COMP1531: Week 5 – HTML & CSS

Isaac Carr

School of Computer Science and Engineering
UNSW

August 24th, 2018

Today

- The Internet
- HTML/CSS
- Very, very basic Flask

WEB PAGE CONSTRUCTION

Material adapted from the COMP1000 reference book “Web 101” by Wendy Lehnert & Richard Kopec, “New Perspectives on HTML, XHTML and XML – Comprehensive”, 3rd Edition by Carey, Patrick, and slides by Bill Wilson

Resources

Textbooks:

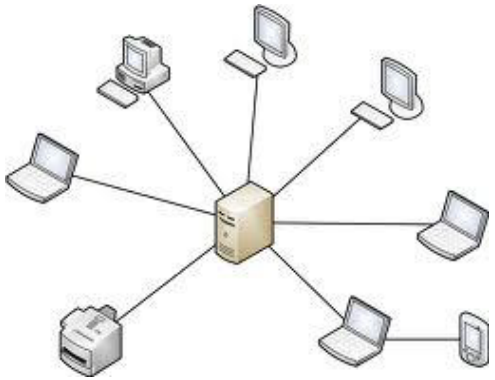
- "Web 101" by Wendy Lehnert & Richard Kopec
- "New Perspectives in HTML, XHTML and XML", 3rd Edition, Cengage Learning by Carey, Patrick (2010)

Web sites:

- World Wide Web Consortium ("W3C") – https://www.w3.org
- W3Schools – https://www.w3schools.com

Basic terms

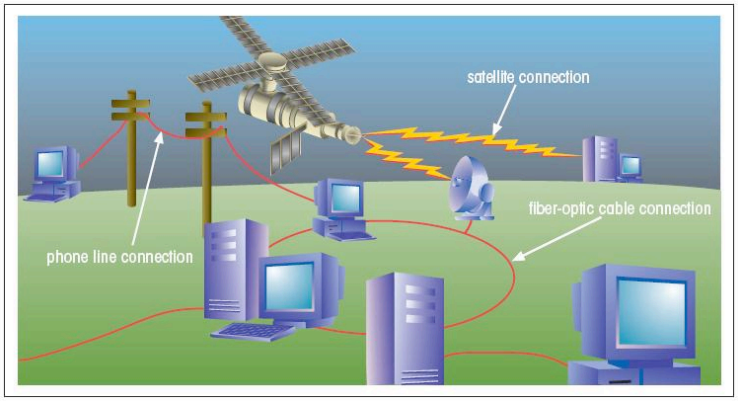
- A network is a structure linking computers together for the purpose of sharing information and services.
- Users typically access a network through a computer called a host or node.
- A node that provides information or a service is called a server.
- A computer or other device that requests services from a server is called a client.
- One of the most commonly used designs is the client-server network.
- The Internet is a massive network of networks, a networking infrastructure.



New Perspectives on HTML, XHTML and XML - Comprehensive, 3rd Edition by Carey

Internet

- Global network of connected computers, governed by a system of standards and rules
- Purpose of connecting computer is to *share information* – through variety of standardised rules and methods known as protocols
  - Emails, FTP
- Uses fibre-optic cables, satellites, phone lines, wireless access points and other telecommunication media



New Perspectives on HTML, XHTML and XML - Comprehensive, 3rd Edition by Carey

World Wide Web (WWW)

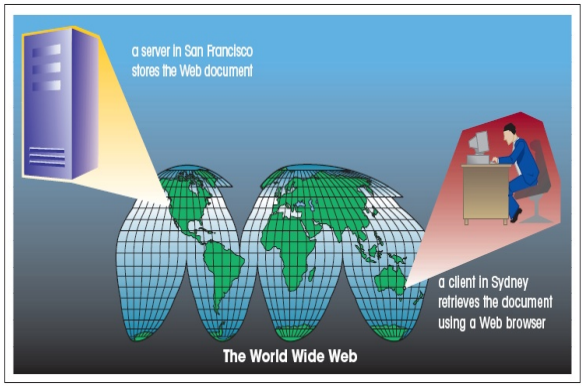
- The World Wide Web, or simply Web, is a way of accessing information over the medium of the Internet.

- It is an information-sharing model that is built on top of the Internet. where information is *not accessed in a linear fashion, but allows documents to be connected using* ***hypertext links*** *forming a huge "web" of connected information.*

- The Web uses the HTTP protocol, only one of the languages spoken over the Internet, to transmit data.

Definitions: Web Pages and Web Servers

- Each document on the World Wide Web is referred to as a Web page.
- Web pages are stored on Web servers, which are computers that make Web pages available to any device connected to the Internet.
- A Web browser retrieves the page from the Web server and renders it on the user's computer or other device.

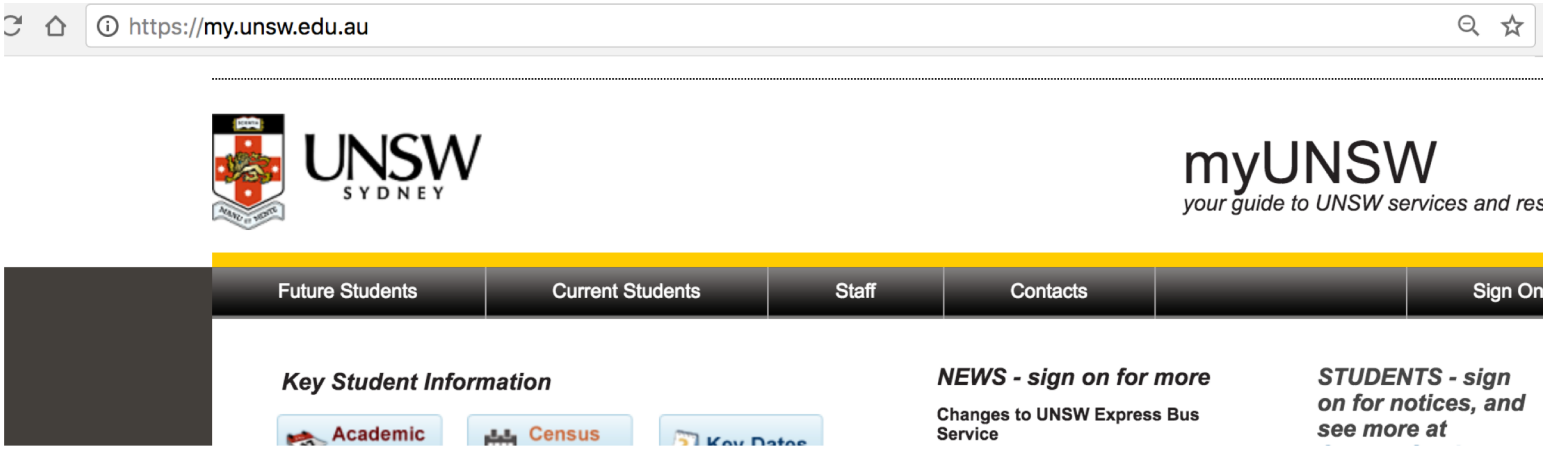


Text: New Perspectives on HTML, XHTML and XML - Comprehensive, 3rd Edition by Carey

Web Pages

- Written in HTML – *HyperText Markup Language*

- Web pages are stored on Web servers (HTTP servers), which are computers that run a special server software that enables communication between computers through the HTTP protocols and these servers make available any web page to any device connected to the Internet
  - e.g. Apache HTTP server, Microsoft IIS server



Web Pages

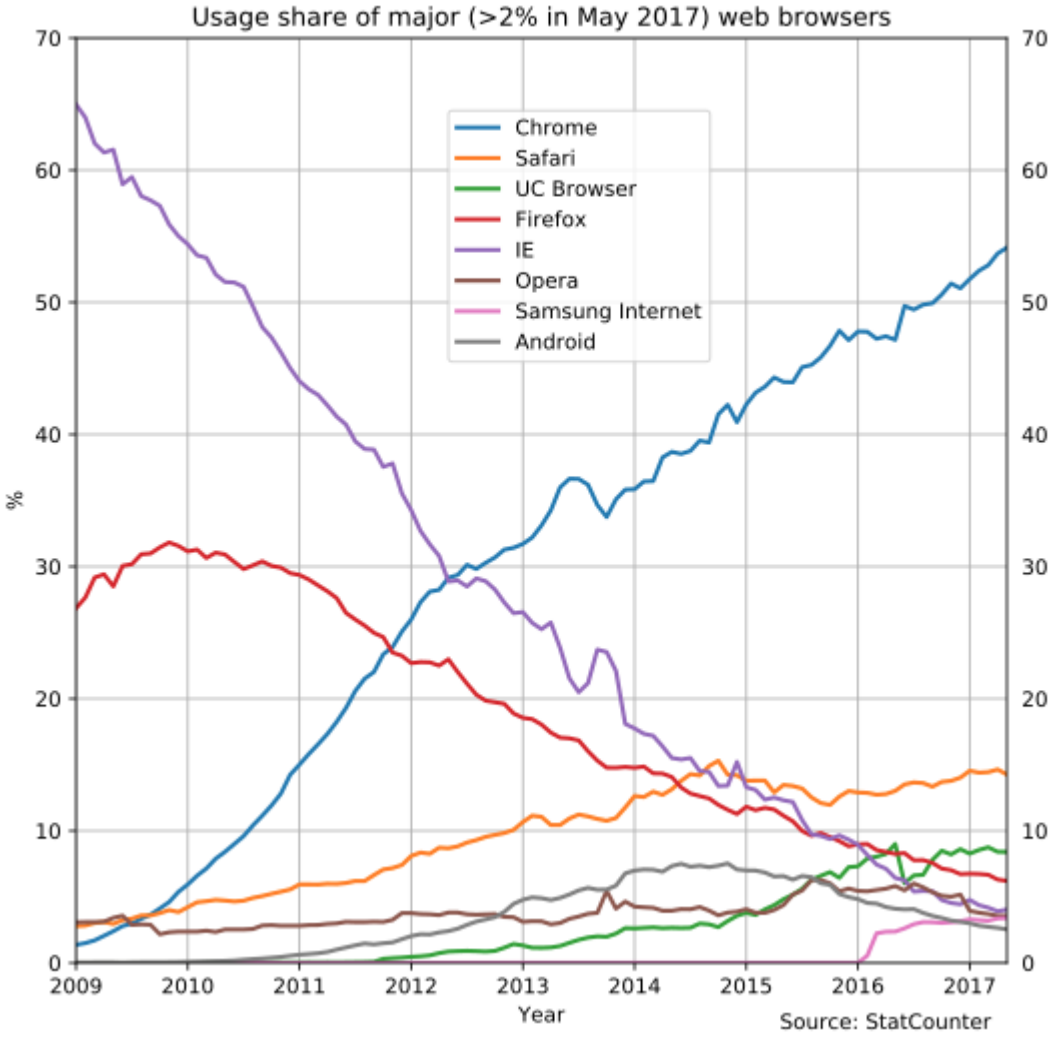
Each web page has a URL – *Uniform Resource Locator* – a special address

https://www.cse.unsw.edu.au/~cs1531/my_fun_page.html

More commonly seen as:

https://www.cse.unsw.edu.au/pages/fun/my_fun_page.html

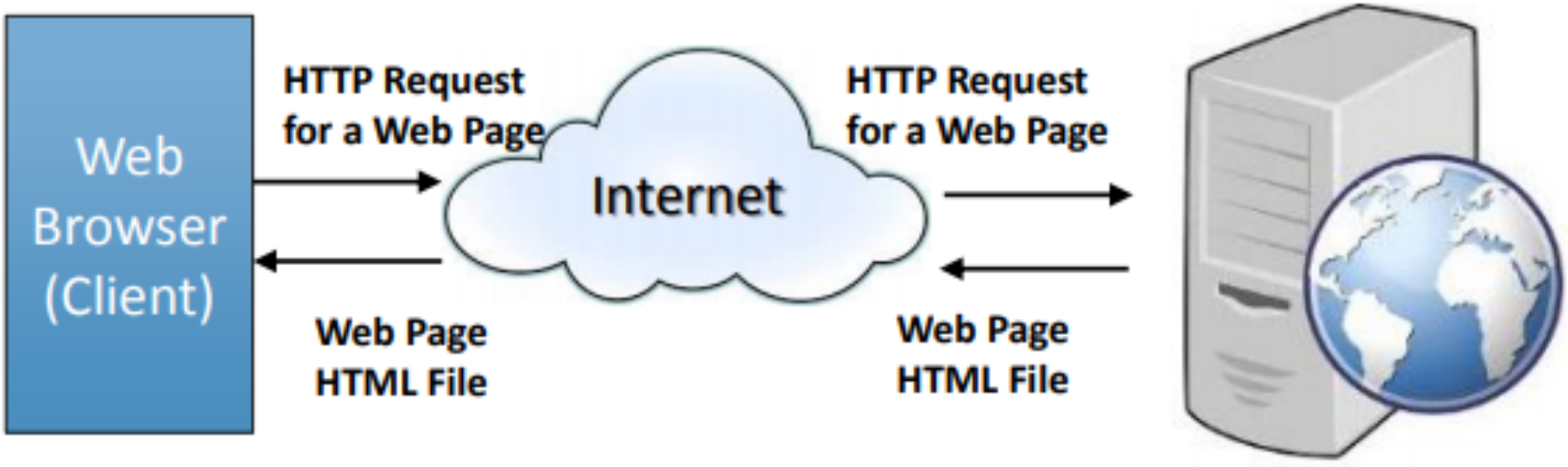
Web Browsers Today



https://en.wikipedia.org/wiki/File:Web_browser_usage_share,_May_2017.svg

Web Architecture

HTTP Request → **GET**, **POST**, PUT, DELETE



HTTP Response → 200, 404

How is a web page assembled?

