIVERSITY

Browsers and HTML



- Basic mode of operation:
- To make this work on various types of client machines requires **a standard language** (protocol) for specifying the formatting of the information in a way that can be sent by the server, and **browsers** capable of interpreting the language faithfully on the various client machines with their various operating systems.
- The standard language used is HTML (or XHTML), the HyperText Markup Language, which is specified by the W3C (World Wide Web Consortium).
  - New versions of HTML are published every so often by the W3C to keep up with the rapidly developing technology.
  - HTML5 (released in October 2014) is the new markup language which has both a regular text/html serialization and an XML serialization.
  - Anyone is free to answer web page requests in any format they want but, since the browser manufacturers (MS, Mozilla and Google) respect HTML specs, a different format will probably not be able to be displayed by many clients.

Murdoch UNIVERSITY

Basic HTML: an example

ICT373T2Ex1.html

```
<!DOCTYPE HTML PUBLIC "EG">
<html>
    <head>
        <!-- A sample document -->
        <title>Example 1</title>
    </head>
    <body>
        <h1>Top-level heading</h1>
        <p>First paragraph of text.</p>
        <ul>    <!-- A bulleted list -->
            <li>First list item.</li>
            <li>Second list item, with a hypertext
                <a href="info.html">link</a>
                to another file.</li>
        </ul>
        <h2>Second-level heading</h2>
        <p>Another paragraph,
<img src="mypic.gif" alt="Picture of me"
align="bottom" > with my picture.</p>
    </body>
</html>
```

Murdoch UNIVERSITY

Multipurpose Internet Mail Extensions (MIME)

- MIME is an Internet standard
- MIME extends the format of email to support
  - Text in character sets other than ASCII
  - Non-text attachments: audio, video, images, application programs..
  - Message bodies with multiple parts
  - Header information in non-ASCII character sets
- Location of the picture: When the browser constructs such a web page it will notice if it has to insert pictures at appropriate places.
- Additional information: another request off to get the image file from a place specified by the SRC tag. The image is decoded and displayed in the right place.
- Image types: The image file will also have to be sent from a server to the client browser. Thus HTTP must be able to cope with (still) images of several different formats (as well as text, special characters, and audio and video).

MIME

- MIME is mainly designed for email protocols (SMTP)
  - Also important for communication protocols, e.g., HTTP, WWW.
- Servers insert the MIME header at the beginning of any Web transmission.
  - each image in a webpage thus requires a new client server connection to be made with a separate message sent each way.
- MIME messages involve several headers giving details about the format and content of the message and then a suitably encoded message body.

HTML: Links

- When the client's user clicks on the hypertext link text (which could also be a picture), the browser arranges for the specified file to be requested from the server, interpreted and displayed.
  - Links to other pages appear in HTML pages within tags (<a…>,)

here is the blue clickable text

and here is the following text.

…or, for a more distant transfer example, …

here is the blue clickable text

- In this example, click on the linking text causes a page from a completely different server to be requested and displayed.
- The server and the particular HTML page kept on it are specified by the string defined as *href* in the opening <a> tag.

Murdoch UNIVERSITY

Uniform Resource Locators (URLs)

- Each page on the web is specified by a unique URL which allows a user of a browser to find it and allows it to be linked to from other pages on the web. e.g.,

http://www.it.murdoch.edu.au/~psd/addr.html

- comes in three parts:
  - a protocol (eg, http)
  - a name of the host server (eg, www.it.murdoch.edu.au)
  - the file specification (eg, ~psd/addr.html)

Murdoch UNIVERSITY

URLs

When such a link is clicked on,

- the browser finds out an IP (Internet Protocol) address of the host server (by asking, across the internet, another machine known as the domain name system or DNS - which has a list of all such names).
- Using this address it can establish a connection with the server using the specified protocol (eg, HTTP) to get the specified file.
- The way the host server actually finds the file varies from server to server (and the client does not need to know the system). Eg,
- *~psd/addr.html* might mean the file called *addr.html* within the directory *public_html* within the home directory of the user *psd*.
- giving no file name usually means the file called *index.html*

Interactivity

- HTML 1.0, the HTML version used by the first browsers (Mosaic), was a one-way language.
  - great interests which it (and the early web) caused, led to demand for full two-way client-server capabilities.
  - People wanted to send information back to the server, for example, to look up information in a central database or even to add information to a central database.
  - There were many commercial (as well as scientific) applications.
- HTML version 2.0 included ***forms***. Forms contain boxes and buttons of a few useful types which can be filled in by the browser user.
  - A **SUBMIT** button causes the collected information to be sent to a specified URL.

