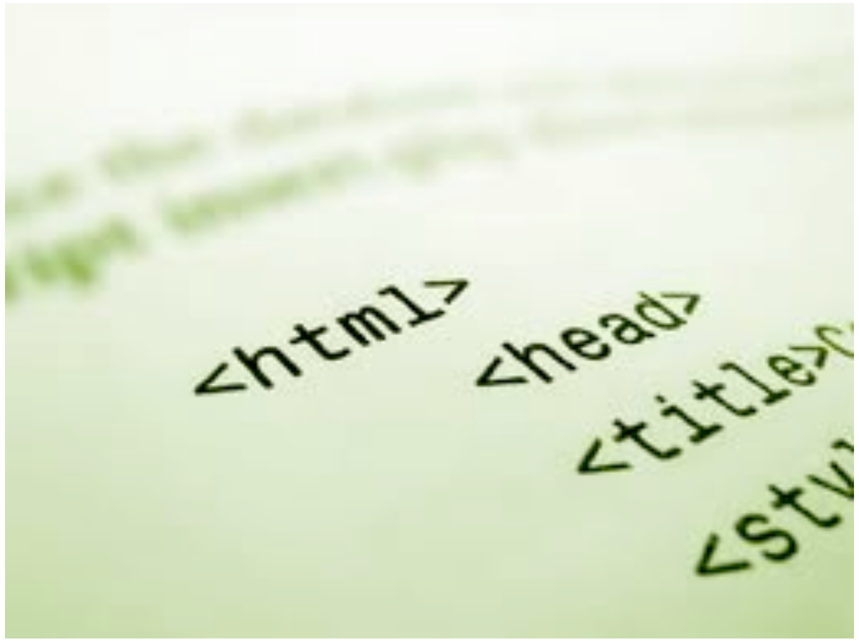
CHAPTER 3

HTML/CSS

HTML, CSS, Page Layout, and Forms



http://laughingsquid.com/wp-content/uploads/css-is-awesome-20090407-142244.jpg

http://stuarteske.com/wp-content/uploads/2011/01/cool-html-codes.jpg

SECTION 1

A brief history of HTML

- Invented by Tim Berners-Lee in 1990 (on a NeXT computer)
- Uses angled brackets (e.g. <html>)
- Designed to be human readable and writable, using a text editor.
- HTML 2.0 (1995)
  - added support for tables.
- HTML 4.0.1 (1999)
  - HTML version used until recently.
- XHTML
  - XML compliant version of HTML
- HTML5
  - Begun in 2004 by Apple, Mozilla, and Opera.
  - HTML 4.0.1 and XHTML were trending towards stricter, consistent syntax.
  - HTML5 is less strict, easier to type, and backwards compatible.
  - HTML5 also introduces several new features for web applications including:
    - Canvas support
    - Application Cache
    - Forms, etc.
- Even though HTML5 has only been recently adopted, it is backwards compatible with earlier browsers.
- Therefore we will focus on HTML5 in this course.

HTML Head and Body

Resources:

- W3C HTML5 specification:
  - http://dev.w3.org/html5/markup/spec.html
- W3C HTML5 Validator:
  - http://validator.w3.org/

The following tags are required for a valid HTML5 document:

- **DOCTYPE:**
  - All HTML documents must begin with a DOCTYPE tag as follows:

```
<!DOCTYPE html>
```

- **<html> tag:**
  - The rest of the html is contained within the <html></html> tags.
  - The <html> tag has two required child tags: <head> and <body>
- **<head>**
  - The <head> tag is used to declare the document's title, character set, other metadata, scripts and styles.
- **<meta charset=utf-8>**
  - Declared within the <head> tag.
- **<title>**
  - Declared within the <head> tag.
  - Specifies the strings displayed in the window title bar.
- **<body>**
  - Contains the remainder of the HTML document.