- A client requests a web page by specifying the URL or clicking on a hyper link

- Browser sends a HTTP request to the server named in the URL and requests for the specific document

- The web server locates the requested file and sends a response.
  - If document is not found, an error message “404, Not found” is returned
  - If the document is found, the server returns the requested file to the browser

- The browser parses the HTML document and assembles the page
  - If the page contains images, the browser requests the server for the image, inserts the image into the document in the position indicated and displays the assembled page

A HTTP Request

HTTP Request Structure

```
1 GET /home.html HTTP/1.1
2 Host: www.yoursite.com
```

- A request message consists of:
  - Request line
    - *Get /home.html HTTP/1.1*
  - Headers
  - An optional message body
- Common HTTP Request methods:
  - HEAD, GET, POST, DELETE

HTTP Response

HTTP Response Structure

```
1 HTTP/1.1 200 OK
2 Date: Sun, 28 Jul 2013 15:37:37 GMT
3 Server: Apache
4 Last-Modified: Sun, 07 Jul 2013 06:13:43 GMT
5 Transfer-Encoding: chunked
6 Connection: Keep-Alive
7 Content-Type: text/html; charset=UTF-8
8 Webpage Content
```

A HTTP Response consists of:

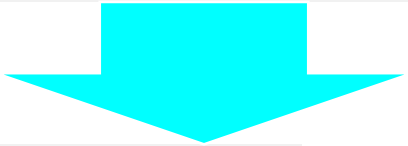
- A status line
- Headers
- Content (*optional)

Status lines:

- Success: 2xx
- Redirection: 3xx
- Client-Error: 4xx
- Server-Error: 5xx

HTTP Request & Response Example

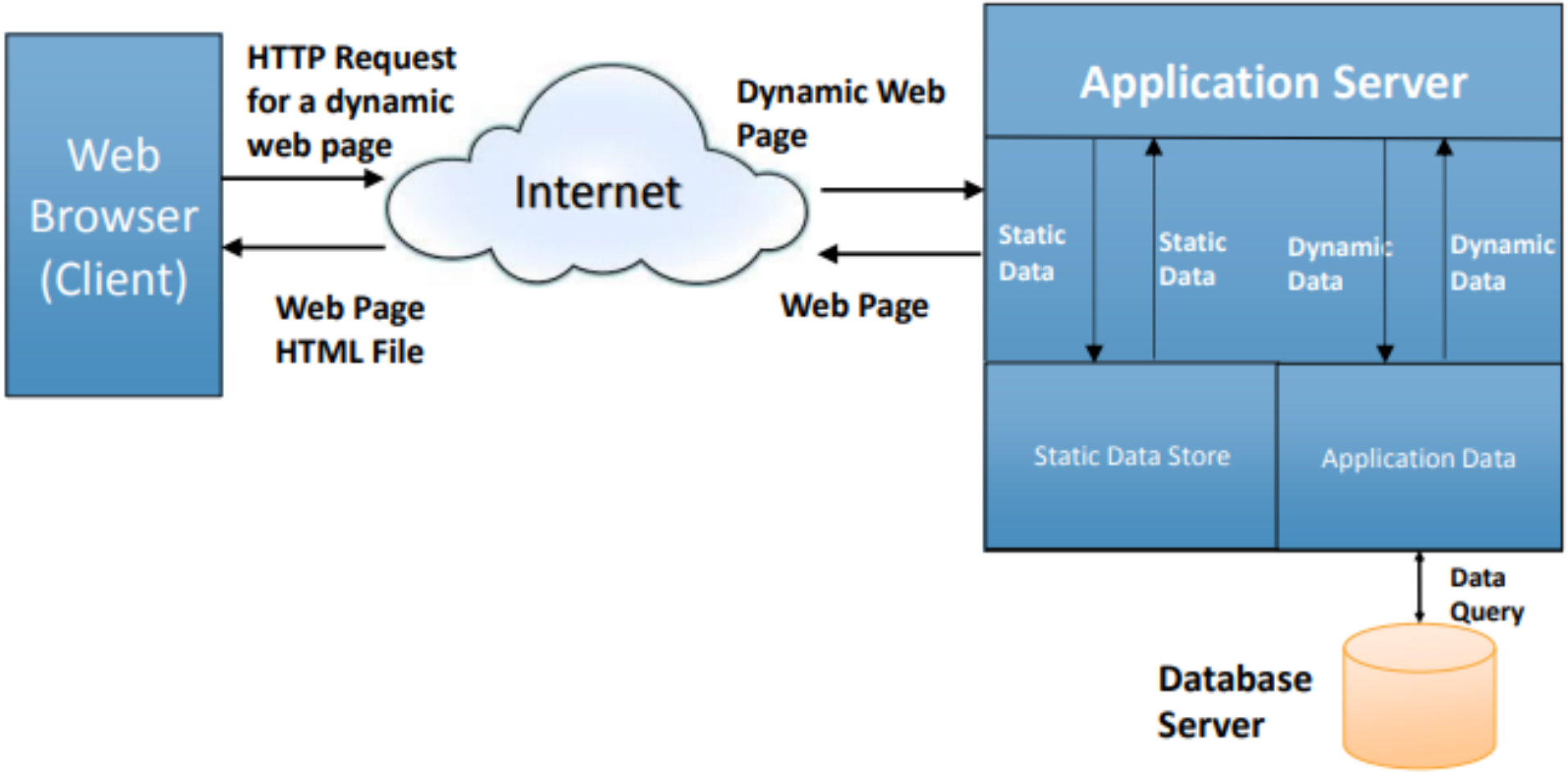
```
GET /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```



```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

```
<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Extended Web Architecture



Web Applications – Server v Client-Side Processing

- Server-Side Processing→ e.g. PHP, Perl, Flask, Ruby on Rails
  - Receives web request
  - Server processes user input, renders the dynamic web page to be returned to the client on browser

- Client-Side Processing → e.g. JavaScript
  - Processing needs to be "executed" by the Browser to:
    - Complete the request for the dynamic page (i.e. valid user input)
    - Create dynamic web page
  - More response UI and lowers the bandwidth cost

HTML – *HyperText Markup Language*

- Web pages are text files written in HTML, a platform-independent, markup language that describes the content and structure of the document

- The textual content is represented by plain text and the structure of the document is made up elements, which are distinct objects in the document like a paragraph, a title and are indicated in the document by HTML tags
  - <element> textual content </element> e.g., ,

- **Not** a programming language

- **NOT** a formatting language i.e., it does not specify how the content should be rendered. The browser on the user's devices parses the HTML tags and renders the content. Use styles for formatting.

HTML – *HyperText Markup Language* - Summarised

- Independent of platform
  - Not particular to Mac or Windows, or mobile or desktop
- Not programming language or formatting language
- Content is represented as plain text
- Structure is indicated by tags - Paragraph

And…

Offt… breathe out

Time for practical stuff!

HTML Tag

- HTML Documents are made of tag elements
  - E.G. - paragraph, <h1> - heading, - divisions
- Tells the Web Browser how we format the Web Page
- Most elements have an opening and closing tag
  - E.G. for paragraph
    - Opening:
    - Closing:
  - Whatever is in-between these tags will get formatted by the web browser as a paragraph
- You can nest tags:
  - E.G: Hi!
    - This is a paragraph that says **Hi** in bold

Structure of Web Page

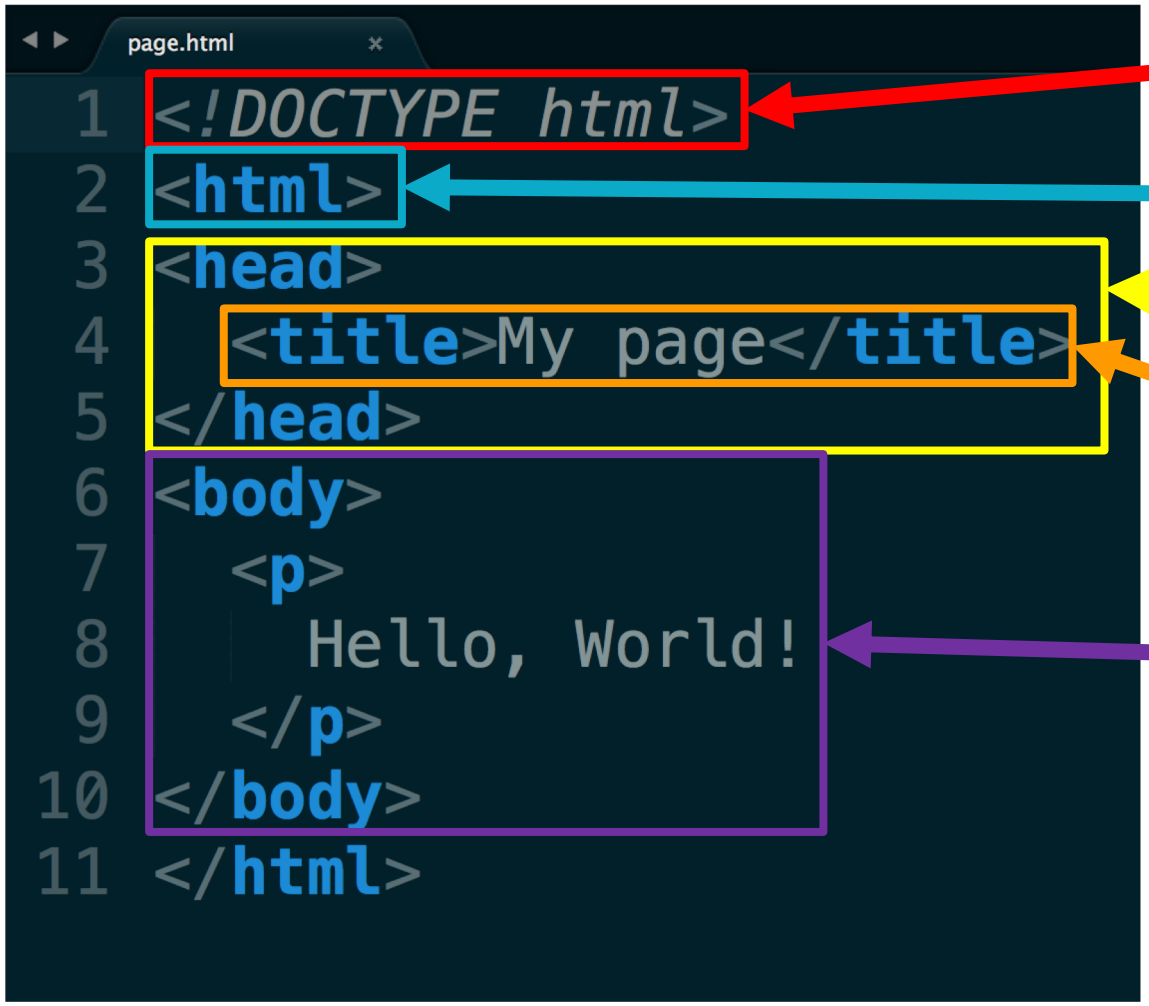
- An HTML document is divided into two main sections: the head and the body that appear inside the html tag.

- The root element: html
  - Everything goes inside this tag

- The head element:
  - It contains information about the document, for example the document title or the keywords.
  - The title element contains the page's title. A document's title is usually displayed in the browser's title bar (or as a title of the tab)
  - The content of the head element is not displayed within the Web page.

- The body element:
  - It contains all of the content to appear on the Web page.
  - It contain code that tells the browser how to render the content.

Web Page Structure



Declaration – tells web browser page is written in HTML

Root tag

Head

Title

Body

HTML Layout and Indentation

page.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>My page</title>
5  </head>
6  <body>
7    <p>
8      Hello, World!
9    </p>
10 </body>
11 </html>
```

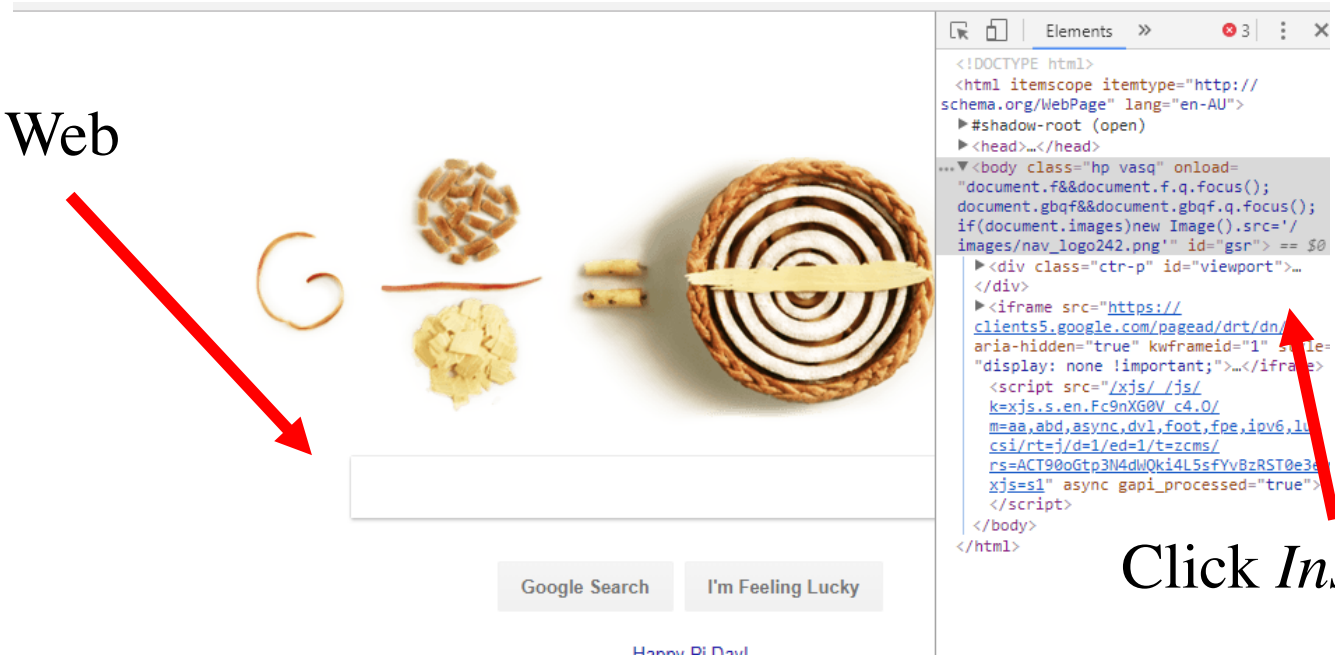
- ← Typical example of basic HTML layout
- You ***should*** use indenting
  - Easier to read
  - Easier to find bugs
  - Shows parts that are *inside* each other
    - "Levels"
  - Maintainable code

A web browser will still work (web page will be displayed) but if anything goes wrong or someone else needs to edit your work, it will be difficult.