Interactivity

- There may be several forms on an HTML page. If it is to allow sending of information, each needs to include
  - a METHOD for communication,
  - a destination URL (which does not have to be on the host server) and
  - a SUBMIT button.

Interactivity: an example

ICT373T2Ex2.html

```
<!DOCTYPE HTML PUBLIC “EG”>

<html>
        <head>
            <title>Example 2</title>
        </head>
        <body>
<hr>
 <form method="GET"
 action= "http://www.it.murdoch.edu.au/cgi-bin/reply1.pl ">

<p> Name:
<input name="name" type="text" size="20"><br>

Operating System: <select name="opsys">
              <option> Unix
              <option> Linux
<option> Mac OS
<option> MS-DOS
<option> Windows
<option> Android
<option> iOS
        </select> </p>

<p><textarea name="comments" rows="4" columns="40">Please write your comments here...</textarea> </p>

<p><input type="submit"> <input type="reset"></p>
         </form>
         </body>
</html>
```

Interactivity

- Press SUBMIT -> the values entered on the form are sent to the server specified in the ACTION URL via the HTTP METHOD (either **GET** or **POST**) specified.
- **GET** - Requests data from a specified resource
- **POST** - Submits data to be processed to a specified resource

Interactivity: requesting data

- The server passes the message on to the file specified in the URL.
- The GET method in HTTP is meant to be only for getting web pages from a server but can be used in a tricky way to pass information to a program on a server.
- A GET message will be in the (ASCII) form "GET *URL*" with obvious meaning.
- If the GET method is used for submission of a form then the input values are actually sent in this message appended after a '?' at the end of the ACTION URL. The server finds the file specified by the URL string before the '?'.
- http://www.it.murdoch.edu.au/cgi-bin/reply1.pl?name=ferdous&opsys=Unix&comments=Hello

Murdoch UNIVERSITY

Interactivity: sending data

- The POST method is more common.
- HTTP's POST was designed for adding info to a web page. The message sent contains a header "POST *URL*" followed by a coded version of the input values.
- The HTTP POST message is capable of sending all sorts of information (text, sound, video) via an encoding to ASCII. To send the input values of a form the encoding is a simple text encoding.
- http://www.it.murdoch.edu.au/cgi-bin/reply1.pl
- In either GET or POST methods the values are put together into one string such as
- **name=john&opsys=Linux&comments=no+comment**

Murdoch UNIVERSITY

GET VS POST

- When the method is GET, all form data is encoded into the URL, appended to the action URL as query string parameters. With POST, form data appears within the message body of the HTTP request.

Self study for fun: http://www.diffen.com/difference/GET-vs-POST-HTTP-Requests

Server-side Programming and CGI

- A common set up for receiving such information is to use the Common Gateway Interface (CGI) provided on all servers.
  - The "**cgi**-**bin**" **directory** is the location you will use to store your Perl or compiled script files. Any files you place in it will be treated as programs (instead of HTML pages or images), and will be "run" by the server instead of displayed normally.
  - When the server gets a request to apply a method to a page in a directory called “cgi-bin”, it interprets the file name as an executable program (or script) and starts it up.
  - The body of the message is passed to the running program as its standard input.
  - The program can do whatever it wants with the input (eg, interact with a local database) and if it produces some text on its standard output then that is sent to the client’s browser to be displayed.

Murdoch UNIVERSITY

Server-side Programming and CGI

- Such a CGI script is commonly written in one of the following languages: Perl, PHP, Python, ASP (and ASP.Net).
- C and C++ can also be used to set up more complicated server programs.
- Java is increasingly used. Java-based Web servers allow you to perform all your server-side programming in java by writing servlets, JSPs and JSFs.

We will return to server-side programming much later in the unit.

Client-side programming: why?

- Sending all data back to the server for all processing is not efficient. For example,
- if there is a simple mistake on the form making it unacceptable then there can be delays while information is sent to the server, checked, and an error information page prepared and sent back.
- This is a waste of resources as the browser is most likely running on a machine quite able to check a form for valid input.
- The solution is ***client-side programming***, i.e.,
  - giving the browser the ability to run programs and allowing the server to send programs along with the HTML page.

