

# An introduction to website design

- Web page design with HTML
- Website construction and management
- Enhancing website usability.

The course will enable you to design webpages and websites, and to set up and manage simple websites.

## *Course plan*

1. Introduction
2. Writing simple Web pages
3. Lists and tables
4. Images and animations
5. Links: connecting webpages
6. Forms: sending data from webpages
7. Advanced forms
8. Writing complete websites
9. Setting up websites on Internet hosts
10. How will people use your site?
11. Helping users understand your site

12. Helping users navigate your site
13. Navigation and location in websites
14. Good and bad practices in website design
15. Summary and next steps

All examples can be downloaded from:

<https://nms.kcl.ac.uk/kevin.lano/cllwd>

## Session 1: Introduction

- Scope of the course
- Internet and Web concepts
- Browsers and HTML
- Websites

## *Scope of the course*

- We will focus on writing web pages + on their usability
- We cover all the key web page elements + show how to use them to build user-friendly pages and sites
- We consider how pages can be effectively combined into websites + how websites are published
- We use many exercises to practise ideas
- Cover HTML, CSS, JavaScript
- We don't cover server-side programming – this would require a further course.

## *Internet and Web concepts*

- *Internet*: global network of connected computers/devices – started in 1969
- *Web*: World-Wide Web (WWW) is all information/data on internet
- *HTML*: structured text containing links to other text/data across internet. Invented in 1989.
- By early 90's many companies were beginning to establish websites
- 90's: websites become more sophisticated, Google, Amazon, etc
- 00's: massive expansion of online services for commercial + social uses
- 10's: some online services replace traditional stores (eg., bookshops, record shops). 4G – Web via mobile devices.

## *Internet and Web concepts*

- WWW originally designed for exchange of scientific information, but soon adopted for commercial and social uses
- Key advantage of websites – they may be accessed from any computer connected to internet, anywhere in world
- Commercial use: as ‘shop window’ advertising the company, + product/service sales
- Social use: to promote/publicise non-commercial organisations and services.

Some example websites: Community organisations ([friendsofcarnegielibrary.org.uk](http://friendsofcarnegielibrary.org.uk)); businesses ([ratrecordsuk.net](http://ratrecordsuk.net)).

The screenshot shows a web browser window with the address bar displaying 'friendsofcarnegielibrary.org.uk'. The page features a navigation menu with links for Home, Join, Aims, Events, Services, Forums, Newsletters, History, and Contact. The main content area is titled 'PUBLIC MEETINGS ON THE FUTURE OF CARNEGIE LIBRARY' and includes a sub-header 'Promoting Use of and Access to the Carnegie Library in Herne Hill. We support the DefendTheTen campaign which is working to save all 10 Lambeth libraries: http://defendthe10-lambeth.org.uk'. A sidebar on the right contains a 'Login' button and a section titled 'The Carnegie on Twitter' featuring a tweet from @CarnegieLib.

Friends of Carnegie Library

*Promoting Use of and Access to the Carnegie Library in Herne Hill. We support the DefendTheTen campaign which is working to save all 10 Lambeth libraries: <http://defendthe10-lambeth.org.uk>*

Home Join Aims Events Services Forums Newsletters History Contact

**PUBLIC MEETINGS ON THE FUTURE OF CARNEGIE LIBRARY**

September 18, 2016 Campaigns, Events, News Nicholas Edwards

With our partners in the Carnegie Library Association CIO, the Friends are arranging two public meetings to discuss plans for the future. The announcement leaflet is [here](#)

Are you a Friend? Login

*The Carnegie on Twitter*

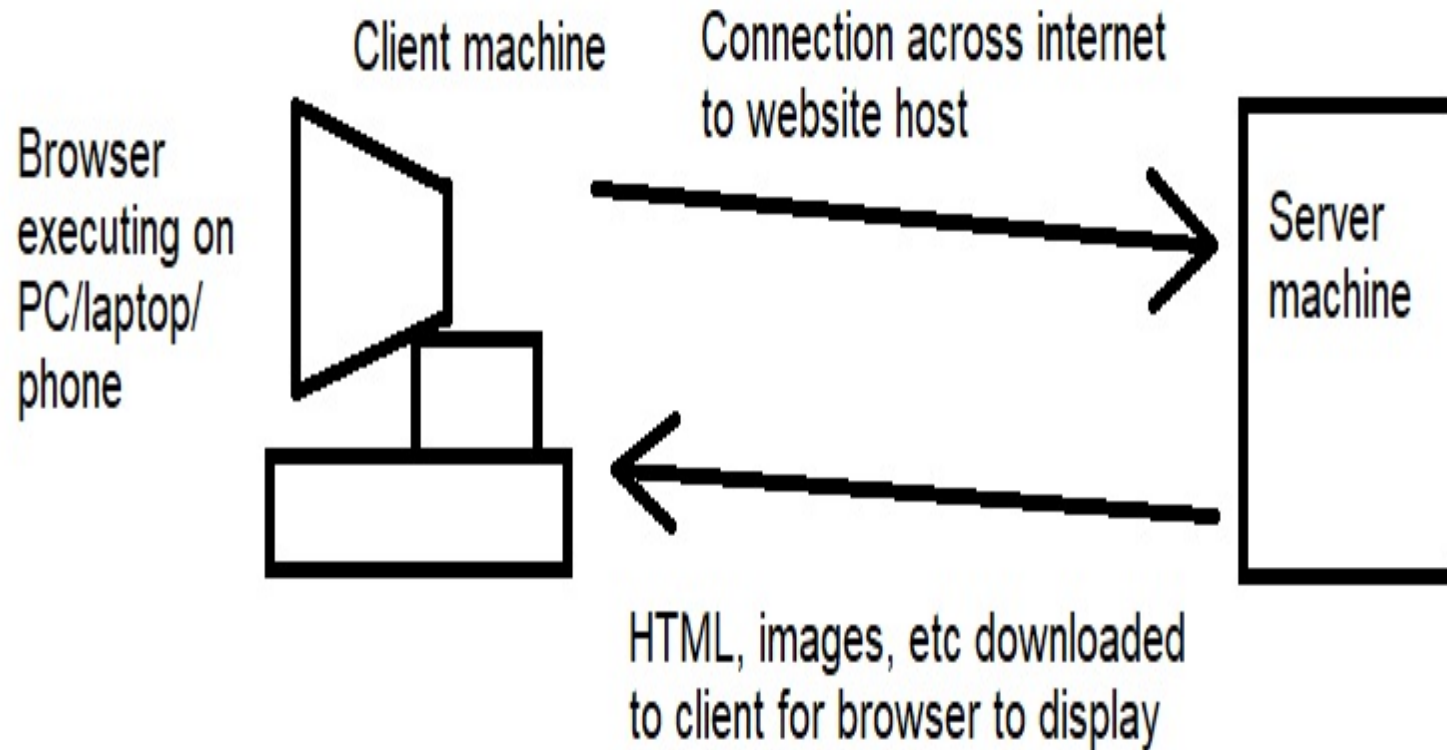
Carnegie Library @CarnegieLib  
This Census-taker by China Mieville  
[ethlibraryreadinggroups.wordpress.com/2016/09/09/thi...](http://ethlibraryreadinggroups.wordpress.com/2016/09/09/thi...)

Home page of Friends of Carnegie Library



## *Internet and Web concepts*

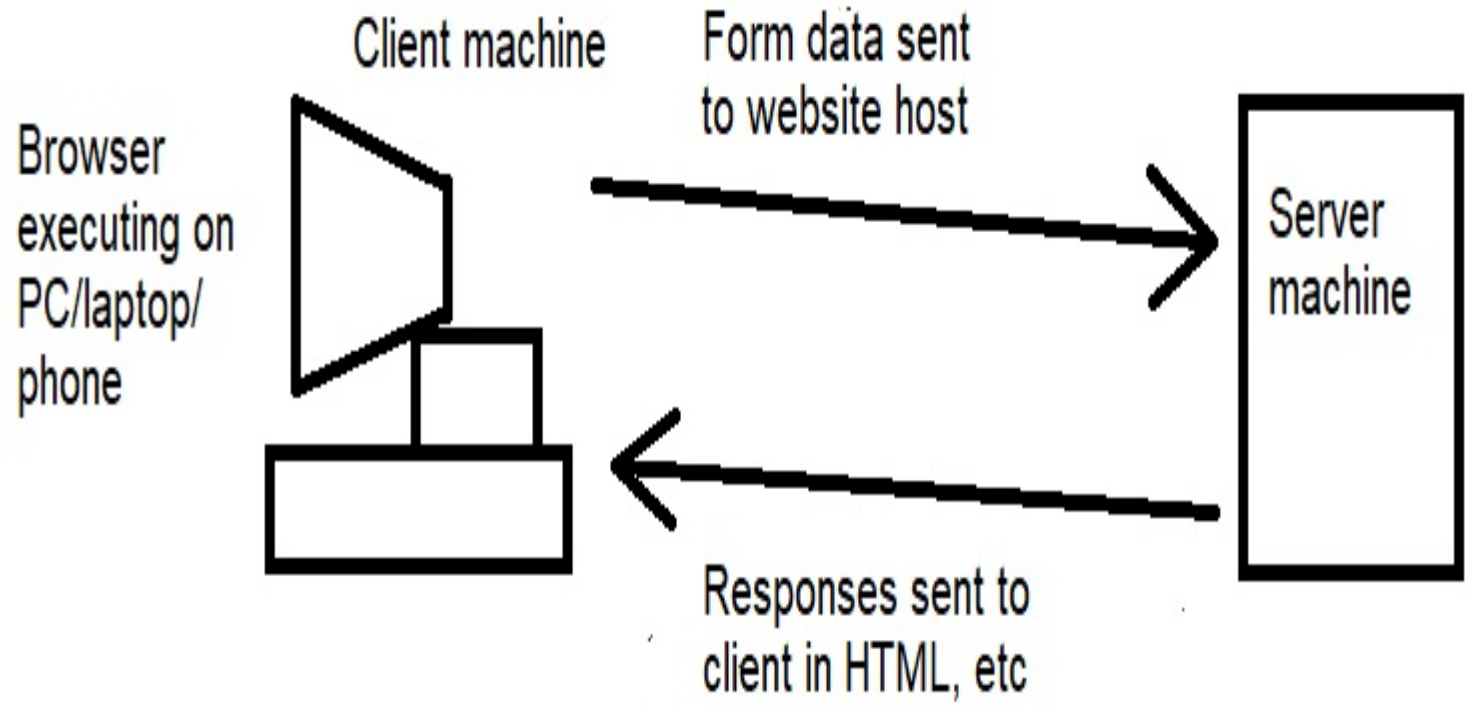
- A website consists of a set of *Web pages*, located (hosted) on a computer (termed the server or host)
- Host may be internal to the owner of the website, or provided by a 3rd-party organisation specialising in hosting sites
- Browsers connect across internet to server, and download HTML and other files (eg., images) to be displayed on client device (your computer/phone).
- As you browse + follow links, so other websites are contacted + provide their data also.



The internet is used to access websites

### *Internet and Web concepts*

- Apart from viewing pages, client can also interact: submitting data via forms, performing searches, buying products, etc
- Browsers can send information and requests from client to server, via internet
- Typically the server responds with more HTML, eg., a confirmation page or page of search results.



Exchange of information between client and server

## *Browsers and HTML*

- The web pages you view in a browser are usually formatted using HTML (Hypertext Markup Language)
- HTML defines the text, images and structure of pages
- Web pages are just text files with a *.html* extension, eg.:  
`index.html`
- You can create web pages with any text editor, following the standard structure of HTML
- Preview your files in any browser
- Knowing HTML enables you to create webpages, and to manage websites.

## *Browsers and HTML*

*example1.html* text file:

```
<html>
```

```
<head>
```

```
<title>Example of a web page</title>
```

```
</head>
```

```
<body>
```

```
<h1>Example of a web page</h1>
```

```
Some text. Text in <b>bold</b>.
```

```
Some text <i>in italic</i>. <p>
```

```
2nd paragraph. Some text
```

```
in <i><b>bold italic.</b></i>
```

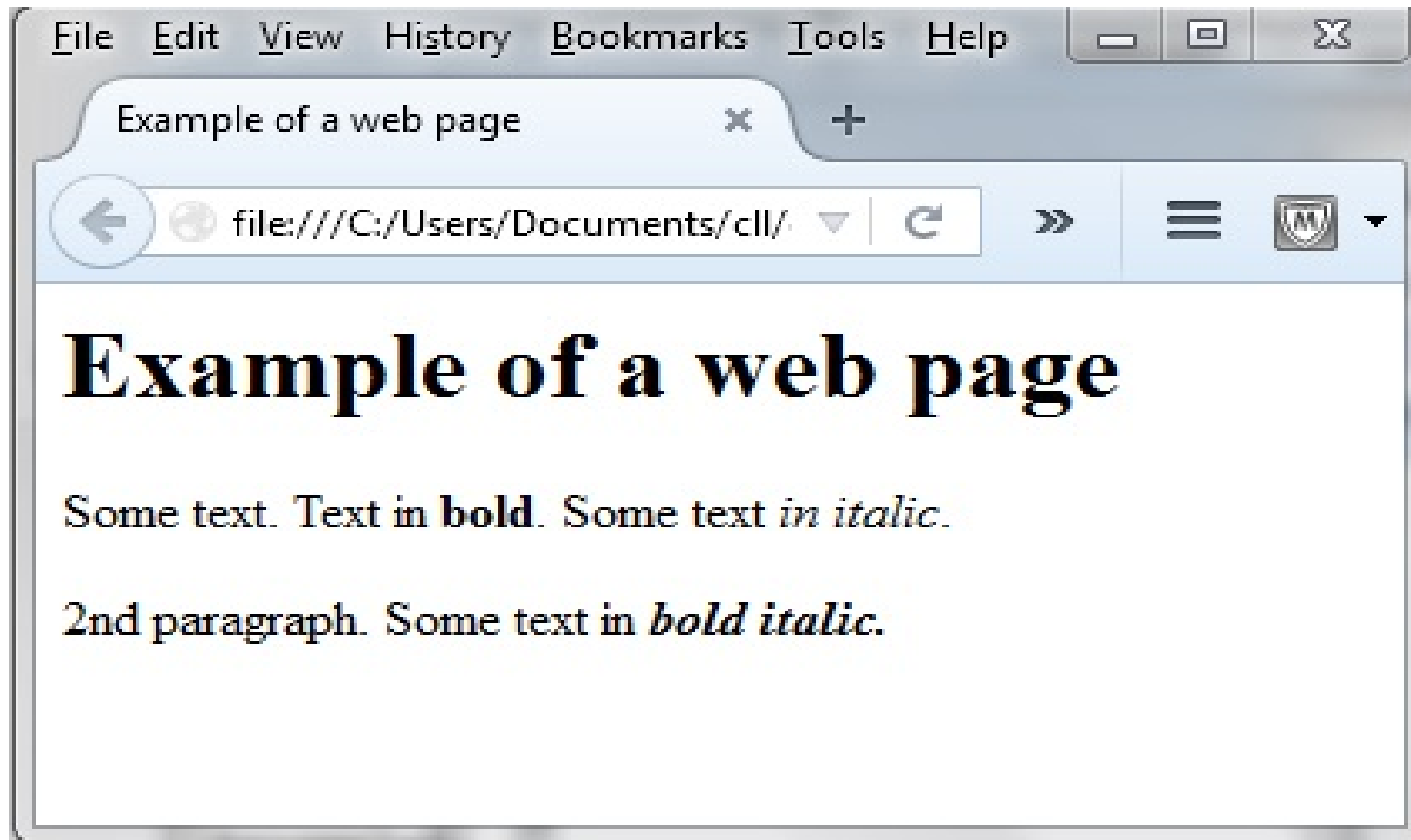
```
</body>
```

```
</html>
```

These files can be edited with WordPad, Notepad or other plain text editor.

View them in any browser – Firefox, Internet Explorer, Chrome, etc.

*www.w3schools.com* gives definitions + many examples of HTML. Also see the glossary.



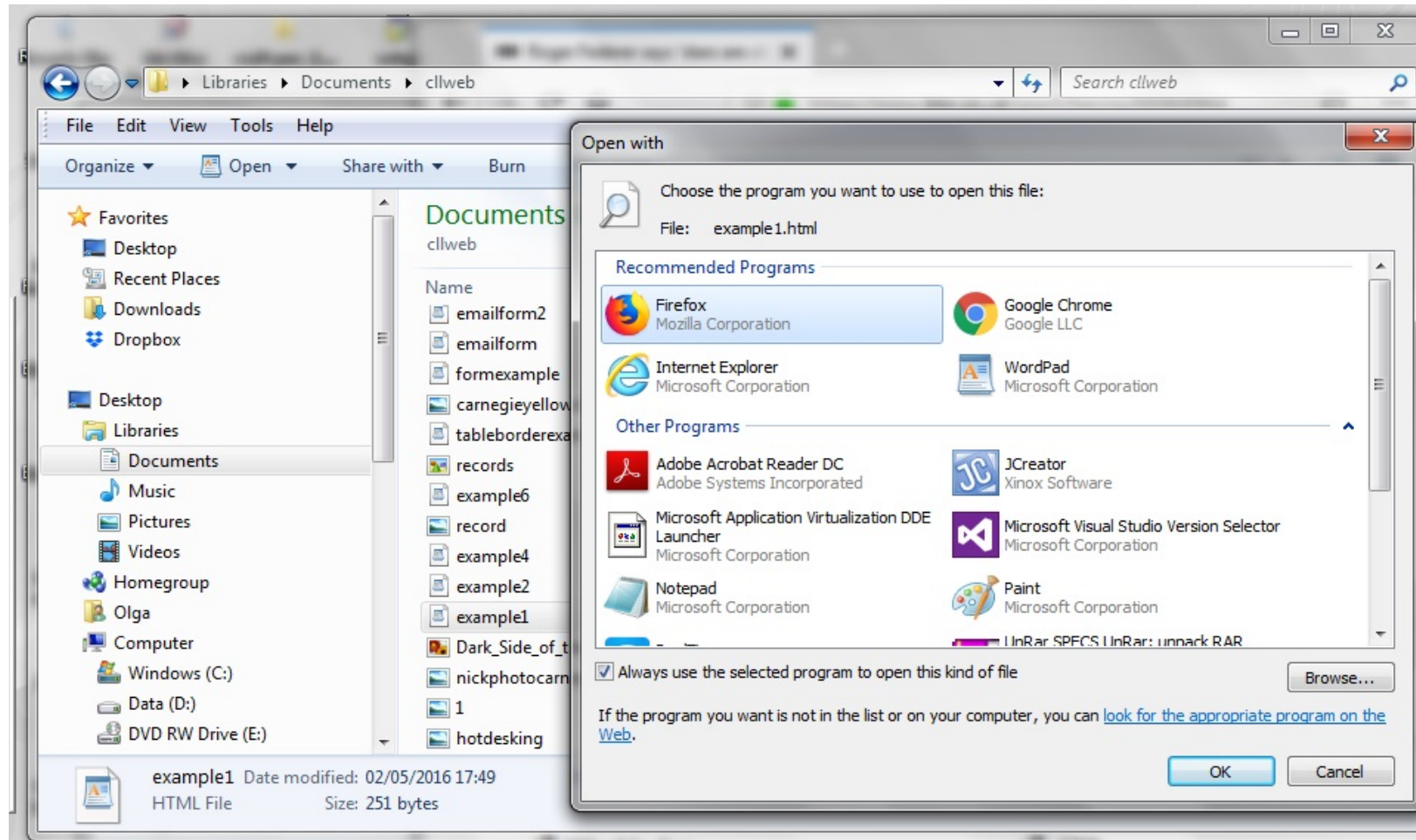
example1.html in browser



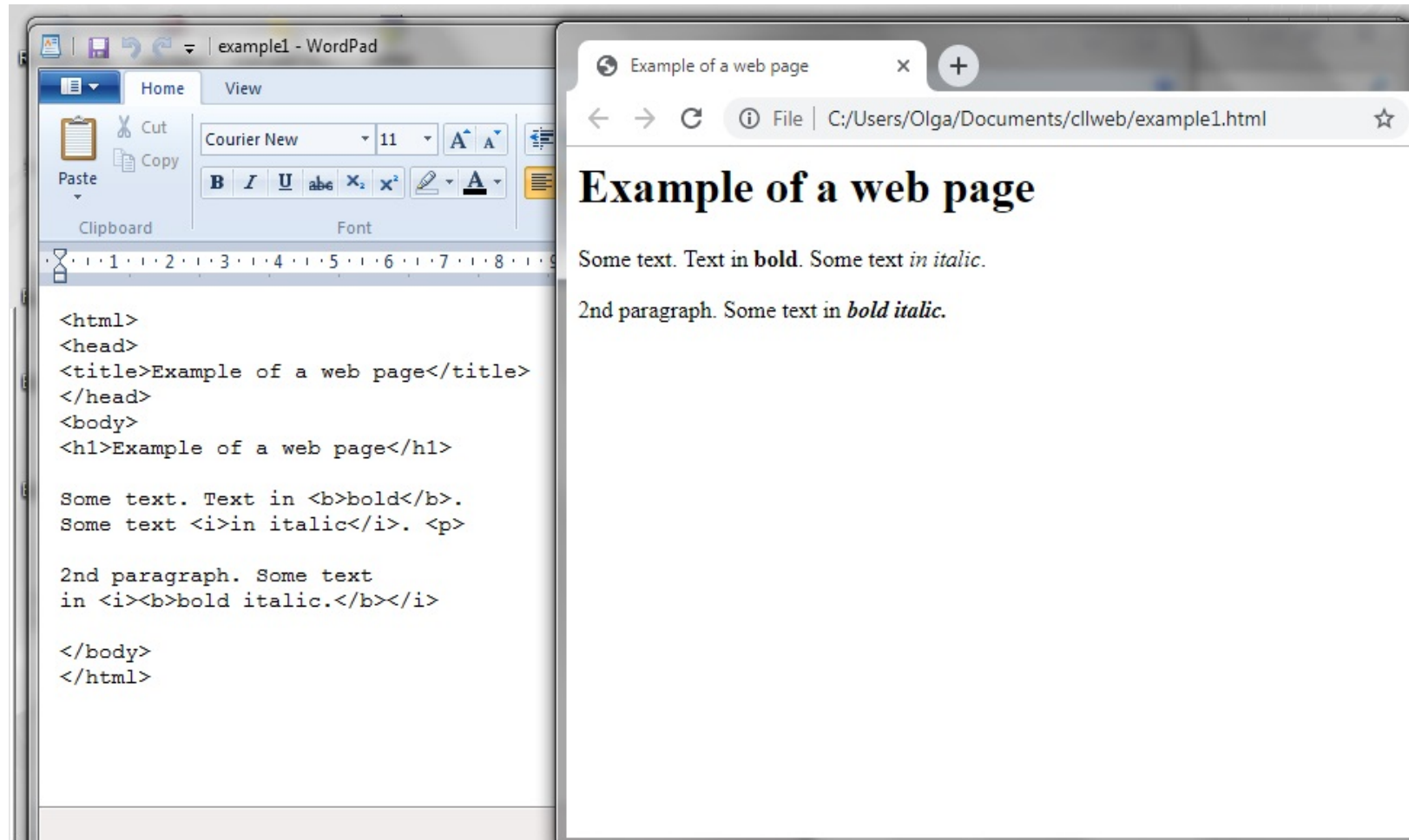
### *Editing web files*

- We will edit HTML files in *WordPad*
- At same time, view them in a browser.

Use “open with” option to open HTML files with WordPad.



Open HTML file in WordPad



Edit in WordPad and view in browser

## Session 2: Writing simple Web pages

- Structure of Web pages
- HTML concepts
- Text formatting and page colours
- Exercise: editing simple Web page

## *Structure of Web pages*

The basic structure of a simple web page is:

```
<html>  
<head>  
<title>Title</title>  
</head>
```

```
<body>  
<h1>Title</h1>
```

All text and content that will be shown by a browser viewing the page.

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

## *Structure of Web pages*

- `<html>` starts the document
- `<head>` starts the page head section, containing a title (this is shown in browser tab):

```
<title>Example of a web page</title>
```

- `</head>` ends the header
- `<body>` starts the body, the main content of page.

## *HTML concepts*

In the body:

```
<h1>Example of a web page</h1>
```

defines a main section header *h1* (largest size – can have *h2*, *h3* etc for subheadings).

Some text. Text in **<b>bold</b>**.

Some text *<i>in italic</i>*. `<p>`

Defines text, with bold, italic display of parts. `<p>` ends the paragraph.

2nd paragraph. Some text

```
in <i><b>bold italic.</b></i>
```

Shows how to have both bold and italic.

Finally, `</body>` ends the body, and `</html>` ends document.

## *HTML concepts*

- HTML instructions are written between `<` and `>` characters – these instructions control how page is displayed by a browser
- Pages always begin with `< html >` and end with `< /html >`
- Usually, a `< tag >` has a following `< /tag >` element
- `< head >` and `< /head >` contain header of the page, such as the title
- `< body >` and `< /body >` contain main content of the page.

Try opening *example1.html* in Internet Explorer, Chrome and Firefox. Are there differences in its appearance?



## *Text formatting and page colours*

- Different tags are used to display information, to emphasise important information
- *h1* is used for main headings in page (like newspaper main headlines). Expect users to focus on these first – so must be clear + relevant
- *h2*, *h3* for subheadings
- Bold, italic for particular emphasis – but avoid underlining because links are usually underlined.

Notice that displayed text is not broken into lines or paragraphs based on the source text lines/paragraphs.

Need explicit `<p>` to make a paragraph break, and `<br>` to break a line.

## *Text formatting and page colours*

Text/background colours should be chosen appropriately and with sufficient contrast for readability.

*example2.html* text file:

```
<html>
<head>
<title>Vinyl Revival</title>
</head>
<body style="background-color: silver; color: blue">
<h1>Vinyl Revival</h1>
```

We sell all styles of LPs + 45s from  
the 1950's onwards.

```
<h2>Subheading 1</h2>
```

Some text in first section. <p>

<h2>Subheading 2</h2>

Some text in 2nd section.

<hr>

Text below horizontal rule.

</body>

</html>

The body `style="..."` instruction sets page text and background colours.

This is an *attribute* of the *body* tag.

# Vinyl Revival

We sell all styles of LPs + 45s from the 1950's onwards.

## Subheading 1

Some text in first section.

## Subheading 2

Some text in 2nd section.

---

Text below horizontal rule.