Seeing the HTML Source

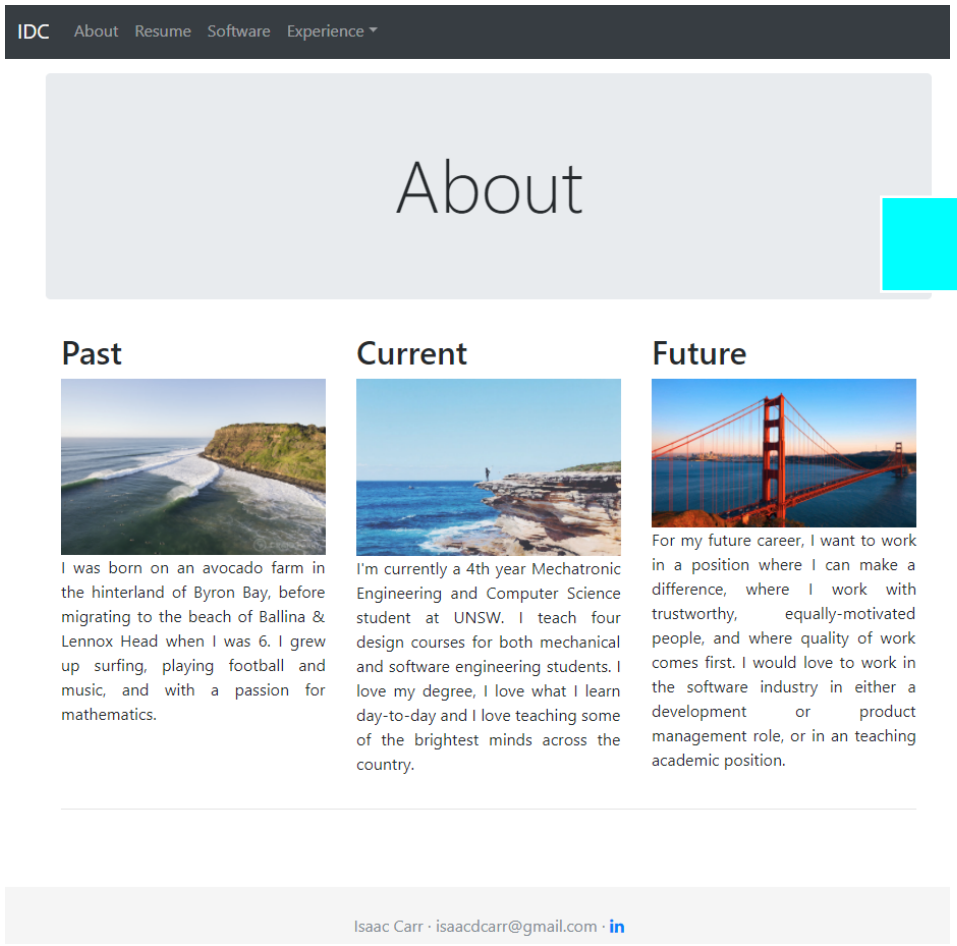
- Right click on any web page, and select "***View Page Source***"
  - Will show the HTML source code of any page
  - Can also press Ctrl+U on Windows
  - On Chrome, you can right click, and select "Inspect" which will show you the source, but in a fancy way. Stick to ***View Page Source***

Google Web page



Click *Inspect*

Page Source – Isaac's Website



```
<!DOCTYPE html>
<html>
<head>
    <title>About | IDC</title>
    <meta charset="utf-8">

    <!-- META TAGS -->
    <meta name="description" content="Website for isaacdcarr@gmail.com">
    <meta name="author" content="isaacdcarr@gmail.com">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- CSS -->
  <link rel="stylesheet" type="text/css" href="css/offcanvas.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css" integrity="sh
PsH8R72JQ3SOdhVi3uxftmaW6Vc51MKb0q5P2rRUpPvrszuE4W1povHYgTpBfshb" crossorigin="anonymous">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-awesome.min.css">
    <link rel="stylesheet" type="text/css" href="css/style.css">

    <!-- JS -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZ
crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper.min.js" integrity="sha384-
vFJXuSJphROIrBnz7yo7oB41mKfc8JzQZiCq4NCceLEaO4IHwicKwpJf9c9IpFgh" crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/js/bootstrap.min.js" integrity="sha384-
alpBpkh1PFOepccYVYDB4do5UnbKysX5WZXm3XxPqe5iKTfUKjNkCk9SaVuEZflJ" crossorigin="anonymous"></script>
</head>
<body>
    <nav class="navbar navbar-expand-md fixed-top navbar-dark bg-dark">
      <a class="navbar-brand" href="/">IDC</a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExampleDefault" aria-controls
aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse">
        <ul class="navbar-nav mr-auto">
          <li class="nav-item">
            <a class="nav-link" href="/about.html">About</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="/resume.html">Resume</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="/software.html">Software</a>
          </li>
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="projects.html" id="dropdown01" data-toggle="dropdown" aria-haspopup="t
            <div class="dropdown-menu" aria-labelledby="dropdown01">
              <a class="dropdown-item" href="/sunswift.html">UNSW Solar Racing Team, Sunswift</a>
              <a class="dropdown-item" href="/tutor.html">Teaching</a>
              <a class="dropdown-item" href="/cuberider.html">Cuberider</a>
            </div>
          </li>
        </ul>
      </div>
    </nav>

    <main role="main" class="container">
        <div class="jumbotron heading-about">
            <center>
                <h1 class="display-3">
                    About
                </h1>
            </center>
        </div>

        <div class="container">
      <div class="row">
        <div class="col-md-4">
          <h2>Past</h2>
          <img src="https://craigparryphotography.com/wp-content/uploads/LINES_LENNOX_HEAD.jpg" alt="Lennox Head" style="wid
          <p style="text-align: justify;">
```

Adding Comments

- The comment tag adds notes to your HTML code

<!– comment -->

- Comments are not displayed by the browser
- Comments can be spread over several lines.
- Comments are useful in documenting your HTML code for yourself and others.
  - So if someone else takes over maintaining a web page, you should not have to sit down with them and explain what you have done
  - So you remember yourself what you have done
- Shown in source code (Ctrl + U)

Marking Elements with Tags

- The core building block of HTML is the tag, which marks the presence of an element.
- A two-sided tag is a tag that contains some document content. General syntax for a two-sided tag:

  <element>content</element>

- A two-sided tag's opening tag ( ) and closing tag ( ) should completely enclose its content.
- Elements can contain other elements.
  - Tags cannot overlap

Welcome to COMP1531 

Welcome to COP1531 

Marking Elements with Tags

- Many tags can have attributes, which modify the behavior of the tag
  - E.G. alignment, line justification, block size
- Each attribute has a value enclosed in double quotes
- Attributes are specified within the opening tag to which they belong:
  - <element attribute="value">content</element>
  - E.G. content
- The closing tag cannot contain any attributes

Block-Level vs Inline Elements

- **Block-level** elements are elements/tags that contain content that is viewed as a distinct block within the Web page.
  - E.G. Headings, paragraphs, tables
  - The browser (usually) inserts a new line between each block-level element

- An **inline** element marks a section of text within a block-level element
  - They can contain other inline elements but can't contain block-level elements
  - E.G. Hi is an inline element

Import Block-Level Elements

- Headings: <hn>*content*</hn>
  - Where *n* is an integer from 1 to 6 and *content* is the text of heading.
  - <h1>This is a heading</h1>
  - h1 is the most significant heading, h2 is a subheading etc.

- Paragraphs: *content*
  - Paragraphs tag - creates a line space ahead of text.

- Division tag: *content*
  - Used to contain content

Block-level List Elements

- HTML supports three kinds of lists: **ordered, unordered,** and **definition**
  - You use an **ordered list** for items that must appear in a numerical order -
  - You use an **unordered list** for items that do not need to occur in any special order -
  - The **description list** contains a list of terms, each followed by the term's description - <dl>

- One **list** can contain another list
  - This is called a **nested list**

Lists

- **Unordered lists:**
  - Created using the <ul> tag
  - The items of the list are created using the <li> tag
- **Ordered lists:**
  - Created using the <ol> tag and has a type attribute
  - The items of the list are created using the <li> tag
- **Description lists:**
  - Created using the <dl> tag
  - It consists of term/definition pairs created using <dt>, <dd> tags

Other Block Elements

- <pre>…</pre> - Preformatted text – retaining all whitespace and special characters
- <address> … </address> - Contact information
- <blockquote>…</blockquote> - an extended quotation
- … - Used to group elements – important when we use CSS
  - We will be using plenty of divs!

Text Formatting – Inline Elements

- … **Defines bold text**
- … *Defines italicized text*
- … **Defines important text**
- … *Defines emphasized text*
- <small>…</small> Defines smaller text
- _… Defines subscripted text
- […] Defines superscripted text
- <ins>…</ins> Defines inserted text
- ~~…~~ ~~Defines deleted text~~
- <mark>…</mark> Defines marked/highlighted text
- <code>…</code> `Defines computer code text`
- <q>…</q> “Defines inline quotes”

http://www.w3schools.com/html/html_formatting.asp

Working with Empty Elements

- An empty element contains no content
- Empty elements appear in code as one-sided tags
  - E.G. <element>
- The one-sided tag to mark a line break is
  - E.G.
- The horizontal rule element places a horizontal line across the Web page
  - E.G. <hr>

Hyperlinks

- Hyperlinks create hypertext, which is the driving force of the web
- Hyperlinks can be used as:
  - Links (interdocument links) – *outside document*
  - Anchors (intradocument links) – *inside document*
- Links start at the source and point to a destination
- Link source may be text or an image, and the destination may be a file, a web page, program, image, video/audio file, etc.
- The source of the link may be shown as visited or unvisited

Hyperlinks

- A hyperlink is created by using the <a> tag.
  - link name here

- The most important attribute of the <a> tag is href
  - href defines where the link's destination can be found

- Example:
  - UNSW

Absolute v Relative URLs

- The URL (Universal Resource Locator) can be *absolute* or *relative*.

- Absolute URLs start with https (or sometimes ftp, mailto, https) and spell out the path to the document being linked, in full.
  - E.G. "http://www.cse.unsw.edu.au/~cs1000/17s1/courseOutline.html"

- Relative URLs often point to a file in the same folder/directory as the page currently being displayed.
  - Solution is here

Hyperlinks – Anchors

- Anchors use the <a> tag to link different sections of the same web page.
- The creation of an anchor requires
  - an element with an id to be defined
    - E.G. <p id="aims">
  - A link to that location in the file
    - <a href="#aims">
- Note the "#" in the href attribute with the same value "aims" which can refer to the top of the same document
- Try: https://www.w3schools.com/html/tryit.asp?filename=tryhtml_links_bookmark

Images

- To include an image in the page, use the <img> tag and the src attribute that specifies where the image is saved
  - Example:
    - <img src="images/night.gif">
  - or
    - <img src="images/night.gif"> </img>

- Other attributes include alt border width height align hspace vspace etc.

Image Attributes

- `src` – file name of the graphic
- `height` – height of the graphic in pixels or %
  - `E.G. height="400" or height="40%"`
- `width` – width of the graphic in pixels
  - Use `height` and `width` attribute to cause the browser to load efficiently otherwise the browser may shift the image position after loading
- `alt`
  - Configures alternate text content for browsers to display
  - Useful for spider programs and vision-impaired people

Image Links

To create an image link use an anchor element to contain an image element

```
1  <a href="https://www.unsw.edu.au">
2    <img
3    src="images/example.png"
4    alt="my photo"
5    height="100%"
6    width="100">
7  </a>
```

Graphics

- Graphics can add a ‘wow’ factor to web pages BUT they can make web pages very slow to load.
- Web browsers load documents, line-by-line, starting at the top of the document. The top of an image begins to display after 50% of it has been read by the browser.
- Deciding which type of image to use depends on the balance between image type, its size and other factors like compression of the image.
- Graphic types commonly used on Web pages:
  - GIF (Graphics Interchange Format)
  - JPEG, JPG (Joint Photographic Expert Group)
  - PNG (Portable Network Graphic)

Working with GIF Images

- **GIF (Graphics Interchange Format)** is the most commonly used image format on the Web
- Compatible with virtually all browsers
- GIF files are limited to displaying 256 colours
- Often used for graphics requiring fewer colours, such as clip art images, line art, logos, and icons
- Images that require more colour depth, such as photographs, can appear grainy when saved as GIF files.