SECTION 2

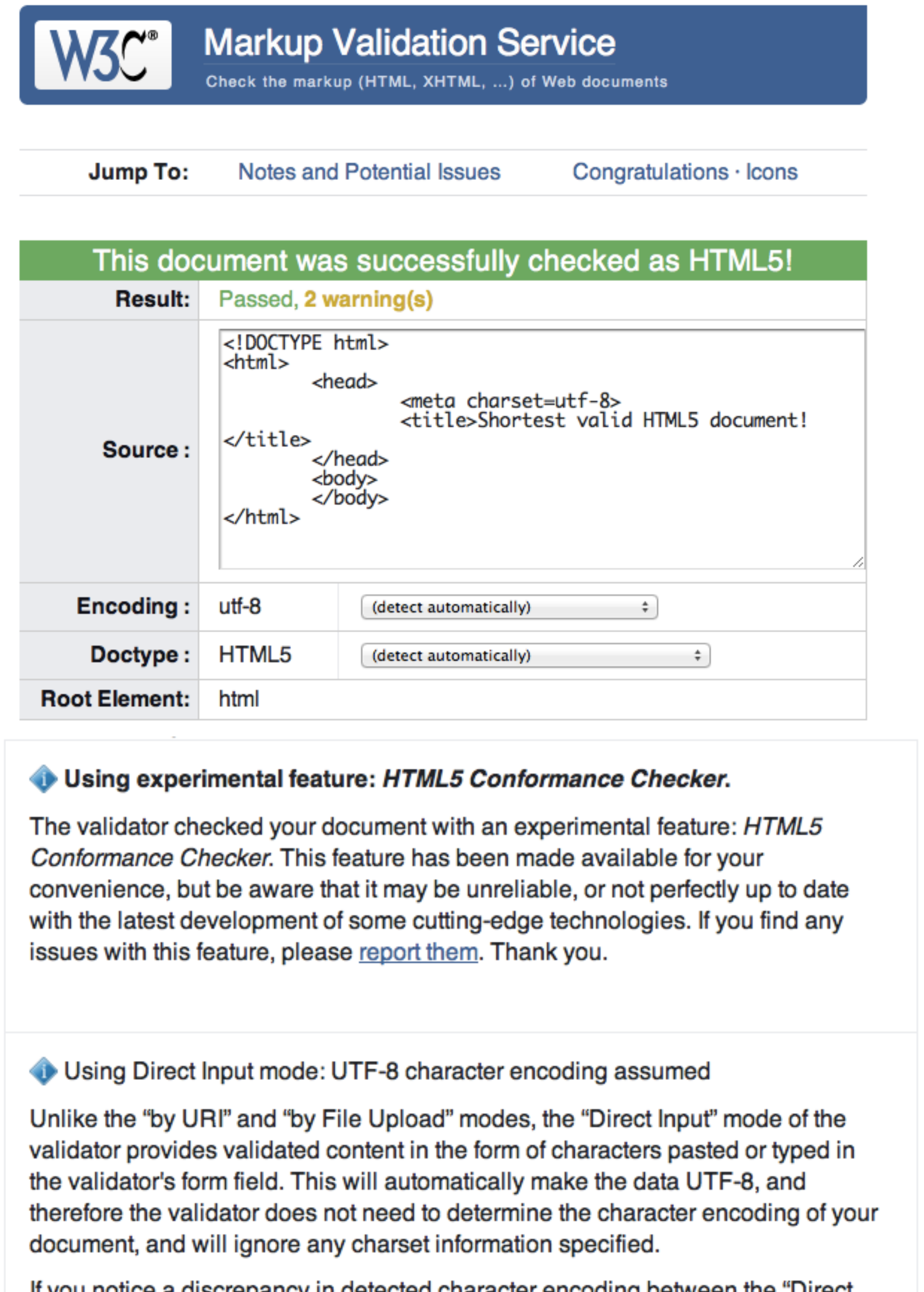
Minimal HTML5 Example

Below is the minimal valid HTML5 document.

You will need to commit to memory these tags.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset=utf-8>
      <title>Shortest valid HTML5 
document!</title>
   </head>
   <body>
   </body>
</html>
```

http://validator.w3.org/



Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents

Jump To: Notes and Potential Issues Congratulations · Icons

This document was successfully checked as HTML5!

Result:	Passed, 2 warning(s)	
Source :	see below	
Encoding :	utf-8	(detect automatically)
Doctype :	HTML5	(detect automatically)
Root Element:	html	

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset=utf-8>
                <title>Shortest valid HTML5 document!
</title>
        </head>
        <body>
        </body>
</html>
```

Using experimental feature: *HTML5 Conformance Checker.*

The validator checked your document with an experimental feature: *HTML5 Conformance Checker*. This feature has been made available for your convenience, but be aware that it may be unreliable, or not perfectly up to date with the latest development of some cutting-edge technologies. If you find any issues with this feature, please report them. Thank you.

Using Direct Input mode: UTF-8 character encoding assumed

Unlike the "by URI" and "by File Upload" modes, the "Direct Input" mode of the validator provides validated content in the form of characters pasted or typed in the validator's form field. This will automatically make the data UTF-8, and therefore the validator does not need to determine the character encoding of your document, and will ignore any charset information specified.

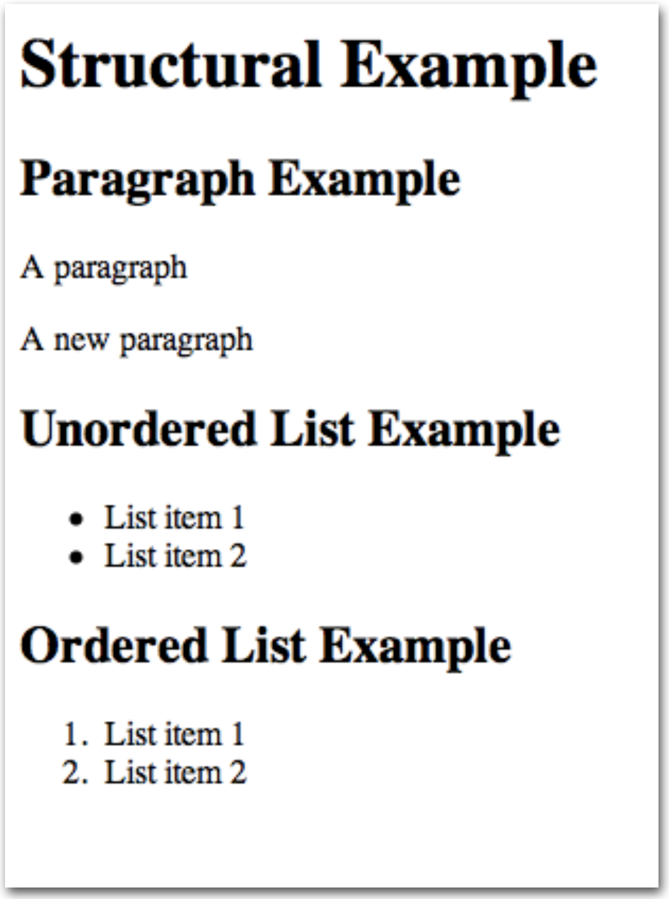
If you notice a discrepancy in detected character encoding between the "Direct

Structural Tags

- Heading tags: **<h1>**, **<h2>**, .. **<h6>**
- Paragraph tag: **<p>** .. **</p>**
- Lists:
  - Unordered list: **<ul>**
  - Ordered list: **<ol>**
  - List items: **<li>**