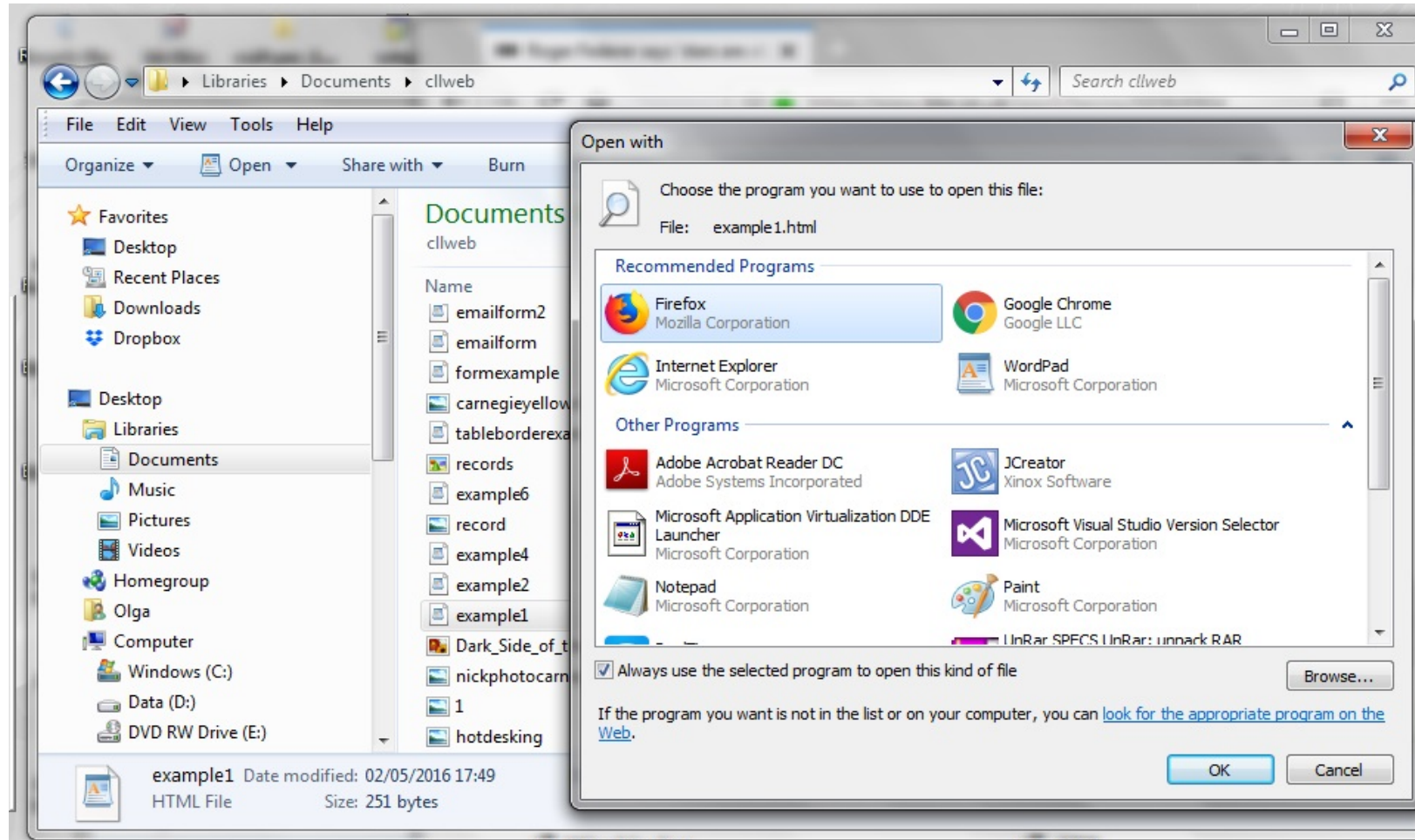
example2.html in browser

## *Text formatting and page colours*

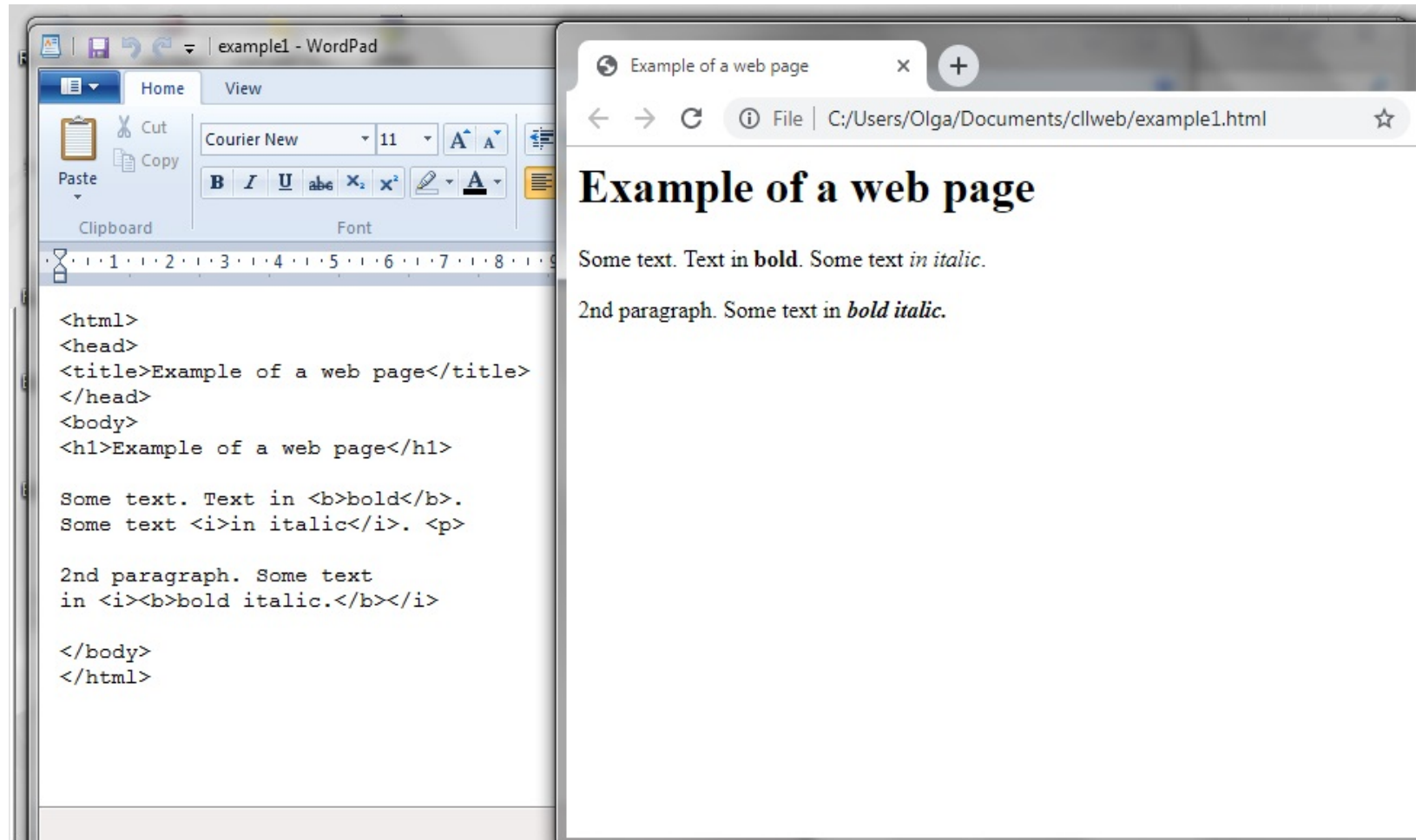
- Many predefined colours: red, green, blue, yellow, pink, silver, etc.
- Colours can also be given by `color="#AA9988"` codes, red-green-blue coding with values 0 to 9, A, B, C, D, E, F. Higher values are lighter.  
`#FF0000` is pure red. `#00FF00` is pure green, etc. (What colour is `#FFFFFF`?)
- It is usual to have the foreground (ie., text) darker than the background – but the reverse can be used for special effect.
- Preview page in different browsers to check your colour scheme works in each.

*Exercise: editing simple Web page*

- Try changing colours of the *example2.html* page in the text file, + reload it in browser.
- Look up possible colours to use in glossary/at [www.w3schools.com](http://www.w3schools.com).
- Try devising your own colours using the # codes.



Open HTML file in WordPad



Edit in WordPad and view in browser



## Session 3: Lists and tables

- Ordered and unordered lists, list styles
- Tables and table formatting
- Exercise: writing tables

### *Lists and list styles*

- Lists are used to show different related subjects
- Can be unordered (bulleted lists) or ordered (numbered)
- Typically each item will have a clickable link to a specialised web page for that topic/product/service
- Lists should not be too long (eg., no more than 7 items)
- Descriptions should be concise + clear.

## *Lists and list styles*

*example3.html* text file:

```
<html>
<head>
<title>Vinyl Revival</title>
</head>
<body style="background-color: silver; color: blue">
<h1>Vinyl Revival</h1>
```

We sell all styles of LPs + 45s from  
the 1950's onwards.

Unordered list:<br>

```
<ul>
<li>Pop
<li>Rock
```

```
<li>Easy listening  
<li>Classical  
</ul>
```

```
Ordered list:<br>
```

```
<ol>  
<li>First item  
<li>Second item  
<li>Third item  
</ol>
```

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

# Vinyl Revival

We sell all styles of LPs + 45s from the 1950's onwards. Unordered list:

- Pop
- Rock
- Easy listening
- Classical

Ordered list:

1. First item
2. Second item
3. Third item

example3.html in browser

## *Lists and list styles*

`<br>` is a line break.

`<ul>` starts an unordered list, each list item started by `<li>`. `</ul>` ends the list.

By default, bullet points are used to display items – but squares or circles could be used instead: `<li type="square">` or `<ul type="circle">`

Ordered list started by `<ol>`, ended by `</ol>`. Items appear with numbers 1, 2, 3, etc – again this format can be altered. `<ol type="I">` gives Roman capital numbering. Other options: i, a, A. It is preferable to define list styles using style sheets (Session 8).



Different list styles

### *Tables and table formatting*

- Tables present information/data in a regular format
- Often used to show search results of products/prices, etc
- Can also be used to organise other page content (especially forms) in a regular layout.

Idea is similar to a Word or Excel table.



## *Tables and table formatting*

A table example is in *example4.html* text file:

```
<html>
<head>
<title>Vinyl Revival</title>
</head>
<body style="background-color: silver; color: blue">
<h1>Vinyl Revival</h1>
```

Table example:<br>

```
<table>
<tr><th>Record</th> <th>Price</th></tr>
<tr><td>Beatles 'Help'</td> <td>&#163; 3.99</td></tr>
<tr><td>Sinatra 'My Way'</td> <td>&#163; 1.99</td></tr>
<tr><td>Moby 'Play'</td> <td>&#163; 2.99</td></tr>
```

```
<tr><td colspan="2">More are available</td></tr>  
</table>
```

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

&#163 is HTML code for pound sign.

`<table>` begins the table, `</table>` ends it.

`<th> Header </th>` is a column header.

`<td> data </td>` is data entry in table.

`<tr> ... </tr>` is a table row. Consists of header cells or data cells.

Try adding another column Condition with values Good, Mint, Fair for the 3 rows.

# Vinyl Revival

Table example:

<b>Record</b>	<b>Price</b>
Beatles 'Help'	£3.99
Sinatra 'My Way'	£1.99
Moby 'Play'	£2.99
More are available	

example4.html in browser

## *Tables and table formatting*

Tables can be given borders to emphasise the structure:

```
<html>
<head>
<title>Vinyl Revival</title>
</head>
<body style="background-color: lime; color: black">
<h1 style="font-family: fantasy">Vinyl Revival</h1>
```

Table with border and fixed

widths:<br>

```
<table border="2">
<tr><th width="200px">Record</th>
      <th width="100px">Price</th></tr>
<tr><td>Beatles 'Help'</td> <td>£ 3.99</td></tr>
```

```
<tr><td>Sinatra 'My Way'</td> <td>#163 1.99</td></tr>
<tr><td>Moby 'Play'</td> <td>#163 2.99</td></tr>
<tr><td colspan="2">More are available</td></tr>
</table>

</body>
</html>
```

The instruction `<th width="200px">Record</th>` sets width of this column to be 200 pixels.

Update your extended example with widths 200, 100 and 80 for the 3 columns. Center the final row text using

```
<center>text</center>
```

The screenshot shows a web browser window with the address bar containing the path `C:\Users\Olga\Documents\cll\example` and the page title `Vinyl Revival`. The main content area has a blue background and features the heading `Vinyl Revival`. Below the heading is the text `Table with border and fixed widths:`. A table with a black border and fixed widths is displayed, containing the following data:

Record	Price
Beatles 'Help'	£3.99
Sinatra 'My Way'	£1.99
Moby 'Play'	£2.99
More are available	

Improved table example in browser

## *Tables and table formatting*

Table alignment can be set in the table style to center all table text.

Column widths can be specified as percentages of table width instead of as absolute values:

```
<html>
<head>
<title>Vinyl Revival</title>
</head>
<body style="background-color: lime; color: black">
<h1 style="font-family: fantasy">Vinyl Revival</h1>

<table border="2" style="text-align: center" width="80%">
<tr><th width="60%">Record</th>
    <th width="40%">Price</th></tr>
<tr><td>Beatles 'Help'</td> <td>£ 3.99</td></tr>
```

```
<tr><td>Sinatra 'My Way'</td> <td>&#163 1.99</td></tr>
<tr><td>Moby 'Play'</td> <td>&#163 2.99</td></tr>
<tr><td colspan="2">More are available</td></tr>
</table>

</body>
</html>
```

Table style `text-align: center` applies to all table data.

`<table ... width="80%">` means table width is 80% of browser window.

`<th width="60%">Record</th>` means column width is 60% of table width.



### *Tables and table formatting*

- In general it is best to use the default settings for tables – header text is centered, cell text is left-aligned
- Fixed width is more usual than variable width
- Borders help to mark out the table from rest of page.

The screenshot shows a web browser window with a single tab titled "Vinyl Revival". The address bar contains the file path "file:///F:/c/l/tableexercisolution.html". Below the address bar is a search bar and a navigation bar with links for "Most Visited", "Getting Started", "Amazon", "eBay", "Suggested Sites", "Web Slice Gallery", and "WildTangent Games". The main content area has a black background and features the heading "Vinyl Revival" in large white text. Below the heading is the text "Table with border and fixed widths:" followed by a table with three columns: "Record", "Condition", and "Price". The table contains three rows of data and a final row with the text "More are available".

Record	Condition	Price
Beatles 'Help'	Good	£ 3.99
Sinatra 'My Way'	Mint	£ 1.99
Moby 'Play'	Good	£ 2.99
More are available		

Table with center alignment and variable width

## *Summary*

- Lists provide structured group of items, with *ul* or *ol* tags
- Tables organise items in regular grid, using *table*, *tr*, *th*, *td* tags
- Many choices for design and layout.

## Session 4: Images and animations

- Adding images to pages
- Using animations and videos
- Exercise: using images and animations

### *Adding images to pages*

- Images are essential part of most pages. *img* or *image* tag
- Can illustrate products, services, provide information or be decorative
- Since image data is transferred from server to client with web page that refers to it, image files should be of minimal size
- Can show low resolution thumbnail image, with high resolution reached by clicking on the thumbnail.

An example image tag is

```

```

## *Adding images to pages*

An image example and animation example are in *example5.html* text file:

```
<html>
<head>
<title>Vinyl Revival</title>
</head>
<body style="background-color: silver; color: blue">
<h1>Vinyl Revival</h1>
```

Image example:<br>

```
<p>
```

Animation example:<br>

```
<p>
```

Image with text:<br>

```
 <h3>Pink Floyd 'Dark Side  
of the Moon'</h3>
```

The classic rock LP from 1973.

Mint-quality vinyl with original  
cover and insert. &#163 15.99.

```
</body>
```

```
</html>
```

 inserts the image from file f.jpg  
into web page, with width 50 pixels.

Try resizing the images to improve the layout.

Notice that animations may play faster/slower in different browsers.

### *Adding images to pages*

`height="100"` can also be specified in `image`, and `title="Spinning record"` for popup text description.

The *img* style can be used to position the image relative to text. Eg., *float : right*, *float : left*.

A thumbnail image can link to the full-sized image:

```
<a href="fullsize.jpg">  
  <image src="thumbnail.jpg" width="30">  
</a>
```

Clicking on the thumbnail image takes browser to full-sized version.



## Vinyl Revival

Image example:



Animation example:



Image with text:



**Pink Floyd 'Dark Side of the Moon'**

The classic rock LP from 1973. Mint-quality vinyl with original cover and insert. £15.99.

example5.html in browser

### *Adding images to pages*

- The images are in *f.jpg*, *f.gif* or *f.png* files
- *jpg* for photographs/high-quality images
- *gif* for logos, animations
- *png* format can be used for either purpose.

`alt="Description"` is used to give description of image, which can be read out by assistance software.

## *Animations and videos*

To insert a video, use the tag:

```
<video src="video.webm" width="400" controls>
```

Message

```
</video>
```

*src* identifies the video file. *controls* inserts play/stop etc controls.

Can also specify poster image by `poster="posterimage.jpg"`

Title as: `title="Title"`

Other formats, such as *mp4*, can also be used.

Not all browsers support *video* tag yet. *Message* appears if there is no support.

## *Animations and videos*

Example page *videoexample.html* with video:

```
<!DOCTYPE html>
<HTML>
<HEAD> <TITLE> Video example </TITLE> </HEAD>
<BODY bgcolor = "white">
<h1> Video example </h1>

<video src="realshort.mp4"  loop  width="400"  controls>
Message
</video>
</BODY> </HTML>
```

## *Animations and videos*

- Images and especially animations and videos can distract users attention from main purpose of a page
- Only include if they are necessary and consistent with aim of page
- Never use *autoplay* on a video.
- A common feature is a rotating ‘banner’ of images – but can be annoying to users!

Likewise for *loop* video option ...

Large images and videos can also slow page loading + require good Internet connection.

*Rotating banner images: loop.html*

```
<html>
<head>
  <title>Image Loop</title>
  <script type = "text/javascript">
    var count = 0;

    function startTimer()
    { window.setInterval("tick()",1000); }

    function tick()
    { count = (count + 1) % 5;
      document.photos.src = count + ".jpg";
    }
  </script></head>
  <body onload = "startTimer()">
```

```
<center>  
<img name = "photos" src = "1.jpg" height = "300">  
</center>  
</body>  
</html>
```

## *Rotating banner images*

- Example of a *script* – small program which executes in browser to read + write web page content dynamically
- Here the *tick()* function executes every second (1000 milliseconds) and increments *count*
- *count* cycles through values 0, 1, 2, 3, 4, 0, 1, ....
- The *img src* in the page body is set successively to *0.jpg*, *1.jpg*, *2.jpg*, etc
- Image files must be identical in width, + similar in resolution/style
- In this example, are images of a park.



## *Summary*

- Images make pages more visually attractive and can give essential information. *img* or *image* tags
- Photographic images with *jpg* or *png* files, animations with *gif* or *png* files
- Videos with *video* tag.

*Exercise: using images and animations*

Write a page extending *example4.html* to show small images of each record in an additional column in the table.

The images are in *help.jpg*, *myway.jpg* and *play.jpg*.

Define the small images as clickable links to the actual image files.

## Session 5: Links: connecting webpages

- Internal links
- External links
- Exercise: linking pages

## *Internal links*

- Links are key concept of WWW and HTML
- Clicking on a link tells browser to load the page referred to by the link
- A browsing session often consists of jumping from page to page by following links
- Links can be internal – to other pages on the same website – or external, to other sites.

When building a website, you design separate pages which cross-reference each other via internal links.

Internal link has *a* (anchor) tag:

```
<a href="page.html">Text label</a>
```

Clicking on the text label opens *page.html* in browser.

## *Internal links*

In the record store, can use internal links to different pages for specialised music categories. Eg., *example6.html* text file:

```
<html>
<head>
<title>Vinyl Revival</title>
</head>
<body style="background-color: silver; color: blue">
<h1>Vinyl Revival</h1>
```

We sell all styles of LPs + 45s from  
the 1950's onwards.

For our current list of products, see  
<a href="products.html">Products</a><p>

Links in a list:<br>

<ol>

<li> <a href="fifties.html">1950's</a>

<li> <a href="sixties.html">1960's</a>

<li> <a href="seventies.html">1970's</a>

</ol>

<hr>

Email link:<br>

<a href="mailto:enquiries@vinylrevival.co.uk">Contact Us</a>

</body></html>

# Vinyl Revival

We sell all styles of LPs + 45s from the 1950's onwards. For our current list of products, see [Products](#)

Links in a list:

1. [1950's](#)
2. [1960's](#)
3. [1970's](#)

---

[Email link:](#)

[Contact Us](#)

example6.html in browser

### *Internal links*

`<a href="products.html">Products</a>` appears in document as Products. When clicked, *products.html* is loaded into browser.

Assumes *products.html* is in same directory as the page that refers to it. If not, can put complete URL:

```
<a href =  
"http://www.vinylrevival.co.uk/products.html">Products</a>  
or a location relative to current page:
```

```
<a href = "../pages/products.html">Products</a>
```

Mail link opens up a mailing program when clicked: `<a href = "mailto:enquiries@vinylrevival.co.uk">Contact Us</a>`



## *Internal links*

Also possible to link to different parts of same page:

```
<body>  
<h1 id="top">Main header</h1>
```

Long page content ...

```
<a href="#top">Go to top</a>  
</body>
```

## *External links*

- Internal links remain within same website
- External links support navigation to another web site, or to another program, such as a mailing program
- For external links, specify complete URL (web location).

External link:

```
<a href="http://xxx.org.uk/page.html">Text label</a>
```

Makes request to *xxx.org.uk* for the *page.html* file.

A request

```
<a href="http://xxx.org.uk/">Text label</a>
```

requests *index.html* from the server of *xxx.org.uk*.

## *Summary*

- Links enable the navigation from one page to a related page. *a* tag with *href*
- Internal links are within one site
- External links go to a different website.

*Exercise: linking pages*

For *exercise6.html*, try writing the linked pages *products.html*, *fifties.html*, etc.

Easiest way to create a new page is to copy an existing page, and rename and edit the copy.

Save all files in the same directory as *exercise6.html* and test that the links work correctly.

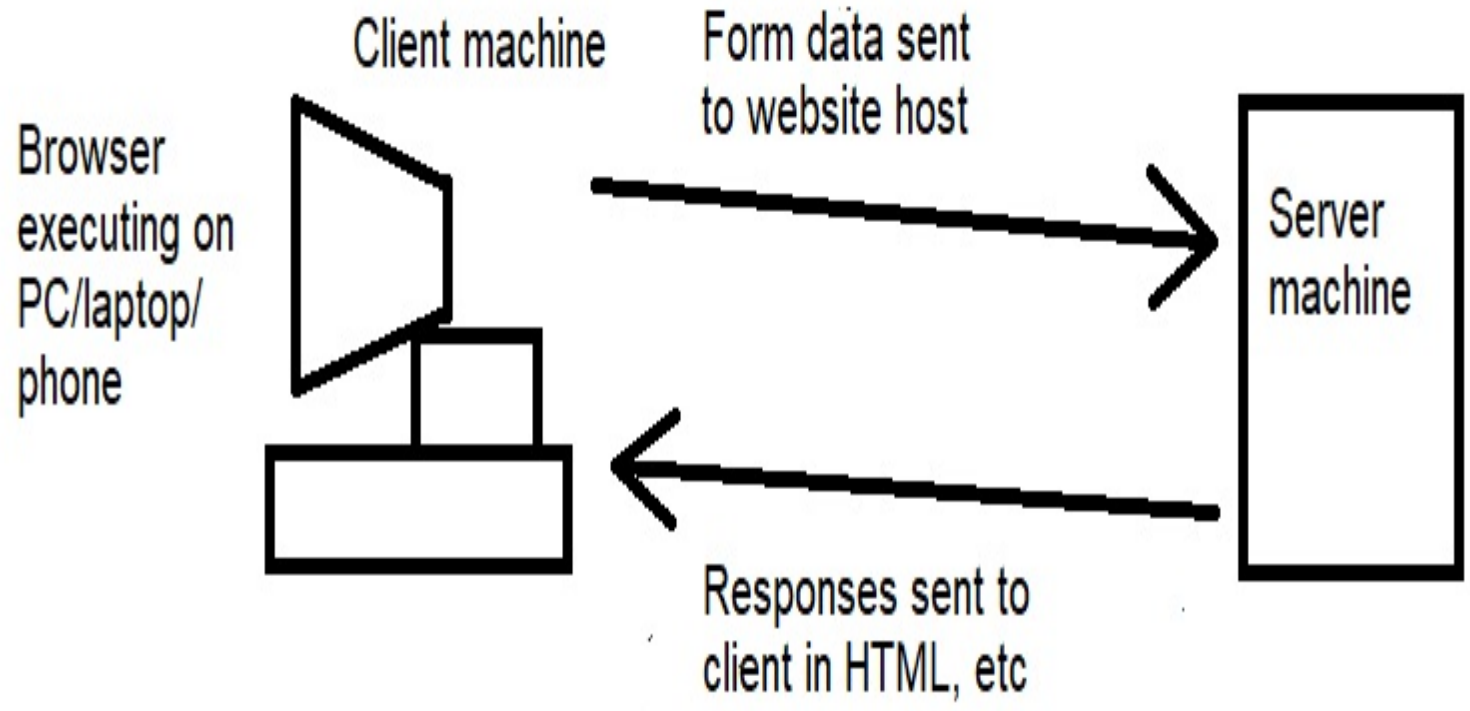
## Session 6: Forms: sending data from webpages

- Form concepts: Post and Get
- Form controls and formatting
- Form layout using tables
- Exercise: design a simple form

## *Form concepts: Post and Get*

- Forms are used to submit requests, to obtain information, etc – maybe even for electronic voting.
- Form tag is *form*, with structure

```
<form method="post"
  action="program URL">
... form content ...
</form>
```
- *method* is the way the form data is submitted (*post* for secure data, *get* otherwise).
- *action* names the server-side program which will process the data.
- content of the form is set of data entry fields with labels.
- Usually a ‘submit’ button is placed at end of the form.



Exchange of information between client and server

## *Form controls and formatting*

A simple form example is the *example7.html* text file:

```
<html>
<head>
<title>Vinyl Revival: Order a record</title>
</head>
<body style="background-color: silver; color: blue">
<h1>Vinyl Revival: Order a record</h1>
```

We sell all styles of LPs + 45s from  
the 1950's onwards.

```
<form method="post"
  action="http://localhost:8080/orders/Order.jsp">
```

```
Product: <input type="text" name="product"/><br>
```



```
Your name: <input type="text" name="name"/><br>
Address: <input type="text" name="address"/><br>
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```

The *name* of an input field is used by the receiving program (eg., *Order.jsp*) to process the submitted form data.

Possible to pre-fill field contents, eg: `<input type="text" name="name" value="John Smith"/>`. But this can be annoying to users!

What happens if *post* is replaced by *get*? Try filling in data + submitting with *get* method.

# Vinyl Revival

We sell all styles of LPs + 45s from the 1950's onwards.

Product:

Your name:

Address:

Submit

example7.html in browser

## *Form controls and formatting*

Forms are started with

```
<form method="post"
  action="http://localhost:8080/...">
```

The `action` identifies a program that will process form data when submitted to server.

Text fields are defined like this:

```
Product: <input type="text" name="product"/><br>
```

The *Submit* button and end of form is defined by:

```
<input type="submit" value="Submit"/>
</form>
```

## *Form layout using tables*

Tables can be used to give neat layout of forms.

*example7table.html* text file:

```
<html>
<head>
<title>Vinyl Revival: Order a record</title>
</head>
<body style="background-color: silver; color: blue">
<h1>Vinyl Revival</h1>
```

We sell all styles of LPs + 45s from  
the 1950's onwards.

