

HTML II & More Git

Programming with
Web Technologies



Auckland
ICT Graduate School

Last time...

- Course introduction
- Intro to HTML
- Forking & cloning from GitLab

Today

More useful HTML tags to build on what we already know

- Tables
- Attributes
- Anchors
- Multimedia

More on git

- Branching & merging

More HTML



Auckland
ICT Graduate School

Tables

- Often we are given data that makes sense to present in a tabular fashion
- Tables are very flexible and have been used for page layout in HTML document for many years
 - No longer recommended as it distracts from the purpose of a table – displaying tabular data
- Trend with HTML5 is to avoid structures like tables for layout, and instead leave that to CSS
 - More on this in future lectures
- Tables make use of a collection of HTML tags

Tables

- `<table>` - Defines an HTML table, and groups the components of the table together
- `<tr>` - A row in the table that contains a number of cells
- `<td>` - A data cell in the table
- `<th>` - A header cell in the table
- `<thead>` - Table header, represents the top-most row of a table
- `<tbody>` - Table body, the main content of a table
- `<tfoot>` - Table footer, represents the bottom-most row of a table
- `<caption>` - A caption to attach or associate with the table, must appear immediately after the opening `<table>` tag

A basic table

```
<table>
  <tr>
    <td>col1row1</td>
    <td>col2row1</td>
    <td>col3row1</td>
    <td>col4row1</td>
  </tr>
  <tr>
    <td>col1row2</td>
    <td>col2row2</td>
    <td>col3row2</td>
    <td>col4row2</td>
  </tr>
</table>
```

| | | | |
|----------|----------|----------|----------|
| col1row1 | col2row1 | col3row1 | col4row1 |
| col1row2 | col2row2 | col3row2 | col4row2 |

Basic table observations

The table will have as many rows as there are `<tr>` elements

The table will have as many columns as the largest number of `<td>` elements inside a `<tr>`

- If one row has five cells, and any others have 3 cells each, the table will have five columns
- Cells are placed left-to-right

Intermediate table

`<table>`



`</table>`

Intermediate table

```
<table>  
  <caption>An example table</caption>
```

An example table

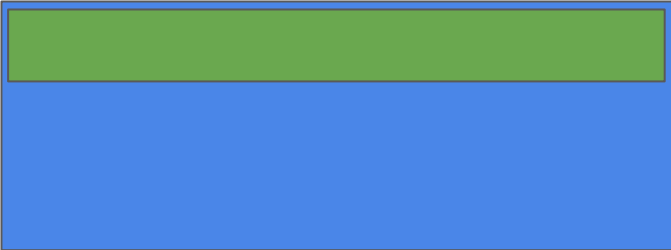


```
</table>
```

Intermediate table

```
<table>  
  <caption>An example table</caption>  
  <thead>  
    <tr>  
  
    </tr>  
  </thead>
```

An example table



||
||
||

```
</table>
```

Intermediate table

```
<table>
  <caption>An example table</caption>
  <thead>
    <tr>
      <th>header 1</th>
      <th>header 2</th>
    </tr>
  </thead>
```

```
</table>
```

An example table

| header 1 | header 2 |
|----------|----------|
|----------|----------|

Intermediate table

```
<table>  
  <caption>An example table</caption>  
  <thead>  
    <tr>  
      <th>header 1</th>  
      <th>header 2</th>  
    </tr>  
  </thead>  
  <tfoot>  
    <tr>  
  
    </tr>  
  </tfoot>  
</table>
```

An example table

| header 1 | header 2 |
|----------|----------|
| | |
| | |

```
</table>
```

Intermediate table

```
<table>
  <caption>An example table</caption>
  <thead>
    <tr>
      <th>header 1</th>
      <th>header 2</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td>footer 1</td>
      <td>footer 2</td>
    </tr>
  </tfoot>
</table>
```

An example table

| header 1 | header 2 |
|----------|----------|
| | |
| footer 1 | footer 2 |

```
</table>
```

Intermediate table

```
<table>
  <caption>An example table</caption>
  <thead>
    <tr>
      <th>header 1</th>
      <th>header 2</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td>footer 1</td>
      <td>footer 2</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>

  </tr>
</tbody>
</table>
```