Text: New Perspectives on HTML, XHTML and XML - Comprehensive, 3rd Edition by Carey

Working with GIFs #2

- Best used for line art and logos
- Maximum of 256 colors
- One color can be configured as transparent
- Uses lossless compression
- Can be animated
- Can be interlaced, progressively displayed 13% to 25% to 100%

Text: New Perspectives on HTML, XHTML and XML - Comprehensive, 3rd Edition by Carey

JPEG/JPG

- JPEG stands for Joint Photographic Experts Group
- Supports up to 16.7 million colours
- Most often used for photographs and other images that cover a wide spectrum of colour
- Usually smaller than their GIF counterparts
- Best used for photographs
- Cannot be animated
- Cannot be made transparent
- Interlaced progressive JPEG

Text: New Perspectives on HTML, XHTML and XML - Comprehensive, 3rd Edition by Carey

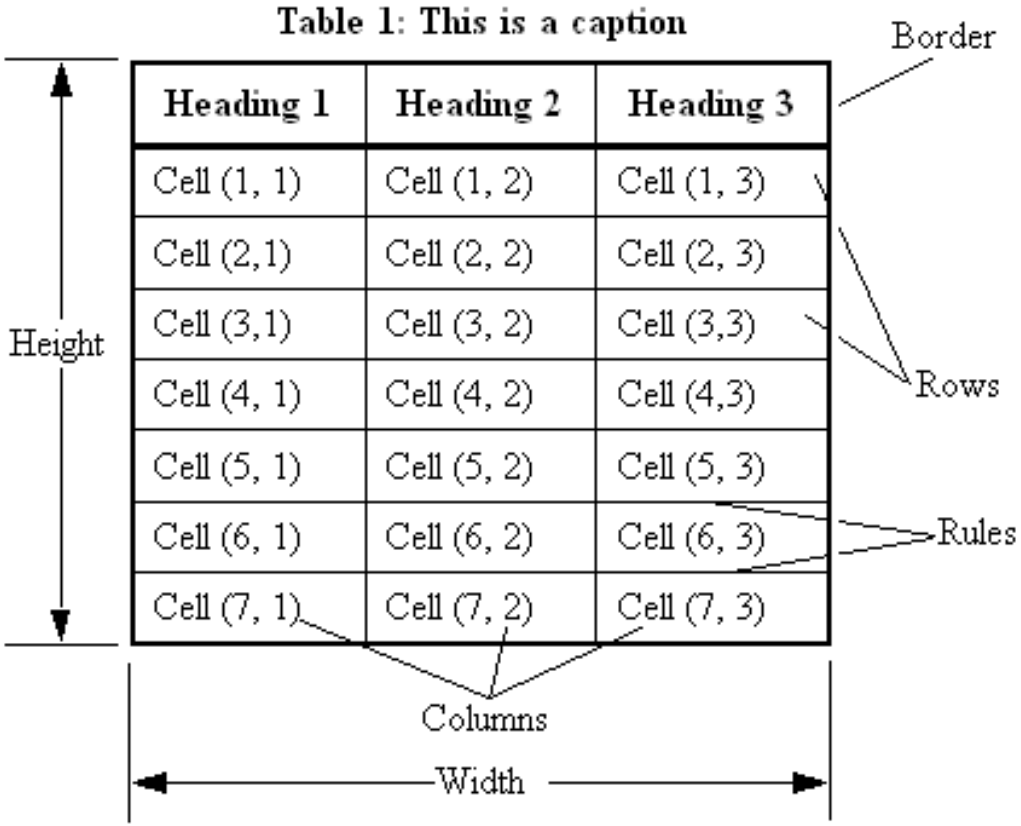
PNG

- A file format called PNG (Portable Network Graphics) has been gaining wider acceptance
- PNG files use a free and open file format and can display more colours than GIFs
- Support millions of colours
- Support interlacing
- Use lossless compression
- Browser support is growing

Text: New Perspectives on HTML, XHTML and XML - Comprehensive, 3rd Edition by Carey

Tables

- Tables can be used to organise and structure content on a web page



Text: New Perspectives on HTML, XHTML and XML - Comprehensive, 3rd Edition by Carey

Web Tables

- Each table in a Web page follows a basic structure consisting of the table element and a collection of table rows nested in the table element and table data nested in each row

```
1  <table>
2    <tr>
3      <td> Cell(1,1) </td>
4      <td> Cell(1,2) </td>
5    </tr>
6    <tr>
7      <td> Cell(2,1) </td>
8      <td> Cell(2,2) </td>
9    </tr>
10 </table>
```

Web Tables

- Can optionally use Table Headings for the first tow
  - `<th>`

```
<table>
    <tr>
        <th>zID</th>
        <th>Name</th>
    </tr>
    <tr>
        <td>z555</td>
        <td>Isaac</td>
    </tr>
    <tr>
        <td>z637</td>
        <td>Matt</td>
    </tr>
</table>
```

Web Tables

- `<table>` - Encloses all other table tags
- `<caption>` - Assigns a title to the table
- `<tr>` - Creates a table row
- `<th>` - Creates a header cell
  - Attributes: `abbr, headers, rowspan, colspan`
- `<td>` - Creates data cells
  - Attributes: `abbr, headers, rowspan, colspan`

Table Structures and Attributes

- Table attributes can be broadly classified as:
  - Non-cell attributes – control properties of the whole table
    - E.G. caption, border
  - Cell attributes – control properties of an individual cell
    - E.G. column span, rowspan, cell padding, cell spacing

Creating a Table Border

- To add a border to a Web table using HTML, use the border attribute

```
<table style="border: valuepx, linetype,
                color;"> </table>
```

  - where *value* is the size of the border in pixels.

- This style attribute will need to be defined for <td> to give the table internal borders.

Text: New Perspectives on HTML, XHTML and XML - Comprehensive, 3rd Edition by Carey

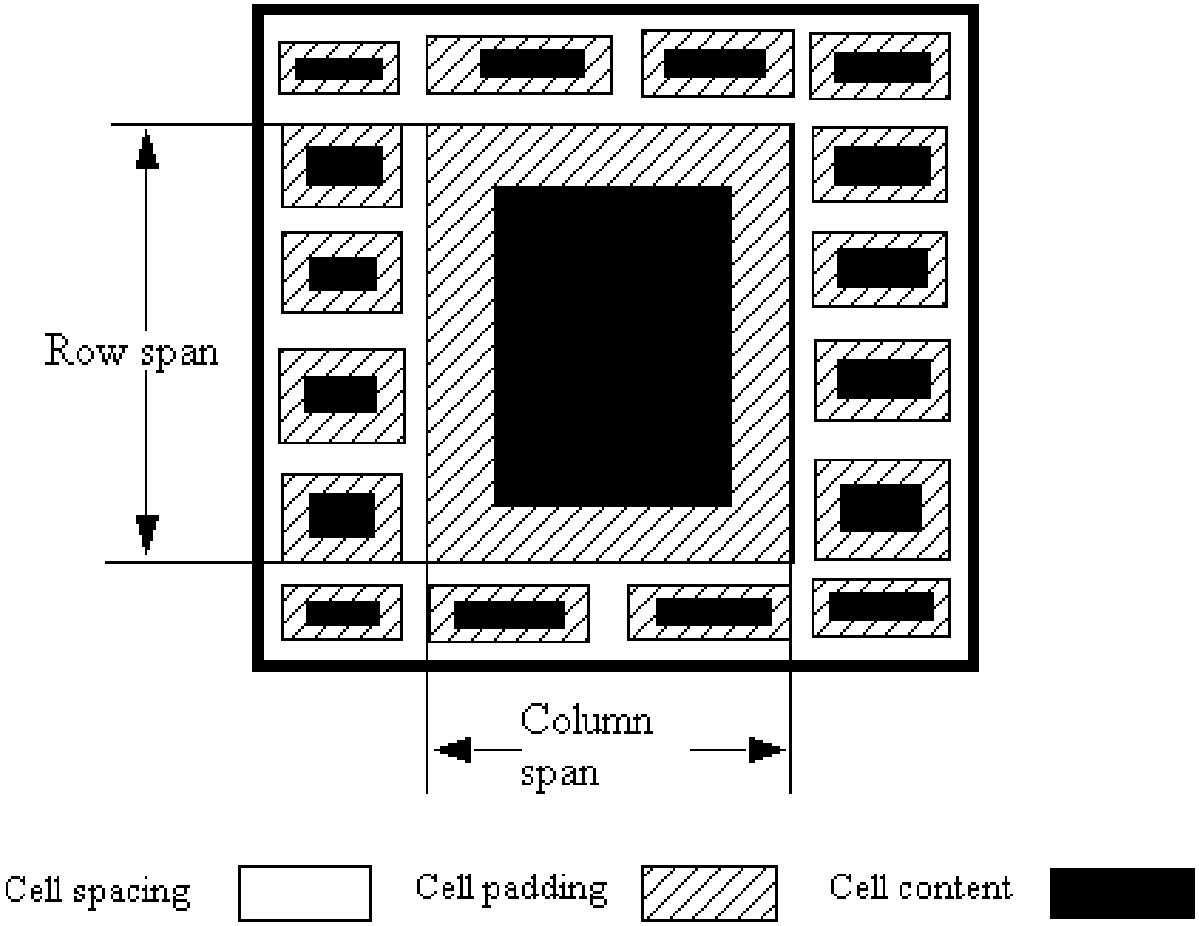
Creating a Table Caption

- To create a table caption, add the caption element directly below the opening <table> tag with the syntax

  <caption>content</caption>

  – where *content* is the content of the table caption

Cell Structure and Attributes



Text: New Perspectives on HTML, XHTML and XML - Comprehensive, 3rd Edition by Carey

Spanning Rows and Columns

- A spanning cell is a single cell that occupies more than one row or one column in the table
- To create a table cell that spans several columns, add the attribute

  colspan="value"

  – to the cell, where *value* is the number of columns covered by the cell
- To create a table cell that spans several rows, add the attribute

  rowspan="value"

  – to the cell, where *value* is the number of rows covered by the cell

Text: New Perspectives on HTML, XHTML and XML - Comprehensive, 3rd Edition by Carey

Formatting Tables with HTML Attributes

- To define the padding within table cells, add the style

  <table style=“ padding: value;"> ... </table>

  – to the table element, where *value* is the size of the padding space in pixels

- To define the space between table cells, add the style

  <table style=“border-spacing: value;"> ... </table>

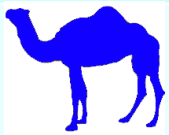
  – to the table element, where *value* is the space between table cells in pixels

Text: New Perspectives on HTML, XHTML and XML - Comprehensive, 3rd Edition by Carey

HTML Character Codes

- Typing < is hard when it indicates a new tag
- HTML character codes are of the form &*name*; or &*number*; here are some more:
  - < is <
  - > is >
  - & is &
  -   is non-breaking space
  - → is ®
  - α is the Greek letter α
- more at http://www.cse.unsw.edu.au/~billw/symbols.html

Summary of Basic HTML elements

This HTML	produces this…
text	paragraph with text in it
- text1 - text2	• text1 • text2
text1 text2	1 text1 2 text2
	
Heading	**Heading**
> & < ≤ ö etc.	> & < ≤ ö etc.
x² x₁	x^2 x_1
x x	*x* **x**
text1 text2	text1 text2

Deprecated Methods and Alternatives

- We are conforming with **HTML5** – which means old elements and attributes have become "deprecated" methods.
- Phasing out formatting commands.
  - E.G.
    - HTML4: <p align="center">
    - HTML5: <p style="text-align: center;">
- Phasing out proceeds *very* slowly, as existing HTML out there on the web depends on the deprecated stuff.
- The current preferred method is to use something like:
  red stuff
  - rather than red stuff
- This style=… scheme is part of **CSS**.

CSS (Cascading Style Sheets)

- CSS, or Cascading Styles Sheets, is a way to style and present HTML. Whereas the HTML is the meaning or content, the style sheet is the presentation of that document.

- Styles have a format of ‘property: value’ and most properties can be applied to most HTML tags.

CSS (Cascading Style Sheets)

- CSS style specifications can be put in three places:
  - **Inline style attributes** (wrapped around the content) blue stuff
  - **Internal HTML style element** in <head>…</head> <style type="text/css"> em {color: red; } </style>
  - **External Style sheet** (.css file) referenced in <head> of html <link type="text/css" rel="stylesheet" href="*my*.css"> The style sheet might specify that <p> text is purple.
  - There are also default styles (defined by the browser).

Inline Style Attributes

- Inside the tag that you are formatting
- Usually used for one off styling
  - E.G. if **zID** is red in this document and nowhere else

Internal HTML Style Element

- Used if focussed on using one document – i.e. don't want to also have a CSS document

External CSS Document

- Preferred method

```
<head>
  <title></title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <table>
    <tr>
      <td>zID</td>
      <td>Name</td>
    </tr>
```

- External CSS File

```
p {
    color: red;
}
```

Linking CSS Stylesheets to your HTML

- To link your .css file, called say my.css, to your .html file, you need to insert the following between <head> and </head> in your html, say after the <title> … </title> element, and before the <style>…</style> element, if any:

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

- Naturally, you'd replace *my*.css by whatever name you choose for your .css file.

CSS

- So, the following HTML …

```
<body>
This is black.
<p>This is purple
        <em>but this is red and
        <span style="color: blue; ">
        this is blue.</span> </em>
</p>
</body>
```

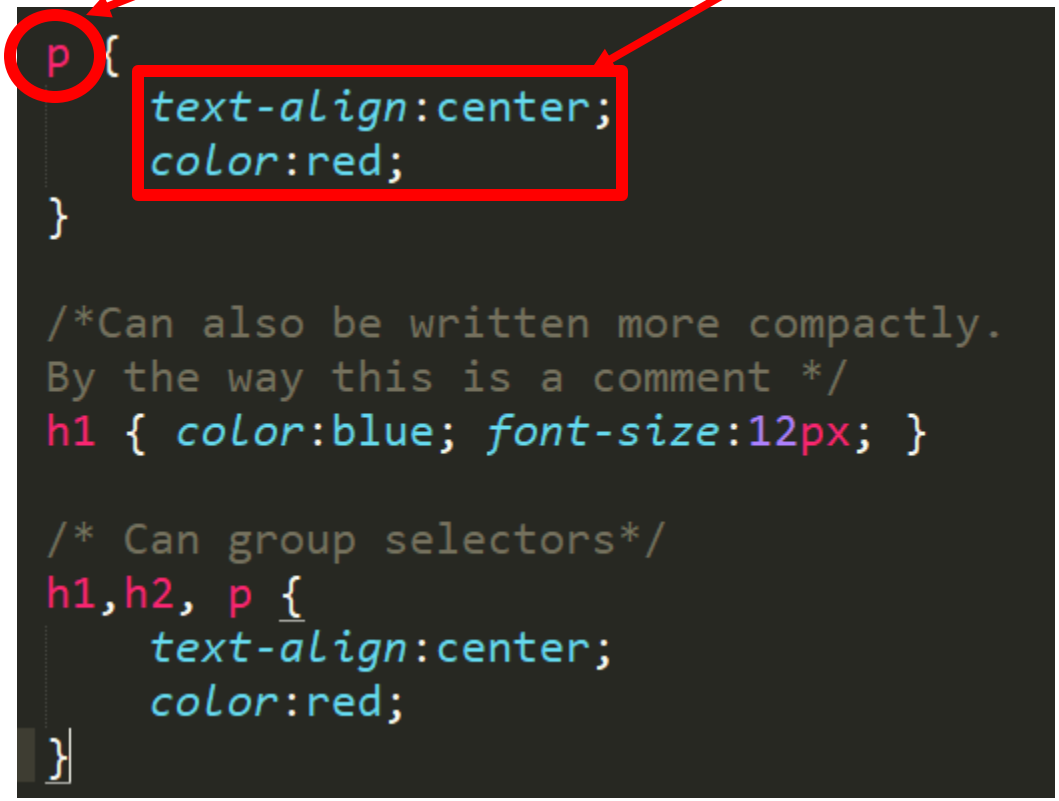
- produces (assuming my.css says <p>is purple and is linked and that we have set to red using an internal style element):

This is black.