Client-side programming

- ***plug-ins:*** extra bits of software that can be added to browser software to give the browser extra capabilities.
- Plug-ins were already being used to cope with all the new formats for sending information across the web, e.g., pictures, audio, video.
  - Instead of getting a new browser every time a new format is available, users could just get the extra plug-in software.
  - The plug-in itself was often able to be downloaded over the web and installed easily.
  - The plug-in could be made available without the involvement (or permission) of the software manufacturer.
  - Several client-side programming languages arrived as plug-ins, e.g., flash player.

Scripting Languages

- These are interpreted client-side programming languages with the source being sent as part of an HTML page.
- The interpreter (either a plug-in or built-in to the browser) runs the code as it displays the HTML page.
- The source code is quick to get from the server (part of the HTML text) and quick to load.
- **Scripting languages** are usually quite simple to learn and understand, allow fast development of software and are designed to solve the kind of programming tasks needed to make nice GUIs for client use.
    - If you need to produce a web-based client-server system then it is a good idea to consider using a scripting language for the client side.

Scripting Languages

- Scripting languages are not general purpose programming languages. Eg, no writing to disk is allowed. Also, they are interpretive languages.
  - JavaScript, the most common, built-in on Netscape, Explorer and Firefox (and nothing to do with JAVA)
  - VBscript (like Visual Basic)
  - Tcl/Tk

JavaScript

- JavaScript can be used to validate a form on the client's browser before it is sent to the server.
  - The script forms part of an HTML page. A chunk can appear in the body for processing during the page's construction.
    - we will just see how to put functions (i.e. procedures) in the head and brief event-activated calls to these procedures in the body.
  - The script is really about the ***objects*** on the page, i.e. **the HTML elements like links, forms and buttons within forms**.
    - It is useful to NAME the objects in their tags.

JavaScript

- ***Events*** happen to the objects on the page as the user interacts with it.
- JavaScript ***event handlers*** allow script to be executed in response to an event. Event handlers include:
  - **onClick** in response to a click on a button
  - **onSubmit** in response to clicking on the submit button on a form
  - **onMouseOver** in response to the cursor being over an object
  - **onChange** in response to the text being changed in a text area.

Functions: alert_example1.html

- To define a **function** we give its name, formal parameters and body within script tags in the head of the HTML page
- We can then simply mention the function name (with arguments) to call it.

```
<!DOCTYPE html>
<html>

<script>
function myFunction() {
    alert("Hello! I am an alert box!");
}
</script>


<body>

<p>Click the button to display an alert box.</p>

<button onclick="myFunction()">Try it</button>


</body>
</html>
```

Functions can **return** values.

Murdoch UNIVERSITY

Function explained

- **alert("text")** pops up a window containing "text" (which can be a number)
- **var variableName = value** create and initialize
- **variableName = value** assignment (can use arithmetic, logic, string operators)
- **if ... else, while, for, switch** similar to C++/Java; Comments as in Java // or /* */
- To get access to user entered information on a document either refer to the .value member of a hierarchy of names, eg, if the form named "myForm" has a text line called "fred", then you can write **x=document.myForm.fred.value**

Function explained

- or, pass an object to a function using the ***this*** variable and look up its value, eg, put
- **x=g.fred.value** in the **function mary(g)** and call by

  <form name="myForm" onSubmit="mary(this)">
- Cancel a submission by returning false on submitting.// isVaild()?
- Focus on and select objects.

Murdoch UNIVERSITY

Form Validation: Batch validation

- **batch validation**. The form is checked only when the user presses the submit button.
- This is similar to what happens with server processing only, but client-side batch processing is much quicker and can allow errors to be indicated without the need for a new page to be loaded.