An example table

| header 1 | header 2 |
|----------|----------|
| | |
| footer 1 | footer 2 |

Intermediate table

```
<table>
  <caption>An example table</caption>
  <thead>
    <tr>
      <th>header 1</th>
      <th>header 2</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td>footer 1</td>
      <td>footer 2</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>data 1</td>
      <td>data 2</td>
    </tr>
  </tbody>
</table>
```

An example table

| header 1 | header 2 |
|----------|----------|
| data 1 | data 2 |
| footer 1 | footer 2 |

Intermediate table observations

- The order of `<tbody>`, `<thead>` and `<tfoot>` inside the `<table>` does not matter
 - `<thead>` will always be rendered as the top-most row in the table, above the `<tbody>`
 - `<tfoot>` will always be rendered as the bottom-most row in the table, below the `<tbody>`
- The order of `<tr>` elements does matter. They will be displayed in the order in which they are declared in their respective sections

HTML attributes

Attributes are name/value pairs that are seen inside the opening tags of HTML elements. We have seen some of these already

```
<html lang="en">
```

```
<meta charset="UTF-8">
```

All HTML elements can have attributes, some just have more than others

HTML attributes

Attributes are used to provide additional information about an element and come in 4 major varieties

- Required – The tag needs these attributes in order to function correctly
- Optional – The tag does not require these attributes to function, but useful functionality can be added using them
- Standard – Available to most tags, provide general functionality
- Event – Will discuss these in future lectures

HTML attributes

- Tables
 - `colspan="2"` – Makes a `<td>` span across multiple columns (merging cells horizontally)
 - `rowspan="2"` – Makes a `<td>` span across multiple rows (merging cells vertically)
- Ordered lists
 - `reversed` – Makes the list numbers count down rather than up (this attribute does not need a value)
 - `type="i"` – Pick the variety of marker to use in the list (roman numerals, alphabet, etc)
- A large variety of attributes exist, and many apply to specific tags
 - Many and difficult to remember options
 - To find the available attributes for a tag, use the [MDN tag reference](#)
 - Locate the tag you are interested in and check the "Attributes" section

Anchors

Also known as links, or hyperlinks, anchors are what allow us to navigate easily from page to page

Generally [look something like this](#), but can be styled to look different

Can link to external pages, or to sections within the same page

Anchor tags

`<a>` - The anchor tag. Marks the content surrounded by the tags as a link. The content can be text or even an image.

Anchors have a required attribute `href` which indicates the address of the resource they link to.

```
<a href="http://www.google.com">Google</a>
```

[Google](http://www.google.com)

href attribute - external

Absolute Address

<http://www.fakewebsite.com/res/2017.html>

Equivalent to specifying a street address in long-form

130 Victoria St

Hamilton

New Zealand

Indicates exactly where to find the page and how to get there

href attribute - external

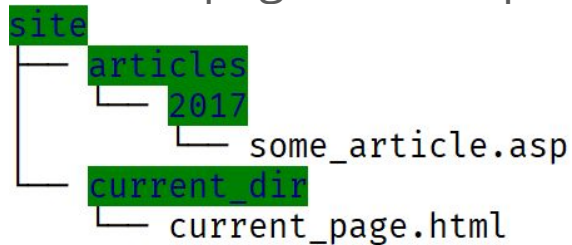
Relative Address

[../articles/2017/some_article.asp](#)

Equivalent to specifying a nearby street address

Go back up the road and take the next right, you will find it at number 17.

Indicates exactly where to find the page with respect to the current page



href attribute - external

Local Link

[something.jsp](#)

Equivalent to sending someone to another house on the same street

You want number 20, right next door

Indicates exactly where to find the page with respect to the current page

```
current_dir  
├─ current_page.html  
└─ something.jsp
```

href attribute - bookmark

A bookmark is an anchor that jumps to a specific part of a web page

You will have encountered one of these if you have used the contents of a Wikipedia page