This is purple *but this is red and this is blue.*

CSS Syntax

- A CSS rule set consists of a selector and a declaration block



```
p {
    text-align:center;
    color:red;
}

/*Can also be written more compactly.
By the way this is a comment */
h1 { color:blue; font-size:12px; }

/* Can group selectors*/
h1,h2, p {
    text-align:center;
    color:red;
}
```

CSS

- CSS is *case sensitive*.
- Some style properties can take a sequence of values:

```
p{
        font-family: “Times New Roman”, Times, serif;
}
```

- This text appears in Times New Roman if available
- The *font-family* property specifies a choice of fonts separated by commas, in priority order.
- Multi-word arguments like “Times New Roman”, must appear in quotes.

Selectors

selector {property$_1$: value$_1$;
property$_2$: value$_2$; …
property$_m$: value$_m$;
}

- The selector will be the name of an HTML element, pseudo-element, class, pseudo-class or id of an element

Selector	Examples
element	p p strong (contextual selector)
pseudo-element	p::first-letter p::first-line
class	.highlight
pseudo-class	a:link a:hover a:active a:visited
id	#para1

Using Element Styles

Aspect	Example
specify	p {font-weight: bold; font-size: 18pt; }
use	content
remark	Modifies every instance of the … tag, unless overridden by an in-line style.

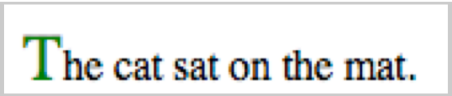
```
<p style="font: 20pt;">
```

This would override the predefine style

Using Contextual Element Styles

Aspect	Example
specify	p b, p strong {color: orange;}
use	contentbold stuffmore content
remark	Modifies instances of the tag, but only if they occur inside a … context, (and unless overridden by an in-line style).
appearance	regular bold and bold inside p tag → **regular bold** and **bold** inside p tag

Using Pseudo-element Styles

Aspect	Example
specify	p::first-letter {color: green; font-size: 24px; }
use	The cat sat on the mat.
appearance	→ 

Using Class Styles

Aspect	Example
specify	.highlight {background-color: green; }
use	content <td class="highlight">content</td> <h1 class="highlight">heading</h1>
remark	Can be used to modify style in any tag: <h1>heading</h1> works as usual, but <h1 class="highlight">heading</h1> highlights the heading in green. You use a class style when different instances of elements in the same document need different formatting.

Using Pseudo-class Styles

Aspect	Example	
specify	a:link {color: blue; } a:visited {color: purple; } a:hover {color: green; size: 150%; } a:active {color: orange; }	
use	The meaning of life can be found here	
appearance	Before visiting:	The meaning of life can be found here.
… tested	After visiting:	The meaning of life can be found here.
… with	When hovering:	The meaning of life can be found here.
… Chrome	Mouse clicked but not released:	The meaning of life can be found here.

Using Element Id Styles

Aspect	Example
specify	#para1 {background-color: blue; }
use	content
remark	Can be used to modify style in any tag: hello works as usual, but hello sets background to blue. An id should be unique within a page, so you should use the id selector when you want to find a single, unique element.

CSS Font Properties

Font Property	Alternatives/Examples
font-family	font-family: Times, serif
font-style	font-style: italic font-style: oblique font-style: normal
font-variant	font-variant: small-caps font-variant: normal
font-weight	font-weight: bold font-weight: bolder font-weight: lighter font-weight: 300 font-weight: normal
font-size	font-size: 14px font-size: 18pt font-size: xx-small | x-small | small | medium | … xx-large font-size: larger | smaller font-size: 80%

CSS Text Properties

Text Property	Examples / Alternatives
line-height	line-height: 1.5 | 18 px | 110% | normal
text-decoration	text-decoration: underline | line-through | overline text-decoration: blink | none
text-transform	text-transform: lowercase | uppercase | capitalize | none
text-align	text-align: left | right | center | justify
text-indent	text-indent: 15px | 7%
white-space	white-space: pre | normal

Colour / Background Properties

Color or Background Property	Examples / Alternatives
color	color: red | rgb(255, 0, 0) | rgb(100%, 0%, 0%) color: #FF0000
background-color	background-color: transparent background-color: alternatives as for color
background-image	background-image: none | url
background-repeat	background-repeat: repeat | repeat-x | repeat-y background-repeat: no-repeat
background-attachment	background-attachment: scroll | fixed
background-position	background-position: top | center | bottom background-position: left| right background-position: 25px 25px ←x y coordinates background-position: 0% 10% ditto

CSS List Properties

List Property	Examples / Alternatives	Meaning
list-style-type	list-style-type: disc list-style-type: circle list-style-type: square list-style-type: decimal list-style-type: lower-roman list-style-type: upper-roman list-style-type: lower-alpha list-style-type: upper-alpha	• item o item ▪ item 1. item i. item I. item a. item A. item
list-style-position	list-style-position: inside (Windows only) list-style-position: outside (Windows only)	1. blah blah blah blah blah 2. more blah 1. blah blah blah blah blah 2. more blah
list-style-image	list-style-image: url list-style-image: none	 blah blah blah

CSS Summary

- Put the main style definitions that apply across all your web pages in a .css file.
- Link this to a particular web page by using a `<link…>` element in the `<head>` of the .html file.
- Also, in the `<head>` of the .html file – possibly put a `<style>` element to hold extra style definitions that apply just to this .html file.
- In the `<body>` of the .html file, here and there: in-line style definitions of a one-off nature.

EXAMPLE

Isaac!!!!!

Welcome to my webpage!

About me!

I'm a lecturer and that thing seems to be cool. Enjoying it so far. My students can be a bit quiet tho. Wish they'd ask me more questions.

Dis me!

Two fun things I do for fun:

1. Surfing
2. Making spreadsheets

i.carr@unsw.edu.au

March, 2018

Webpage Layout

- Multiple columns are created by using or <table> elements.
  - CSS are used to position elements
- Normal Flow
  - In all examples, we have seen that the browser displays elements in the order they are coded in the Web page document
- Nested Flow
  - It is possible to nest one element box within another using the tag.

Two Columns using <table>

```
<h1>My webpage!</h1>
<table>
  <tr>
    <td>
      <h1>Col 1</h1>
      <p>This is the first column</p>
    </td>
    <td>
      <h1>Col 2</h1>
      <p>This is the second column</p>
    </td>
  </tr>
</table>
</ul>
```

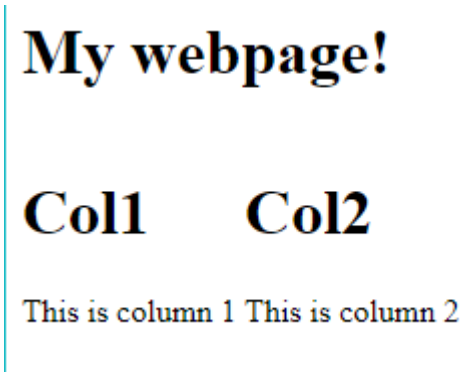
My webpage!

Col 1 **Col 2**

This is the first column This is the second column

Two Columns using

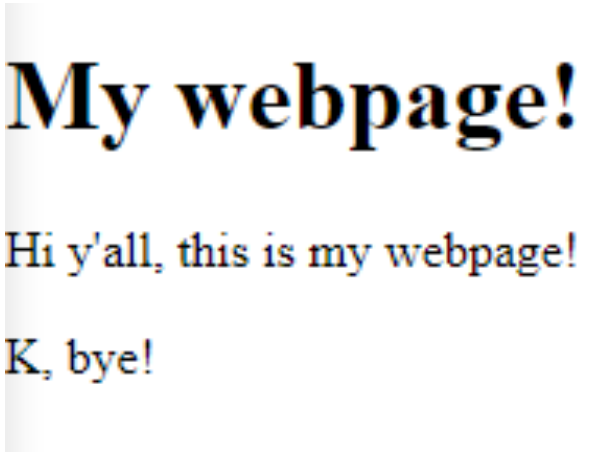
```
<!DOCTYPE html>
<html>
<head>
  <title>My page</title>
  <style type="text/css">
    .column {
      display: inline-block;
    }
  </style>
</head>
<body>
  <h1>My webpage!</h1>
  <div class="column">
    <h1>Col1</h1>
    <p>This is column 1</p>
  </div>
  <div class="column">
    <h1>Col2</h1>
    <p>This is column 2</p>
  </div>
</body>
</html>
```



Normal Flow

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>My page</title>
5  </head>
6  <body>
7    <h1>My webpage!</h1>
8    <p>
9      Hi y'all, this is my webpage!
10   </p>
11   <p>
12     K, bye!
13   </p>
14 </body>
15 </html>
```

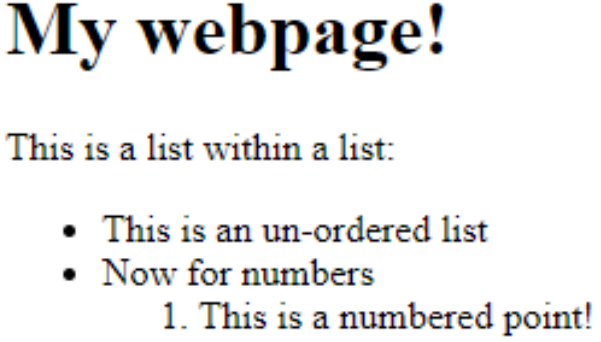
- One element after another
- Will appear vertically down the page in order
- Simple



Nested Flow

```
<!DOCTYPE html>
<html>
<head>
  <title>My page</title>
</head>
<body>
  <h1>My webpage!</h1>
  <p>This is a list within a list:</p>
  <ul>
    <li>This is an un-ordered list</li>
    <li>
      Now for numbers
      <ol>
        <li>This is a numbered point!</li>
      </ol>
    </li>
  </ul>
</body>
</html>
```

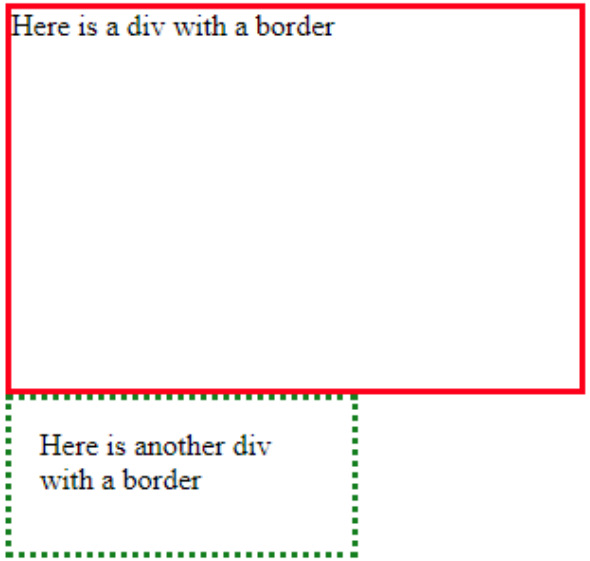
- Higher complexity
- More useful in practice



Normal Flow

- In this example the first div appears on the page then

```
<div style = "border-color: red;
   border-width: 3px; border-style:
   solid; height: 200px; width: 300px;
   ">
   Here is a div with a border
</div>

<div style = "border-color: green;
   border-width: 3px; border-style:
   dotted; height: 50px; width: 150px;
   padding: 15px;">
   Here is another div with a border
</div>
```

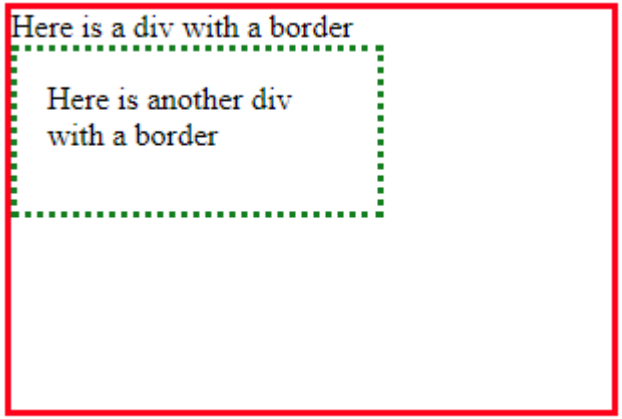


See normal.html

Nested Flow

- In this example the first div appears on the page with the second one nested INSIDE it.

```
<div style = "border-color: red;
   border-width: 3px; border-style:
   solid; height: 200px; width: 300px;
   ">
   Here is a div with a border


  <!-- This div is inside the other
   div -->
  <div style = "border-color: green;
   border-width:  3px; border-style:
   dotted; height: 50px; width: 150px;
   padding: 15px;">
       Here is another div with a
   border
   </div>
</div>
```



See nested.html

Positioning

- Elements can be positioned
  - top, bottom, left, and right properties.
  - properties will not work unless the position property is set first.
  - They work differently depending on the positioning method.

- HTML elements are positioned **static** by default.
  - positioned according to the normal flow of the page.
  - Static positioned elements are not affected by the top, bottom, left, and right properties.

- Elements can also be positioned **relative** or **absolute**

Relative Positioning

- A relative positioned element is positioned relative to its normal position.