```
<form method="post"
  action="http://localhost:8080/orders/Order.jsp">
```

```

<table>
<tr><th width="100px"> </th>
    <th width="200px"> </th></tr>
<tr><td>Product:</td>
    <td><input type="text" name="product"/></td></tr>
<tr><td>Your name:</td>
    <td> <input type="text" name="name"/></td></tr>
<tr><td>Address:</td>
    <td> <input type="text" name="address"/></td></tr>
<tr><td> </td>
    <td><input type="submit" value="Submit"/></td></tr>
</table>
</form>
</body></html>

```

Format is one row for each text field. Label of fields go in 1st column, field + button go in 2nd.

# Vinyl Revival

We sell all styles of LPs + 45s from the 1950's onwards.

Product:

Your name:

Address:

Submit

Form with table layout

## *Submitting forms by email*

Form action can also send an email:

```
<form action="mailto:your.email@com?subject=Register"
  method="post">
<table width="250">
<tr><td>Full name<br>
  <input name="name" type="text" size="90"></td></tr>
<tr><td>Address<br>
  <input name="address" type="text" size="90"></td></tr>
<tr><td>Phone<br>
  <input name="phone" type="text" size="50"></td></tr>
<tr><td>Email<br>
  <input name="email" type="text" size="50"></td></tr>
<tr><td><input type="submit" value="Submit"></td></tr>
</table>
</form>
```

Please sign up here

file:///C:/Users/Documents/cil/emailform.html

Most Visited Getting Started Amazon eBay

Please sign up here:

**Full name**  
nnhh

**Address**  
oopp

**Telephone**  
ddf

**Email**  
dde

Submit

Email form



### *Submitting forms by email*

```
<form action="mailto:your.email@com?subject=Register"
  method="post" id="mailform">
  <table width="250">
  <tr><td>Full name<br>
    <input name="name" type="text" size="90"></td></tr>
  <tr><td>Address<br>
    <textarea name="address" rows="5" cols="90" form="mailform">
    </textarea></td></tr>
  <tr><td>Phone<br>
    <input name="phone" type="text" size="50"></td></tr>
  <tr><td>Email<br>
    <input name="email" type="text" size="50"></td></tr>
  <tr><td><input type="submit" value="Submit"></td></tr>
  </table>
</form>
```

### *Text areas*

The address field uses a text area:

```
<tr><td>Address<br>  
<textarea name="address" rows="5" cols="90" form="mailform">  
</textarea></td></tr>
```

This has 5 rows and 90 character spaces in each row.

Please sign up here:

**Full name**

**Address**

**Telephone**

**Email**

Email form with text area

## *Summary*

- Forms enable the submission of data from client to server.  
*form* tag
- Different kinds of data input fields, usually with *input* tag
- Tables can be used to give regular format for forms.

*Exercise: form design*

Extend *example7table.html* to include a credit card number field and a field for the payment amount.

Use the *get* form method to test that your form submission works correctly.

## Session 7: Advanced forms

- Different input controls: text, radio, checkbox, pulldowns, file upload
- Scripts and data verification
- Exercise: writing forms with controls

### *Different input controls*

- Different input controls can be used to enter different kinds of data
- Selection lists (pulldowns) used to choose one option from a list (eg., a country from list of all countries)
- Radio buttons also allow 1 choice from several
- Checkboxes allow multiple choices of several options
- File selectors enable file upload from client to server.

## *Selection lists/pulldowns*

Lists of options to select from:

```
<label for="make">Make:</label>
<select id="make" name="make">
  <option value="dolby">Dolby</option>
  <option value="sony">Sony</option>
  <option value="hitachi">Hitachi</option>
</select>
```

Selection lists/pulldown lists for user to select one option. Typical example: list of countries in address details.

List options in alphabetic order to make navigation easier.



## *Radio buttons*

Radio buttons also permit one choice from a number of options:

```
<label for="media">Media:</label>
<input type="radio" name="media"
  value="vinyl" checked>Vinyl<br>
<input type="radio" name="media"
  value="CD">CD<br>
<input type="radio" name="media"
  value="tape">Tape<br>
```

The *checked* option is initially shown as selected.

Which is better, selection list or radio buttons?

## *Checkboxes*

Checkboxes: multiple choices from options:

```
<label for="subscribe">Subscribe for news of:</label>
<input type="checkbox" name="subscribe"
  value="4050s">1940's/50's music<br>
<input type="checkbox" name="subscribe"
  value="6070s">1960's/70's music<br>
<input type="checkbox" name="subscribe"
  value="8090s">1980's/90's music<br>
```

# Vinyl Revival

We sell all styles of LPs + 45s from the 1950's onwards.

Make:

Media:  Vinyl

CD

Tape

Subscribe for news of:  1940's/50's music

1960's/70's music

1980's/90's music

Product:

Your name:

Address:

Example form with options

## *Uploading files*

Common situation is to upload client files to server:

```
<!DOCTYPE HTML>
<html><head>
<body>
<form method="POST" action="serverprogram"
  enctype="multipart/form-data">
<input type="file" name="thefile" style="width:350px">
<input type="submit" value="Upload">
</form></body></html>
```

Input type “file” is a file chooser box.

`enctype="multipart/form-data"` is necessary for forms which upload files.

### *Scripts and data verification*

- Usually some fields of a form are mandatory – must be filled in before it is sent
- Convention is to use \* in label of mandatory fields
- A validation check can be coded (in Javascript) that prevents submission unless these fields are filled.

## *Scripts and data verification*

*validation.html* contains example validation code. Eg., to test if a field is empty:

```
<head> <title>Registration</title>
<script type = "text/javascript">

    function checkName()
    { if (document.User.name.value.length > 0)
      { return true; }
      else
      { window.alert("Name cannot be empty!");
        return false;
      }
    }
}
</script>
</head>
```

```
<body>
<h1>Register for property updates</h1>

<form name = "User"
  action =
    "http://localhost:8080/servlet/RegisterUserServlet"
  method = "POST">

<p>
<strong>Your name*</strong>
<input name = "name" type = "text"></p>

<p>
<strong>Email address*</strong>
<input name = "email" type = "text"></p>
```

```
<p>
<strong>What type of property are you
looking for?</strong><br>
Detached
<input name = "type" type = "radio" value = "detached">
Semi-detached
<input name = "type" type = "radio"
  value = "semi" checked>
Terraced
<input name = "type" type = "radio"
  value = "terraced">
Flat
<input name = "type" type = "radio"
  value = "flat">
</p>
```



```
<p>  
<strong>How many bedrooms?</strong><br>  
<input name = "bedrooms" type = "text" size = "1"></p>
```

```
<p>  
<strong>Maximum price?</strong><br>  
<input name = "maxprice" type = "text"></p>
```

```
<p>  
<strong>Minimum price?</strong><br>  
<input name = "minprice" type = "text"></p>
```

```
<p>  
<strong>What area (postcode)?</strong><br>  
<input name = "area"  
  type = "text" size = "4">
```

```
</p>
```

```
<input type = "submit" onclick = "return checkName()"  
      value = "Register">
```

```
</form>
```

```
</body>
```

```
</html>
```

### *Form validation*

- *checkName()* tests if the contents of the *name* field of the *User* form is non-empty
- This is done by testing the length of string contents *document.User.name.value* of the field
- If the length  $> 0$  then *true* is returned, otherwise a warning message is shown + *false* returned
- The submit button calls *checkName()* when pressed – if *false* then submission is not made, otherwise it proceeds.

The image shows a web browser window with two tabs: "Sign out" and "Registration". The address bar displays "file:///C:/Users/Olga/Document". The main content area features a registration form titled "Register for property updates". The form includes the following fields and options:

- Your name\***: A text input field.
- Email address\***: A text input field.
- What type of property are you looking for?**: Radio buttons for "Detached", "Semi-detached", "Terraced" (selected), and "Flat".
- How many bedrooms?**: A text input field.
- Maximum price?**: A text input field.
- Minimum price?**: A text input field.
- What area (postcode)?**: A text input field.
- Register**: A button at the bottom of the form.

Form with validation checks

*More complex validation tests*

```
<head> <title>Registration</title>
```

```
<script type = "text/javascript">
```

```
function isNumber(str)
```

```
{ if (isNaN(parseFloat(str)))
```

```
  { return false; }
```

```
  return true;
```

```
}
```

```
function isInteger(str)
```

```
{ return isNumber(str) &&
```

```
  str.indexOf(".") == -1;
```

```
}
```

```
function isEmail(str)
```

```
{ var ind = str.indexOf("@");
```

```
    return (ind > 0 && ind < str.length - 1);
}

function checkName()
{ if (document.User.name.value.length > 0)
  { return true; }
  else
  { window.alert("Name cannot be empty!");
    return false;
  }
}

function checkEmail()
{ if (isEmail(document.User.email.value)) { }
  else
  { window.alert("Not an email address: " +
```

```
        document.User.email.value);
    }
}

function checkMinprice()
{ if (isInteger(document.User.minprice.value))
  { document.User.minprice.value =
    parseInt(document.User.minprice.value); }
  else
  { window.alert("Not an integer: " +
    document.User.minprice.value);
  }
}

function checkMaxprice()
{ if (isInteger(document.User.maxprice.value))
```

```
{ document.User.maxprice.value =
  parseInt(document.User.maxprice.value); }
else
{ window.alert("Not an integer: " +
  document.User.maxprice.value);
}
}

function checkBedrooms()
{ if (isInteger(document.User.bedrooms.value))
  { document.User.bedrooms.value =
    parseInt(document.User.bedrooms.value);
    var bedroomsx = parseInt(document.User.bedrooms.value);
    if (bedroomsx <= 0)
    { window.alert("Bedrooms must be > 0"); }
  }
}
```



```
else
{ window.alert("Not an integer: " +
                document.User.bedrooms.value);
}
}

function formSubmit()
{ if (checkName()) {}
  else
  { return false; }

  var maxp = parseInt(document.User.maxprice.value);
  var minp = parseInt(document.User.minprice.value);
  if (minp >= 0 && maxp >= minp) { }
  else
  { window.alert("Prices not valid!");
```

```
        return false;
    }
    return true;
}
</script>
</head>

<body>
<form name = "User"
    action =
        "http://localhost:8080/servlet/RegisterUserServlet"
    method = "POST">

<p>
<strong>Your name</strong>
<input name = "name" type = "text"></p>
```

```
<p>
<strong>Email address</strong>
<input name = "email" type = "text"></p>
```

```
<p>
<strong>What type of property are you
looking for?</strong><br>
Detached
<input name = "type" type = "radio" value = "detached">
Semi-detached
<input name = "type" type = "radio"
value = "semi" checked>
Terraced
<input name = "type" type = "radio"
value = "terraced">
Flat
```

```
<input name = "type" type = "radio"  
  value = "flat">  
</p>
```

```
<p>  
<strong>How many bedrooms?</strong><br>  
<input name = "bedrooms" type = "text" size = "1"></p>
```

```
<p>  
<strong>Maximum price?</strong><br>  
<input name = "maxprice" type = "text"></p>
```

```
<p>  
<strong>Minimum price?</strong><br>  
<input name = "minprice" type = "text"></p>
```

```
<p>
<strong>What area (postcode)?</strong><br>
<input name = "area"
  type = "text" size = "4">
</p>

<input type = "submit" onclick = "return formSubmit()"
  value = "Register">
</form>
</body>
</html>
```

### *Form validation*

- *str.indexOf(s)* returns -1 if *s* does not occur as a substring of *str*, otherwise it returns position of first occurrence of *s*.

Positions start at 0 on left end of string.

Eg: "abba".indexOf("ba") is 2.

- *isInteger* tests if *str* is a number and has no .
- *isEmail* tests that @ occurs in *str*, and not as first or last character.

### *Summary*

- Data input of one of a fixed number of choices: radio buttons or selection lists.
- Data input of one or more of a fixed number of choices: checkboxes
- Scripts can be used to check data before it is sent.

*Exercise: writing forms with controls*

- Modify *validation.html* to have a selection field instead of a text field for *area*. The selection should list the SW postcodes (SW1 to SW21).
- Modify *validation.html* to have an additional textarea field to specify particular required property features.
- Add a radio button field *position* with three options *Cash buyer*, *Selling property*, *Other*. The label is “Your position”.



## Session 8: Writing complete websites

- Using meta tags to promote pages and sites
- Using style sheets to define common style for a website
- Simple website structures
- Exercise: writing style sheets

## *Metatags*

- HTML elements listed in `< head >` section of web page – not shown in browser
- Can describe document: author, keywords, organisation + other properties of page
- Used by search engines when responding to searches, eg., via Google or Yahoo
- *description* attribute often used to give short description of page.

## *Style sheets*

- In addition to \*.html files, can write \*.css files to apply a consistent style (colours, fonts, etc) to all pages of a website
- \*.css files are called *style sheets*
- Style sheets consist of set of style rules, defining how specific HTML elements should be styled.
- Put .css files in same directory as the .html files that use them.

### *Style sheet example*

This sheet puts border around all paragraphs, and makes their text red (test.css):

```
p {border-style: solid; border-width: 2px; color: red}
```

In the `< head >` part of a web page, include line:

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

The definitions of test.css then apply to the web page.

Example file using .css and meta tags:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="author" content="K. Lano">
<meta name="description" content="Vinyl Revival Website">
<meta name="keywords" content="Vinyl Revival, Records, Music, LPs, 45s, 78s">
<link rel="stylesheet" type="text/css" href="test.css">
</head>
<body>
<p>A test paragraph to show the effect of the style sheet.
All paragraphs will appear in this format.</p>

<p>We sell all styles of LPs + 45s from
the 1950's onwards. </p>

<p>For our current list of products, see
```

```
<a href="http://www.vinylrevival.co.uk/products.html">Products</a></p>
```

```
Links in a list:<br>
```

```
<ol>
```

```
<li> <a href="fifties.html">1950's</a>
```

```
<li> <a href="sixties.html">1960's</a>
```

```
<li> <a href="seventies.html">1970's</a>
```

```
</ol>
```

```
<hr>
```

```
Email link:<br>
```

```
<a href="mailto:enquiries@vinylrevival.co.uk">Contact Us</a>
```

```
</body>
```

```
</html>
```

The screenshot shows a web browser window with the address bar displaying 'file:///C:/Users/Olga/Documents/cil/test.htm'. The browser's toolbar includes a search box and several icons. Below the toolbar, there are several navigation links: 'Most Visited', 'Getting Started', 'Amazon', 'eBay', 'Suggested Sites', 'Web Slice Gallery', and 'WildTangent Games'. The main content area of the page contains three paragraphs, each enclosed in a red border. The first paragraph reads: 'A test paragraph to show the effect of the style sheet. All paragraphs will appear in this format.' The second paragraph reads: 'We sell all styles of LPs + 45s from the 1950's onwards.' The third paragraph reads: 'For our current list of products, see [Products](#)'. Below these paragraphs, there is a section titled 'Links in a list:' followed by a numbered list: '1. [1950's](#)', '2. [1960's](#)', and '3. [1970's](#)'. Underneath the list is a section titled 'Table with border and fixed widths:' followed by a table with three columns: 'Record', 'Condition', and 'Price'. The table contains three rows of data: 'Beatles 'Help'' (Good, £ 3.99), 'Sinatra 'My Way'' (Mint, £ 1.99), and 'Moby 'Play'' (Good, £ 2.99). Below the table, there is a link: 'More are available'. At the bottom of the page, there is a section titled 'Email link:' followed by a link: '[Contact Us](#)'.

file:///C:/Users/Olga/Documents/cil/test.htm

Most Visited Getting Started Amazon eBay Suggested Sites Web Slice Gallery WildTangent Games

A test paragraph to show the effect of the style sheet. All paragraphs will appear in this format.

We sell all styles of LPs + 45s from the 1950's onwards.

For our current list of products, see [Products](#)

Links in a list:

1. [1950's](#)
2. [1960's](#)
3. [1970's](#)

Table with border and fixed widths:

Record	Condition	Price
Beatles 'Help'	Good	£ 3.99
Sinatra 'My Way'	Mint	£ 1.99
Moby 'Play'	Good	£ 2.99

More are available

Email link:  
[Contact Us](#)

Web page with style file

## *Style sheets*

More complex example:

```
p {border-style: solid; border-width: 2px; color: red}
```

```
table { border: 1px solid black; text-align: center }
```

```
tr:hover { background-color: #8585FF }
```

```
th {  
    background-color: #4CAF50;  
    color: white;  
}
```

This center-aligns all table text, changes table row colour when mouse hovers over it, and sets white on green colours for table header.



## *Style sheets*

For lists can write style specification:

```
ul { list-style-type: square; }
```

Makes all unordered lists use squares for list items.

But possible to selectively apply styles to elements: by naming a *class* of elements, rules can be specified just for that class.

Eg:

```
<ul class="circlelist">  
<li>Item 1  
<li>Item 2  
</ul>
```

CSS rule specific to these lists is:

```
ul.circlelist { list-style-type: circle; }
```

Other *ul* elements will have the general *ul* style.

## *Style sheets*

Additionally, can label specific elements by an *id*:

```
<ul id="menubar">
<li><a class="menuitem"
  href="index.html" shape="rect">Home</a></li>
<li>...</li>
</ul>
```

CSS rule specific to *menubar* element:

```
#menubar
{ list-style-type: none; }
```

```
#menubar li
{ float: left; }
```

Specific style for *menubar* and its list and *li* elements.

## *Style sheets*

- General format of CSS style rule is

Selector

```
{ property1 : value1;
```

```
  ...
```

```
  propertyN : valueN;
```

```
}
```

- *Selector* identifies what element(s) the style settings should apply to
- Any number of properties can be set – for colours, fonts, sizes, borders, spacing, etc.

## *Style sheets*

*Selector* can be:

- *tag* – applies to all elements with an HTML tag such as *h1* or *p*
- *#idvalue* – applies to one element with unique *idvalue*  
(*id* = "*idvalue*")
- *.classname* – applies to all elements with *class* = "*classname*".

Can combine these: *p.classname* means all paragraphs with *class*="classname"

*tag1 tag2* means all *tag2* elements nested within a *tag1* element.

*tag1, tag2* means all *tag1* and all *tag2* elements.

## *Style sheets*

Properties include

- *color, background, background-color, background-image, background-repeat, background-position*
- *height, width*
- *border, border-style, border-width, border-color, border-radius*
- *margin, margin-top, margin-right, margin-bottom, margin-left*
- *padding, padding-top, padding-right, padding-bottom, padding-left.*

## *Style sheets*

Element borders can be specified for each border separately, or for all borders.

```
p { border-style: dotted dashed solid double;  
    border-width: 5px 10px 5px 10px;  
    border-color: red blue green orange; }
```

Defines *top* border as dotted, red and 5px, *right* as dashed, blue and 10px, *bottom* as solid, green and 5px, and *left* as double, orange and 10px.

## *Style sheets*

If all 4 borders have same property value, write:

```
p { border-style: double;  
    border-width: 10px;  
    border-color: blue; }
```

Can combine all properties in single *border* property:

```
p { border: 10px double blue; }
```

## *Style sheets*

Element margins define space around element, *outside* its borders.

```
p { margin: 10px 50px 20px 50px; }
```

Defines *top* margin as 10px, *right* as 50px, *bottom* as 20px, and *left* as 50px.

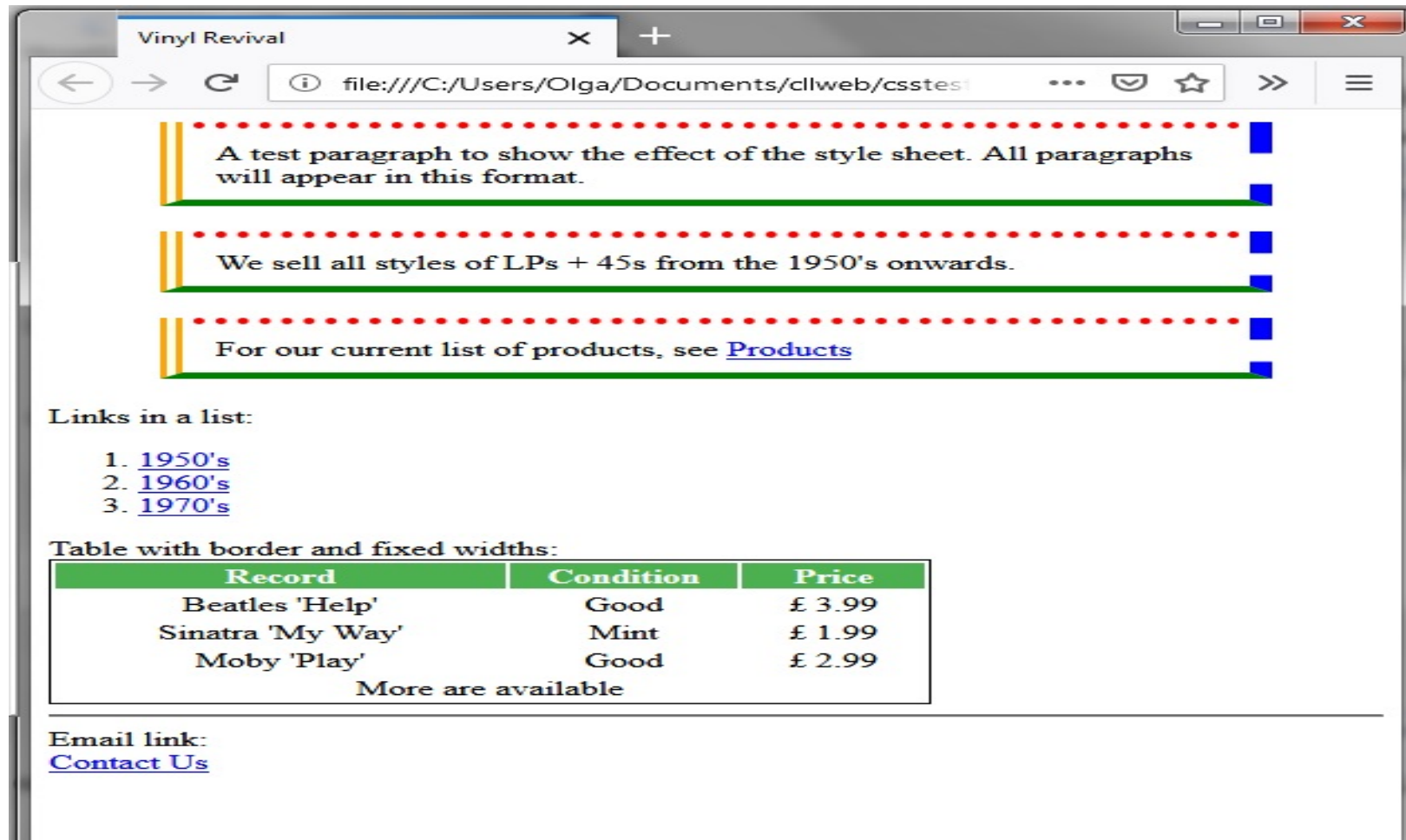
If all 4 sides have same margin property value, write:

```
p { margin: 20px; }
```

In contrast, *padding* inserts space *within* borders, around any content:

```
p { padding: 10px 15px 10px 5px; }
```





Style sheet test of borders, margins, padding

## *Style sheets*

- Possible to have several style sheets, eg., one defining colours for elements, another defining size and shape of elements, etc
- *layout.css* defines some standard layouts
- *colourscheme.css* defines colour choices
- *style.css* defines text styles and formatting.

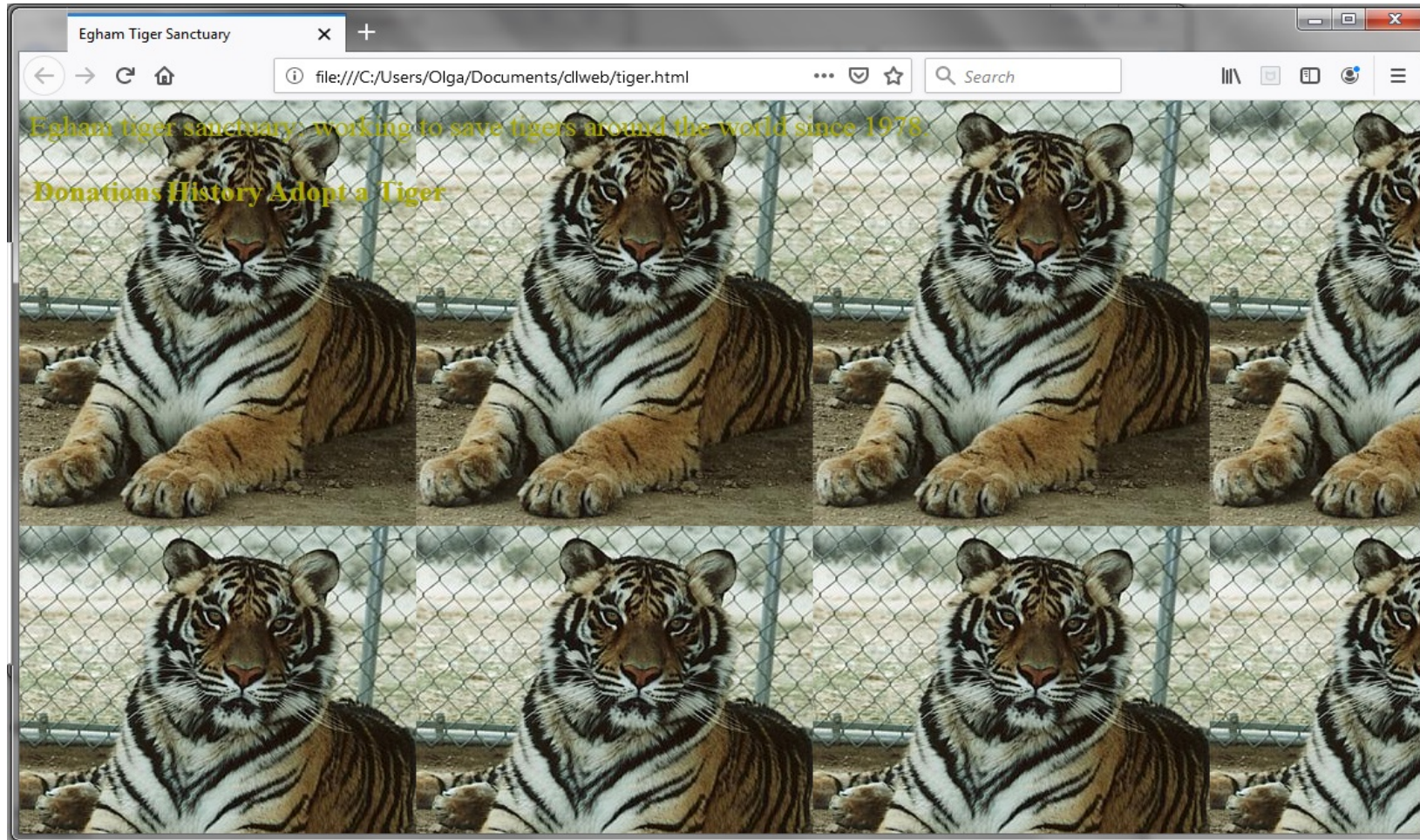
## *Style sheets*

Also possible to define a style local to a single page, in a *style* element:

```
<html>
<head>
<title>Egham Tiger Sanctuary</title>
<style type = "text/css">
body { background-image: url(tiger.jpg); }
</style>
</head>
<body>
<p style = "font-size: 18pt; color: #AAAA00">
Egham tiger sanctuary: working to save tigers
around the world since 1978. </p>
```

```
<p>  
<table style = "font-size: 18pt; color: #AAAA00">  
<th>Donations</th> <th>History</th> <th>Adopt a Tiger</th>  
</table></p>  
</body>  
</html>
```

By default, background images are repeated over page 'tiled'.