Bookmarks are created by setting the `id` attribute of an element

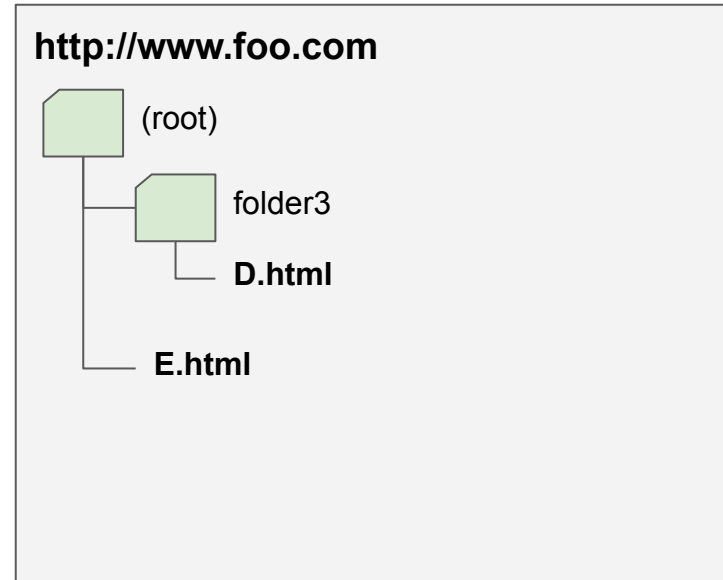
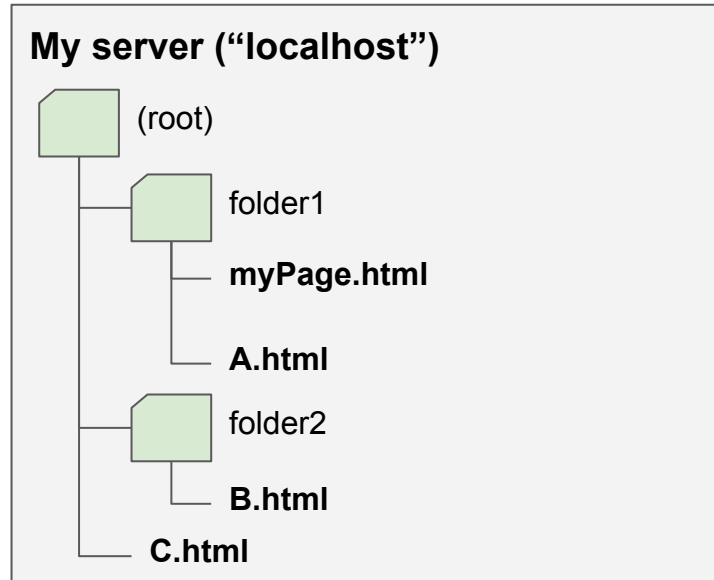
```
<h2 id="example_bm">A heading</h2>
```

Anchors can then link to the bookmark

```
<a href="#example_bm">Link to heading</a>
```

Quiz

- Within myPage.html below, what would be the value of the href attribute required to link to each of the other html pages in the diagram?



Images

`` - The image element. This tag does not need to be closed

Images have a required attribute `src` which, like the `href` attribute used with anchors, indicates where the image can be found.

```

```



THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Images - attributes

```

```



```

```



```

```



Images - attributes

What if the image file isn't available, or the user is visually impaired (using a screen reader)?

The `alt` attribute provides a textual description

```

```



THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

U of W logo

Images - file types

Browsers usually support: GIFs, PNGs, JPEGs/JPGs, BMP and ICO; but not TIFF, JPEG2000, DjVu, etc

JPEGs are a **Lossy** format, with the quality settings set in software

Choose JPEGs for photos and photo-like images

GIFs & PNGs are **Loss-less** formats, PNG usually produces smaller file sizes

Choose GIFs or PNGs for flat colour, graphic text, simple logos and screenshots. Choose PNG if you want transparency

GIFs can also be animated, PNG animation (APNG, MNG) is not supported by browsers, so simple image animation on the Web has to be GIF

Images - more points

You can resize a large image to fit in a smaller area using the width and height attributes, however this is not always wise

You are still downloading a large photo, which can be slow

Sometimes you are better off creating a smaller "thumbnail" version of the image and linking to the larger version

```
<a href="fullsize_image.png">  
    
</a>
```


Multimedia - audio and video

`<audio>` - A container for audio files. The audio tag represents the controls, or player for audio-only files

`<video>` - A container for video files. The video tag represents the controls, or player for audio/visual files

`<source>` - Represents a possible source location for audio or video files. Source elements should be placed inside the appropriate `<audio>` or `<video>` tags

Multimedia - using the tags

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

```
<video controls width="640" height="480">  
  <source src="myvideo.mp4" type="video/mp4">  
  Your browser does not support the video element.  
</video>
```

Why multiple `<source>`'s

Not all browsers across the operating systems support the same audio and video formats

The browser will try to play each source until it finds one that it can handle.