```
<h2>This is a heading with no
  position</h2>
<h2
  style="position:relative;left:50px;">
This heading is relatively positioned
  according to its normal position
</h2>
```

- left, right, top, bottom

This is a heading with no position

This heading is relatively positioned according to its normal position

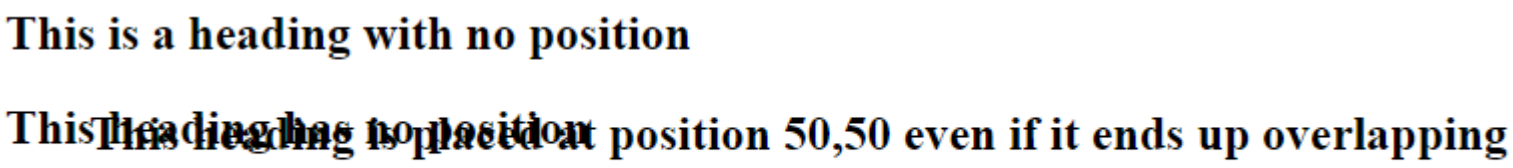
See rel.html

Absolute Positioning

- Absolutely positioned elements are removed from the normal flow.
  - Relative to the first parent element with a position other than static of just the <html> block if no non-static parents exist.
  - The document and other elements behave like the absolutely positioned element does not exist.
  - Absolutely positioned elements can overlap other elements.

Absolute Positioning Example

```
<h2>This is a heading with no position</h2>
<h2 style="position:absolute;left:50px;
                                top:50px">
   This heading is placed at position
  50,50 even if it ends up overlapping
</h2>
<h2> This heading has no position</h2>
```



See abs.html

Floating an Element

- An element can be pushed to the left or right, allowing other elements to wrap around it by using the float property
  - float: left
  - float: right
  - float: none
- Float is useful for images and layouts

Floating Elements

- A floated element will move as far to the left or right as it can within its containing element.
- If you place several floating elements after each other, they will float next to each other if there is room.
- The elements before the floating element will not be affected.
- The elements after the floating element will flow around it.
  - If an image is floated to the right, a following text flows around it, to the left

Demo

```
5    <style>
6      img { float: right;  margin: 0 0 10px 10px;}
7    </style>
8  <body>
9  <p>
10   In this example, the image will float to the right in the paragraph, and the
     text in the paragraph will wrap around the image.
11 </p>
12 <p>
13   <img src="UNSW.png" alt="unswm" width="80" height="50">
14   Text text text text text text text text text Text text text text text text
     text text text Text text text text text text text text text Text text text
     text text text text text text Text text text text text text text text text
     Text text text text text text text text text Text text text text text text
     text text text Text text text text text text text text text text Text text
     text text text text text text.
15 </p>
```

In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.

Text text text text text text text text text Text text text text text text text text text Text text text text text text text text text Text text text text text text text text text Text text text text text text text text text Text text text text text text text text text Text text text text text text text text text Text text text text text text text text text Text text text text text.

See unsw.html

Layers

- CSS allows creating and manipulating layers in a web page using z-indexes –
  - The "*z-index*" property, used to arrange the overlapping elements inside a document.
  - Lower z-indexes are displayed on a lower layer
  - Z-index only applies to positioned elements

```
<!DOCTYPE html>
<html>
<head>
  <title>My page</title>
  <style type="text/css">
    div { font-family: Arial, Helvetica, sans-serif;
          position: relative; font-size: 48px;
    }
    div.toplayer { color: blue; }
    div.bottomlayer { top: -50px; left: 5px; }
  </style>
</head>
<body>
  <div id="layer1" class="toplayer" style="z-index: 2; ">
    COMP1000
  </div>
  <div id="layer2" class="bottomlayer" style="z-index: 1; ">
    COMP1000
  </div>
</body>
</html>
```

See layers1.html

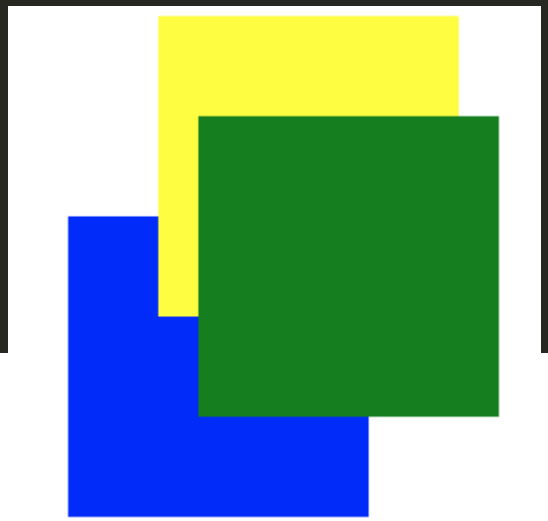
Layers

- The HTML on the last slide produces this:

- is specified to have position: relative
- then div.bottomlayer is given a top of -50px and a left of 5px … relative offsets from where they would placed normally
  - The result is that they the two divs overlap. (N.B. Font size is given as 48px.)
- The z-index values of 1 and 2 in the body specify which is on top (the highest z-index value).

Layers

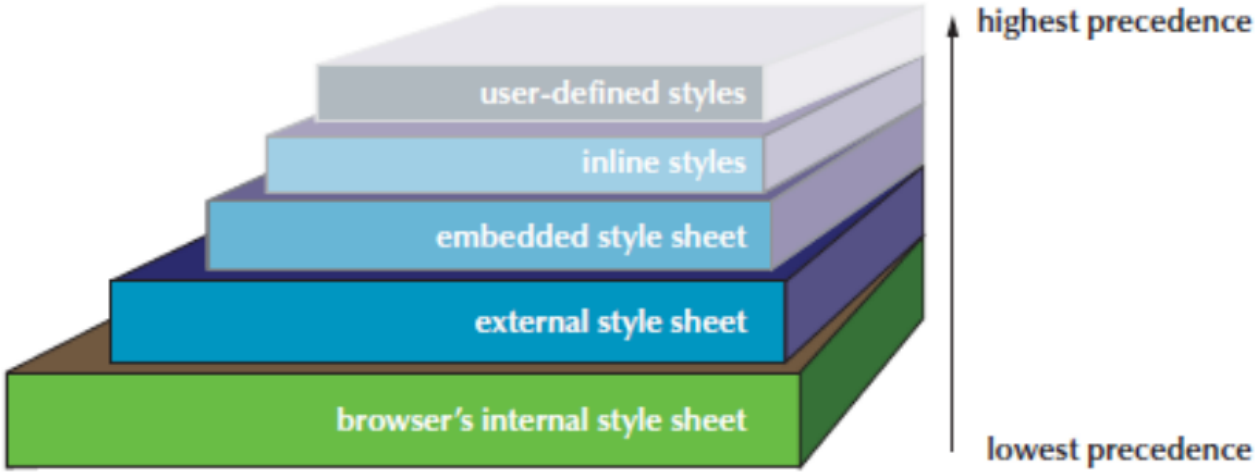
```
<!DOCTYPE html>
<html>
<head>
  <title>My page</title>
  <style>
    div.middlelayer {background-color:yellow; width:150px; height:150px;
    position:relative; top:0px; left:80px;z-index:2}
    div.bottomlayer {background-color:blue; width:150px; height:150px;
    position:relative; top:-50px; left:35px;z-index:1}
    div.toplayer {background-color:green; width:150px; height:150px;
    position:relative; top:-250px; left:100px;z-index:3}
  </style>
</head>
<body>
  <div class="middlelayer"></div>
  <div class="bottomlayer"></div>
  <div class="toplayer"></div>
</body>
</html>
```



- See layers2.html

Understanding Cascading Order

- Style precedence – The rule that determines which style is applied when one or more styles conflict
- Web-author defined styles take precedence over browser in-built styles
- The last declared takes precedence, when conflicting styles are in same order



Applying style to a specific element ID

- To create a style for a specific page element, declare as – #id { color:red } where id is the specific id of the page element

- E.G., if you had the element <h2> as:

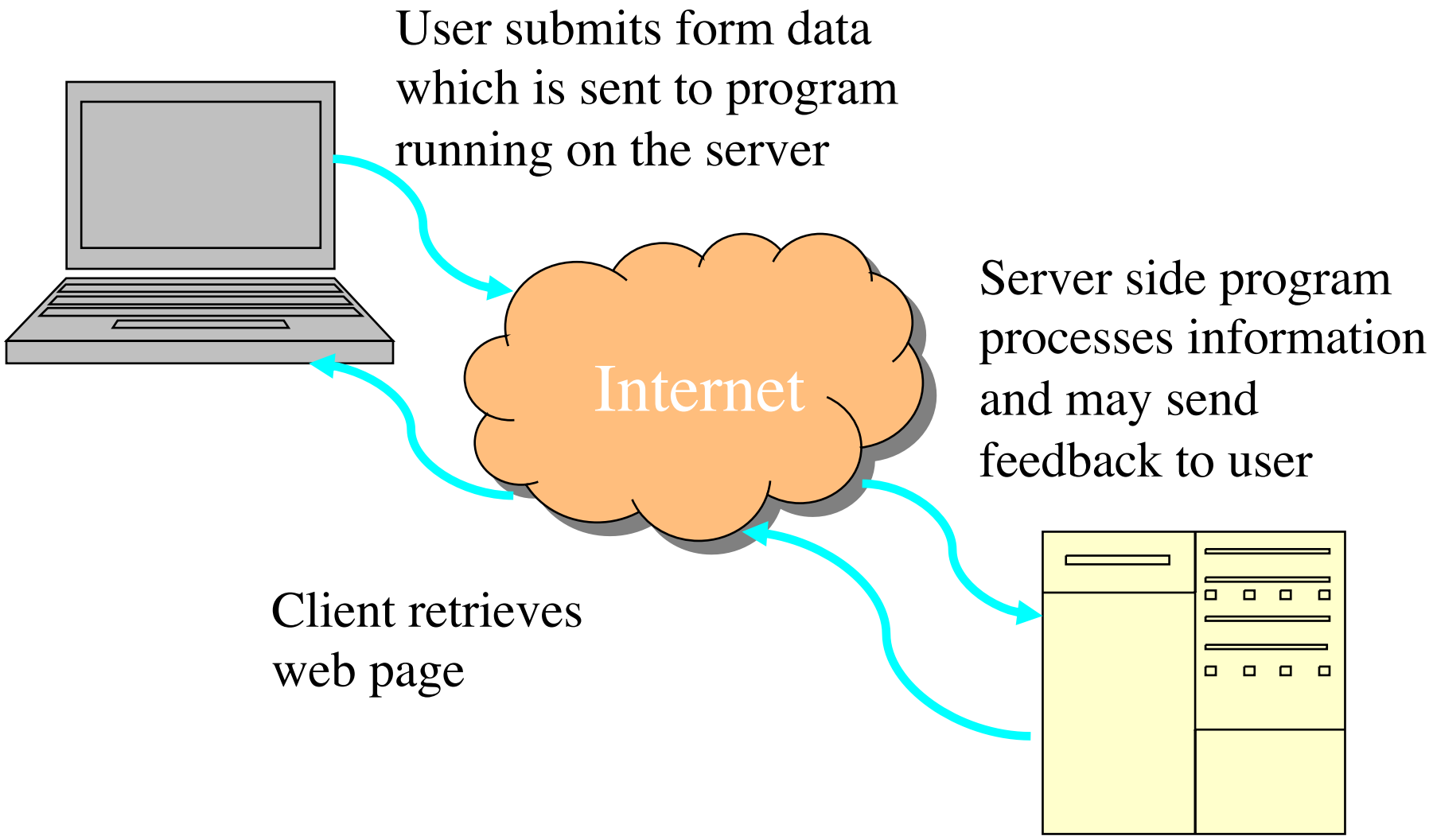
  <h2 id="sub-heading">Course Outline</h2>

- then to apply a specific style to the element, define a rule as: #paraheading { color: red }

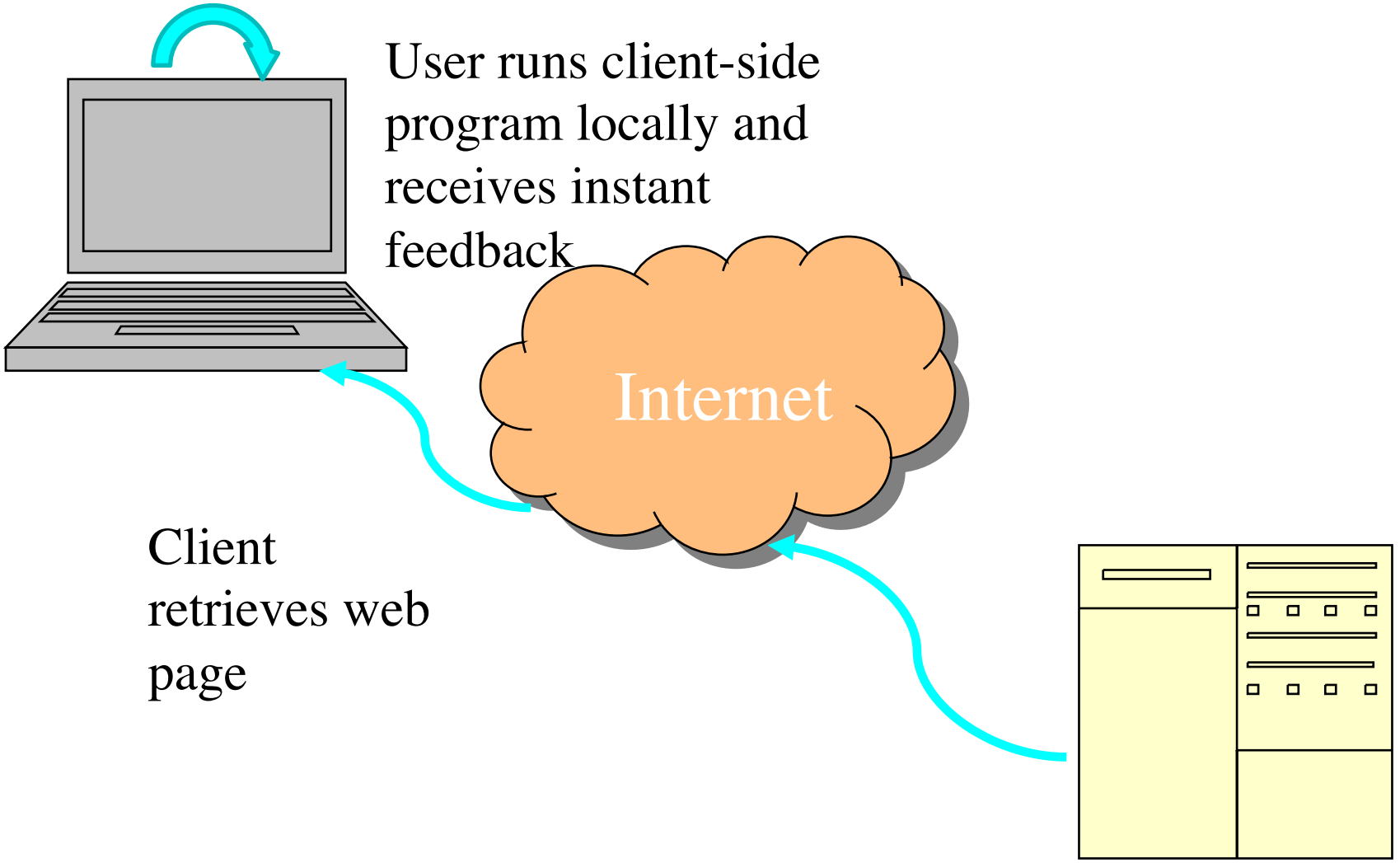
Introduction to Forms

- Forms allow interaction with a webpage
- While HTML supports the creation of forms, it does not include tools to process the information.
  - The information can be processed through a program running on a Web server.
    - Most forms have a submit and reset button.
    - The information is sent using name/value pairs to the server for processing
  - Interaction is also possible on the client side

Server Side Processing



Client Side Processing



Forms

- The `<form>` element marks a section of the Web page that will contain normal content, HTML formatting, and extra HTML elements called *controls* which are used to capture information entered by the viewer of the page.