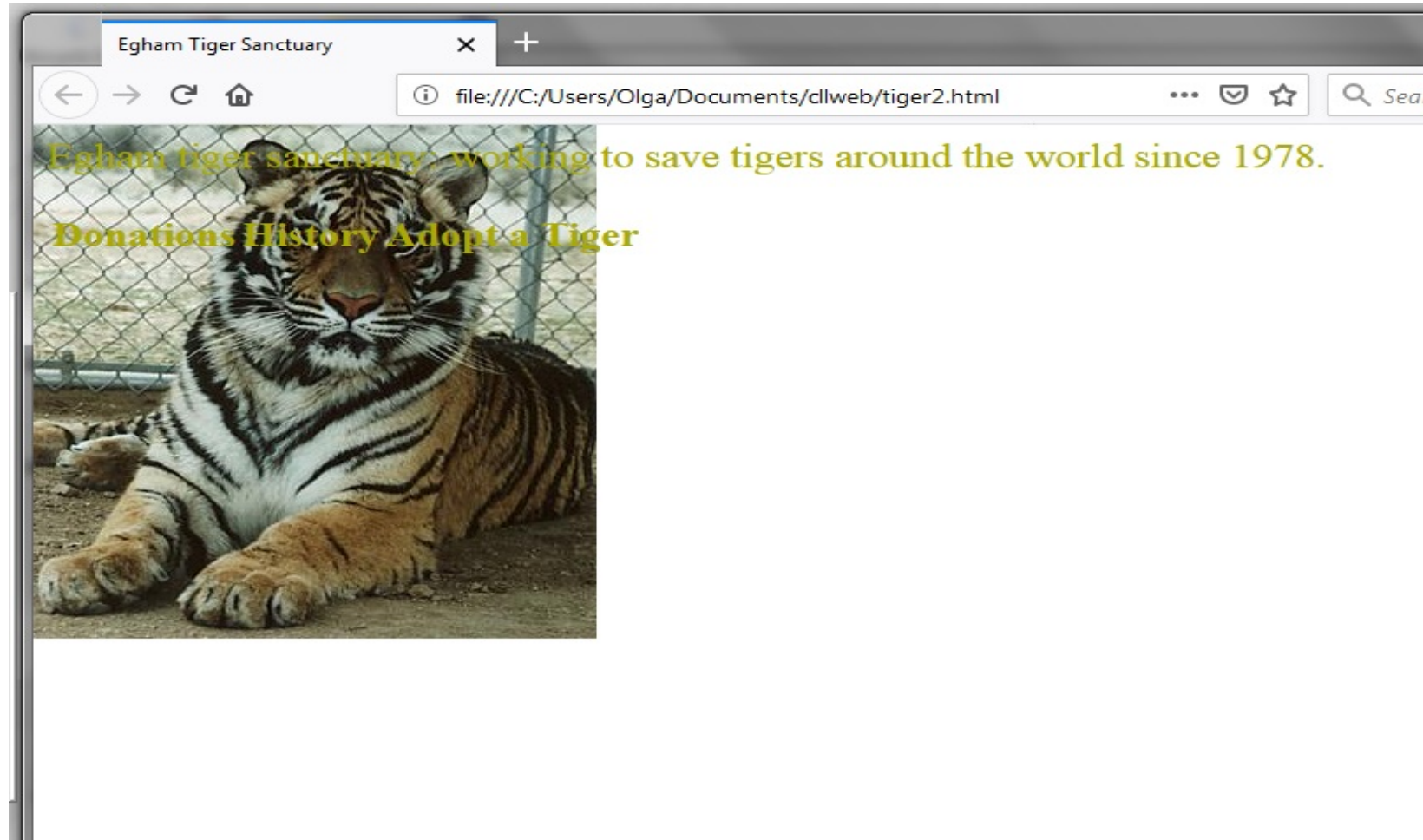


Page with background image style

## *Style sheets*

To avoid tiling, use *background-repeat* property set to *no-repeat*:

```
<html><head>
<title>Egham Tiger Sanctuary</title>
<style type = "text/css">
body
{ background-image: url(tiger.jpg);
  background-repeat: no-repeat;
}
</style>
</head>
```

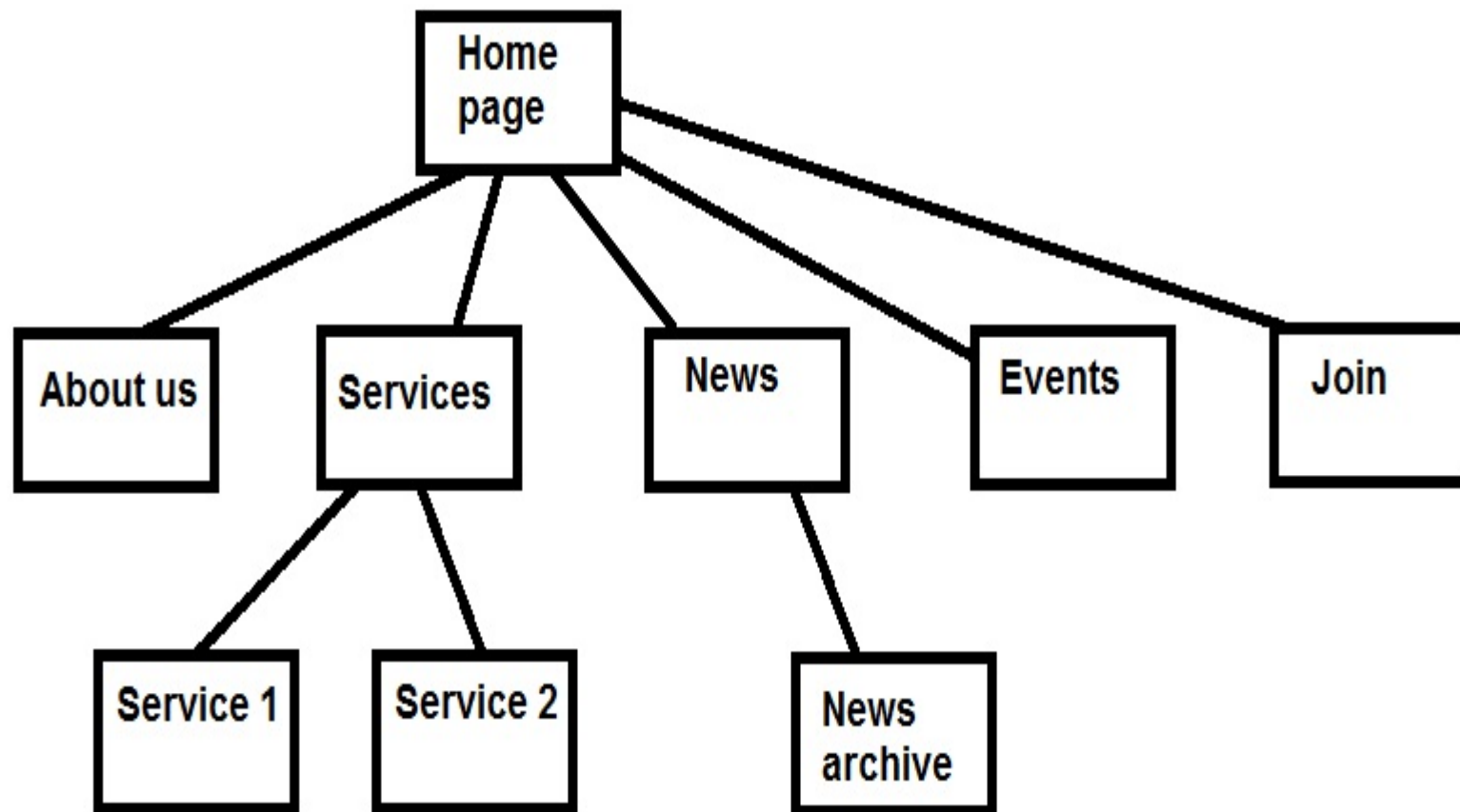


Improved page with non-repeated background

### *Simple website structures*

- A typical structure is a ‘tree’ of pages, starting at root of tree (*index.html*)
- One click/link from root are main branches, eg., pages *about.html*, *products.html*, *services.html*, *contact.html*
- Each branch in turn may subdivide with links to more specialised pages.





Typical website structure

## *Summary*

- Metatags enable promotion of your site by defining non-visible properties of pages for search engines to read
- Style sheets define common settings for page layouts and formats across one website
- Standard website structure is a hierarchical tree of increasingly specialised pages as you descend the tree.

*Exercise: writing style sheets*

Design a style sheet where all paragraphs have a silver background, and all tables have a light blue background.

All headers (h1, h2, etc) should be white text on dark blue background.

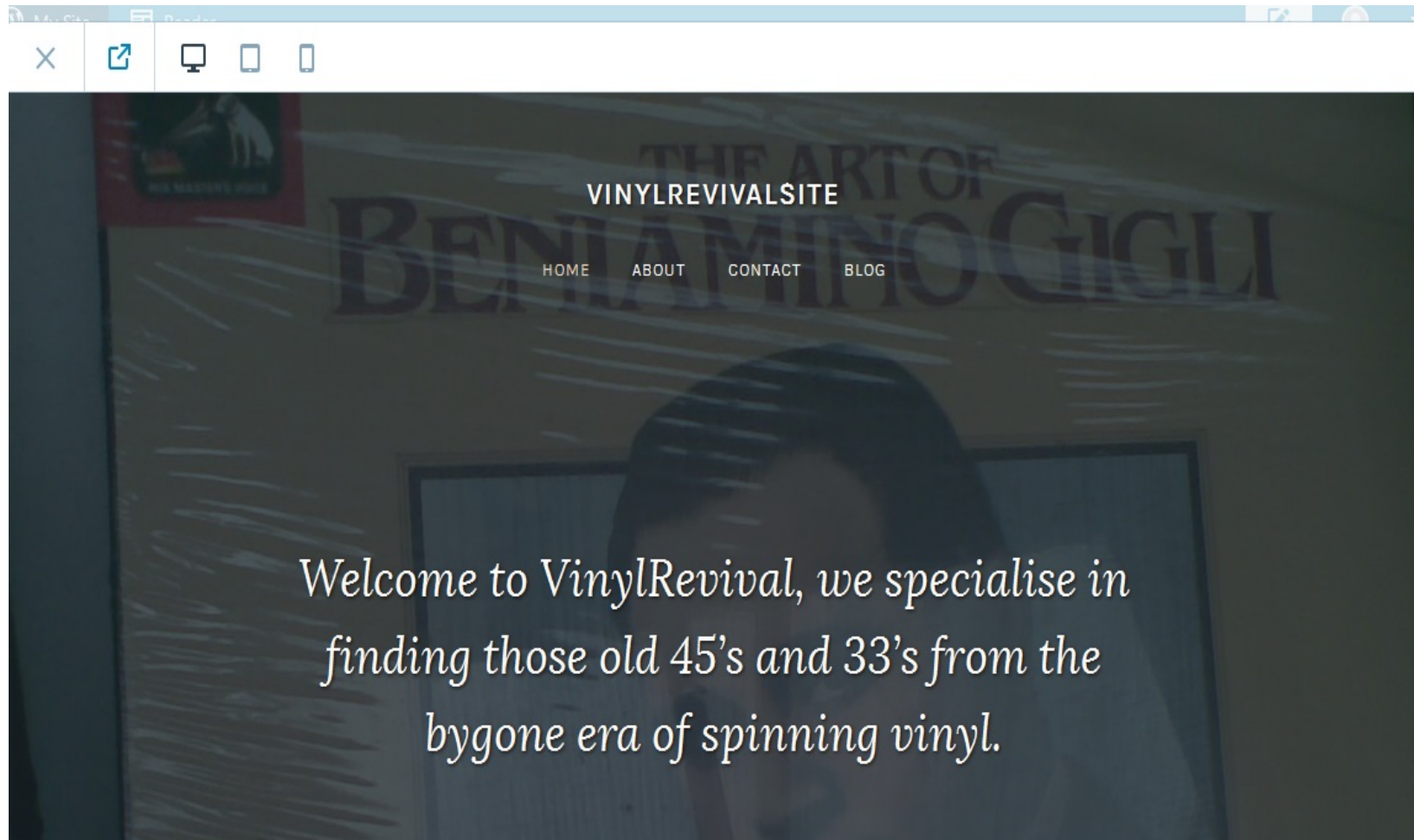
Test your style sheet with an example web page.

## Session 9: Setting up websites on Internet hosts

- Templated websites
- Customised websites
- Different website styles
- Exercise: designing a website

## *Templated websites*

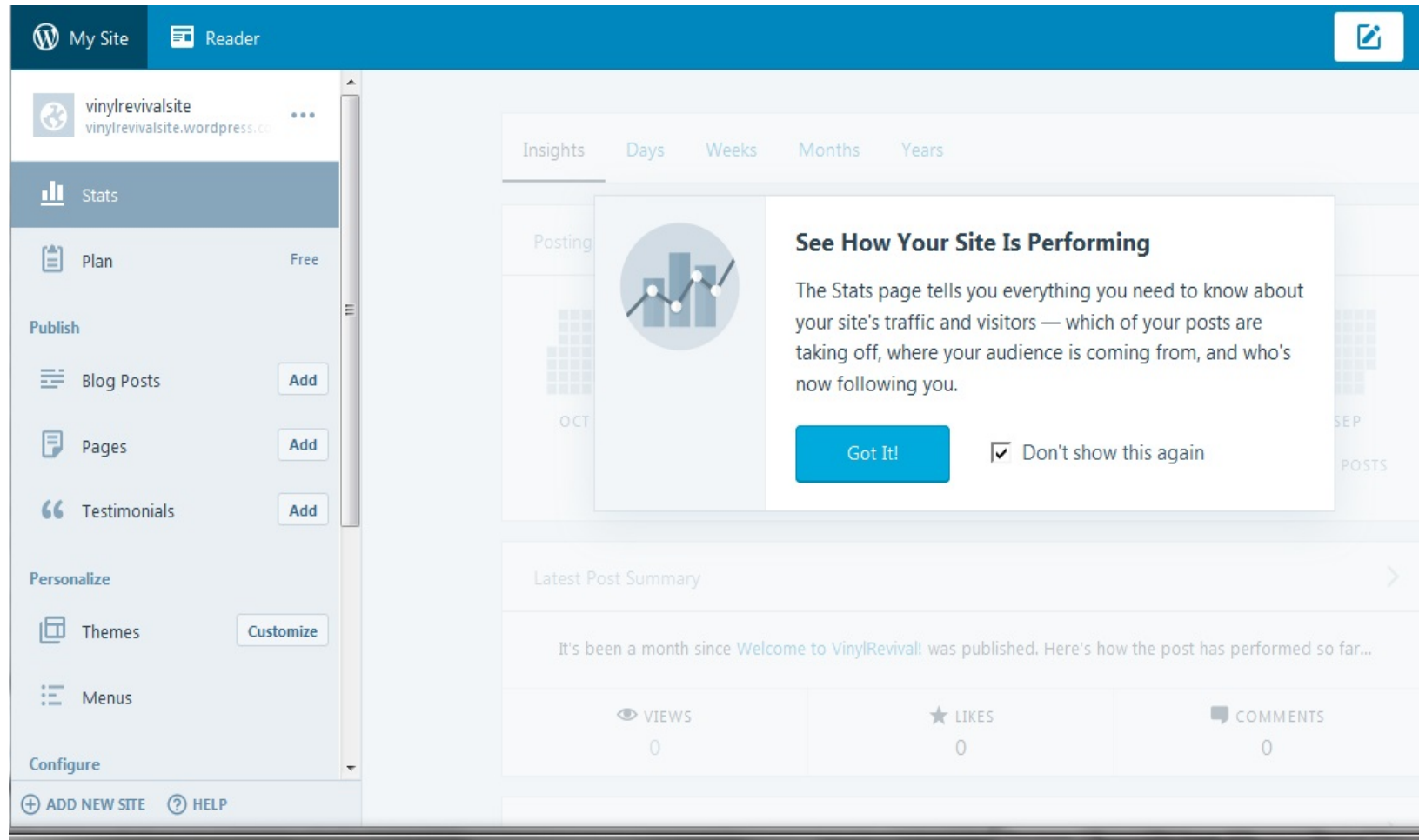
- Typically, either a web host provider has a pre-defined structure for your website, or allows you to freely define your own structure + pages.
- Templated sites example: [wordpress.com](https://wordpress.com)
- Low cost website hosting using templates for site and pages – you can edit details and content of pages
- Eg., [vinylrevivalsite.wordpress.com](https://vinylrevivalsite.wordpress.com)
- Blog style – may not be suitable for commercial sites.



Wordpress site



Wordpress editing



Wordpress site management. Pages add option →



The screenshot shows the WordPress editor interface. At the top, there's a blue header with 'My Site' and 'Reader' tabs. Below that, a left sidebar contains a 'BACK' button, the site name 'vinylrevivalsite', a 'DRAFT SAVED' notification, and buttons for 'Preview' and 'Publish'. The main content area shows the title 'Easy Listening' and the URL 'https://vinylrevivalsite.wordpress.com/easy-listening'. A rich text editor toolbar is visible above the text. The text content lists several LPs:

- Deanna Durbin "Something in the wind" Brunswick 03819-A
- Deanna Durbin "It's only love" Brunswick 03819-B
- Georges Guetary "Where Flamingos Fly" (78) Columbia DB2362
- Lee Lawrence "More than a millionaire" (78) Columbia DB3645

At the bottom right of the editor, it says '38 WORDS'.

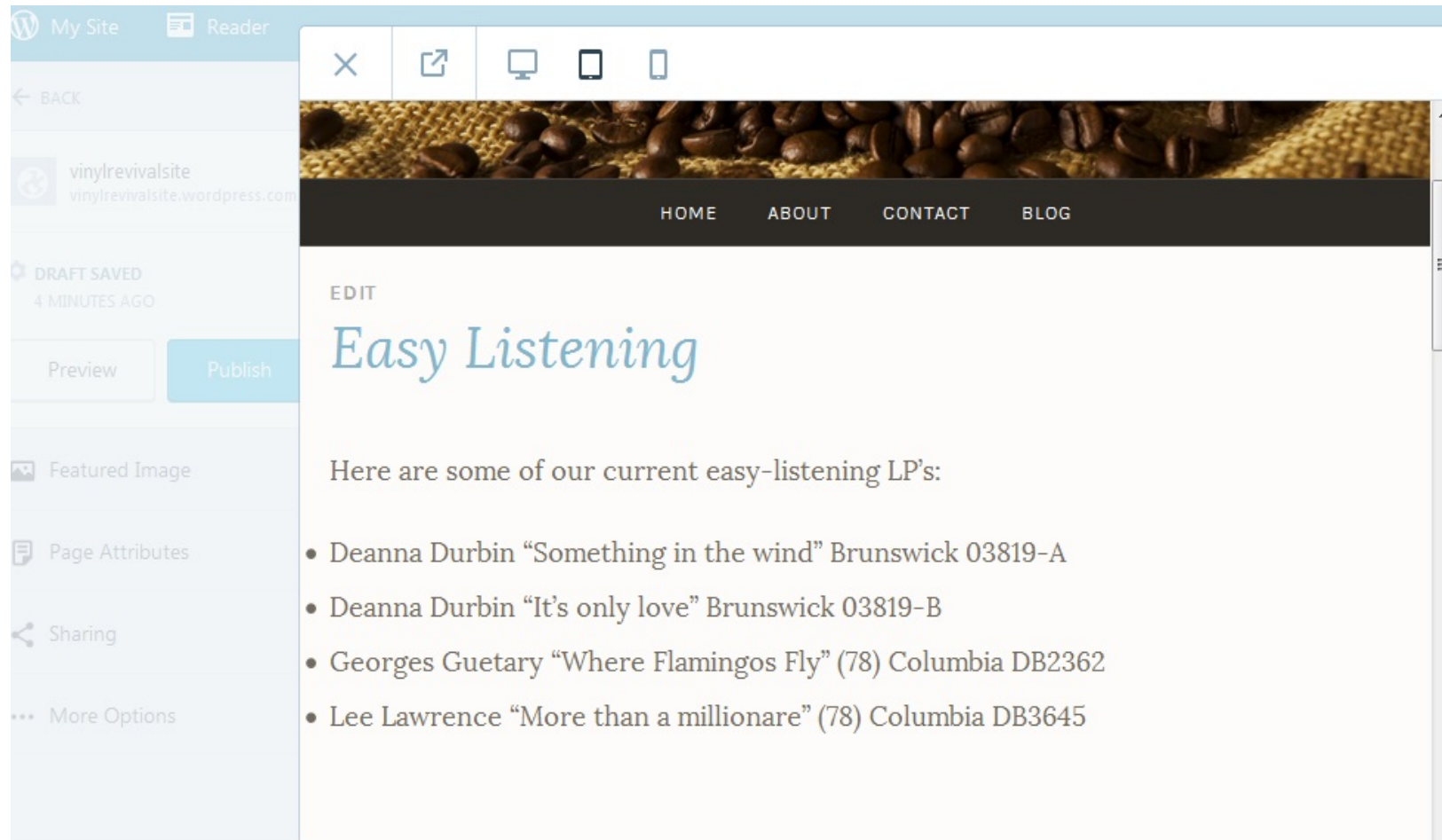
Adding a web page (Word editor)

The screenshot shows the WordPress editor interface. At the top, there is a blue header with 'My Site' and 'Reader' on the left, and a user profile icon on the right. Below the header, the left sidebar contains a 'BACK' button, the site logo 'vinylrevivalsite', a 'DRAFT SAVED' notification, and buttons for 'Preview' and 'Publish'. The main content area is titled 'Easy Listening' and shows the URL 'https://vinylrevivalsite.wordpress.com/easy-listening'. Below the URL, there are 'Visual' and 'HTML' tabs, with 'HTML' selected. The content in the HTML editor is as follows:

```
Here are some of our current easy-listening LP's:  
  
<ul>  
<li> Deanna Durbin "Something in the wind" Brunswick 03819-A  
  
<li> Deanna Durbin "It's only love" Brunswick 03819-B  
  
<li> Georges Guetary "Where Flamingos Fly" (78) Columbia DB2362  
  
<li> Lee Lawrence "More than a millionaire" (78) Columbia DB3645  
</ul>  
  
&nbsp;
```

At the bottom right of the editor, there is a '38 W' indicator.

Adding a web page (HTML editor)



Preview of new web page

### *Simple websites with WordPress*

- Images + other attributes of pages can be edited
- Pages can be linked from other pages `<a href="easy-listening.html">Easy listening</a>`
- Widgets – such as Twitter feeds – can be added. (Customise; add widget; Twitter timeline).

The screenshot shows a browser window with the URL `https://vinylrevivalsite.wordpress.com/wp-admin/customize.php?url=https%3A%2F%2Fen.support.wo`. The browser's address bar includes a search field and a list of "Most Visited" sites: Getting Started, Amazon, eBay, Suggested Sites, Web Slice Gallery, and WildTangent Games. The main content area displays a WordPress customization interface. On the left, a "Twitter Timeline: Follow me ..." widget configuration panel is open, showing a "Saved" button and various settings: Title (Follow me on Twitter), Maximum Width (300), Height (600), # of Tweets Shown (3), Widget Type (Profile), Twitter Username (frndsofcarnegie), and a "No Header" layout option. On the right, a preview of the website is shown with the header "VINYLREVIVALSITE" and navigation links for HOME, ABOUT, CONTACT, and BLOG. The main content area features a large image of a vinyl record with the text "Welcome to VinylRevival, we spend our time finding those old 45's and 33's from the bygone era of spinning vinyl." and a footer that reads "Tweets by Friends of Carnegie Library".

Defining a Twitter widget

### *Customised websites*

- Web hosting company provides you with a directory where you can upload/edit HTML files, images, etc
- You need to design the structure of the site (which pages link to which, what are the menus, etc) and all page layout + content
- Suitable for more expert web developers, gives complete freedom for designs + style.

Should have consistent style of colour, layout, text, images, on every page. Stylesheets can ensure this.

## *Customised websites*

Some standard pages should be present:

- *index.html* – the home page, introduces your organisation/company, has menu/links to rest of site. Twitter feed can go here.

Business sites can have product search prominently on this page (eg., amazon.com).

Member organisations could have *Join* button, *Donate* button, etc.

- *news.html*, *events.html* – for community groups/organisations
- *contacts.html* – details of how to contact the organisation; addresses, maps, etc.

Example: [friendsofcarnegielibrary.org.uk](http://friendsofcarnegielibrary.org.uk)

The screenshot shows a web browser window with the address bar displaying "friendsofcarnegilibrary.org.uk". The page features a navigation menu with links for Home, Join, Aims, Events, Services, Forums, Newsletters, History, and Contact. The main content area includes a large heading "Friends of Carnegie Library" and a sub-heading "Promoting Use of and Access to the Carnegie Library in Herne Hill. We support the DefendTheTen campaign which is working to save all 10 Lambeth libraries: http://defendthe10-lambeth.org.uk". A sidebar on the right contains a "Login" button and a section titled "The Carnegie on Twitter" featuring a tweet from @CarnegieLib.

Friends of Carnegie Library

Promoting Use of and Access to the Carnegie Library in Herne Hill. We support the DefendTheTen campaign which is working to save all 10 Lambeth libraries: <http://defendthe10-lambeth.org.uk>

Home Join Aims Events Services Forums Newsletters History Contact

**PUBLIC MEETINGS ON THE FUTURE OF CARNEGIE LIBRARY**

September 18, 2016 Campaigns, Events, News Nicholas Edwards

With our partners in the Carnegie Library Association CIO, the Friends are arranging two public meetings to discuss plans for the future. The announcement leaflet is [here](#)

Are you a Friend? Login

*The Carnegie on Twitter*

Carnegie Library @CarnegieLib  
This Census-taker by China Mieville  
[ethlibraryreadinggroups.wordpress.com/2016/09/09/thi...](http://ethlibraryreadinggroups.wordpress.com/2016/09/09/thi...)

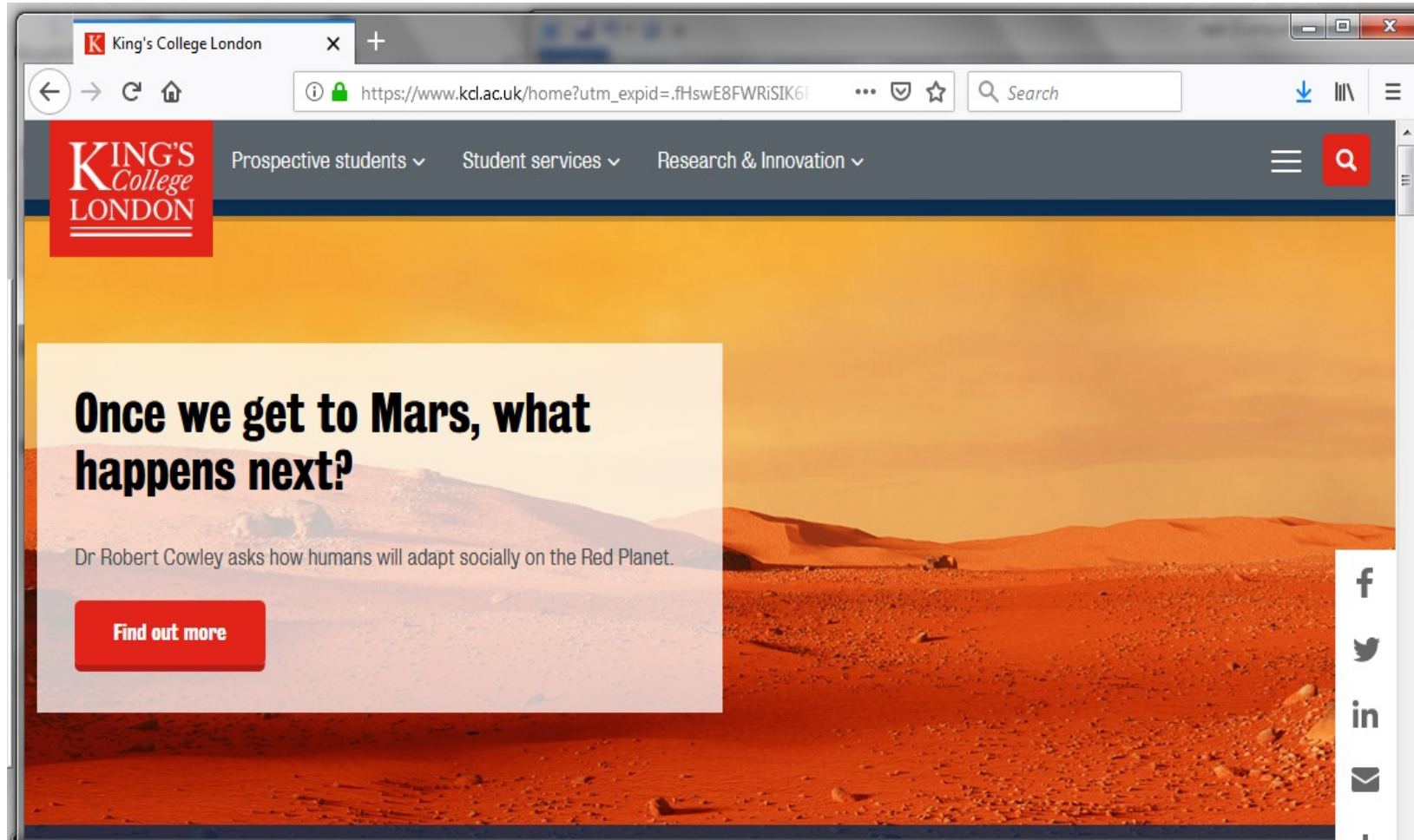
Home page of Friends of Carnegie Library



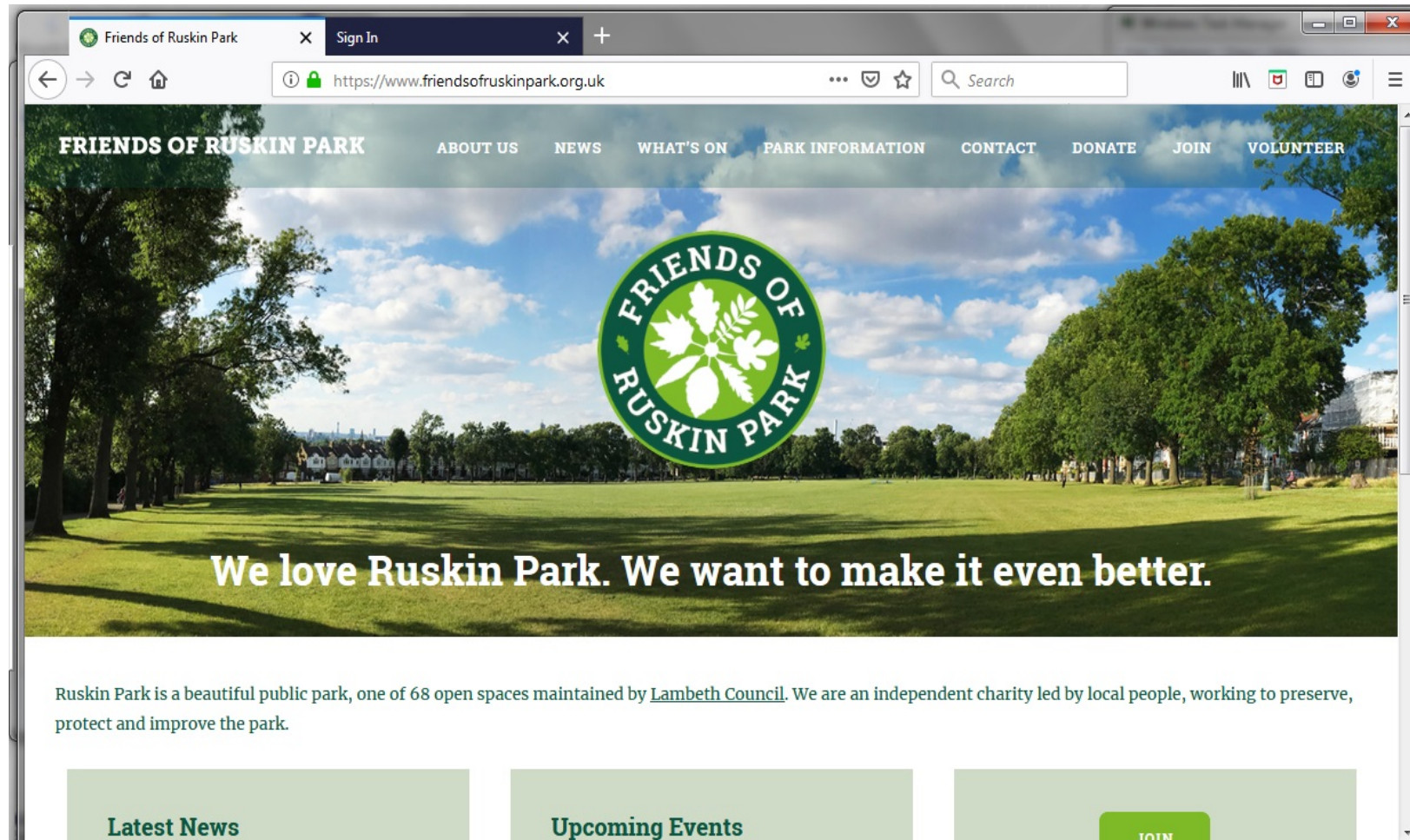
### *Different website styles*

Distinction between text-focussed and image-focussed presentation.