Structural Example:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset=utf-8>
      <title>Structural Example</
   title>
   </head>
   <body>
      <h1>Structural Example</h1>
      <h2>Paragraph Example</h2>
      <p>A paragraph</p>
      <p>A new paragraph</p>
      <h2>Unordered List Example</
   h2>
      <ul>
         <li> List item 1
         <li> List item 2
      </ul>
      <h2>Ordered List Example</h2>
      <ol>
         <li> List item 1
         <li> List item 2
      </ol>
   </body>
</html>
```



Structural Example

Paragraph Example

A paragraph

A new paragraph

Unordered List Example

- List item 1
- List item 2

Ordered List Example

1. List item 1
2. List item 2

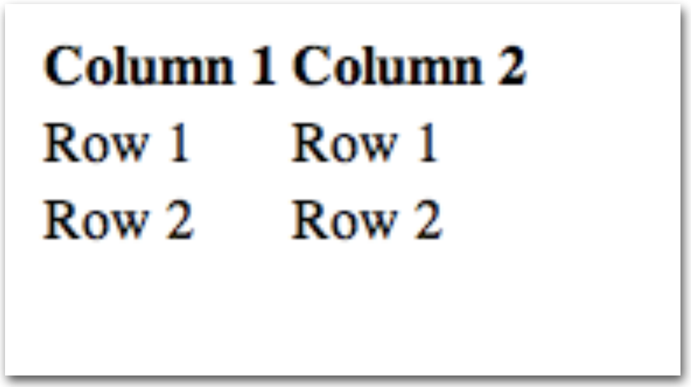
Tables

- Table tag: **<table>**
- Table row: **<tr>** .. **</tr>**
- Table header cell: **<th>** .. **</th>**
- Table data cell: **<td>** .. **</td>**

Table Example:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset=utf-8>
      <title>Table Example</title>
   </head>
   <body>
      <table>
         <tr>
            <th>Column 1</th>
            <th>Column 2</th>
         </tr>
         <tr>
            <td>Row 1</td>
            <td>Row 1</td>
         </tr>
         <tr>
```

```
            <td>Row 2</td>
            <td>Row 2</td>
         </tr>
      </table>
   </body>
</html>
```



Column 1	**Column 2**
Row 1	Row 1
Row 2	Row 2

Links

- **Anchor tag: <a href=**URL> .. **</a>**
  - Creates a link which will load the resource at the URL when the link is clicked.
- **Absolute URL:**
  - http://griffith.edu.au/ict/home.html
- **Relative URLs:**
  - Same directory:

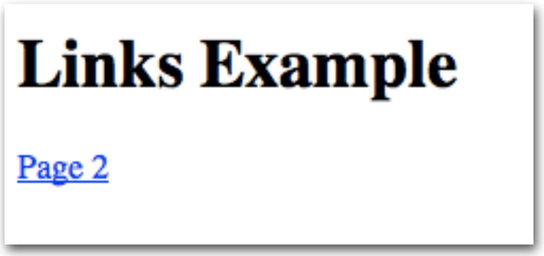
    <a href="home.html">
  - Subdirectory:

    <a href=" ict/home.html">
  - Parent directory:

    <a href="../home.html">

Links Example:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset=utf-8>
   <title>Links Example</title>
</head>
<body>
   <h1>Links Example</h1>
   <a href="page2.html">Page 2</a>
</body>
</html>
```



Links Example

Page 2

SECTION 6

Logical Styles

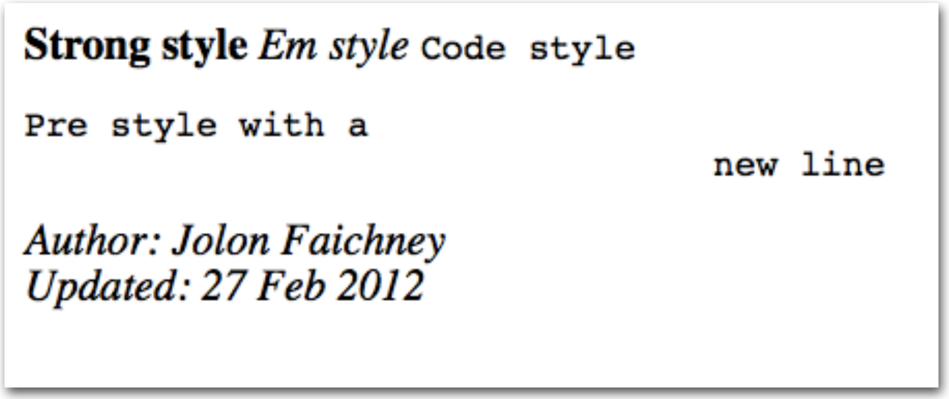
Logical instead of physical styles should be used in HTML, the style can be changed using CSS.

- **<strong>**
  - instead of <b>
- **<em>**
  - instead of <i>
- **<code>**
  - instead of <tt>
- **<pre>**
  - similar to <code> but also preserves whitespace.
- **<address>**
  - Defines the contact information for the author, usually in the header or footer.

Logical styles example:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset=utf-8>
      <title>Logical styles
example</title>
   </head>
   <body>
      <strong>Strong style</strong>
      <em>Em style</em>
      <code>Code style</code>
      <pre>Pre style with a
```

```
            new line</pre>
      <address>
         Author: Jolon Faichney<br>
         Updated: 27 Feb 2012
      </address>
   </body>
</html>
```



```
Strong style Em style Code style

Pre style with a
                           new line

Author: Jolon Faichney
Updated: 27 Feb 2012
```

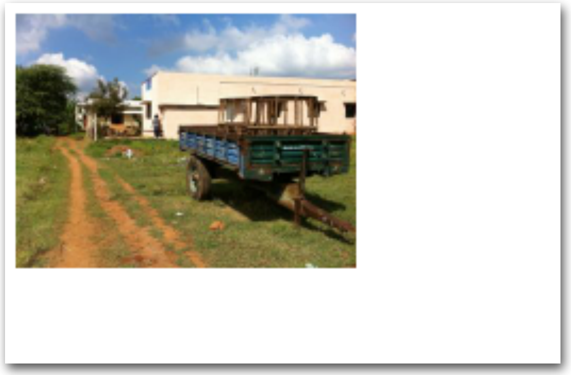
Images

<img src="photo.jpg" alt="Photo">

- **alt** is important for visually impaired users or if there is trouble loading the image.
- **width** and **height** may be specified, useful for scaling the image on screen:
  - <img src="photo.jpg" **width=50 height=50**>

Image Example:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset=utf-8>
      <title>Image Example</title>
   </head>
   <body>
      <img src="photo.jpg" width=160
height=120 alt="India">
   </body>
</html>
```



SECTION 8

CSS

- CSS - Cascading Style Sheets
- CSS allows the document *style* to be specified independent of the document *structure*.
- CSS can be placed in three locations:
    - External file
    - Embedded
    - Inline
- CSS can be placed in an **external** file ending with .css and linked using the <link> tag in the head of the document:

```
<link rel="stylesheet" type="text/
css" href="wp.css">
```

This is the recommended way to specify styles within a document.

- CSS can be placed within a <style> tag **embedded** in the <head> of the document:

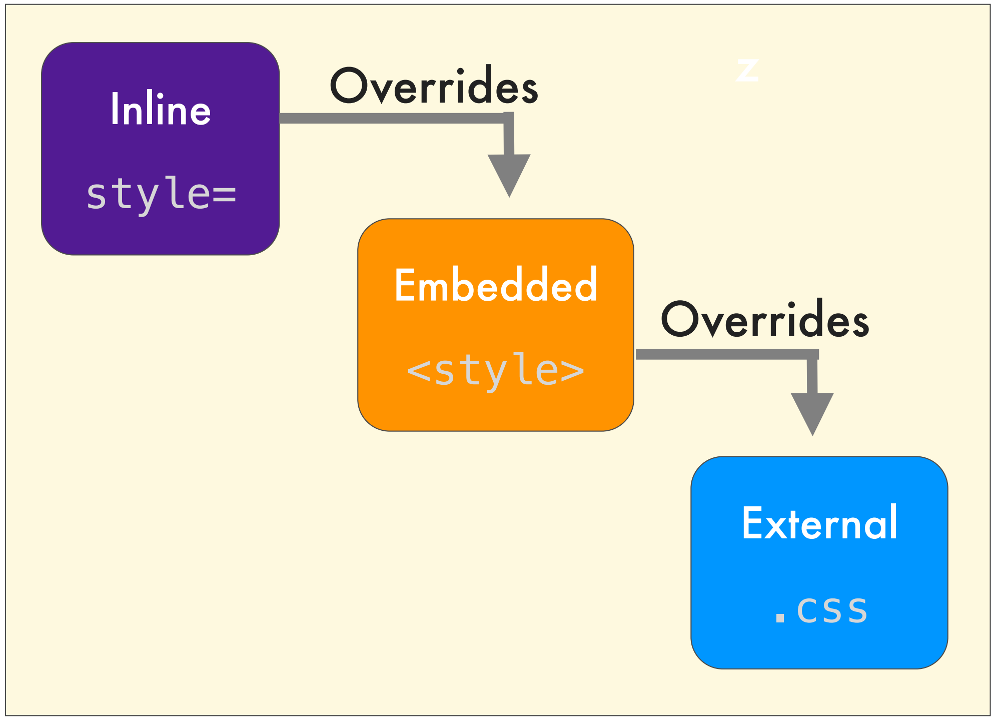
```
<style type=”text/css”>
   ...
</style>
```

- Style may be specified **inline** using the style attribute:

```
<h1 style=”Color: red”> ... </h1>
```

Cascading Style Sheets

- Styles cascade with external styles being overridden by embedded styles, and both being overridden by inline styles.



Common styles:

```
/* level 1 headings are in bold font */

h1 { font-weight: bold; }
```

```
p  { font-family: verdana, sans-serif;
color: blue; background-color: white; }