```
<form action="URL for action" method="post">
…
</form>
```

  - The *URL* gives the name of the program on some **server** that will process the input.
  - The *method* says how the info will be sent to this program.
- Do not need to specify action or method for client-side only interaction

Forms

- This course will not go into "server-side" actions.
- However, interaction is possible on the client side using JavaScript.
  - *Not Java – Java is a programming language*
- It needs JavaScript to function in reality.
- Form control elements include:
  - buttons (several types)
  - text input boxes
  - selection lists
  - password fields
  - text areas

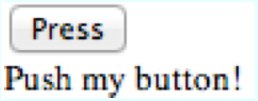
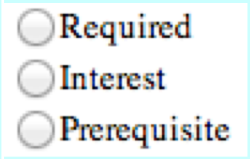
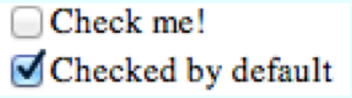
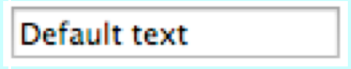
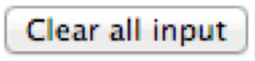
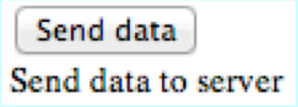
Input Attributes

- type: specifies what type of input it is
  - E.G. text, button, checkbox etc
- name: names the form element to indentify the associated data for later processing by the server or (JavaScript)
- value: text or numeric value for the text displayed on the button, or the text inside the text input etc
- id: a unique id that can be used by css or javaScript

Buttons

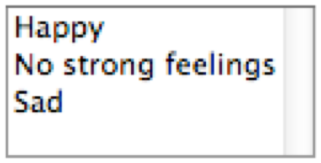
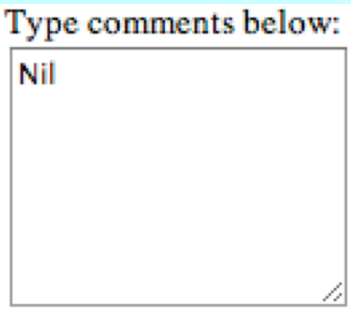
- Buttons are a type of control element that performs an action
- Buttons are created using the <input> tag
- Types of buttons:
  - Command button
    - For client-side processing
    - `<input type="button" value="text">`
  - Submit button
    - To submit date to the server
    - `<input type="submit" value="text">`
  - Reset button
    - Restore forms to their original default values
    - `<input type="reset" value="text">`

Form elements

Type	Looks like	HTML
All of these must appear inside an HTML <form>…</form> element		
button		<input type="button" name="btn1" value="Press">Push my button!
radio buttons		<input type="radio" name="radio1" value=1>Required <input type="radio" name="radio1" value=2>Interest <input type="radio" name="radio1" value=3>Prerequisite
check box		<input type="checkbox" name="check1" value="A">Check me! <input type="checkbox" name="check2" value="B" checked=true>Checked by default
text		<input type="text" name="text1" value="Default text" size=14>
reset		<input type="reset" value="Clear all input" name="clear">
submit		<input type="submit" value="Send Data" name=send>Send data to server

Using the same name for all the buttons in a set of radio buttons makes them mutually exclusive: clicking one "unclicks" any others.

More form elements

Type	Looks like	HTML
select		<select> <option value="1">Happy</option> <option value="2">No strong feelings</option> <option value="3">Sad</option> </select>
select		<select size="3"> <option value="1">Happy</option> <option value="2">No strong feelings</option> <option value="3">Sad</option> </select>
textarea		Type comments below: <textarea name="area1" cols=15 rows=7>Nil</textarea>

Demo

```
<form id="myform">
  Name: <input type="text" maxlength="10"/><br/>
  Male: <input type="radio" name="gender" value="male"/>
  Female: <input type="radio" name="gender" value="female"/><br/><br/>
  Dog:  <input type="checkbox" name="pet" value="dog"/>
  Cat: <input type="checkbox" name="pet" value="cat"/><br/> <br/>

  <select>
    <option value="1">Happy</option>
    <option value="2">No strong feelings</option>
    <option value="3">Sad</option>
  </select> <br/><br/>

  <select size="3">
    <option value="1">Happy</option>
    <option value="2">No strong feelings</option>
    <option value="3">Sad</option>
  </select> <br/><br/>

  Type comments below:
  <p>
    <textarea name="area1" cols=15 rows=7>Nil</textarea>
  </p>
  <br>
</form>
```

Name:

Male: Female:

Dog: Cat:

Happy

Happy
No strong feelings
Sad

Type comments below:

Nil

Label, Fieldset and Legend Form Elements

- fieldsets allow you to organise option buttons into a group.

```
<fieldset id="someId">
 <legend>someText</legend>
  . . .
</fieldset>
```

  - The legend tag is optional and adds a caption to the field set

- You can also link a label with an associated text element for scripting purposes

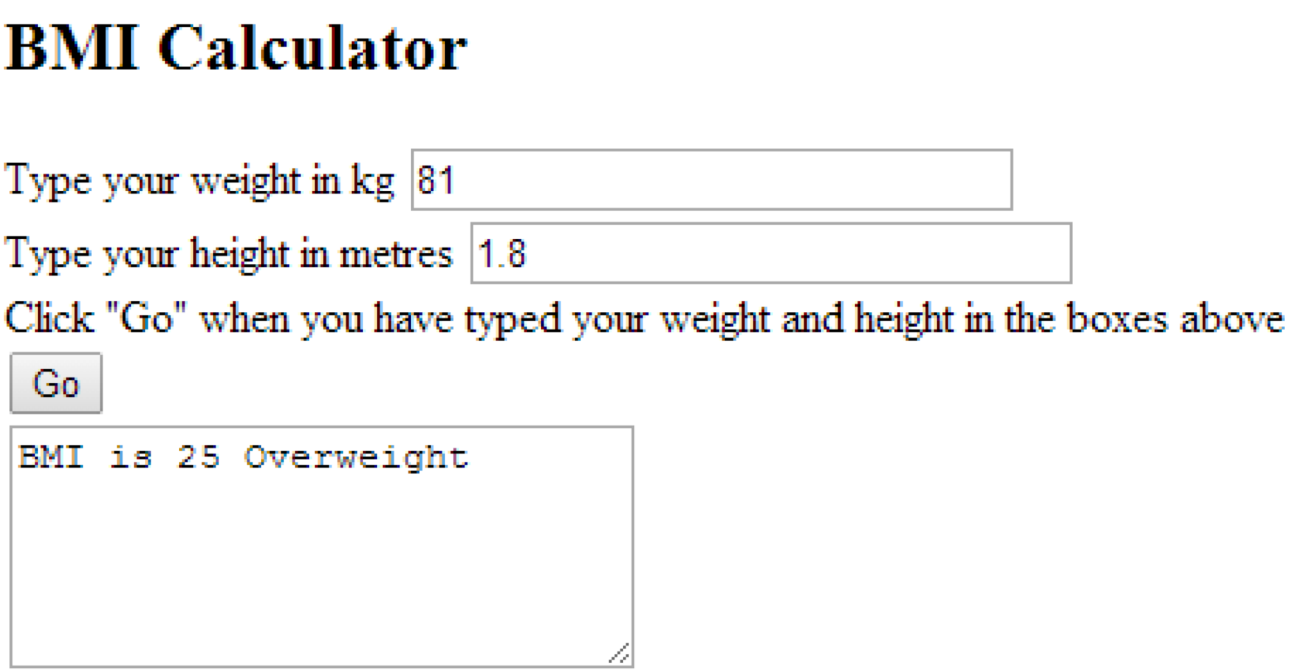
```
<label for="id">label text</label>
```

  - Where id is the value of the id attribute for a field's control element, and label text is the text of the label

Demo

```
 7  <form id="myform">
 8    <fieldset>
 9      <legend>This is the legend</legend>
10      <label>First Name: <input type="text" name="first"></label>
11      <br/>
12      <label for="last"> Last Name: </label>
13      <input type="text" id="last" name="last">
14    </fieldset>
15  </form>
```

BMI Example


BMI Calculator
Type your weight in kg
81
Type your height in metres
1.8
Click "Go" when you have typed your weight and height in the boxes above
Go
BMI is 25 Overweight

Audio in HTML5

- Before HTML5, there was no standard for playing audio files on a web page.
  - audio files had to be played with a plug-in (like flash). However, different browsers supported different plug-ins.
- HTML5 defines a new element which specifies a standard way to embed an audio file on a web page: the <audio> element.

Audio Example

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

- The control attribute adds audio controls, like play, pause, autoplay and volume.
- You should also insert text content between the <audio> and </audio> tags for browsers that do not support the <audio> element.
- The <audio> element allows multiple <source> elements. <source> elements can link to different audio files. The browser will use the first recognized format.
- The autoplay control

Video in HTML5

- Before HTML5, there was no standard for showing videos/movies on web pages.

- Before HTML5, videos could only be played with a plug-in (like flash). However, different browsers supported different plug-ins.

- HTML5 defines a new element which specifies a standard way to embed a video or movie on a web page: the <video> element.

Video Example

```
<video width="320" height="240"
controls>
  <source src="movie.mp4"
type="video/mp4">
  <source src="movie.ogg"
type="video/ogg">
Your browser does not support the
video tag.
</video>
```

Image Maps

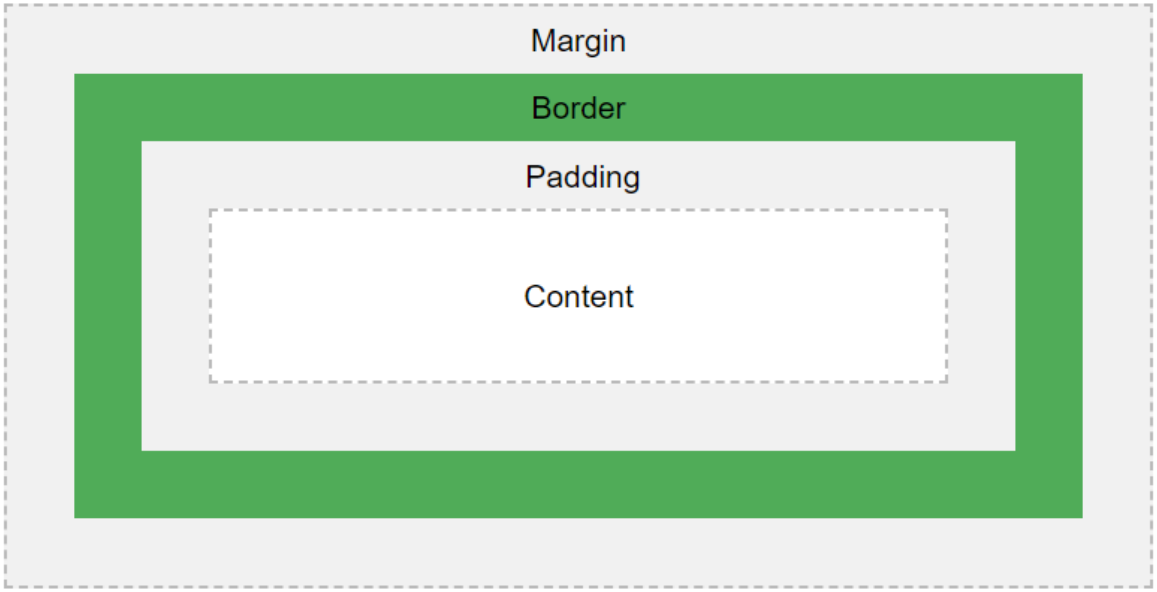
- It is possible to make one image link to several pages, depending on where the image is clicked
  - image mapping.
- You simply specify which areas of the image should link to where.
- Example:

```
<img src="some.gif" usemap = #mymap border=0>
<map name=mymap>
<area shape="rect" coords="0,0,30,30"
  href="http://www....">
<area shape="circle" coords="50,50,20"
  href="http://www....">
</map>
```

Demo:
http://www.w3schools.com/tags/tag_map.asp

HTML Box Model

- Content: content of box where text and images appear
- Padding: clears an area around the content (transparent)
- Border: goes around padding + content
- Margin: Clears an area outside the border (transparent)



- https://www.w3schools.com/css/css_boxmodel.asp

Someone has done it for you already…

- Bootstrap: https://getbootstrap.com/
- Materialize: http://materializecss.com/
- Semantic UI: https://semantic-ui.com/

For Bootstrap, you just need to include this link in your <head> tag:

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/cs
s/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSA
wiGgFAW/dAiS6JXm" crossorigin="anonymous">
```

Find on Bootstrap Website

Chrome Developer Tools

- https://www.elegantthemes.com/blog/resources/why-you-should-start-using-chrome-developer-tools-right-now

Introduction to JavaScript

- JavaScript is a scripting language used to make web pages more interactive

- It is one of the **3 languages** all web developers must learn
  - **HTML** to define the content of web pages
  - **CSS** to specify the styling/layout of web pages
  - **JavaScript** to program the behaviour of web pages

Introduction to JavaScript

- 3 ways to incorporate JavaScript into a website
  - Inline as an attribute of specific element

```
<input type="button" name="mybutton"
  value="Press" onclick="alert('You
  pressed me!')">
```

  - Internal: Put JavaScript inside a script element in the html header

```
<script>alert("Hello!");</script>
```