Example: batch validation: **batch validation.html**

```
<html>
        <head>
                <title>Batch Example</title>
                <script language="JavaScript">

                function validate(form) {
                        if (form.name.value == "") {
                                alert("The name field is required.")
                                form.name.focus()
                                form.name.select()
                                return false
                        }
                        return true
                }
</script>
</head>
        <body>

<form method="GET" action="http://www.it.murdoch.edu.au/cgi-bin/reply1.pl"
                                onSubmit="return validate(this)">
<p>Name: <input name="name" type="text" size="20"></p>
<p><textarea name="comments" rows="4" columns="40">Please write here...</textarea></p>
<p><input type="submit"> </p>
</form>
</body>
</html>
```

Form Validation Examples

- From the user's point of view, it is even better to use "**real-time validation**" and actually check the form as individual events occur.
- However, it is easy for a user to bypass the real-time error messages and submit a bad form.
  - best to use both batch validation and real-time validation.
- Note that even with such client-side validation in force, the server program still needs to check input as well, since messages can be sent to it from other "clients".
- Note that the next example uses a non-primitive type of JavaScript object and so uses the keyword **new** (see Java or C++). Arrays have to be created this way too.

Validation example: **validation_example.html**

```
<html>

<head>

<title>Validation Example</title>

<!-- we only check for numbers in the fields-->

<script language="JavaScript">
     today = new Date()

function isNotEmpty(field){
    var inputStr = field.value
    if (inputStr == "" || inputStr == null ) {
        alert("This field requires an entry.")
        field.focus()
        field.select()
        return false
    }
    return true
}
```

```
function isNumber(field){
    if (isNotEmpty(field)){
        var inputStr = field.value
        for (var i = 0; i < inputStr.length; i++){
            var oneChar=inputStr.substring(i,i+1)
            if (oneChar < "0" || oneChar > "9"){
                alert("Numbers only.")
                field.focus()
                field.select()
                return false
            }
        }
        return true
    }
    return false
}
```

Murdoch UNIVERSITY

Validation example

```
function validate(form) {
    if  (isNumber(form.day) && isNumber(form.month) && isNumber(form.year))
        return true
    else
        return false
}

function makeToday(form) {
    form.day.value=today.getDate()
    form.month.value=today.getMonth()+1
    form.year.value=today.getYear()+1900
}

</script>
</head>
```

```
<body>

<form method="GET" action="http://www.it.murdoch.edu.au/cgi-bin/reply1.pl"
    onSubmit="return validate(this)">

    <p>Please enter the date of your next birthday.</p>

    <p>Day(1-31): <input name="day" type="text" size="10"
    value="1" onChange="isNumber(this)"></p>

    <p>Month(1-12): <input name="month" type="text" size="10"
    value="1" onChange="isNumber(this)"></p>

    <p>Year(2015-2016): <input name="year" type="text" size="10"
    value="2016" onChange="isNumber(this)"></p>

    <p>
        <input type="submit">
        <input type="reset">
        <input type="button" value="today" onClick="makeToday(this.form)">
    </p>

</form>

</body>

</html>
```

Further Client-Side Programming

- The general purpose programming language JAVA also allows client-side programming via its ***applet*** objects. An applet is a program which runs under a Web browser (both Firefox and Explorer are JAVA enabled).
- The compiled applet forms part of a web page and is downloaded as part of getting the page (just as a picture may be downloaded). It can be immediately activated and the running program can do all sorts of useful things from providing a diverting animation in one corner of the page to input validation to managing the whole user interface.
- As a powerful programming language with good GUI capabilities, JAVA allows the designer of web-based systems to construct powerful client-side programs.
- After we see the basics of JAVA in general over the next few weeks, we will return to look in more detail at the use of applets and consider issues of speed of downloading and security (with the client machine running a program which it got off any old web page).
- It is worth emphasizing that scripting languages have advantages of ease of use and speed over JAVA as client-side programming languages. But they can not do everything.

Extra Information is available at ...

- The Manual for Javascript (Netscape):

https://developer.mozilla.org/en/docs/Web/JavaScript/Guide

- Another Javascript 1.5 reference: http://www.webreference.com/javascript/reference/core_ref/
- HTML details (W3C's home page for the HTML Activity): http://www.w3.org/html/
- HTML5 details: http://www.w3.org/TR/html5/

Murdoch UNIVERSITY

Summary

- Client-Server architecture using WWW..
- Web to support a client-server system.
- Concepts of objects and events on an HTML page (as used by JavaScript).
- Use of batch and run-time validation.
- Simple batch and run-time validation using JavaScript.

Murdoch
UNIVERSITY

Summary

Summary

- Client-Server architecture using WWW..
- Web to support a client-server system.
- concepts of objects and events on an HTML page (as used by JavaScript).
- use of batch and run-time validation.
- simple batch and run-time validation using JavaScript.


Murdoch
UNIVERSITY