By providing multiple options, we can hopefully get the media to play for everyone

If all else fails, we can display an error message

Multimedia - patents

Variability of codec support in browsers result of patents (and "politics" to some extent)

Audio

MP3 Support fairly widespread in browsers now

Video

Video codec support more variable than audio

Branching & Merging with Git



Auckland
ICT Graduate School

Basic git workflow (from last time)

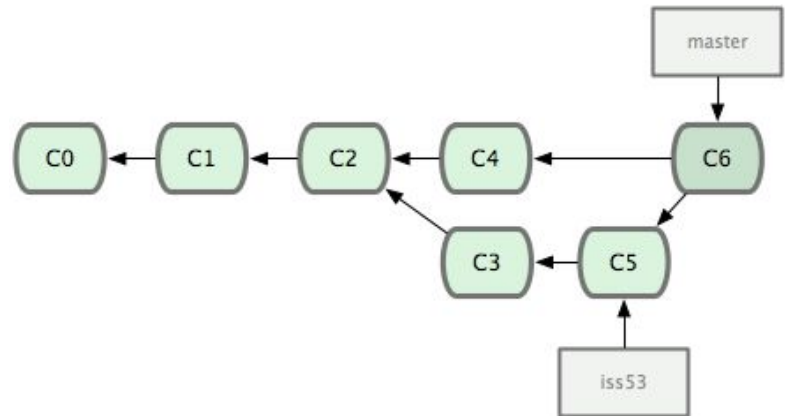
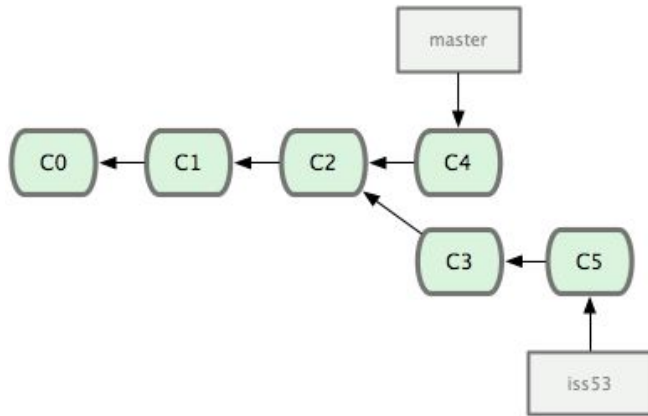
1. Initialize a repository with `git init`, or obtain an existing one with `git clone`
2. Create or modify files
3. Track and stage new files or changes using `git add`
4. Commit your changes with `git commit`
5. Push your changes back to your hosting remote with `git push`
6. Loop back to #2

Branches

- Imagine that you have been given a large task to do in a code project
 - This will take several days, and your code may not compile at some points
 - You need to do other smaller tasks at the same time
 - You need to be able to produce a running solution at any moment, even if it doesn't run completely correctly
- Maybe make a copy of your project and do the big change there, and do the small always-compiling work on the original
 - We have effectively separated our tasks, at the expense of some disk space
 - But when we are done with the big task, how do we get it back into the original project
 - Manual copy and paste? How can you make sure you copy all of the code?
- Can git give us a better option?

Branches

Branches allow you to diverge from your 'mainline' development and make changes that don't affect that mainline, while maintaining all history and relationships. This allows you to 'merge' these changes together easily at a later date



Creating and switching branches

- `git branch`
 - List all branches, and indicate the currently checked-out branch
- `git branch <branch-name>`
 - Create a new branch from the currently checked out commit, with the specified `branch-name`
- `git checkout <branch-name>`
 - Switch to the branch specified by `branch-name`. You will need to commit (or stash) any unstaged changes in the current branch before you do this
- `git checkout -b <branch-name>`
 - Create a new branch from the currently checked out commit, with the specified `branch-name`, and immediately switch to it