- Blog – linear sequence of articles. Comment facility. Eg., [ukpollingreport.co.uk](http://ukpollingreport.co.uk)
- Newspaper – structured into subpages with articles of different priority. Eg., [bbc.co.uk](http://bbc.co.uk), [wikipedia.org.uk](http://wikipedia.org.uk)
- Image focussed – large banner image/images and minimal text. Eg., [kcl.ac.uk](http://kcl.ac.uk), [ucl.ac.uk](http://ucl.ac.uk), [amazon.co.uk](http://amazon.co.uk)



kcl.ac.uk homepage



Friends of Ruskin park homepage

## *Different website styles*

Style depends on who are the users + how they want to use your site.

- Blog – for users who want to get news/information in specific area + join discussions
- Newspaper – for readers of news/information across range of areas
- Image focussed – to attract casual readers to explore more about the organisation/services/products.

However, image-focussed presentation can be obstruction to users who want specific information/service.

In this case *essential* to have effective search or other access to specific pages/services. Eg., amazon.co.uk

*Exercise: designing a website*

- Design the structure of a website for a local park – the website should inform readers of the park facilities, location, access and history
- Decide what the style of the site should be + what are the main pages
- Decide how pages should be linked/accessed
- Draw sketches of the pages and of the website structure.

## *Summary*

- The web host (wordpress, in our example) may provide page and website templates, which you can edit, customise + load with your own content
- Other hosts may only provide a directory into which you place your own HTML files. Navigation starts from `index.html` page.
- Think carefully about the style of your site – who is it aimed at + how would they use it?

## Session 10: How will people use your site?

- Concepts of website usability
- How do people actually use websites?
- Design and navigation choices
- Exercise: usability review.

### *Concepts of website usability*

- Usability – degree to which it is possible to access required information/services from the site without excessive effort
- General principles of usability for websites are:
  - Minimise effort needed to understand and use pages
  - Minimise effort needed to navigate the site
  - Obey conventions of use
  - Remove un-necessary and unclear text/content
  - Take into account accessibility needs of users



## *Concepts of website usability*

Make website as easy as possible for a user to use, for the range of uses that you intend.

- Example: purchasing specific product from a commercial site usually involves (i) search; (ii) selection from search results; (iii) registration; (iv) payment.
- The fewer steps + the simpler each step is, the more usable the site.
- (i) Search must be accurate + complete; (ii) allow ordering/filtering of results; (iii) skip unless essential, or do with (iv).

An alternative to search is navigation via links to specific kinds of product – suitable if relatively few categories + products.

### *Concepts of website usability*

- Example: Community site – user wants to find out when/where a service is available
- Involves either a search or navigation to specific page.
- (i) If by search, then this must be accurate + complete; (ii) if by navigation, appropriate link labels must be used + should be easy to find on home page/subpages; (iii) back button should work in usual way to allow alternative navigations; same search/navigation capabilities should be present on all pages.

### *Concepts of website usability*

- From experience of using websites, users learn various conventions + expectations
- Whenever possible, your site should follow such conventions
- Eg.: links are underlined and in different colour to normal text
- Don't confuse users by (a) underlining non-links; (b) not underlining links – unless they are obviously clickable buttons/menu items
- Search should have a text field + button labelled *Search* (or magnifying glass icon).

The screenshot shows a web browser window with the address bar displaying "friendsofcarnegielibrary.org.uk". The page features a navigation menu with links for Home, Join, Aims, Events, Services, Forums, Newsletters, History, and Contact. The main content area includes a large heading "Friends of Carnegie Library" and a sub-heading "Promoting Use of and Access to the Carnegie Library in Herne Hill. We support the DefendTheTen campaign which is working to save all 10 Lambeth libraries: http://defendthe10-lambeth.org.uk". A sidebar on the right contains a "Login" button and a section titled "The Carnegie on Twitter" featuring a tweet from @CarnegieLib.

Friends of Carnegie Library

Promoting Use of and Access to the Carnegie Library in Herne Hill. We support the DefendTheTen campaign which is working to save all 10 Lambeth libraries: <http://defendthe10-lambeth.org.uk>

Home Join Aims Events Services Forums Newsletters History Contact

**PUBLIC MEETINGS ON THE FUTURE OF CARNEGIE LIBRARY**

September 18, 2016 Campaigns, Events, News Nicholas Edwards

With our partners in the Carnegie Library Association CIO, the Friends are arranging two public meetings to discuss plans for the future. The announcement leaflet is [here](#)

Are you a Friend? Login

*The Carnegie on Twitter*

Carnegie Library @CarnegieLib  
This Census-taker by China Mieville  
[ethlibraryreadinggroups.wordpress.com/2016/09/09/thi...](http://ethlibraryreadinggroups.wordpress.com/2016/09/09/thi...)

Home page of Friends of Carnegie Library

## *Concepts of website usability*

### Other conventions:

- Larger font headings are the most important (as in a newspaper)
- Successful purchase should be followed by confirmation page (or clear error message if something went wrong)
- Buttons should look like buttons + have relevant label
- *Home* link/button takes user to *index.html*, the default landing page of site
- *Back* link/button takes user one step back in the link navigation they have followed.
- A menu option/link entitled *Topic* should actually take user to a page named *Topic*, about the *Topic*.

*How do people actually use websites?*

- Generally 2 categories of user: (i) casual – came to the site by accident/just looking out of casual interest; (ii) focused – users with definite goal.
- When considering usability, take account of both kinds of user, but (ii) may be priority for a commercial site/site offering specific service.
- Run through scenarios of use of your site, taking role of user – or get independent tester to do this.
- Focussed users will only read minimal amount of text + scan content to find what they want. Ignore rest of content.
- When reviewing site wrt a user goal, ask “What % of this page is actually useful to me to achieve this goal?”

*How do people actually use websites?*

- Since many users *scan* pages instead of *reading* them, design pages that can be scanned easily!
- Reduce user effort by making page structure and options obvious
- Make clear what the site is about
- Make clear what are the most important elements/links/options
- Make clear what is clickable
- Navigation by search should be in prominent place (eg., near top + center or right of page)
- Navigation menu/list of links should also be prominent (eg., separate section on left of page, or menubar across page under the header).

## *Design and navigation choices*

- Some design aspects are in conflict – rotating banner images may be attractive to casual users, but frustrating to focussed users. They take up much space + are distracting.
- There is a temptation to put lots of content on pages – especially the home page – but this creates visual clutter and confusion.
- Carefully separate content into different pages with specific scope.
- Avoid very wide flat website structures (eg., 20+ links from home page to specific pages) or very narrow deep structures (requiring a long sequence of clicks to get to specific topic).
- Avoid a maze of many interconnected pages in which users can get lost.



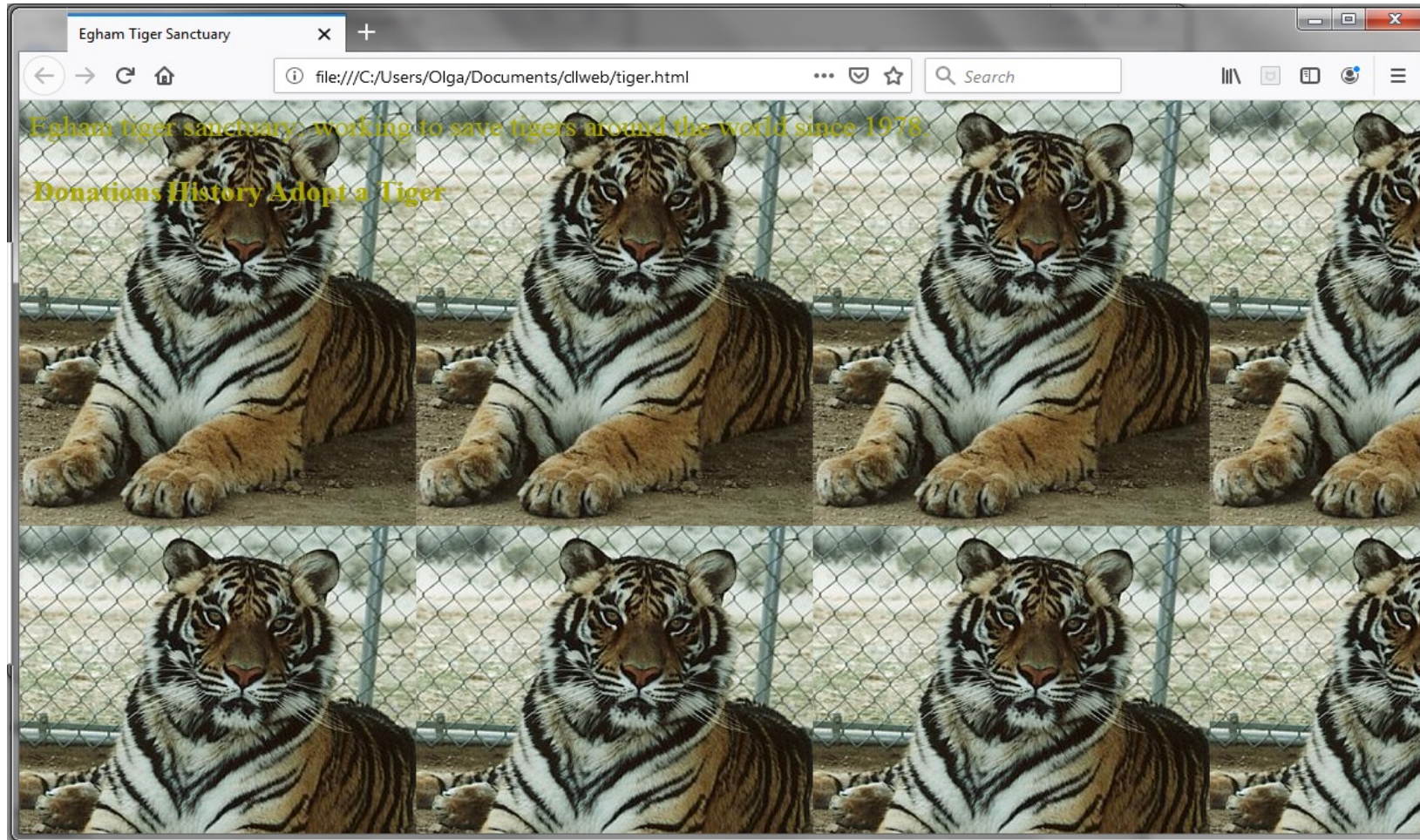
## *Design and navigation choices*

- Use a consistent set of colours, and not too many different colours. Eg., can use various shades of green with same red/blue element:  $xyyxx$  where  $y > x$ .
- Don't pre-fill forms: can be annoying to users.
- Don't duplicate the same navigation/actions in different places on one page – eg., both a button and a link which do same thing.
- Avoid large numbers of menu options/links.
- Provide clear feedback to any actions – eg., mousing over a clickable item should result in visible change in it.
- Use clear wording: “Pay” or “Make Payment” clearer than “Complete Order”.

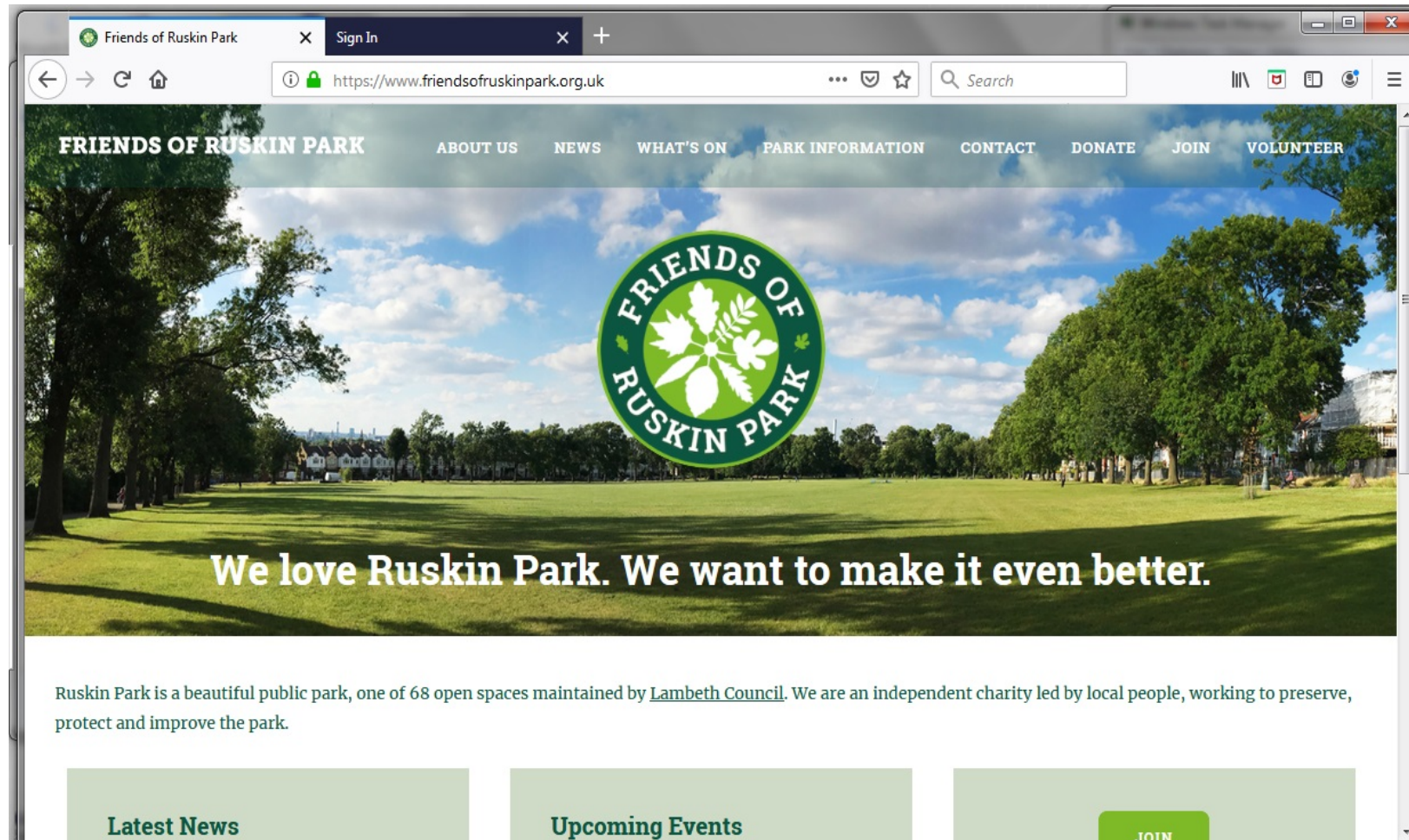
## *Design and navigation choices*

Background images can make any text hard to read.

```
<html>
<head>
<title>Egham Tiger Sanctuary</title>
<style type = "text/css">
body { background-image: url(tiger.jpg); }
</style>
</head>
<body>
<p style = "font-size: 18pt; color: #AAAA00">
Egham tiger sanctuary: working to save tigers
around the world since 1978. </p>
</body>
</html>
```



Page with background image style



Friends of Ruskin park homepage

## *Summary*

- Websites should be usable – otherwise they will not be used
- Take account of how people will use the site
- Take advantage of conventions + simplify + clarify content + presentation
- Give clear instructions and feedback for any actions.

*Exercise: usability review*

- Look at the Friends of Ruskin Park site and review it for usability
- Try to find out
  1. when the park opens and closes
  2. how to book a tennis court
  3. about the history of the park
  4. who are the trustee members of the Friends.
- Identify good features of the site
- Identify aspects that you think could be improved.

## Session 11: Helping users understand your site

- Text and explanations
- Presentation choices
- Frames and divisions
- Exercise: design homepage of community site

### *Text and explanations*

- Generally try to minimise text on home page, to reduce clutter. But it has to communicate the identity + purpose of site to users.
- Some text is essential: site id/name, tagline, welcome blurb
- Site id is the company name/logo – normally in top left corner of *each* page
- Tagline – a brief slogan to distinguish the site and make clear its purpose. Can be placed at top of home page, to right of logo
- Welcome blurb – short paragraph explaining what the site is and how to use it. Again, in header section of home page. Eg., below tagline, centered, in distinctive font/colour.

Placing these items at top of page avoids clutter in the functional body of page. Frequent visitors can ignore them.



Friends of Carnegie Library... x +

friendsofcarnegielibrary.org.uk

Most Visited Getting Started Amazon eBay Suggested Sites Web Slice Gallery WildTangent Games

# Friends of Carnegie Library



*Promoting Use of and Access to the Carnegie Library in Herne Hill. We support the DefendTheTen campaign which is working to save all 10 Lambeth libraries: <http://defendthe10-lambeth.org.uk>*

Home Join Aims Events Services Forums Newsletters History Contact

## PUBLIC MEETINGS ON THE FUTURE OF CARNEGIE LIBRARY

🕒 September 18, 2016 📁 Campaigns, Events, News 👤 Nicholas Edwards

With our partners in the Carnegie Library Association CIO, the Friends are arranging two public meetings to discuss plans for the future. The announcement leaflet is [here](#)

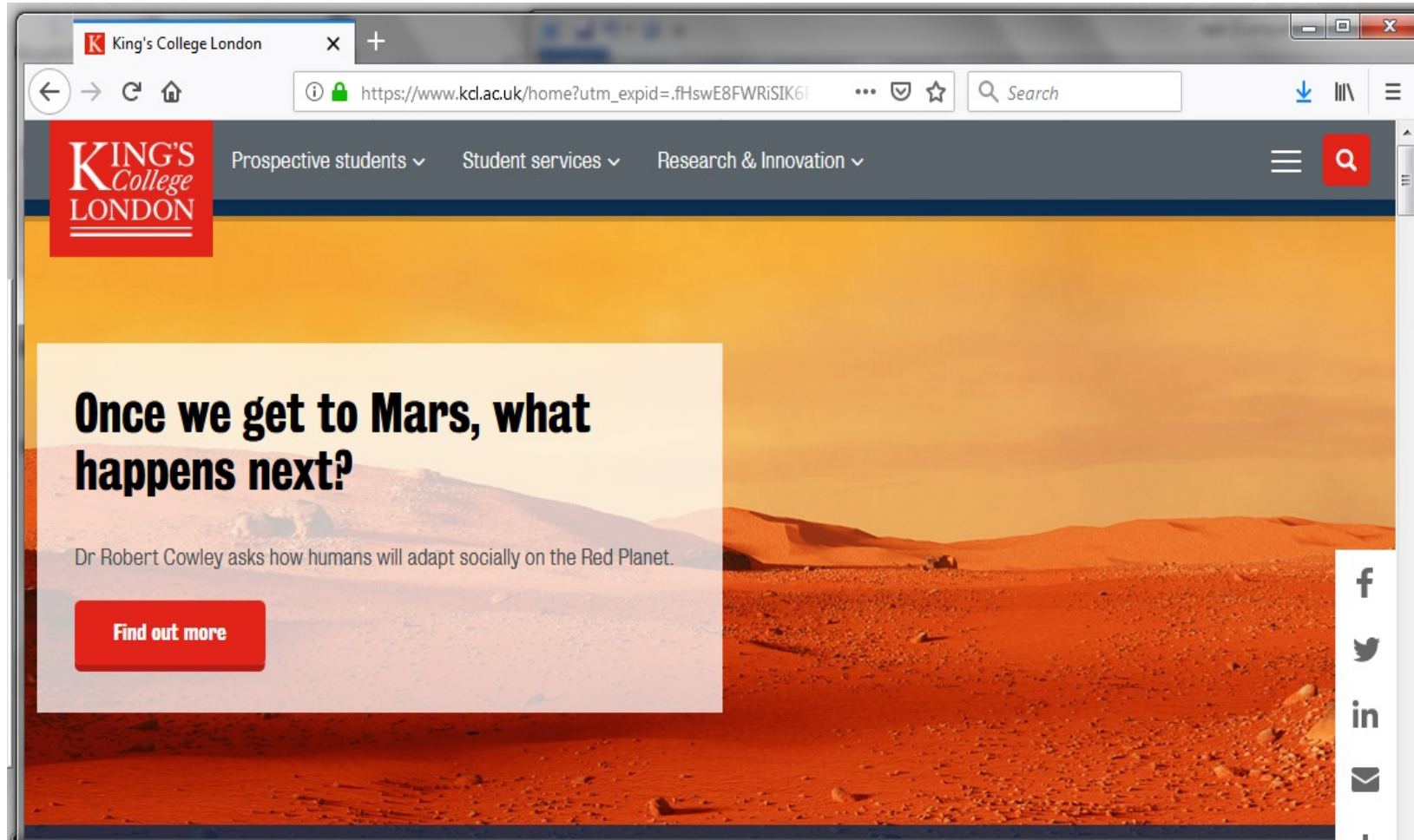
Are you a Friend? Login

### The Carnegie on Twitter

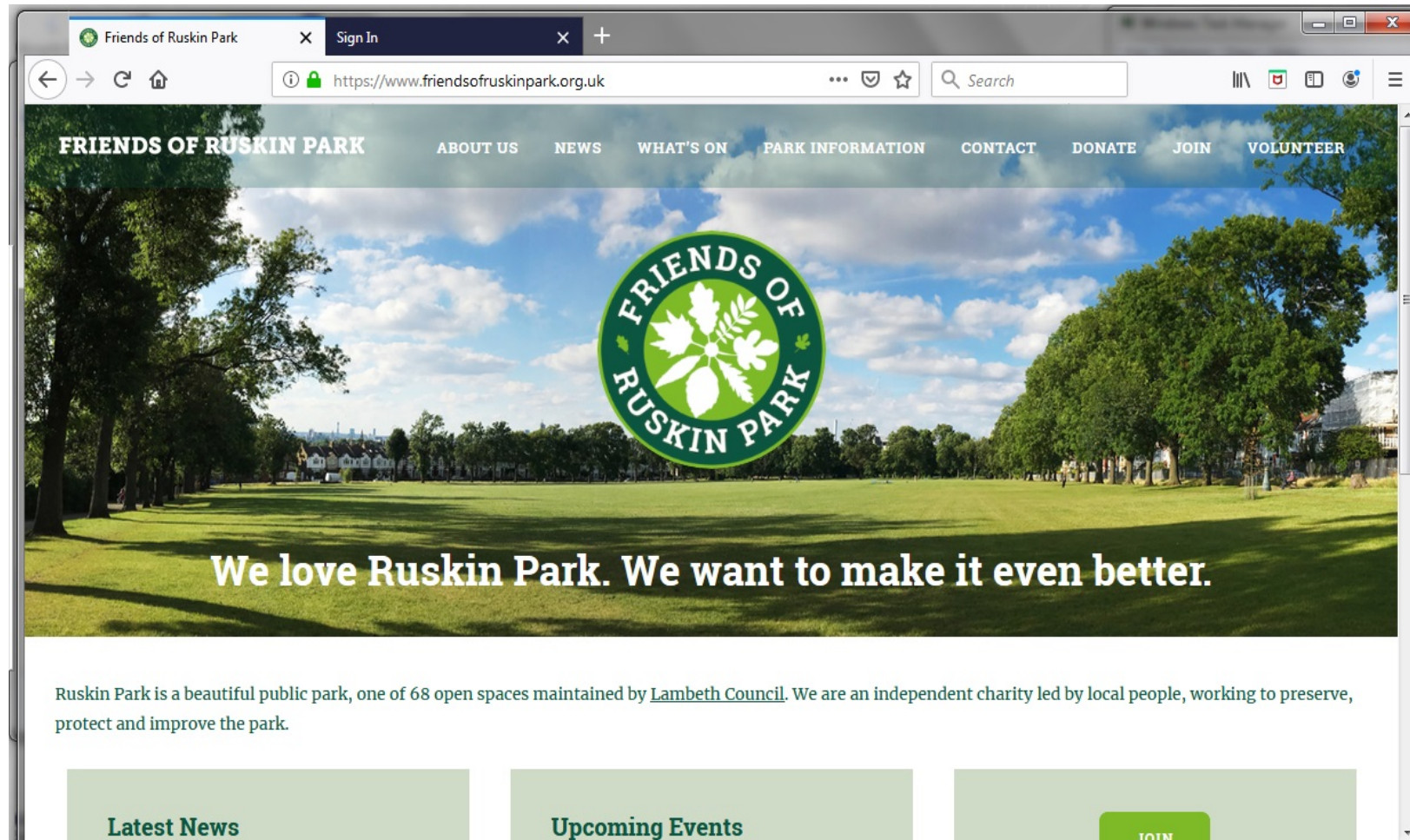
 Carnegie Library @CarnegieLib

This Census-taker by China Mieville  
ethlibraryreadinggroups.wordpress.com/2016/09/09/thi...