  - External: Write it in an external text file with a .js extension and set the src attribute of the script tag

```
<script src="myscript.js"></script>
```

Events

- Events are things like *mouse-clicks* and *mouse-rollovers* that can be detected by the computer and acted on.
- Some work with most HTML elements, some with few.
- Simple example: <input type="button" name="mybutton" value="Press" onclick="alert('You pressed me!')">
- If such a button is pressed, this happens:

Common event attributes

Attribute	Description	Works with …
ONLOAD	document is loaded	BODY
ONUNLOAD	document is unloaded	BODY
ONCHANGE	element changed, loses focus	TEXTAREA, INPUT TYPE=text, SELECT
ONMOUSEOVER	cursor moves over element	A, IMG, form control elements, AREA
ONMOUSEOUT	cursor moves off element	A, IMG, form control elements, AREA
ONMOUSEMOVE	cursor moves	A, IMG, form control elements, AREA
ONCLICK	mouse button clicked	A, IMG, form control elements, AREA
ONDBLCLICK	… double-clicked	A, IMG, form control elements, AREA
ONMOUSEDOWN	… pressed	A, IMG, form control elements, AREA
ONMOUSEUP	… released	A, IMG, form control elements, AREA
ONKEYDOWN	key pressed on keyboard	form control elements, AREA
ONKEYUP	… released	form control elements, AREA
ONKEYPRESS	… pressed and released	form control elements, AREA
ONBLUR	element loses focus	TEXTAREA, INPUT TYPE=text, SELECT

Important JavaScript syntax

- Each statement (aka command) is a single line that gives an action for the browser to take.
  - A semicolon ends each statement
- A string is any sequence of text characters, such as “Hello” or “Happy Holidays!”
  - Must be enclosed within either double **or** single quotations
  - “\n” is a newline character
  - + is an operator that concatenates two strings
- // This is a comment
- /* This is also a comment */

Output

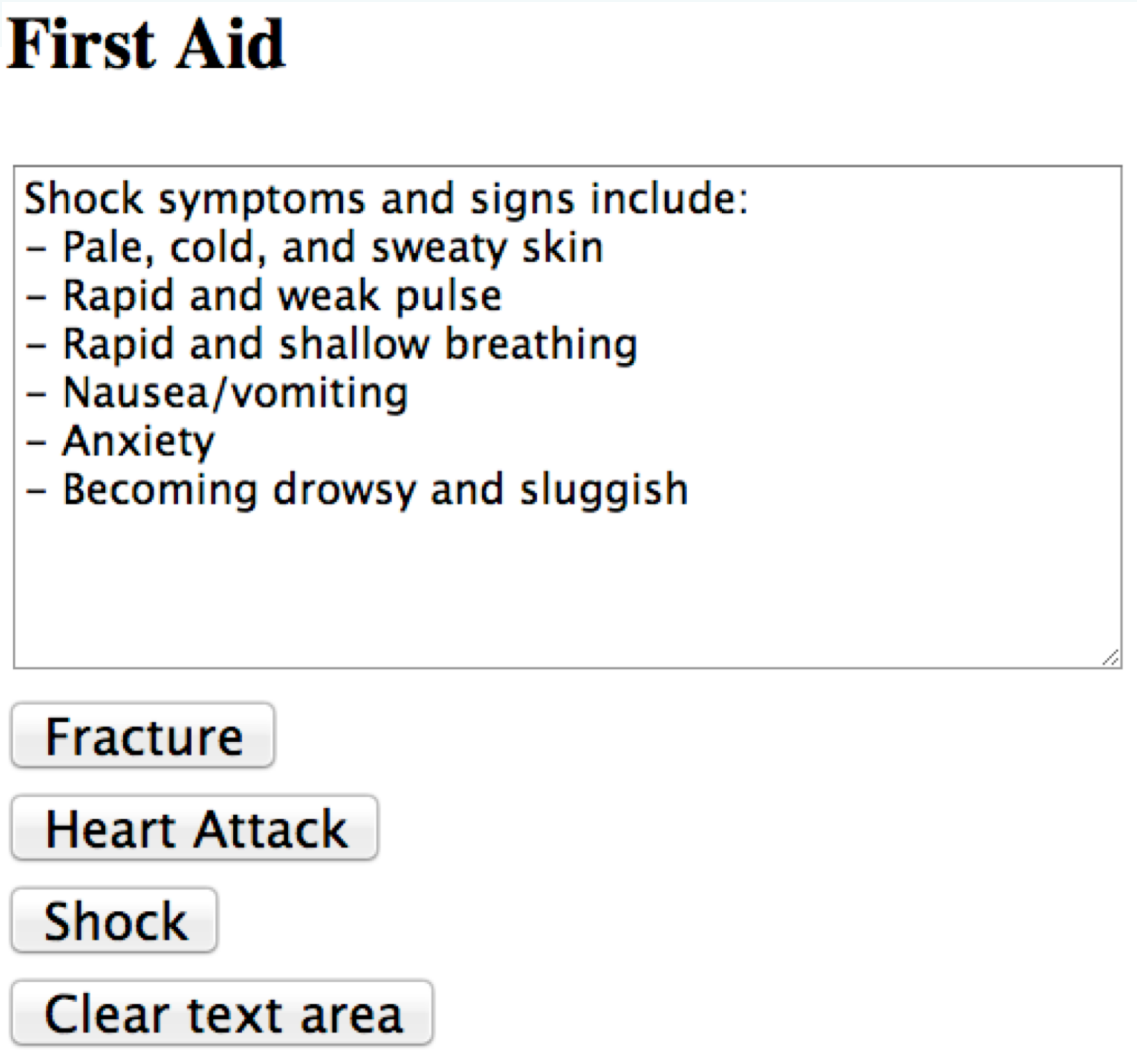
- Message Box:
  - alert(“Hello it is “ + Date());
- Write to document:
  - document.write(“Hello it is “ + Date());
- Write to an element from within a form
  - <textarea name=“area1”>
  - area1.value = “Hello”;
  - form1.area1.value=“Hello”;

Demo

```
<h3>First Aid</h3>
<form name="form1">
<textarea name="area1" cols=40 rows=10>Select a first-aid area
</textarea>
<br />
<input type
 onclick="area1.value='Symptoms and signs of bone fracture include:\n-
Pain\n- Swelling\n- Deformity\n- Loss of function\n- Shock'"
<br />
<!– insert more buttons here: e.g. for , heart attack, shock … -->
<input type="reset" value="Clear text area" name="clear">
</form>
```

See first_aid.html

HTML on previous slide produces...



Points to note in the Javascript example

- `onclick="area1.value='Symptoms …\n…'`
  `+ area1.value "`
  - `onclick` event occurs when mouse is clicked
  - `area1.value`: name of the textarea previously defined
  - `'Symptoms…'` : use ' since this is already inside "…"
  - `\n` : newline character: means "start new line"
  - `"…" + area1.value` : concatenates (joins) 2 strings of characters

Variables and Expressions

- `var age;`

  `age = 17;`
  - This is a numeric variable
- `var greeting = "Happy Birthday!";`
  - This is a string variable
- `var passed = true;`
  - Note this is a boolean variable – not a string as true is not inside quotes
  - booleans values are **true** or **false**
- `var total = 1 + age;`
- `var name = "John";`
- `var message = greeting + " " + name +"\n";`
  - Note "\n" is a newline character

Functions

- A function is a collection of commands that performs an action or returns a value
  - A function has a name to identify it
  - Parameters are input values used by the function
  - A function can return a value

```
function functionName(parameters){
        . . . . .
        return value;
}
```

Sample Function

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function sayHello()
        {
            alert("Hello World!");
        }
    </script>
</head>

<body>
    <input type="button" value="Press"
  onclick="sayHello();">
    </input>
</body>
</html>
```

See hello_world.html

Sample Function with input

```
<head>
    <script>
        function sayHello(title,name)
        {
            alert("Hello " + title + " " + name);
        }
    </script>
</head>
<body>
    <input type="button" value="Press1"
        onclick="sayHello('Dr','Phil');"> </input>
    <input type="button" value="Press2"
        onclick="sayHello('Mrs','Marsh');"> </input>
</body>
```

See title.html

Sample Function with return

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function calc(num)
        {
            return num*2;
        }
    </script>
</head>
<body>
    <input type="button" value="Press1"
        onclick="var x = calc(10); alert('The answer is ' +
   x);">
    <input type="button" value="Press2"
        onclick="alert('The answer is ' + calc(-1));">
   </input>
</body>
</html>
```

See calc.html

Comparisons

- Can use comparison operators

Operator	Description
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal

- Eg x == 10

Logical Operators

- Can perform operations on boolean expressions

Operator	Description
&&	and
||	or
!	not

- Eg (x >= 0 && x <= 100)

If Conditions

if (condition){

code to be executed if condition is true

}

For example:

```
if(age < 18){
    alert("You can't come into the
 club!");
}
if(userName.value == "){
    alert("Please enter your name");
}
```

If-else conditions

if (condition) {

code to be executed if condition is true

} else {

code to be executed if condition is not true

}

For example:

```
if(mark >= 50){
    alert("You passed!");
} else {
    alert("You failed");
}
```

If…else if…else statements

```
if (condition1) {
    code to be executed if condition1 is true
} else if (condition2) {
    code to be executed if condition2 is true
} else {
    code to be executed if neither condition1 nor
  condition2  is true
}
```

See demoGrade.html

Visibility

- Layers have a `visibility` property, with possible values of `visible` (the default) or `hidden`.
- Buttons, Javascript and `<div>` can then be used to control what parts of a web page are visible.
- The essence of the example on the next slide:
  - the second layer starts out hidden (`<style="visibility: hidden; ">`)
  - The button `onclick` action toggles its `visibility` using the Javascript `if` *condition action1*; `else` *action2*
  - Notice that == (as opposed to just =) is used for the equality test in the condition part of the "if"

Layers

```
<style type="text/css">
    div.see { color: black; }
    div.hide { color: blue; }
</style>
…

<div id="layer1" class="see">Test Answers</div>
<div id="layer2" class="hide" style="visibility: hidden; ">
      The answer to every question is 42.
</div>

<form name="form1">

<input name="showhide" id="sh1" value="Show/Hide"
type="button"

onclick="if(layer2.style.visibility=='hidden'){layer2.style.vi
sibility='visible';}else {layer2.style.visibility='hidden';}">
</form>
```

Geez Isaac… styling looks hard. You think I'm some sort of designer or sumthin?

HTML/CSS Summary

You should feel somewhat comfortable with HTML5/CSS.

- https://www.w3schools.com/html/default.asp
- https://www.w3schools.com/css/default.asp

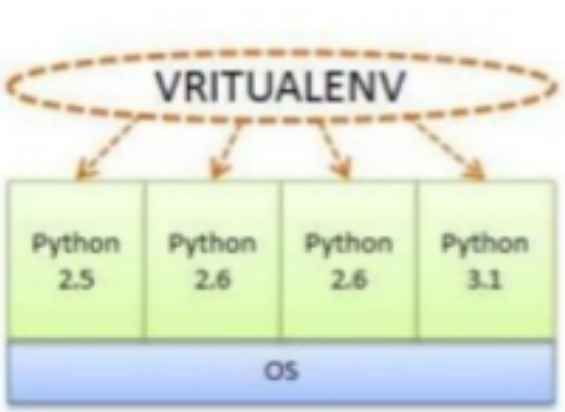
Flask

- A **micro** web framework written in Python, developed by Armin Ronacher
- AS a micro-framework, aims to provide a simple, solid core but designed as an *extensible* framework
  - i.e. not native support for databases, authenticating users
  - But these services are available through *extensions* that integrate with the core package
- As a developer, you can choose which packages to include or write your own customer extensions
- Two main dependencies:
  - Werkzeug which supports routing (request + response), utiling functions such as debugging and WSGI (Web Server Gateway interface – a standard interface between web server and web applications)
  - Jinja2, a powerful templating language to render dynamic web pages

Install Flask using Virtualenv

```
pip3 install virtualenv
virtualenv --version
```

```
$ mkdir myproject
$ cd myproject
$ virtualenv myproject
(myproject) $
```




```
$ myproject/bin/activate # OSX
> myproject/scripts/activate # win
```

```
$ myproject/bin/deactivate # OSX
> myproject/scripts/deactivate # win
```

```
(hello) $ pip3 install flask
```

Simple Flask Application

Route Decorator – binds function to a URL

View function to render HTML page to browser

Running a Flask app

```
# Import Flask Library
from flask import Flask

# create a Flask application instance
app = Flask(__name__)

# define a route through the app.route decorator
@app.route("/")
def index():
    return '<h1> Hello World </h1>'

# launch the integrated development web server
# and run the app on http://localhost:8085

if __name__=='__main__':
   app.run(debug=True,port=8085)
```

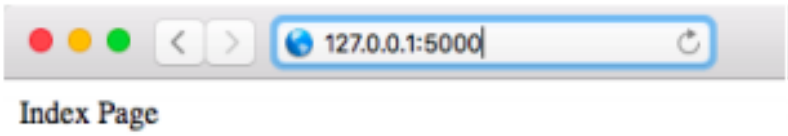
```
(hello) $ python hello.py
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 281-144-438
 * Running on http://127.0.0.1:8085/ (Press CTRL+C to
quit)
127.0.0.1 - - [18/Jul/2017 09:10:19] "GET / HTTP/1.1"
200 -
127.0.0.1 - - [18/Jul/2017 09:10:19] "GET /favicon.ico
HTTP/1.1" 404 -
```

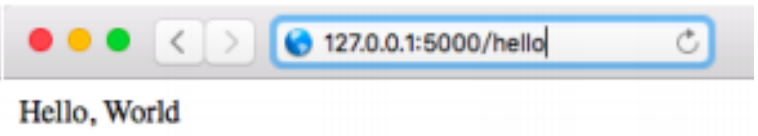
Simple Flask Application

@app_route Function Decorator

The @app_route decorator or annotation is used to specify python functions to be executed for a specific URL path of the web application, i.e., to bind a function to an URL

```python
#!/usr/bin/env python

from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return 'Index Page'

@app.route('/hello')
def hello():
    return 'Hello, World'

if __name__ == '__main__':
    app.run()
```





GO FORTH AND BUILD FUNCTIONAL, BEAUTIFUL WEB PAGES