/* text in class="alert" elements has
colour red and size large */

.alert { color: red; font-size: large; }

/* text in paragraphs in elements with
class="body" have style italic */

.body p { font-style: italic; }

/* text in elements with id="footer"
have weight bold, style italic */

#footer { font-weight: bold; font-style:
italic; }

/* tables have solid thin blue borders
*/

table { border: solid thin blue; }
```

- Stylesheets are capable of much more than this, including the specification of absolute and relative positions of elements on a page, allowing them to replace the use of tables. In particular, the float position indicates that other elements may flow around this one.

- For example, style sheets can be used to specify multi-column layouts, avoiding the use of frames (which can't be bookmarked) and tables (which are very inaccessible).

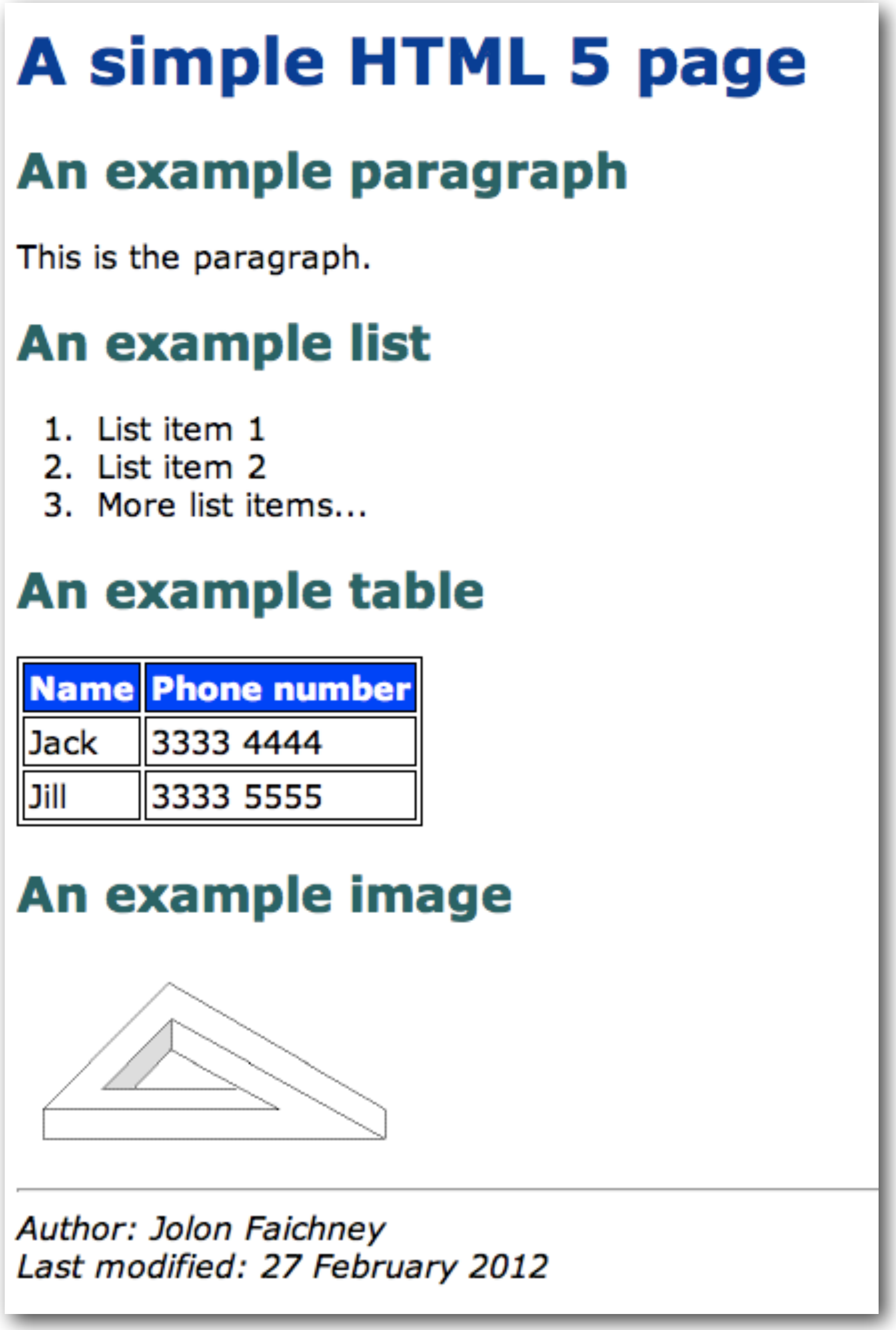
Simple HTML5 example

A simple HTML5 example is provided below:

http://www.ict.griffith.edu.au/teaching/2503ICT/Examples/simple-html5/

It demonstrates the following:

- A linked CSS file (wp.css)
- Headings
- Paragraph
- Ordered list
- Table
- Image
- Address block



A simple HTML 5 page

An example paragraph

This is the paragraph.

An example list

1. List item 1
2. List item 2
3. More list items...

An example table

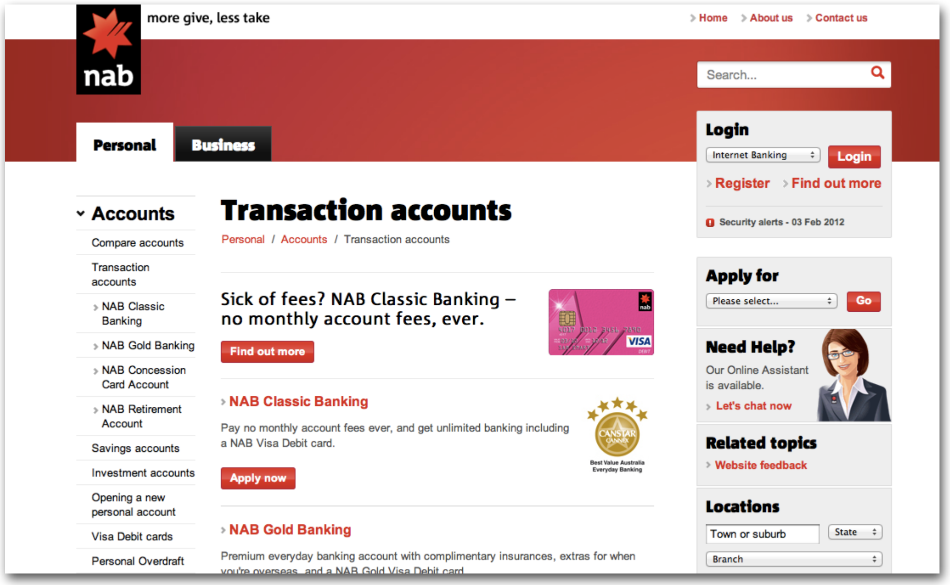
Name	Phone number
Jack	3333 4444
Jill	3333 5555

An example image

Author: Jolon Faichney
Last modified: 27 February 2012

SECTION 10

Page Layout



Sick of fees? NAB Classic Banking – no monthly account fees, ever.

Pay no monthly account fees ever, and get unlimited banking including a NAB Visa Debit card.

Premium everyday banking account with complimentary insurances, extras for when you're overseas, and a NAB Gold Visa Debit card.

- There is widespread agreement between popular Web site developers about page layout. They all have the following structure:
  - A full-width header containing site and user identification on the left and login and help links on the right, and optionally a second row containing links to very common operations.
  - A left column containing links to common operations.
  - A centre column containing the main text or body of the page.
  - An optional right column containing additional site-dependent information.
  - A full-width footer containing "fine print": information about the organisation, policies, site map, etc.
- All developers should follow this standard practice for page layout, even if using a single-column layout.

SECTION 11

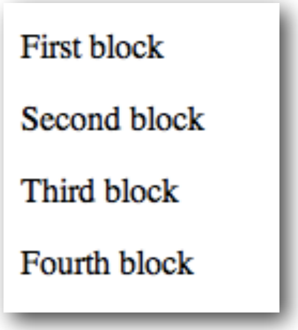
CSS Page Layout

- CSS Page Layout is covered by other courses.
- We will cover what is required in this course to achieve a multicolumn layout.
- Block elements vs. inline elements
    - HTML elements can be either block or inline.
    - Block elements are on their own line.
    - Inline elements flow and wrap from left to right and down the page.
- CSS can be used to change the layout of block and inline elements.
- The <div> tag is used to create an arbitrary block element which can contain other HTML elements.

Simple Div Example:

```
<!DOCTYPE html>


<html>
<head>
   <meta charset=utf-8>
   <title>Simple Div Example</title>
   <link rel="stylesheet"
type="text/css" href="ex1.css">
</head>
<body>
   <div id="firstblock">
      <p>First block</p>
   </div>
   <div id="secondblock">
      <p>Second block</p>
   </div>
   <div id="thirdblock">
      <p>Third block</p>
   </div>
   <div id="fourthblock">
      <p>Fourth block</p>
   </div>
</body>
</html>
```


First block

Second block

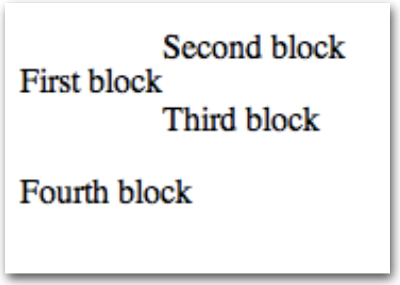
Third block

Fourth block

CSS Float

- HTML elements can be set to float left or right
- Float left CSS:

```
#firstblock {
    float: left;
}
```

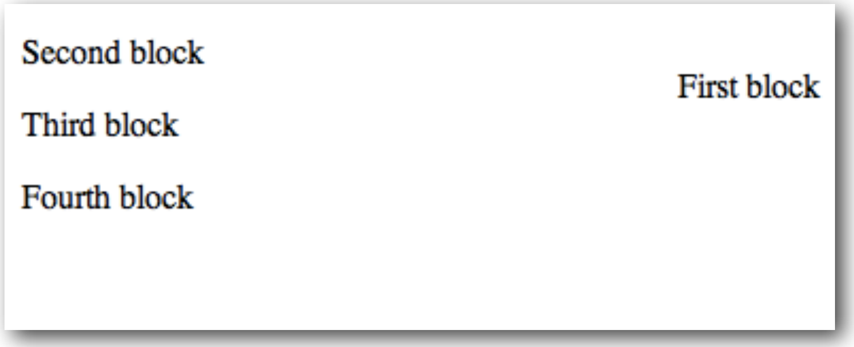

Second block
First block
Third block

Fourth block

- First block floats to the left.
- Second and third blocks wrap down the right of the first block.
- Fourth block continues underneath.

- Float right CSS:

```
#firstblock {
    float: right;
}
```



Second block

Third block

Fourth block

First block

Multicolumn Layout

- Multicolumn layouts can be fixed width or fluid width.
- Websites are commonly fixed width today due to the width of screens and difficulty in keeping the appearance consistent across different widths.
- A multicolumn layout can be achieved using the float left and right css.
- To make the content centred within the browser window the following CSS can be used:

```
#centerContent{
   width: 1000;
   margin: auto;
}
```

Two Column CSS Example:

```
<!DOCTYPE html>


<html>
   <head>
      <meta charset=utf-8>
      <title>Two column example</
title>
```

```
      <link rel="stylesheet"
type="text/css" href="ex2.css">
   </head>
   <body>
      <div id="centreContent">
            <div id="leftColumn">
                Left Column
            </div>
            <div id="centreColumn">
                Centre Column
            </div>
        </div>
   </body>
</html>
```

ex2.css:

```
#centreContent {
    width: 1000px;
    margin: auto;
}

#leftColumn {
    width: 300px;
    float: left;
    background: #ddf;
}

#centreColumn {
    background: #dfd;
}
```

Alternatively a fluid width example:

http://www.ict.griffith.edu.au/teaching/2503ICT/Examples/css/ex1/

Fluid width example CSS:

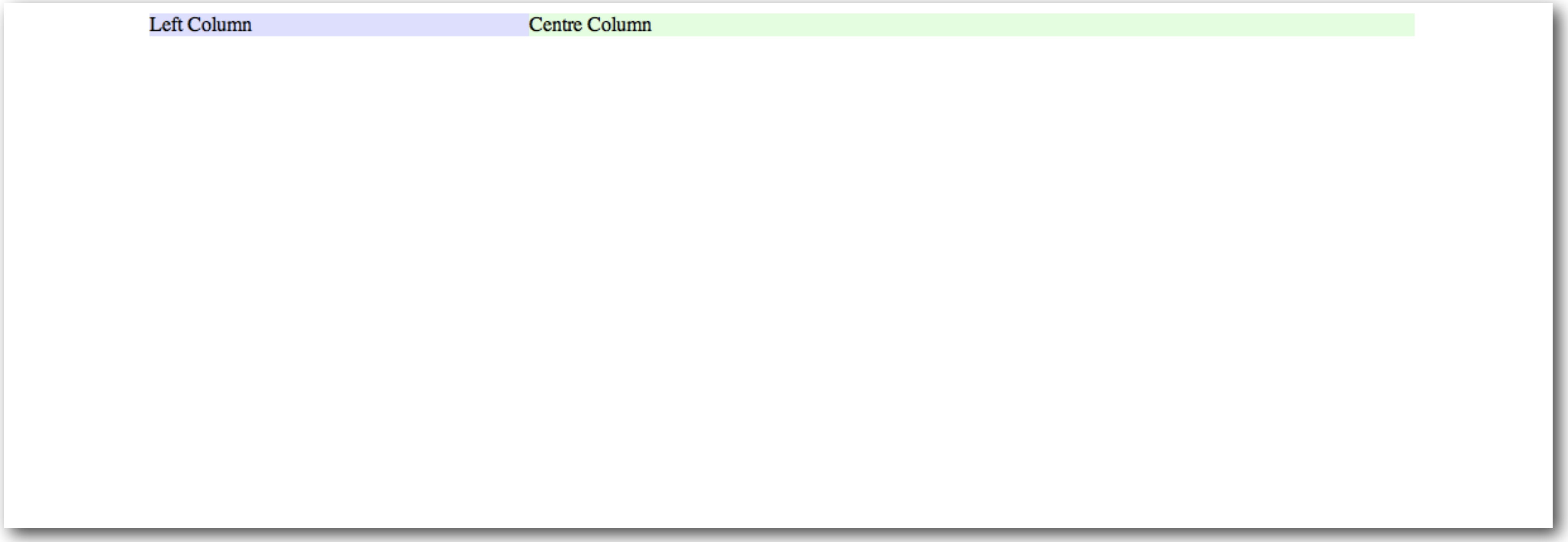
http://www.ict.griffith.edu.au/teaching/2503ICT/Examples/css/ex1/ex1.css

And a more complex example:

http://www.ict.griffith.edu.au/teaching/2503ICT/Examples/css/ex2/

Complex CSS example:

http://www.ict.griffith.edu.au/teaching/2503ICT/Examples/css/ex2/ex2.css



http://www.ict.griffith.edu.au/teaching/2503ICT/Examples/css/fixedTwoColumn/TwoColumnCSSExample.html

Forms

- Forms allow users to enter data which can be sent to a server.
- A form consists of two types of elements:
    - **Input controls** - that allow the user to enter information.
    - **Action buttons** - that allow the user to submit or reset the form.
- A form has an **action** which is the URL to send the form data to when the form is submitted.

HTTP Methods

- HTTP has two request methods: GET and POST.
- A form can be configured to use either method.
- If GET is used the form data is sent as URL parameters.
- If POST is used the form data is sent in the body of the HTTP request.

GET vs. POST:

- GET requests can be bookmarked.
- POST requests hide sensitive information.
- If a GET request is reloaded by the user the same action will occur twice.

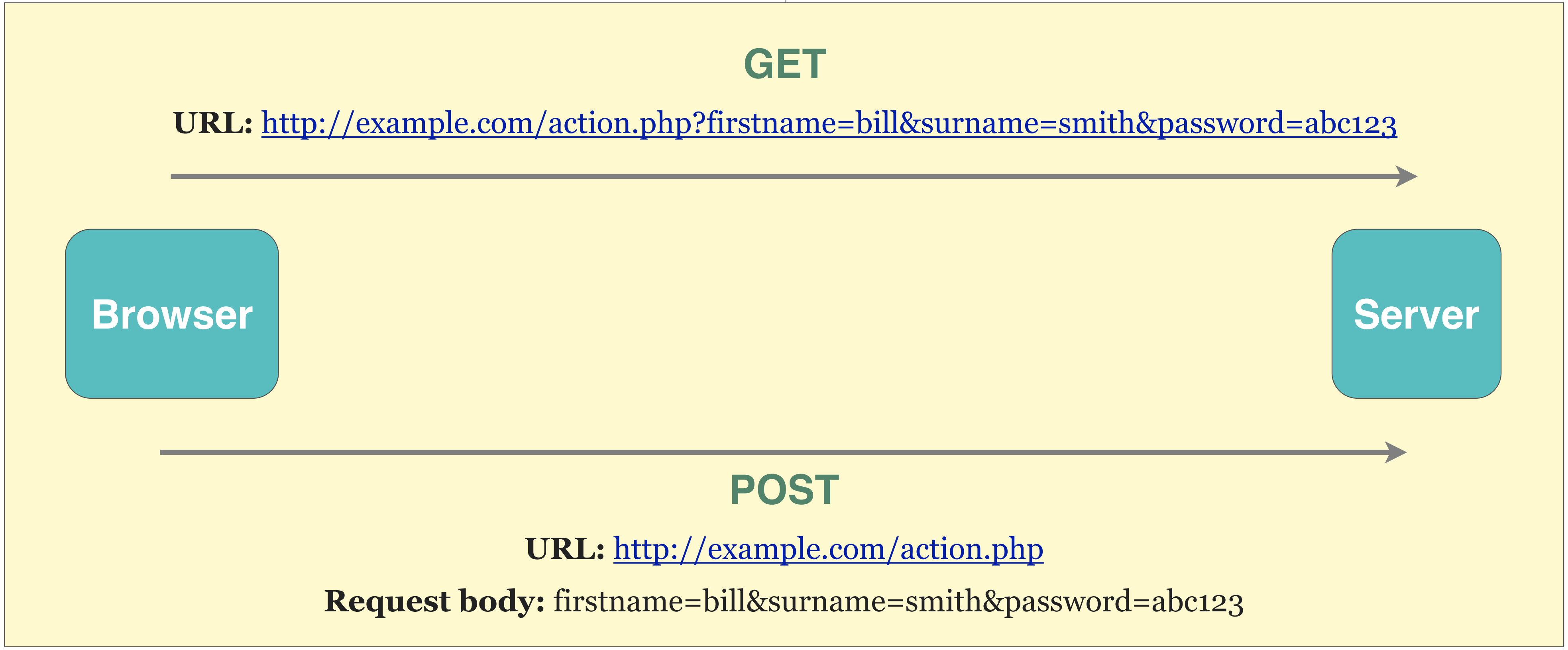
- A POST request reloaded by the user will generally cause the browser to ask the user if they want to resend the form data.

When to use GET:

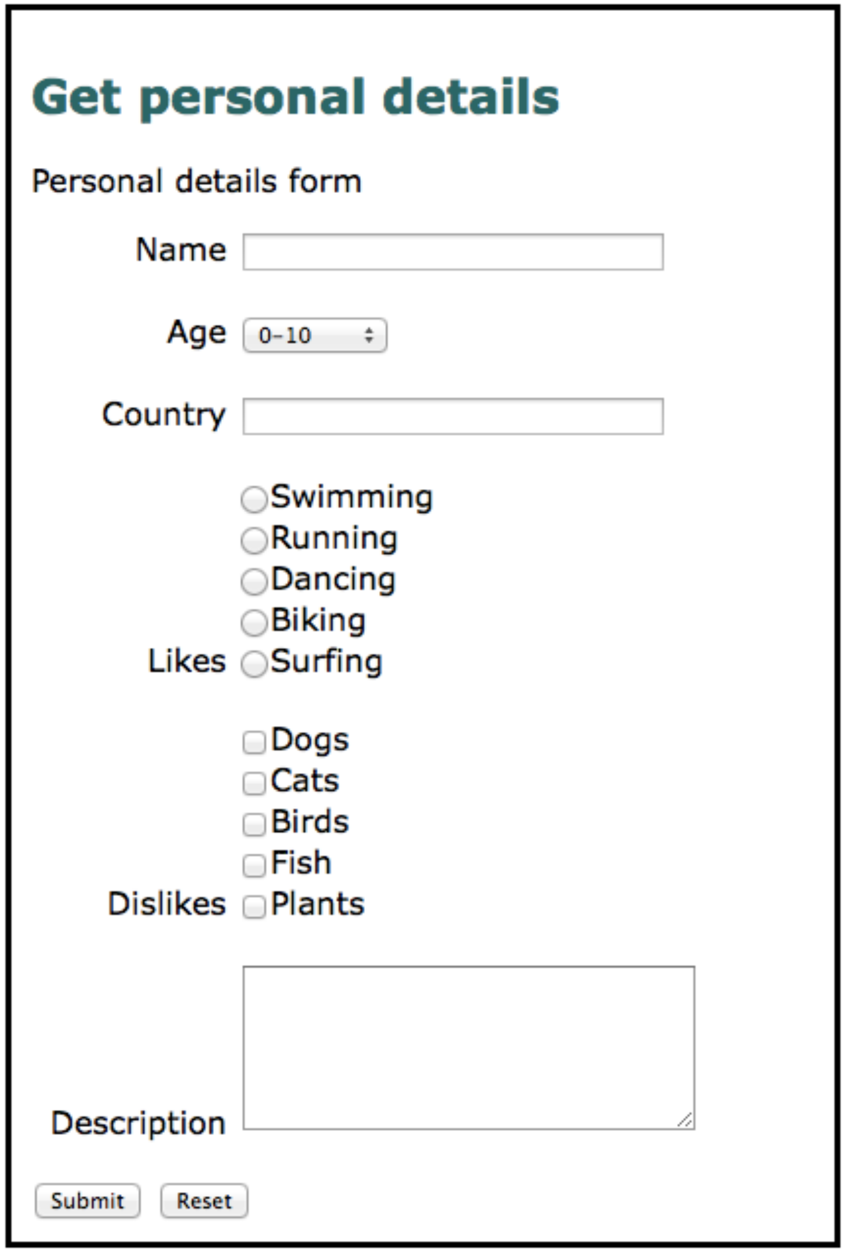
- When retrieving information.
- For example: search results.

When to use POST:

- When updating the server.
- For example: adding, deleting, or changing data.
- For uploading large amounts of data.
- For example: uploading images.


GET
URL: http://example.com/action.php?firstname=bill&surname=smith&password=abc123
Browser
Server
POST
URL: http://example.com/action.php
Request body: firstname=bill&surname=smith&password=abc123

Example Form: http://www.ict.griffith.edu.au/teaching/2503ICT/Examples/detailsCssForm/



Get personal details

Personal details form

Name	
Age	0-10
Country	
Likes	○ Swimming ○ Running ○ Dancing ○ Biking ○ Surfing
Dislikes	☐ Dogs ☐ Cats ☐ Birds ☐ Fish ☐ Plants
Description	