Home page of Friends of Carnegie Library



kcl.ac.uk homepage



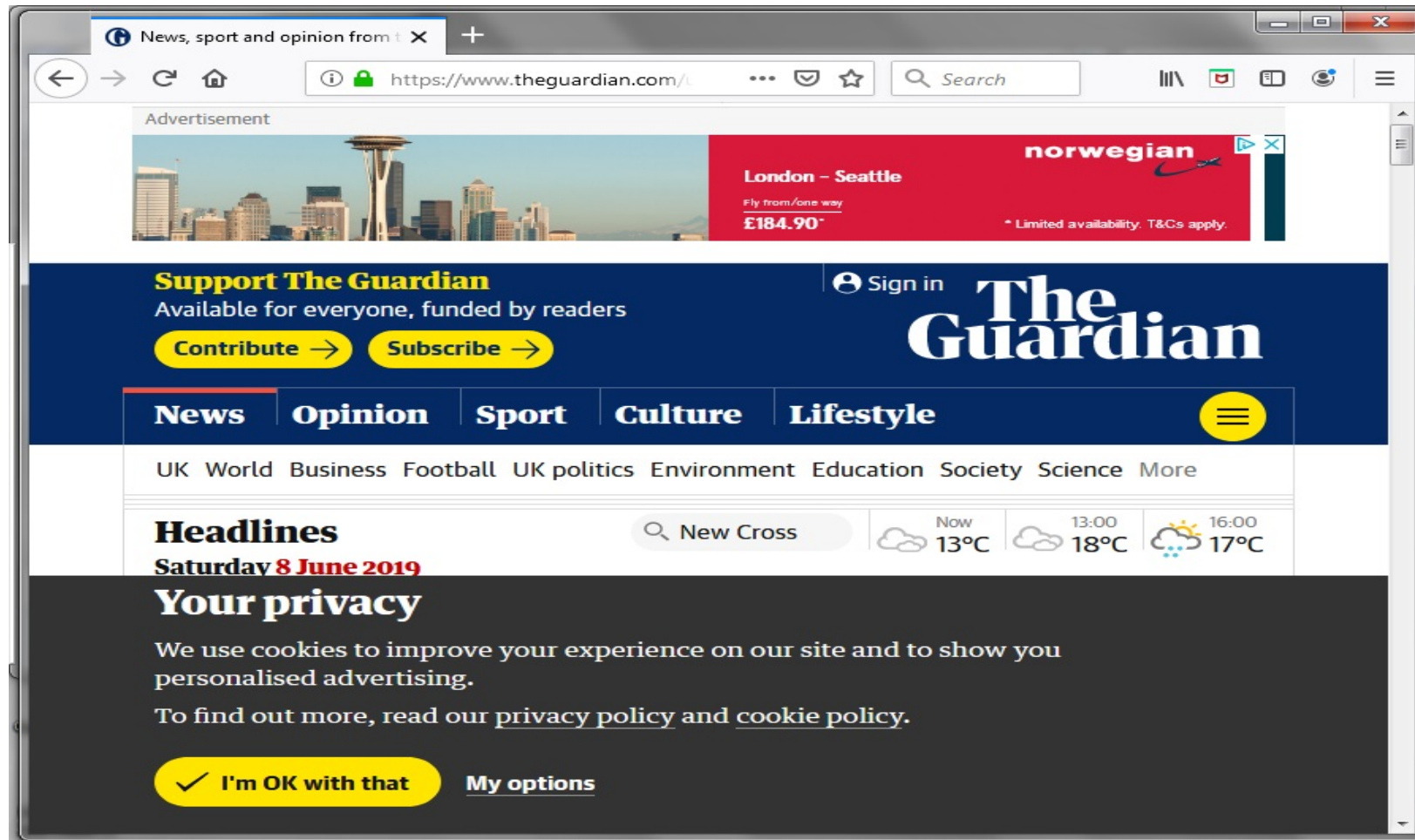
Friends of Ruskin park homepage

### *Text and explanations*

- For *VinylRevival*, could design a logo using a record image + the name
- Organisations may already have a logo, which can be used.
- A tagline for *VinylRevival* could be “Buy and sell those bygone records”
- Again, an organisation may have an established mission statement, but this may need to be cut/adapted to work on a web page – eg., no more than 8 words. Purpose of site may also be different to the organisation’s general mission
- Good taglines emphasise the specific merits of the site, eg.: “The largest UK trading site for vintage records”.

### *Text and explanations*

- For *VinylRevival*, welcome blurb could be “Welcome to VinylRevival, we stock 1000’s of vinyl LP’s, 45’s & even 78’s! You can also sell your own vinyl records in our marketplace.”
- Clear + to point about what site is + what it offers
- Make blurb distinctive but clearly part of the site. It must not look like a banner ad (which sometimes appear in heading space of home pages)
- Importance is mainly for new users, returning users will ignore it.



Guardian site with banner ad + disclaimer

### *Presentation choices*

- Colour schemes: blues, greys to convey a professional business-like impression. Greens, orange more friendly and relaxed. Colour scheme may be mandated by existing business/organisation branding.
- Fonts: Arial and other sans-serif fonts are more readable than serif fonts (Times Roman, etc).
- Flashing animations/rapidly changing images can create negative impression (“if they’re trying so hard to sell me this, something must be wrong with it”).

## *Frames and divisions*

- Frames and divisions subdivide a page into logically separate areas
- Divisions can include:
  - Header – containing menubar, logo, blurb, search box and other persistent header content
  - Left column for navigation via links
  - Right column for twitter feed/social media links
  - Main central area for main page content
- Each division can have separate colour scheme/format – although these should be compatible within one page + the same from page to page.

For mobile devices the side frames can be omitted and width of screen reduced.



## *Frames and divisions*

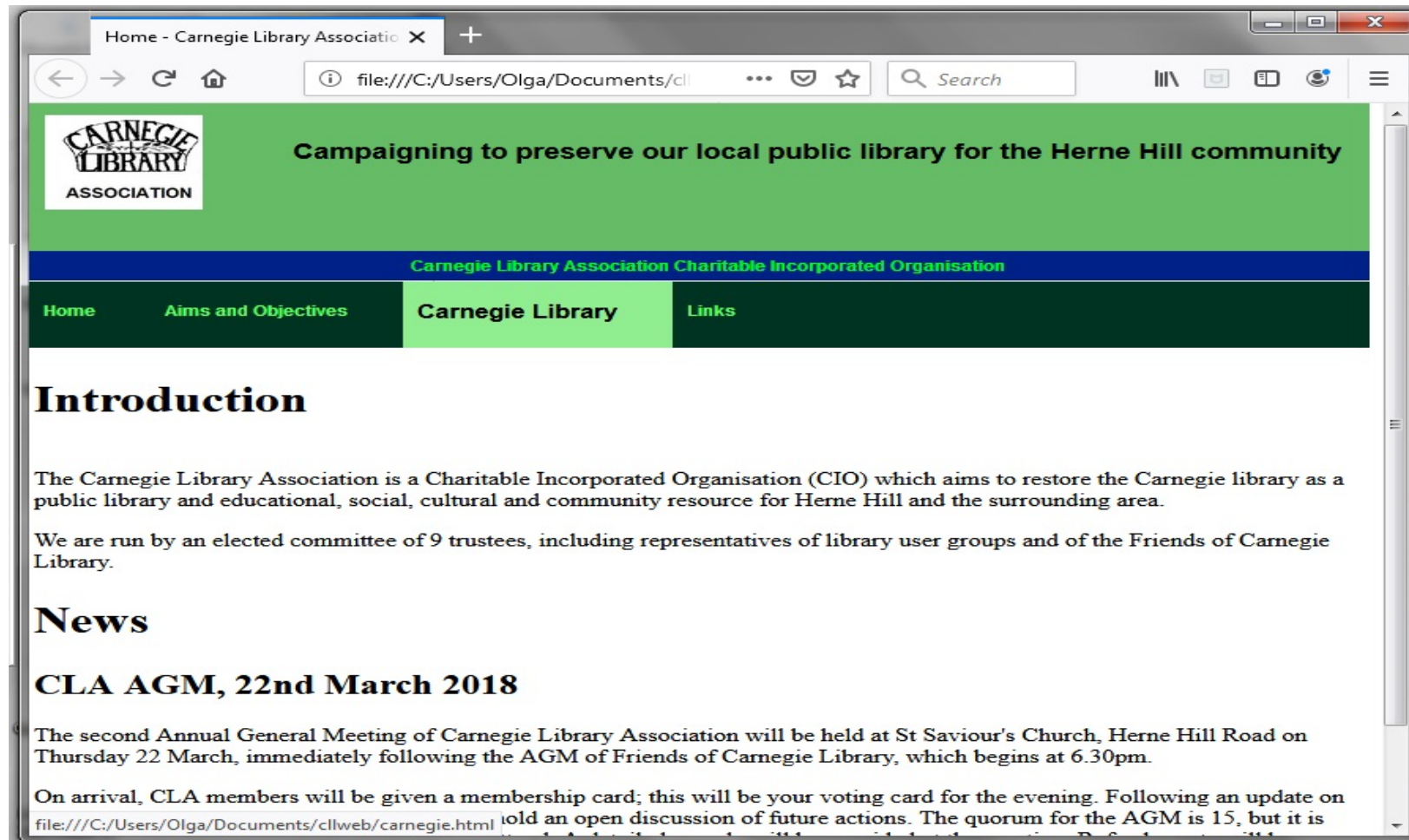
- Frames and divisions can be nested – divisions within divisions  
– eg., logo and blurb divisions within header division:

```
<div id="header">  
  <div id="logo">  
    <a href="index.html">  
      </a>  
  </div>  
  
  <div id="blurb">Campaigning to preserve our local  
    public library for the Herne Hill community  
  </div>  
</div>  
  
<div id="slogan">Carnegie Library Association CIO</div>
```

```
<div id="navigation">

<ul id="menubar">
<li><a class="menuitem" href="index.html"
      shape="rect">Home</a></li>
<li><a class="menuitem" href="trustees.html"
      shape="rect">Aims and Objectives</a></li>
<li><a class="menuitem" href="carnegie.html"
      shape="rect">Carnegie Library</a></li>
<li><a class="menuitem" href="links.html"
      shape="rect">Links</a></li>
</ul>
</div>
```

The logo can also be a link to home page.



CLA homepage with menubar

### *Frames and divisions*

- Left side of page often used for detailed navigation
- Main content area for important news/featured products, etc
- Right side for Twitter feed, other social media.

Friends of Carnegie Library... x +

friendsofcarnegielibrary.org.uk

Most Visited Getting Started Amazon eBay Suggested Sites Web Slice Gallery WildTangent Games

# Friends of Carnegie Library



*Promoting Use of and Access to the Carnegie Library in Herne Hill. We support the DefendTheTen campaign which is working to save all 10 Lambeth libraries: <http://defendthe10-lambeth.org.uk>*

Home Join Aims Events Services Forums Newsletters History Contact


## PUBLIC MEETINGS ON THE FUTURE OF CARNEGIE LIBRARY

🕒 September 18, 2016 📁 Campaigns, Events, News 👤 Nicholas Edwards

With our partners in the Carnegie Library Association CIO, the Friends are arranging two public meetings to discuss plans for the future. The announcement leaflet is [here](#)

[Are you a Friend?](#) [Login](#)

### The Carnegie on Twitter

 Carnegie Library @CarnegieLib

This Census-taker by China Mieville  
ethlibraryreadinggroups.wordpress.com/2016/09/09/thi...

Home page of Friends of Carnegie Library

## *Summary*

- Logo, tagline and blurb identify site + its purpose to users
- Pages should have clear organisation + structure
- Header area usually for consistent identification + navigation elements
- Other areas for detailed navigation, main content, social media/links.

*Exercise: design homepage of community site*

- Imagine you are designing a site for a park users group to give information about a small park in an urban area – it has a playground, sports pitch + pond
- Your site aims to encourage local residents to use the park + tell them about its services + history. In addition, to encourage people to volunteer to help care for the park.
- Design the homepage structure, with tagline and blurb.

## Session 12: Helping users navigate your site

- Menubars and tabs
- Search boxes
- Exercise: add a menubar to the community site



## *Menubars and tabs*

List main sections of site (primary navigation).

- Common navigation option is list of links in left hand column of pages
- But takes up considerable page space – particularly significant for mobile devices
- A menubar or tabs in page header is more compact
- Must ensure navigation is the same from every page – identical options in same order + same format

## *Menubars and tabs*

- Essentially, menubars are horizontally-formatted lists of links, with *hover* defined to change colour of item
- Each item could have a submenu – in vertical formatting – beneath it
- Order of items must be same on every page
- Order is usually that more frequently-used options on left of less used options
- Labels must be clear + concise (usually one word)
- Link is to a page with same name/topic as the label.

## *Menubars and tabs*

Style sheet for menubar:

```
#menubar {  
    margin : 0;  
    padding : 0;  
    list-style-type: none;  
    border-top: 1px solid white;  
}
```

```
#menubar a {  
    margin: 0;  
    padding: 0px 35px 0px 9px;  
    display: block;  
    float: left;  
    line-height: 50px;  
    text-align: center;
```

```
height: 56px;  
}
```

Sets format for menubar and the links within it.

Within menubar, list style is none (no bullets) and links are displayed floating left.

## *Menubars and tabs*

Style sheet for menu items:

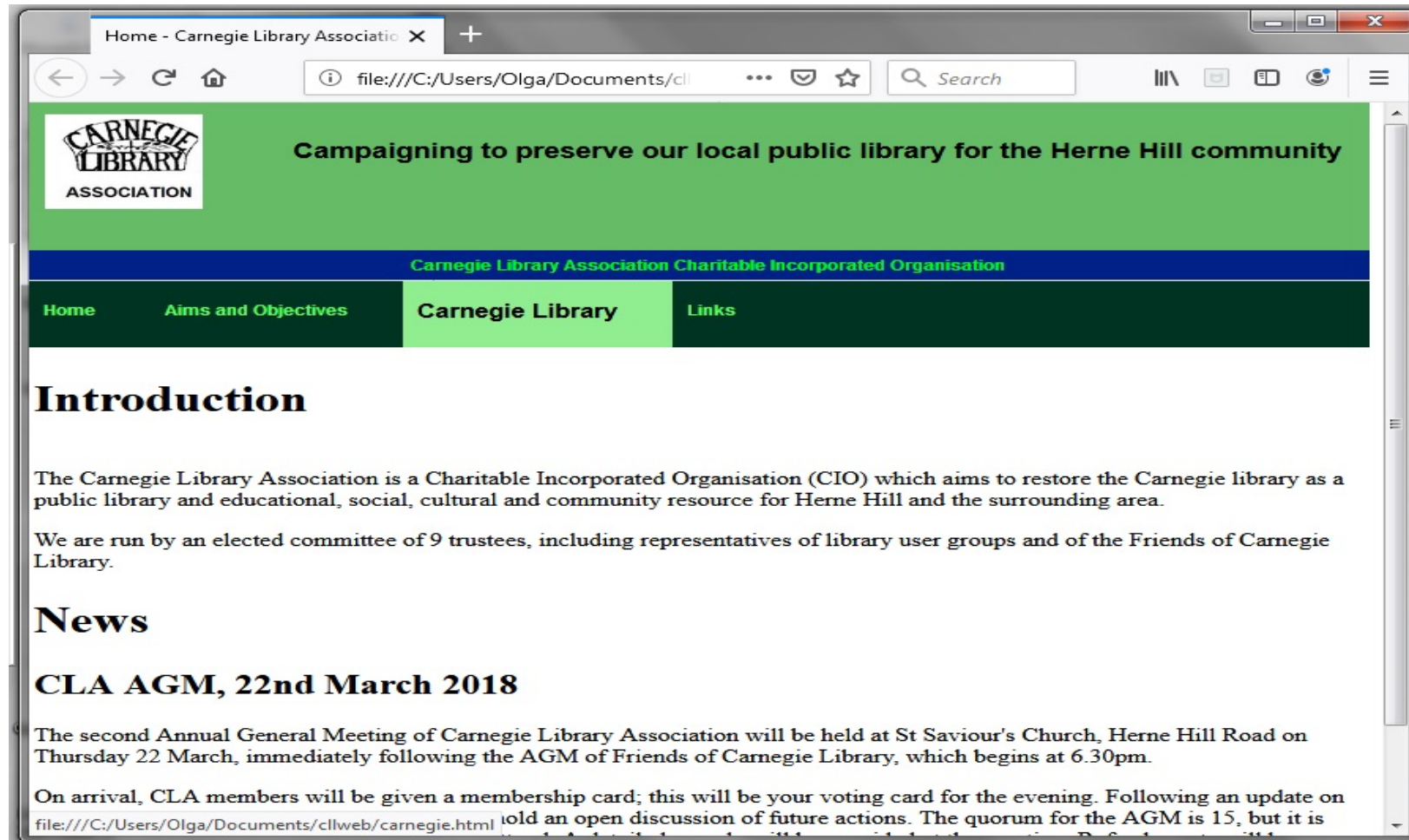
```
#menubar li {  
    float: left;  
    margin: 0px 0 0 0;  
    padding: 0;  
}  
  
#navigation {  
    background-color: #003322;  
}  
  
a.menuitem {  
    background: no-repeat;  
    background-position: top left;  
    color: #66ff66;
```

```
}
```

```
a.menuitem:hover {  
    background-color: lightgreen;  
    color: black;  
}
```

Normal background colour for navigation is dark green/blue, so mousing over menu items will show clear change.

An alternative is to use a table to format the menu items.



CLA site with menubar

## *Menubars and tabs*

Can also enlarge font on hover:

```
a.menuitem {  
    font-family: Arial,sans-serif;  
    font-style: normal;  
    font-size: 12px;  
    font-weight: bold;  
    text-decoration: none;  
    background-position: top left;  
}
```

```
a.menuitem:hover {  
    font-family: Arial,sans-serif;  
    font-style: normal;  
    font-size: 16px;  
    font-weight: bold;
```



```
text-decoration: none;  
background-position: top left;  
}
```

But this is perhaps more useful for tabs.

### *Menubars and tabs*

- Tabs are web version of physical notepad/organiser tab dividers
- Same position as menubar, same principles for labels
- Clicking on a tab brings it and its page 'forward' to the browser window
- Intuitive to use and gives clear page location
- Has become less popular as more complex to code than simple menubars.

## *Menubars and tabs*

- A basic tab approach is to distinguish three states for menu items:
  1. Item is for current page: has same background colour as current page + text in contrasting colour, emphasised relative to other menu items
  2. Item is for different page + not under mouse: different background colour to page + text not emphasised
  3. Item for different page + under mouse: different background colour to page + other menu items + text emphasised.
- Implement with extra class *selected* for item of current page.

## *Menubars and tabs*

Additional styles:

```
a.selected {  
    background-color: white;  
    color: #005500;  
}
```

```
a.selected {  
    font-family: Arial,sans-serif;  
    font-style: normal;  
    font-size: 16px;  
    font-weight: bold;  
    text-decoration: none;  
    background-position: top left;  
}
```

The *selected* element has no hover behaviour.

## *Menubars and tabs*

```
<!DOCTYPE html>
<html><head>
<title>Trustees - Carnegie Library Association CIO</title>
<link rel="stylesheet" type="text/css" href="layout.css">
<link rel="stylesheet" type="text/css" href="colourscheme.css">
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body id="main_body">
<div id="header">
  <div id="logo"><a href="indextabs.html">
    </a></div>
  <div id="blurb">Campaigning to preserve our local public
  library for the Herne Hill community
  </div>
</div>
</div>
```

```
<div id="slogan">Carnegie Library Association  
Charitable Incorporated Organisation</div>
```

```
<div id="navigation">
```

```
<ul id="menubar">
```

```
<li><a class="menuitem" href="indextabs.html"  
  shape="rect">Home</a></li>
```

```
<li><a class="selected" href="trusteestabs.html"  
  shape="rect">Aims and Objectives</a></li>
```

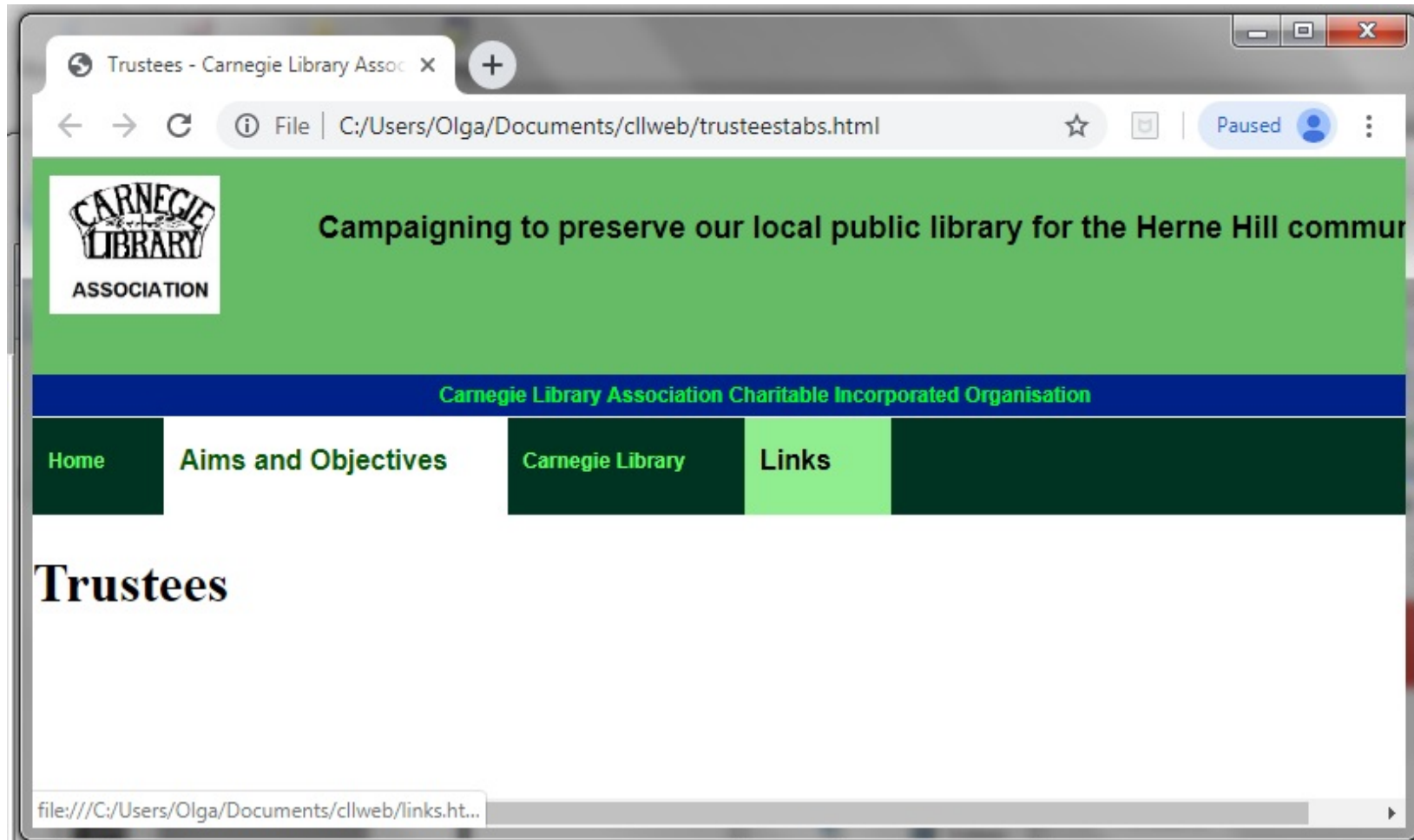
```
<li><a class="menuitem" href="carnegietabs.html"  
  shape="rect">Carnegie Library</a></li>
```

```
<li><a class="menuitem" href="links.html"  
  shape="rect">Links</a></li>
```

```
</ul>
```

```
</div>
```

```
<div id="main_container">  
<h1>Trustees</h1>  
</div></body></html>
```



CLA site with tabs

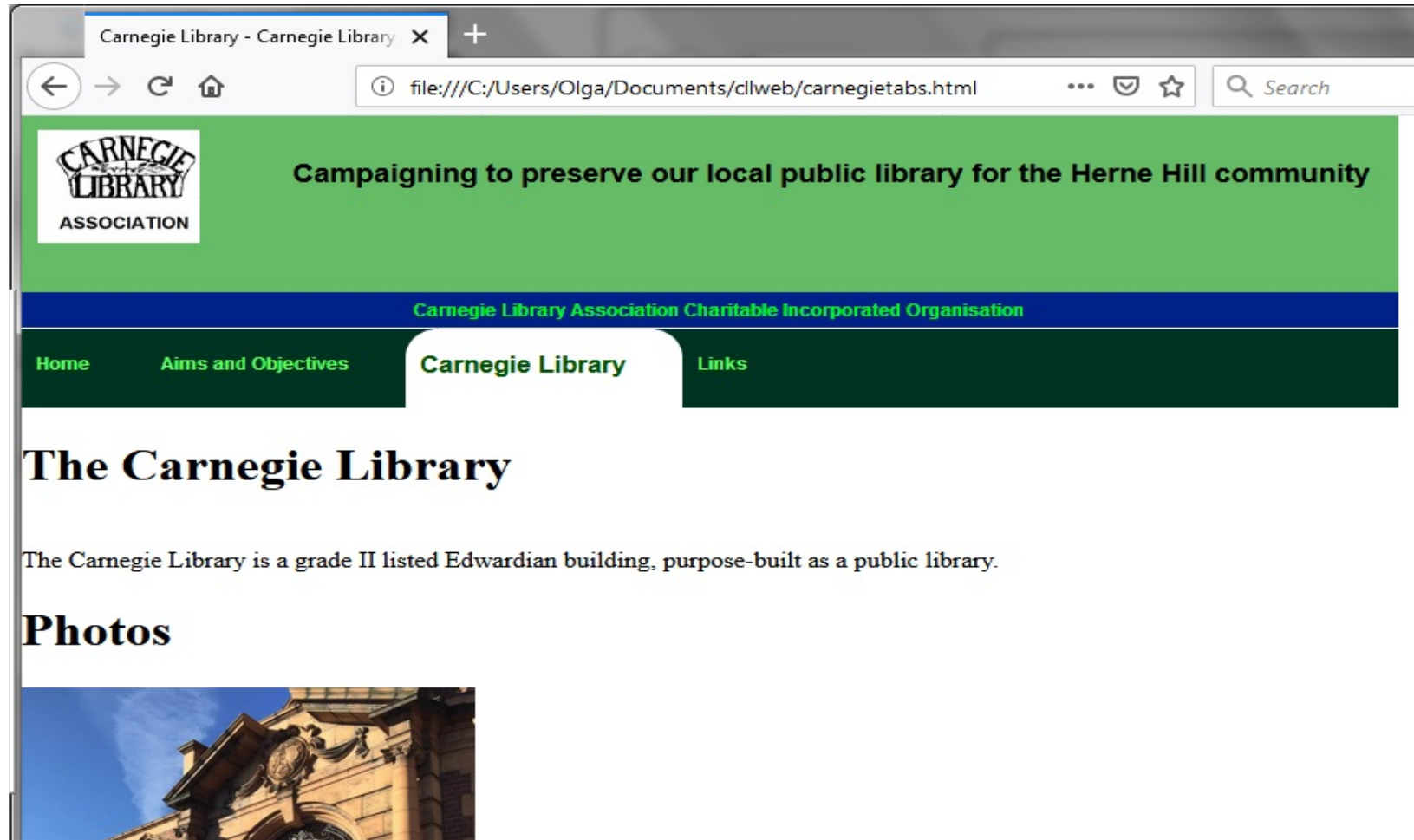


## *Menubars and tabs*

Curved tabs can be defined by using the CSS *border-radius* property:

```
a.selected {  
    font-family: Arial,sans-serif;  
    font-style: normal;  
    font-size: 16px;  
    font-weight: bold;  
    text-decoration: none;  
    background-position: top left;  
    border-radius: 20px 20px 0px 0px;  
}
```

The top left and top right corners of selected item are rounded with radius 20px, the bottom corners are normal (square).



Curved tabs

### *Search boxes*

- Simplest way of finding something in a site is a search
- Accessible to any user – they do not need to navigate
- Sites such as Amazon have made search the principal navigation option
- But – search must be effective and return good quality results – very frustrating for users to get long list of irrelevant results (or no results at all!).

## *Search boxes*

- Users expect these will look like a text field with a button labelled *Search*, or a magnifying glass icon.
- Calling *Search* something else – like *Find* – can be confusing
- Complicating the search with lots of options is also confusing – your search algorithm should be intelligent enough to find relevant results
- Filtering options can be provided on search results page (eg., news items versus events, etc – corresponding to where result was found).

## *Search boxes*

- Search usually implemented on server side – eg., by program that searches application databases or all web pages of the site
- Alternatively, by a custom Google search (fee-paying)
- One of simplest ways to implement search is to call Google search from your search. This expects a parameter with name  $q$  for the search term
- Define search as a form with text input field  $q$ , and submit button labelled *Search*
- Use table to format it vertically or horizontally
- Form action invokes Google search.

## *Search boxes*

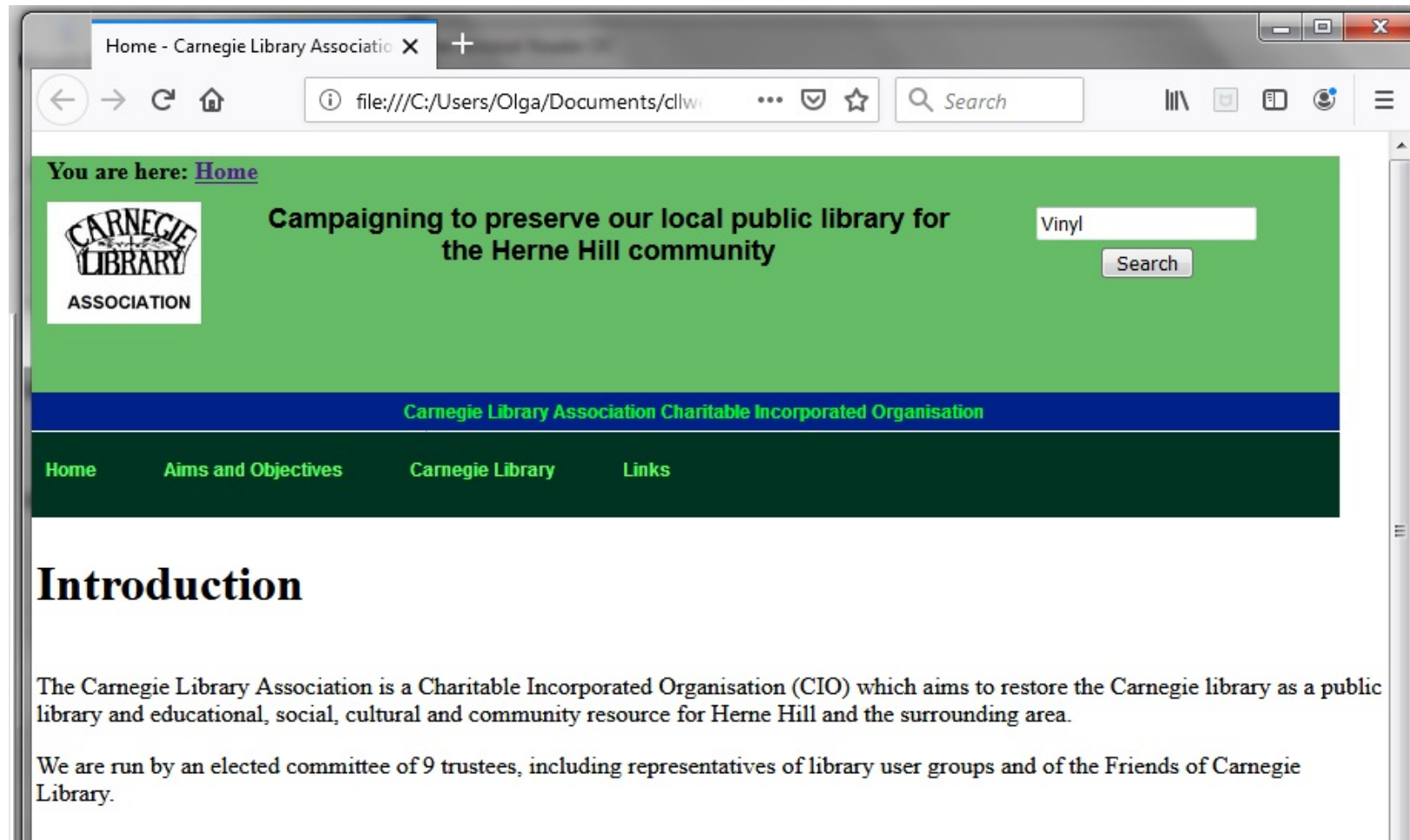
```
<!DOCTYPE html>
<html><head>
<title>Home - Carnegie Library Association CIO</title>
<link rel="stylesheet" type="text/css" href="layout2.css">
<link rel="stylesheet" type="text/css" href="colourscheme.css">
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body id="main_body">
<div id="header">
  <div id="logo"><a href="indexsearch.html">
    </a></div>

  <div id="blurb">Campaigning to preserve our local public
    library for the Herne Hill community</div>
  <div id="search">
```

```
<form method="get"
  action="https://www.google.com/search">
  <table><tr><td>
    <input type="text" name="q"></td></tr>
    <tr><td><input type="submit" value="Search"></td></tr>
  </table></form></div>
</div>
```

```
<div id="slogan">Carnegie Library Association
  Charitable Incorporated Organisation</div>
```

...



Site with simple search



## *Search boxes*

New style items:

```
#header {  
    position: relative;  
    padding: 0;  
    margin: 0;  
    width: 850px;  
    height: 155px;  
    top: 0px;  
    left: 0px;  
}
```

```
#blurb {  
    clear: both;  
    position: absolute;  
    top: 30px;
```

```
    left: 150px;  
    width: 450px;  
    height: 70px;  
    text-align: center;  
}
```

```
#search {  
    clear: both;  
    position: absolute;  
    top: 30px;  
    left: 650px;  
    width: 100px;  
    height: 70px;  
    float: left;  
    text-align: center;  
}
```

```
#logo {  
    width: 100px;  
    height: 100px;  
    position: absolute;  
    top: 30px;  
    left: 10px;  
}
```

These layout specifications place search to right of blurb in header.

## *Search boxes*

An alternative is to insert a specific website name (such as your website) into the Google search:

```
<script>
function processQuery()
{ document.searchform.q.value =
  "https://www.vinylnet.co.uk/ " +
  document.searchform.qry.value;
  document.searchform.qry.value = "";
  return true;
}
</script>

...

<div id="search">
```

```
<form name="searchform" method="get"
      action="https://www.google.com/search">
<table>
  <tr><td><input type="text" name="qry"></td></tr>
  <input type="hidden" name="q">
  <tr><td><input type="submit"
        onclick="processQuery()" value="Search"></td></tr>
</table></form></div>
```


The *hidden* field *q* is written by the *processQuery()* function and submitted to Google – includes website address you want to search. The *qry* field data is erased.

Home - Carnegie Library Associatio X +

file:///C:/Users/Olga/Documents/cllweb/indexsearch.html

Search

You are here: [Home](#)



**Campaigning to preserve our local public library for the Herne Hill community**

Carnegie Library Association Charitable Incorporated Organisation

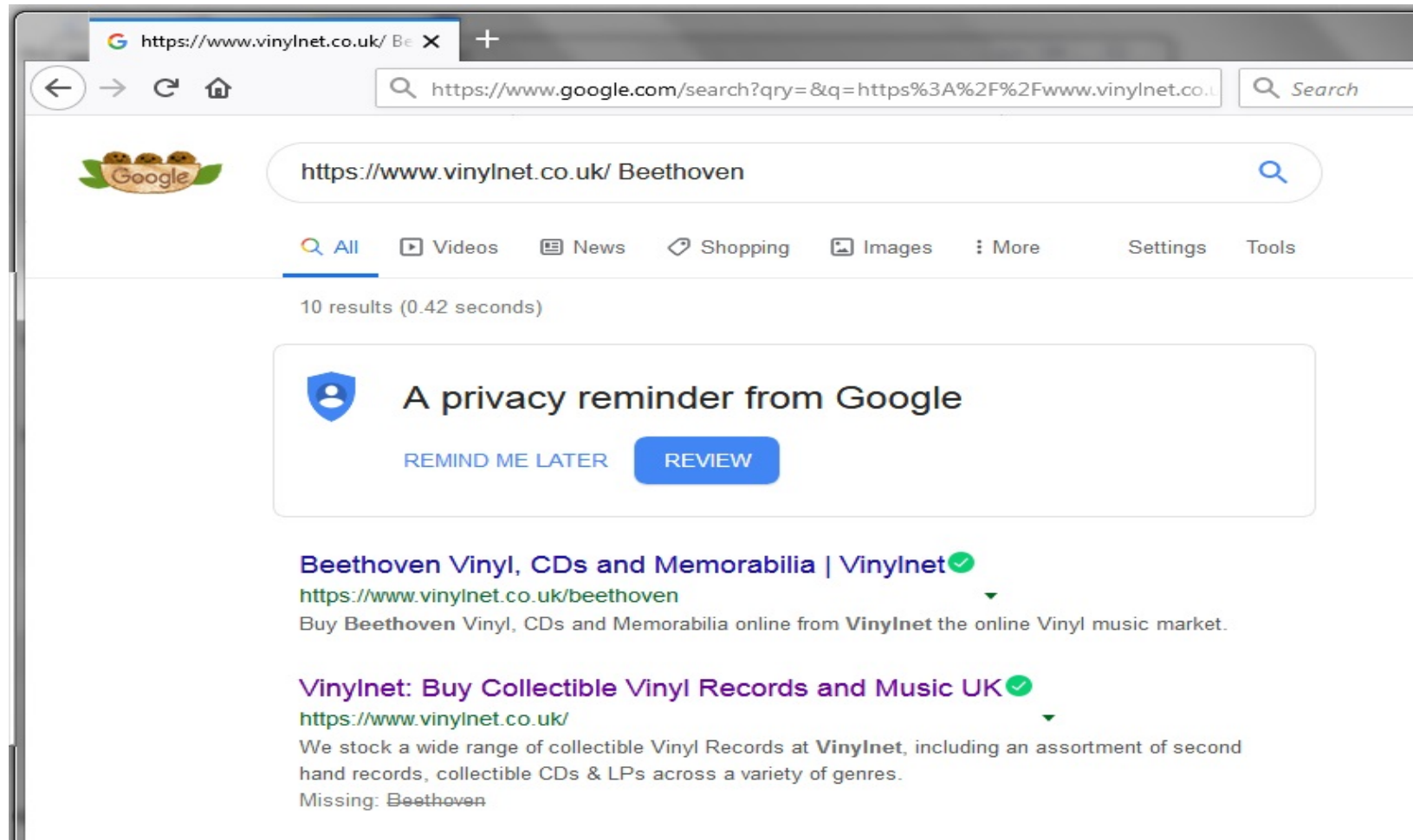
[Home](#)   [Aims and Objectives](#)   [Carnegie Library](#)   [Links](#)

## Introduction

The Carnegie Library Association is a Charitable Incorporated Organisation (CIO) which aims to restore the Carnegie library as a public community resource for Herne Hill and the surrounding area.

We are run by an elected committee of 9 trustees, including representatives of library user groups and of the Friends of Carnegie Libr

Search query



Search result

## *Summary*

- Menubars and tabs enable users to select specific area of site that interests them/is relevant to their goal
- Menu options should be named clearly + take user to correct place
- Navigation content + format should be consistent on all pages
- Search should provide meaningful results.



*Exercise: add a menubar to the community site*

- Add a menubar to the park site, below the blurb
- Choose suitable options for the menu items, and a colour scheme, defined in the *colourscheme.css* style sheet

## Session 13: Navigation and location in websites

- Persistent navigation and site identifiers
- Page names
- Breadcrumbs
- Exercise: design navigation for community website

### *Persistent navigation and site identifiers*

- Important to offer same navigation options in same way on different pages – to avoid user confusion
- Site identifier should appear on every page – can be more prominent on home page
- Navigation options help users to locate themselves relative to site – get global view of what is present in site
- Navigation can suggest an order/priority for using site (eg., options *News, Events, Join, Blog* on a community site).

## *Page names*

- Page names are like aisle signs in a supermarket, or section identifiers in a magazine
- Page name should be immediately apparent from the page
- Essential to help user locate themselves
- Large font – ie., *h1* – left aligned at head of main content.

## *Breadcrumbs*

- Show sequence of pages passed through in site to reach current page
- Labelled *You are here* : followed by list of page names (as links) separated by > symbol. Final name is current page
- Enable user to retrace their steps + try alternatives
- Same idea is used in file system navigation in Windows.

For example:

You are here: Home > Buy > Fifties > **Bing Crosby**

Best location for this is at very top of page, above header, so does not interfere with main navigation + content.

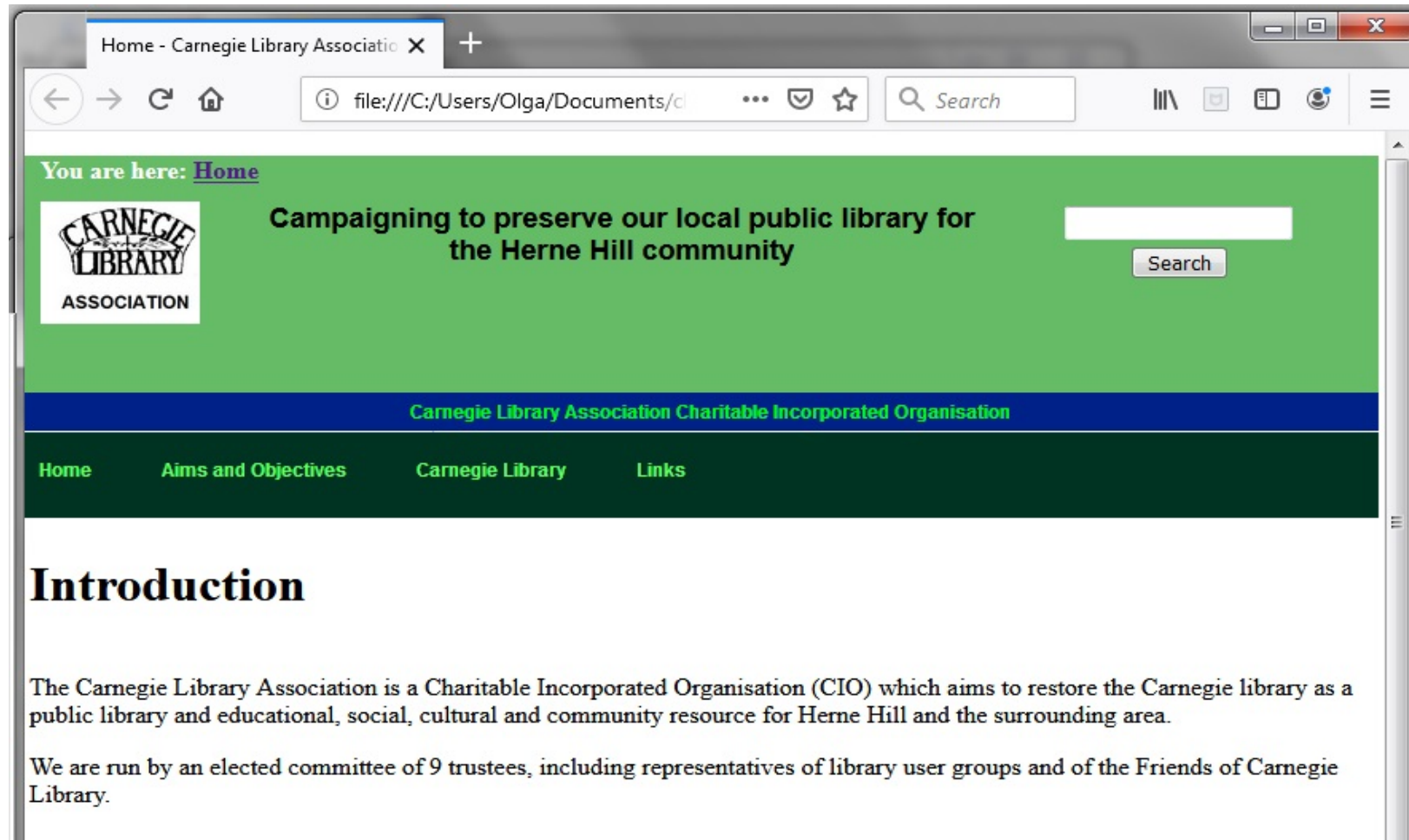
## *Breadcrumbs*

```
<!DOCTYPE html>
<html><head>
<title>Home - Carnegie Library Association CIO</title>
<link rel="stylesheet" type="text/css" href="layout2.css">
<link rel="stylesheet" type="text/css" href="colourscheme.css">
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body id="main_body">
<div id="header">
  <div id="breadcrumbs"><strong>You are here:</strong>
    <a href="indexsearch.html"><strong>Home</strong>
  </a></div><p>

  <div id="logo"><a href="indexsearch.html">
    </a></div>
```

```
<div id="blurb">Campaigning to preserve our local public  
  library for the Herne Hill community</div>
```

...



CLA site with breadcrumbs

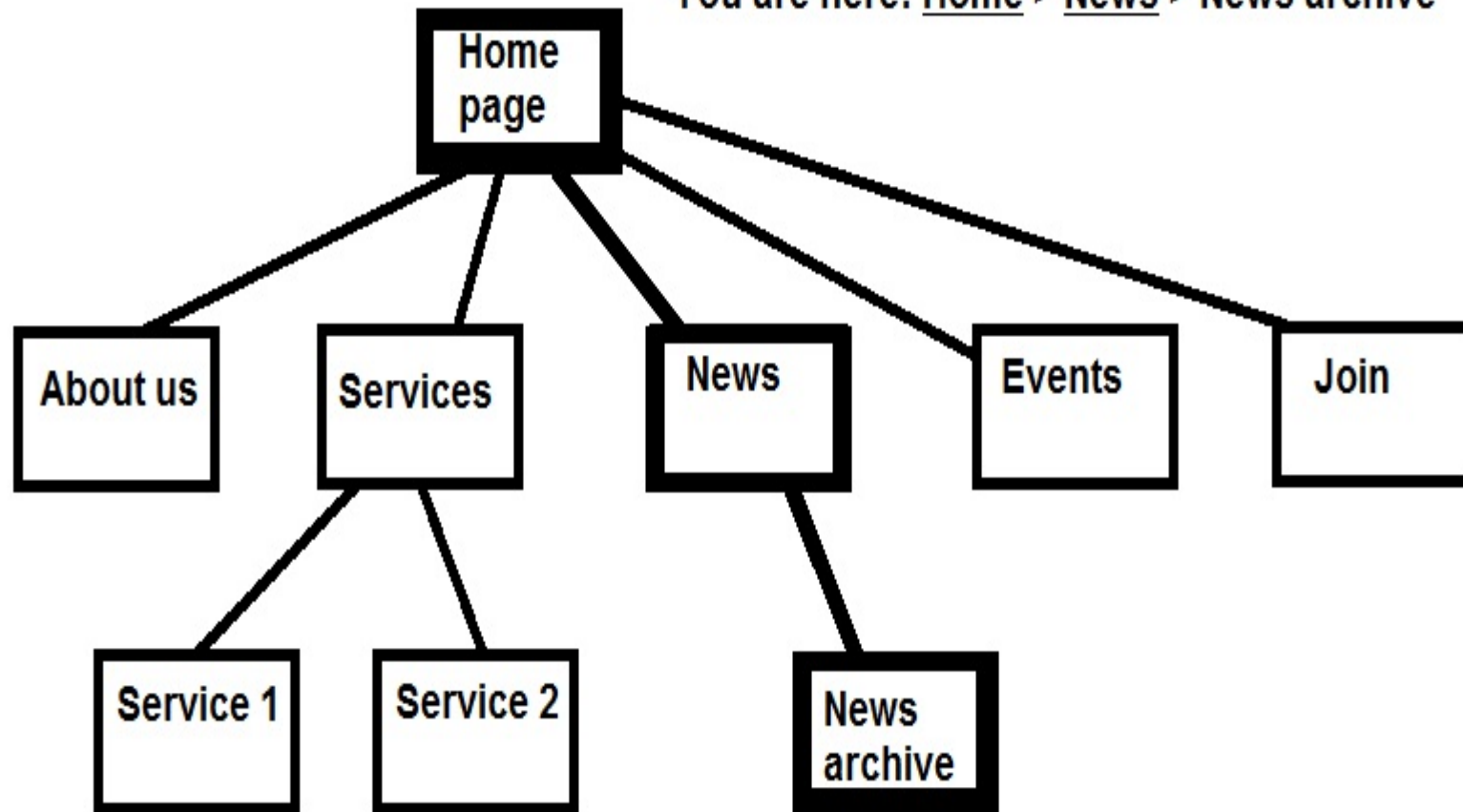


## *Breadcrumbs*

Additional style definitions:

```
#breadcrumbs {  
    width: 700px;  
    height: 25px;  
    position: absolute;  
    top: 0px;  
    left: 10px;  
    color: white;  
}
```

You are here: [Home](#) > [News](#) > News archive



Navigation with breadcrumbs

## *Summary*

- Users should be able to see clearly where they are within a site
  - what specific page, and how this is located within the site
- Page names must be clear + meaningful
- Breadcrumbs or similar mechanism useful to show location within site.

*Exercise: design navigation for the community site*

- Add pages for each of the sections in the menubar of the park site
- Include suitable navigation and location information on each of these pages.

## Session 14: Good and bad practices in website design

- Rollovers and popups
- Pulldowns
- Issues for commercial websites

## *Rollovers and popups*

- Mechanisms for adding extra content to pages, which appears based on cursor position
- Eg., an explanation of a menu item could pop up when cursor is over the item
- Can be useful for new visitors to a site, but distracting for experienced users
- Need to ensure that popup appears close to item it refers to
- Simple rollovers such as changing item colour/font with *hover* are ok
- Popups should be used sparingly + be small and non-intrusive, eg., *title* text for images/tables.

## *Rollovers and popups*

Define popup help for a form field:

```
<html> <head>
<title>Popup test</title>
<link rel="stylesheet" type="text/css" href="popups.css">
<script language = "JavaScript">
function cube()
{ var x =
    parseInt(document.form1.field1.value);
  document.form1.field2.value = "" + x*x*x;
}
</script>
</head>

<body>
<form name = "form1"
```

```
    action = "">

<p class="tooltip">
<strong>Enter integer to be cubed:</strong>
<input name = "field1" type = "text">
<span class="tooltiptext">Enter a whole number, positive
or negative</span></p><br>

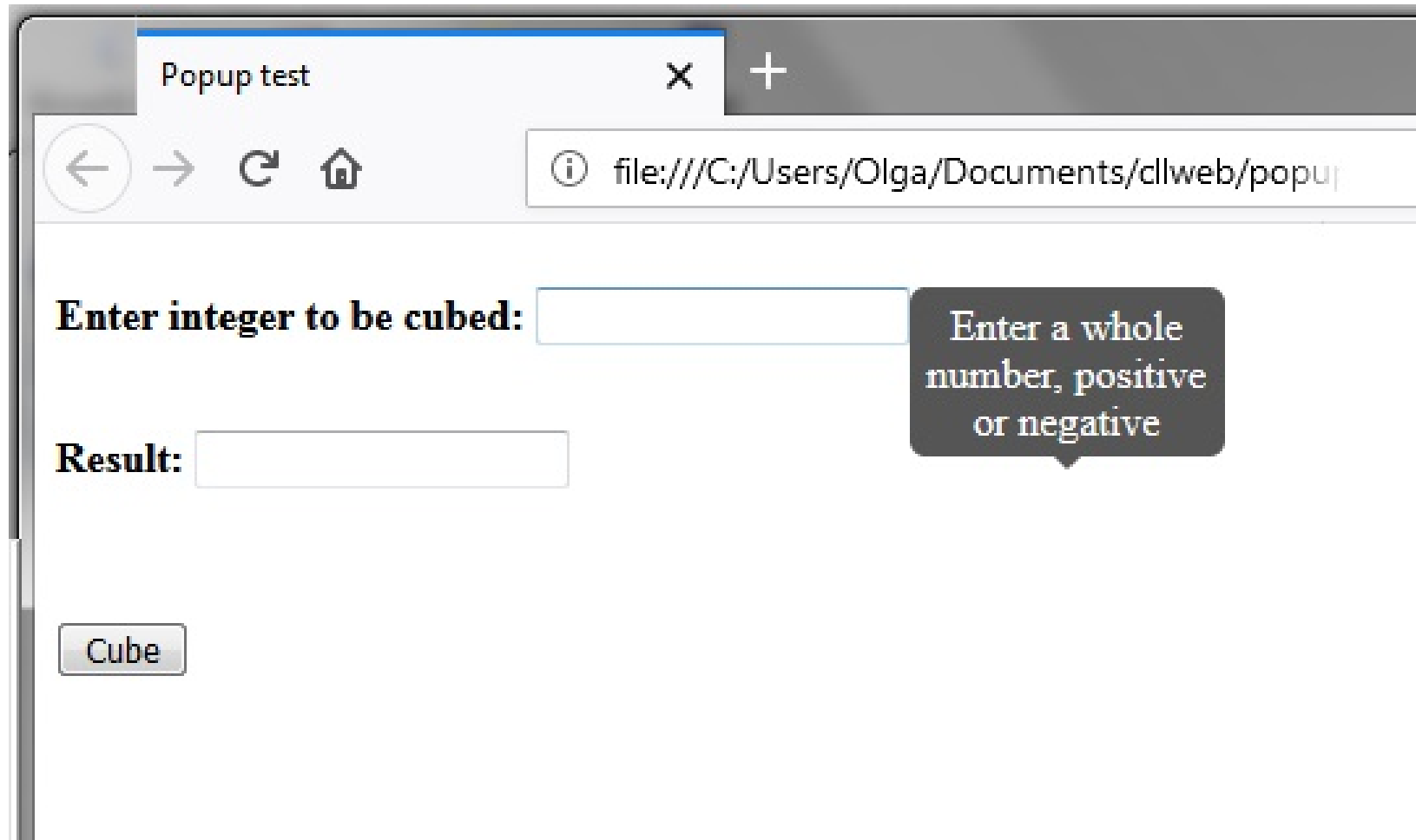
<p><strong>Result:</strong>
<input name = "field2" type = "text"></p><br>

<p><input type = "button"
    value = "Cube" onclick = "cube()"></p>
</form>
</body></html>
```



### *Rollovers and popups*

- The *tooltip* class is for any item which can have a mouseover popup defined for it
- The inner *tooltiptext* element gives the text that will popup on mouseover
- Here, tries to clarify meaning of form field – but preferable to have clear label instead!



Popup help on form field

## *Rollovers and popups*

Styles defined in *popup.css*:

```
.tooltip {  
    position : relative;  
    display : inline-block;  
}  
  
.tooltip .tooltiptext {  
    visibility : hidden;  
    width: 120px;  
    background-color: #555555;  
    color: white;  
    text-align: center;  
    padding: 5px 0;  
    border-radius: 6px;
```

```
position: absolute;
z-index: 1;
bottom: 125%
left: 50%
margin-left: -40px;

opacity: 0;
transition: opacity 0.3s;
}

.tooltip .tooltiptext::after {
content: "";
position: absolute;
top: 100%;
left: 50%;
margin-left: -5px;
```

```
border-width: 5px;  
border-style: solid;  
border-color: #555555 transparent transparent transparent;  
}
```

```
.tooltip:hover .tooltiptext {  
  visibility: visible;  
  opacity: 1;  
}
```

## *Popups*

- The tooltip text is normally hidden and transparent (*opacity* : 0)
- White text on grey background, above any background (z-index : 1)
- On mouseover (hover) of the tooltip element, the tooltip text will become visible and opaque.

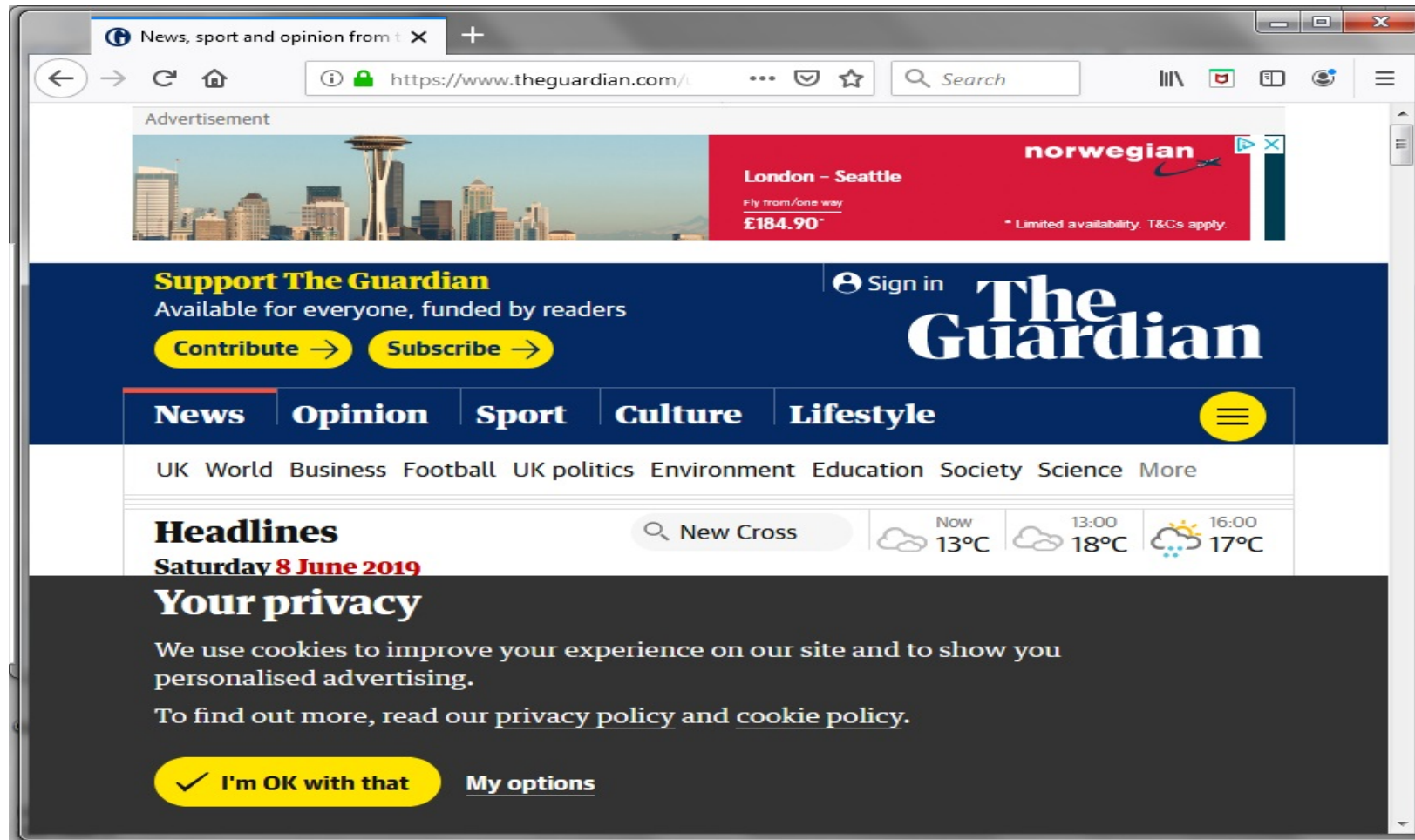
## *Pull downs*

- The *select* form control. Also called “selection lists” and “drop-down menus”
- Advantage: saves space – items to be selected only appear when the selection arrow is pressed
- But also means the items are hidden – can’t be seen by scanning page
- Can be difficult to use for general lists of items
- Most effective for standard lists (eg., country names) in alphabetical order – can use key shortcuts to jump around list
- Invisibility is advantage if list is dynamically updated by server – user isn’t distracted by updates.