```
<!DOCTYPE html>
<html>
<head>
    <title>Get personal details</title>

    <meta charset="utf-8">

    <!-- External stylesheet -->
    <link rel="stylesheet" href="styles/wp.css" type="text/css">
</head>

<body><div class="content">
    <h2>Get personal details</h2>

    <p>
   <caption>Personal details form</caption>
    <form method="get" action="show_details.php">

            <label> Name </label>
```

```
        <input type="text" name="name" size=30> <br><br>
        <label> Age </label>
        <select name="age">
            <option>  0–10 </option>
            <option> 11–20 </option>
            <option> 21–30 </option>
            <option> 31–40 </option>
            <option> 41–50 </option>
            <option> Over 50 </option>
        </select> <br><br>
        <label> Country </label>
        <input type="text" name="country" size=30> <br><br>

        <label> Likes </label>
     <div class="choices">
        <input type="radio" name="likes" value="swimming">Swimming<br>
        <input type="radio" name="likes" value="running">Running<br>
        <input type="radio" name="likes" value="dancing">Dancing<br>
        <input type="radio" name="likes" value="biking">Biking<br>
```

```
            <input type="radio" name="likes" value="surfing">Surfing
         </div><br><br>
            <label> Dislikes </label>
      <!-- The value of dislikes is an array of checked values -->
         <div class="choices">
                <input type="checkbox" name="dislikes[]"
value="dogs">Dogs<br>
                <input type="checkbox" name="dislikes[]"
value="cats">Cats<br>
                <input type="checkbox" name="dislikes[]"
value="birds">Birds<br>
                <input type="checkbox" name="dislikes[]"
value="fish">Fish<br>
                <input type="checkbox" name="dislikes[]"
value="plants">Plants
         </div><br><br>
            <label> Description </label>
            <textarea name="description" rows=6 cols=30></textarea><br><br>
            <input type="submit" name="submit" value="Submit">
            <input type="reset"  name="reset"  value="Reset">
      </form>
```

```
</div></body>
</html>
```

wp.css additions:

```
.content{
   width: 400px;
   margin: auto;
   border-style: solid;
   border-color:black;
   padding: 10px;
}

label{
   width: 100px;
   display: inline-block;
   text-align: right;
}
```

```
.choices{
   display: inline-block;
}
```

Types of form elements:

- text fields (<input type="text" size=nchars>), for single lines of text
- passwords (<input type="password" size=nchars>), for passwords
- text areas (<textarea cols=nchars rows=nrows>), for multiple lines of text
- radio buttons (<input type="radio">) for selecting one alternative
- checkboxes (<input type="checkbox">) for selecting multiple alternatives
- selections (<select>) for selecting from a drop-down list of options
- buttons (<button type="submit/reset/button">) for initiating actions or submitting the form
- files (<input type="file">) for sending files

Radio buttons and checkboxes:

- Radio buttons/checkboxes with the same name attribute will be a part of the same control.

SECTION 13

HTML Authoring Principles

- Write **standard** HTML. Refer to the W3C HTML home page (http://www.w3.org/) for details.
- Use **validators** to check that your HTML documents conform to the recommendations. For example, use the W3C Markup Validation Service (http://validator.w3.org) to check your HTML documents.
- Use HTML to describe the **structure** - and only the structure - of your documents. For example, use `<em>` not `<i>`, use `<strong>` not `<b>`, use `<code>` not `<tt>`, use `<pre>` not `&nbsp;`, don't use `<font>`.
- Use cascading style sheets (CSS) to describe the **presentation** of your documents.
- Use **metadata** to facilitate automated processing of your documents.
- Learn about the Web Accessibility Initiative (http://www.w3.org/WAI/) to make your HTML documents **accessible** to all regardless of disability or presentation device. Follow the guidelines (http://www.w3.org/TR/WCAG10/) and checklist (http://www.w3.org/TR/WCAG10/full-checklist.html).
- Write HTML documents for **reuse**. Include comments. Use indentation. Make the source readable. The HTML document prepared with your fancy WYSIWYG editor today will be edited by your colleague (or yourself) with notepad tomorrow!

Review Questions

- Which version of HTML will we be using in this course?
- What comes after !DOCTYPE in html5?
- Which is the preferred approach:
  - <b>Some important text</b>
  - <strong>Some important text</strong>
- What is the value of the `type` attribute of a HTML form *submit* button?