### *Issues for commercial websites*

- Sites may need to carry advertising for 3rd parties – adverts often placed above heading at top of home page, may be several on rotation
- Eg: theguardian.com
- Distracting and may discourage users – organisation has to balance income stream against user retention
- Some ads may mislead users into thinking they are part of site “Click here!” messages, etc
- Likewise cross-promotion of other sites – must be related/complementary + adverts not too intrusive/confusing.





Guardian site with banner ad + disclaimer

### *Issues for commercial websites*

- Avoid asking for too much user information when it is not necessary – eg., making someone fill in their email before they can search
- In UK/EU GDPR requires users explicitly agree to site retaining information about them
- Any retained information must be securely stored – never store plain-text personal information in files/databases – must always be encrypted
- Never send plain-text passwords by email!
- Use https not http for any pages processing confidential data.

### *Issues for commercial websites*

- Avoid using jargon/acronyms which general users would not know
- Eg., “Mint condition” means in original wrapper, unopened – ordinary users may not know this
- Especially navigation items must be very clear. “Buy” and “Sell” are better than “Record store” and “Marketplace”
- Avoid writing very long pages which require lots of scrolling.
- Sites should work in all reasonable browsers. Saying “Only works in IE version 9.0” will drive users away.
- Do usability testing with range of testers – may be some ‘obvious’ errors in your site which you don’t perceive.

## *Summary*

- Popups can give useful additional information – but use sparingly
- Don't overuse pulldowns
- For commercial sites need to consider issues of advertising + cross-promotion
- All sites need to consider security + GDPR issues when dealing with user data.

## Session 15: Summary and next steps

- Course summary
- Further technologies: Javascript, AJAX, server-side programming
- Resources

### *Course summary*

- We have covered all of the key HTML elements used to write web pages
- We have used both commercial and non-commercial site examples to illustrate how individual pages and sites can be constructed
- We've covered the main issues of web usability and given guidelines to improve usability.

### *Further technologies*

- Javascript: add coding to web pages to change content dynamically, validate input + provide advanced interaction
- AJAX: Javascript-based technology for making web interfaces highly responsive – parts of a page can be dynamically updated from server independently of other parts
- Server-side programming: code on server which handles web requests (eg., form submissions) + processes data + provides responses as web pages.

## *Javascript*

- Functions defined in *script* element of page *head*
- Functions can read and write page data (eg., in form fields)
- Functions triggered by page events (pressing buttons, mouse movements or page loading): *onclick*, *onmouseover*, *onmouseout*, *onkeydown* (some key is pressed), *onload*, etc.



## *Javascript*

Basic statements are assignments

```
variable = value;
```

Computes *value* and stores result in *variable*.

Conditionals:

```
if (condition)
{ statements1 }
else
{ statements2 }
```

If *condition* is true, executes *statements1*, otherwise executes *statements2*.

Bounded loop:

```
for (var x = a; x <= b; x++)  
{ statements }
```

Repeats *statements* for  $x = a, a + 1, \dots, b$ .

Functions defined as

```
function name(parameters)  
{ code }
```

Can return values to caller by statement

```
return value;
```

Functions called with notation *name(arguments)*.

## *Javascript*

```
<html> <head>
<title>Sums of numbers in 1..10</title>

<script language = "JavaScript">
  document.writeln("<table border = '1' width = '60%'>");
  document.writeln("<tr><td width = '30%'>Number</td> " +
    "<td width = '30%'>Sum</td></tr>");
  var sum = 0;
  for (var i = 1; i <= 10; i++)
  { sum = sum + i;
    document.writeln("<tr><td> " + i +
      " </td><td> " + sum + " </td><tr>");
  }
  document.writeln("</table>");
  // end of main function
```

```
</script>  
</head> <body> </body>  
</html>
```

This code is written as sequence of statements in *script* section.

*document.writeln(s)* writes text *s* to the page.

```
for (var i = 1; i <= e; i++)  
{ code }
```

repeats *code* for  $i = 1, 2, \dots, e$ .

Effect is to write a table with fixed content to the page.

The image shows a screenshot of a web browser window. The title bar reads "Sums of numbers in 1..10". The address bar shows the file path "C:/Users/Olga/Documents/c...". The browser is in a "Paused" state. The main content area displays a table with two columns: "Number" and "Sum". The table contains the following data:

Number	Sum
1	1
2	3
3	6
4	10
5	15
6	21
7	28
8	36
9	45
10	55

Sum table example

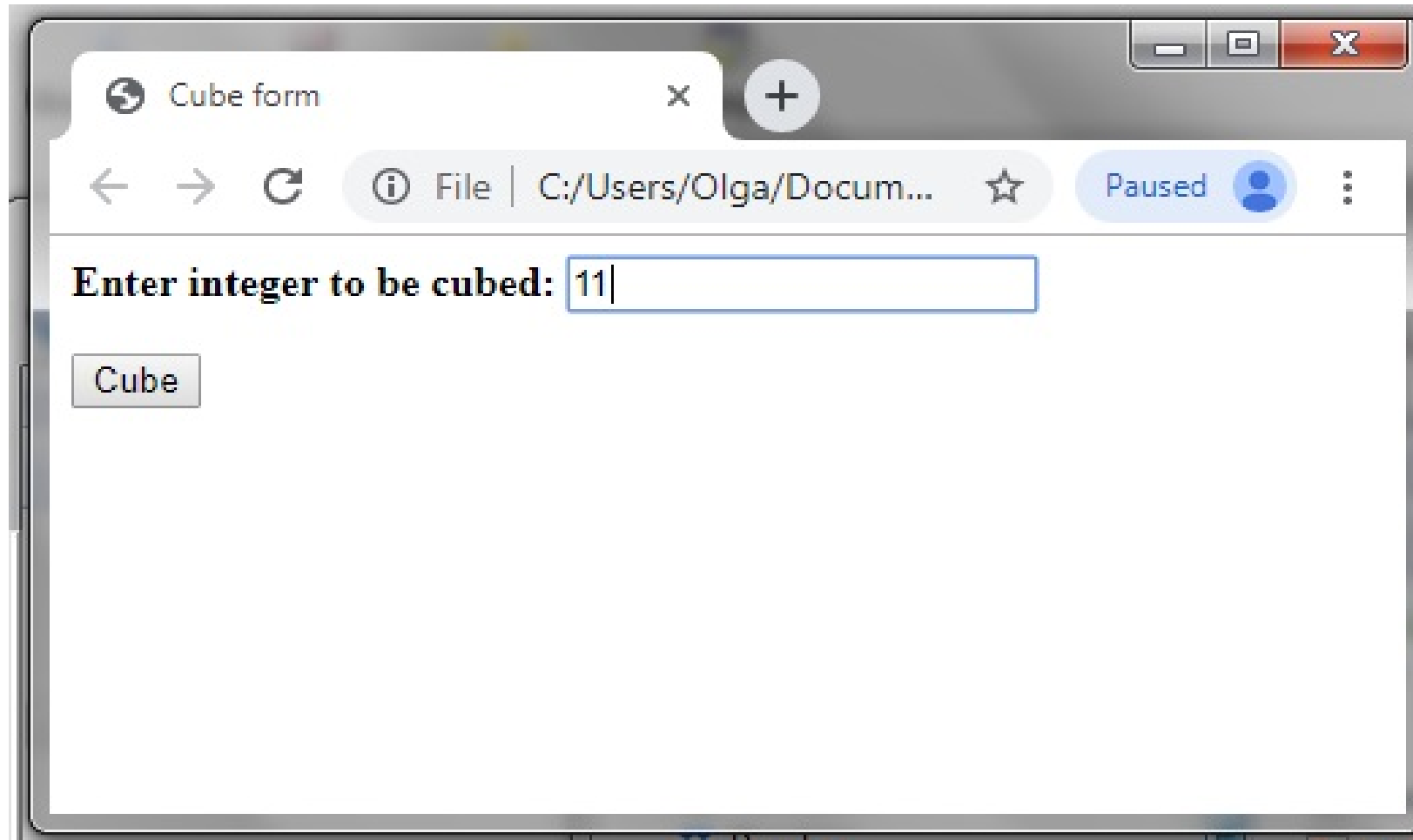
## *Javascript*

```
<html> <head>
<title>Cube form</title>
<script language = "JavaScript">
function cube()
{ var x =
    parseInt(document.form1.field1.value);
  document.writeln("<h1>The cube is: " +
                    x*x*x + "</h1>");
}
</script>
</head>
<body>
<form name = "form1" action = ""><p>
<strong>Enter integer to be cubed:</strong>
<input name = "field1" type = "text"></p>
```

```
<p>  
<input type = "button"  
  value = "Cube" onclick = "cube()"></p>  
</form>  
</body></html>
```

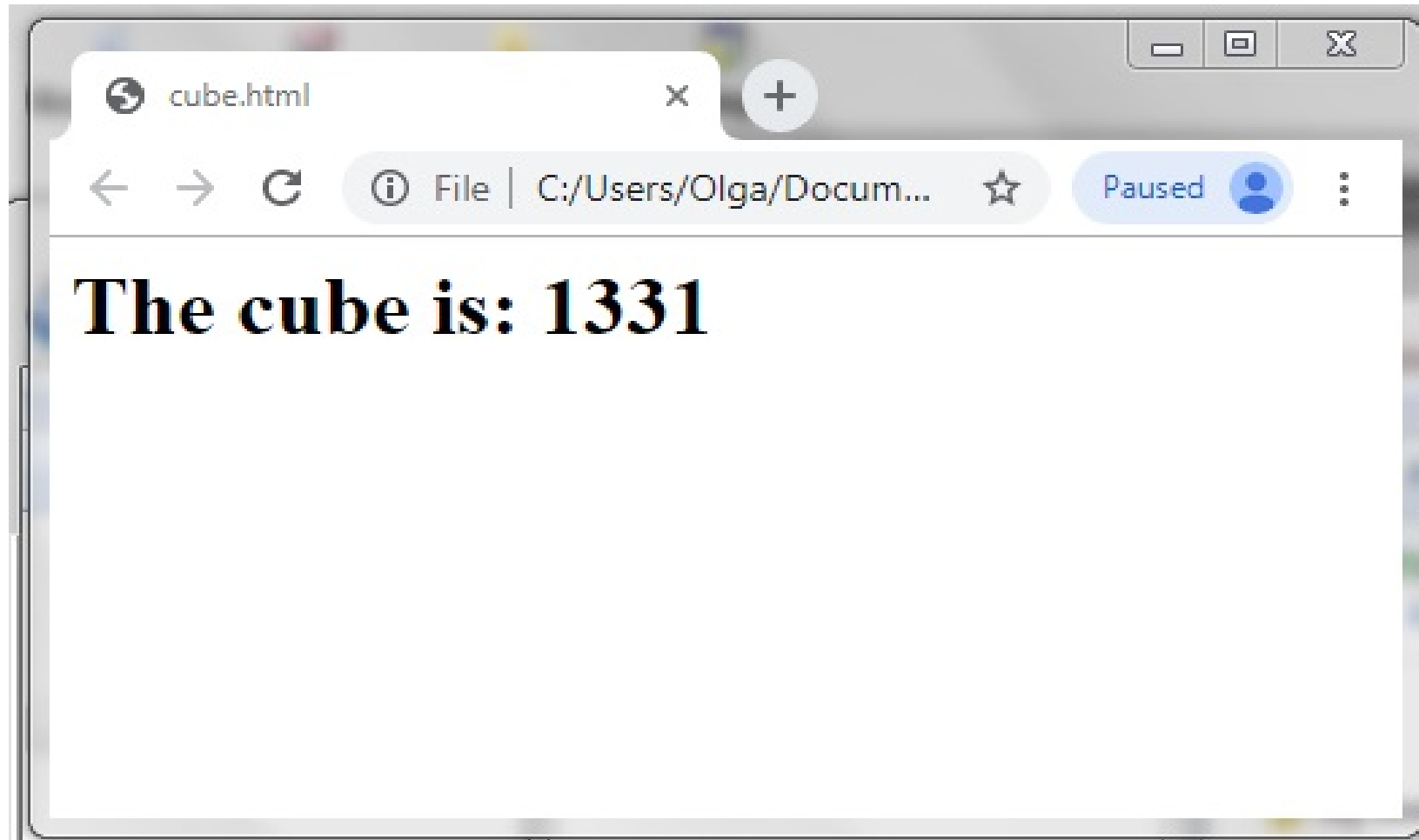
Reads value from form field when button is pressed, converts it to integer  $x$ , then prints cube  $x * x * x$  to page.

Here, the code is defined inside a function *cube()*.



Cube example





Cube example result

## *Javascript*

```
<html> <head>
<title>Factorial form</title>
<script language = "JavaScript">
function factorial()
{ var x =
    parseInt(document.form1.field1.value);
    document.writeln("<h1>The factorial is: " + fact(x) + "</h1>");
}

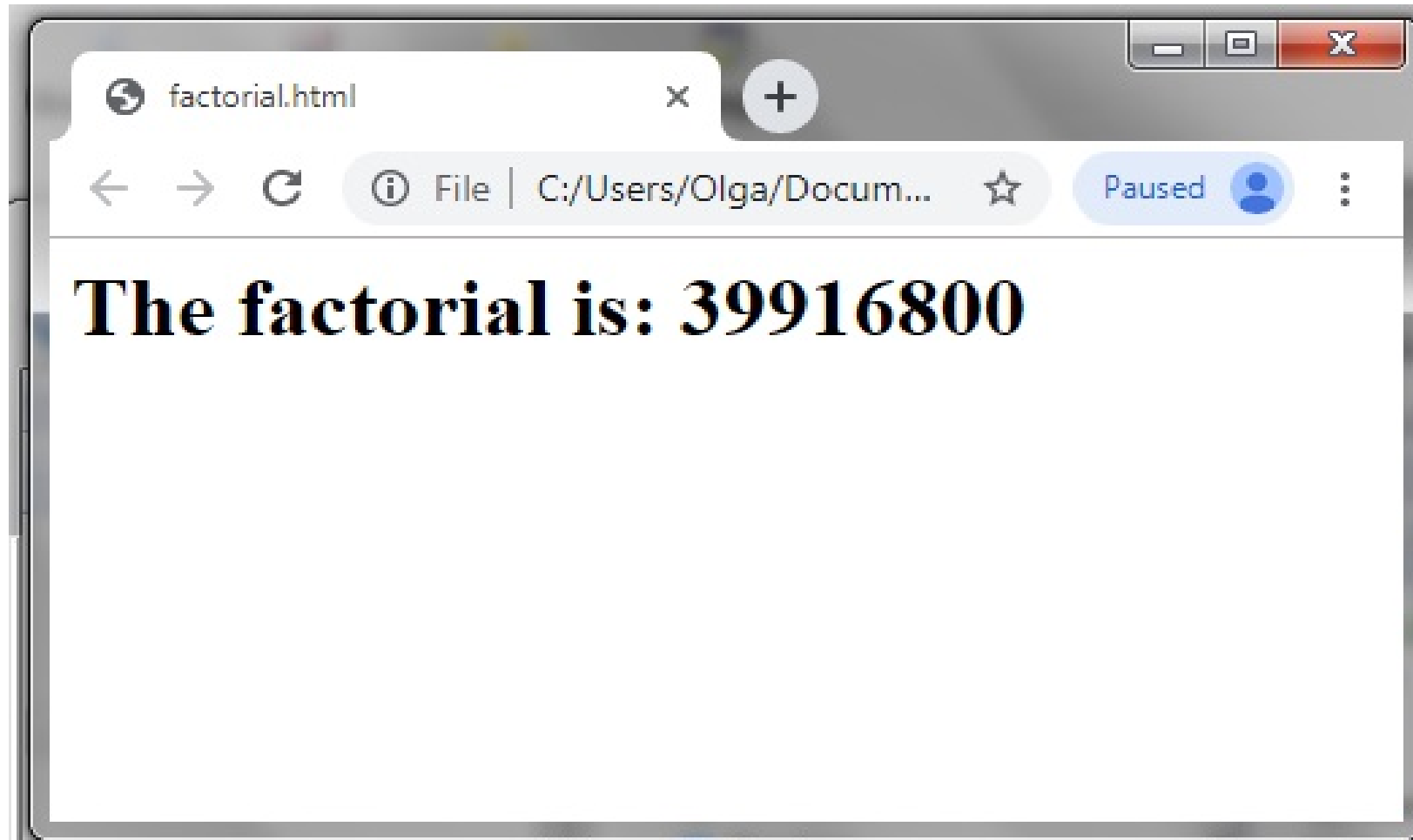
function fact(x)
{ if (x <= 1)
    { return 1; }
  return x*fact(x-1);
}
</script>
```

```
</head>

<body>
<form name = "form1" action = "">
<p><strong>Enter integer:</strong>
<input name = "field1" type = "text"></p>
<p><input type = "button"
  value = "Factorial" onclick = "factorial()"></p>
</form>
</body></html>
```

Reads form field value, calculates its factorial and writes back to the page.

Here there are 2 functions, one is recursive – calling itself in a loop.



Factorial example

## *Javascript*

Alternative approach is to update one part of form to show the result:

```
<html> <head>
<title>Cube form</title>
<script language = "JavaScript">
function cube()
{ var x =
    parseInt(document.form1.field1.value);
  document.form1.field2.value = "" + x*x*x;
}
</script>
</head>

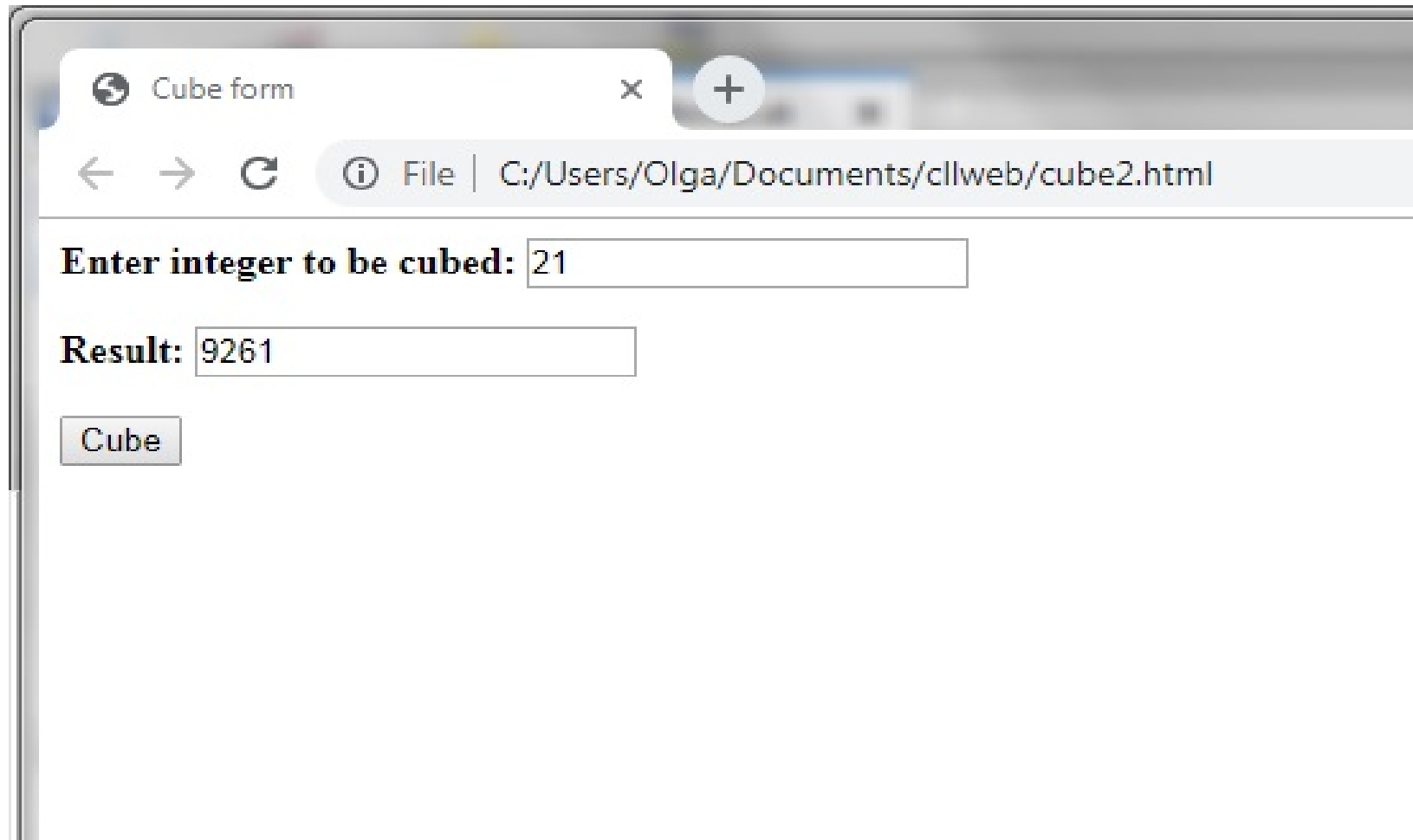
<body>
<form name = "form1"
```

```
    action = "">

<p>
<strong>Enter integer to be cubed:</strong>
<input name = "field1" type = "text"></p>

<strong>Result:</strong>
<input name = "field2" type = "text"></p>

<p>
<input type = "button"
  value = "Cube" onclick = "cube()"></p>
</form>
</body></html>
```



Cube with partial page updates

## *Javascript*

Or to write to *innerHTML* (content of) a specific element:

```
<html> <head>
<title>Factorial form</title>
<script language = "JavaScript">
function factorial()
{ var x =
    parseInt(document.form1.field1.value);
  document.getElementById("field2").innerHTML = "" + fact(x);
}

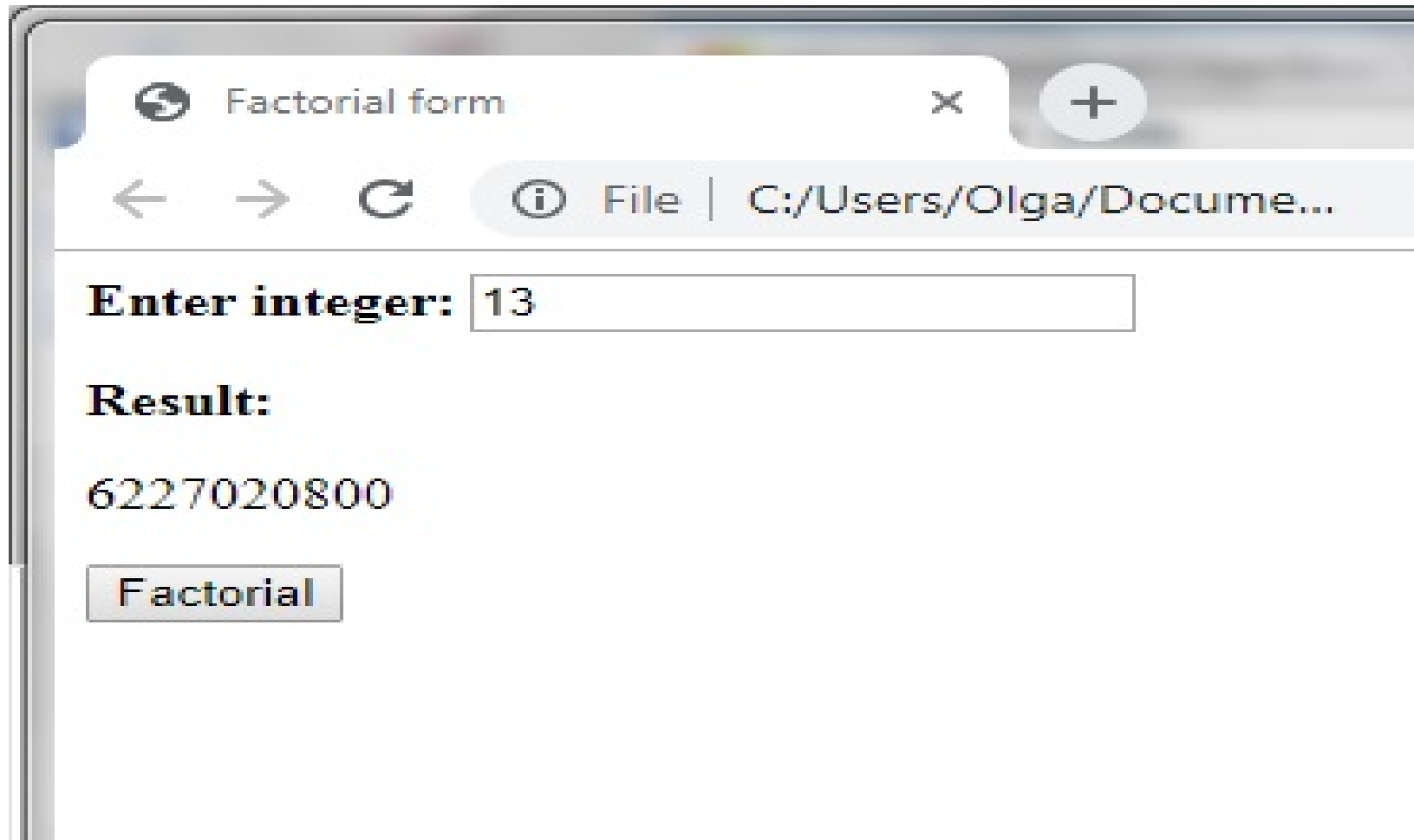
function fact(x)
{ if (x <= 1)
  { return 1; }
  return x*fact(x-1);
}
```



```
</script>
</head>

<body>
<form name = "form1" action = "">

<p>
<strong>Enter integer:</strong>
<input name = "field1" type = "text"></p>
<strong>Result:</strong>
<p id = "field2"></p>
<p><input type = "button"
  value = "Factorial" onclick = "factorial()"></p>
</form>
</body></html>
```



Factorial with partial page update

## *Javascript*

Javascript can also be defined in separate *.js* files and included in the HTML:

```
function factorial()
{ var x =
    parseInt(document.form1.field1.value);
  document.getElementById("field2").innerHTML = "" + fact(x);
}

function fact(x)
{ if (x <= 1)
  { return 1; }
  return x*fact(x-1);
}
```

## *Javascript*

The *.js* file is linked in the *script* tag:

```
<html> <head>  
<title>Factorial form</title>  
<script language = "JavaScript" src="factorial3js.js">  
</script>  
</head>
```

```
<body>  
<form name = "form1" action = ""><p>  
<strong>Enter integer:</strong>  
<input name = "field1" type = "text"></p>  
<strong>Result:</strong>  
<p id = "field2"></p>  
  
<p><input type = "button"
```

```
value = "Factorial" onclick = "factorial()"></p>  
</form>  
</body></html>
```

The same script file can be linked to from several HTML files.

### *Further resources*

We hope you have enjoyed this course. The following are useful for further study:

- HTML reference: <https://www.w3schools.com>
- Course resources:  
<https://nms.kcl.ac.uk/kevin.lano/c11wd>
- User-centered design: John Cato “User-centered web design”, Addison-Wesley.