KUAS
KYOTO UNIVERSITY OF ADVANCED SCIENCE
京都発世界人財

京都先端科学大学

Information Literacy 13

Web Content

HTML, CSS, and a teeny bit of JavaScript

Ian Piumarta

Faculty of Engineering, KUAS

Web pages contain four kinds of information

the words, images, and other actual content

the structure of the document: sections, tables, ...

its visual appearance and layout: fonts, colours, ...

behaviour: how the page responds to user input

plain text plus markup equals Web page

content: text and media

- plain text
- media (images, audio, video)

```
Welcome to
The Wonderful World
of Kittens web site!
<img src="cat666.jpg"/>
```

structure: HTML markup

- headings, paragraphs
- lists, tables, etc.
- hyperlinks

```
<h1>Introduction</h1>
<p>Cats are awesome.</p>
<ul>
  <li>They're furry.</li>
  <li>They purr.</li>
</ul>
<p>How <i>cute</i> is
that?</p>
```

style affects appearance, scripts add behaviour

appearance: CSS style

- fonts, colours
- decorations
- page layout

```
h1 {
    font-size:    200%;
    font-weight:  bold;
}
table {
    margin:      10pt;
    background:  #eee;
}
```

behaviour: JS scripts

- UI behaviour
- dynamic content

```
<button onclick="alert('Thanks!')">
  Click me!
</button>
```

pairs of tags create a tree of document elements

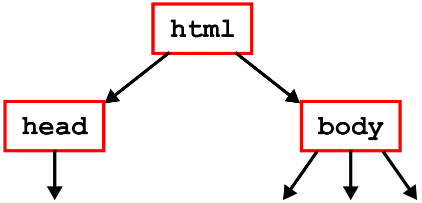
a page is a *tree* of elements (just like a file system)

a *start tag* begins an element

an *end tag* ends an element

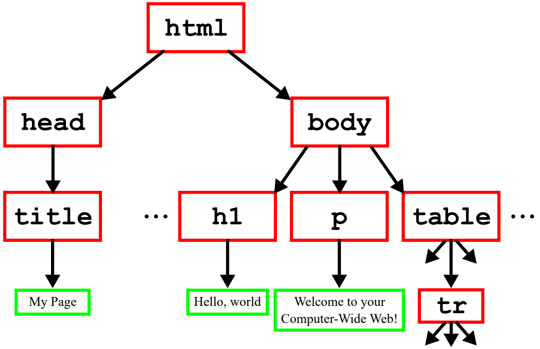
everything between the start and end tags are its children

```
<html>    ← start tag of html
    <head>   ← start tag of head
        ...
    </head>  ← end tag of head
    <body>   ← start tag of body
        ... ... ...
    </body>  ← end tag of body
</html>  ← end tag of html
```



anatomy of a web page

```
<!DOCTYPE html>    ← says what the file contains
<html>             ← root of the HTML document
  <head>           ← meta data
    information
    about the document
  </head>
  <body>           ← content
    text, media,
    and markup
  </body>
</html>
```



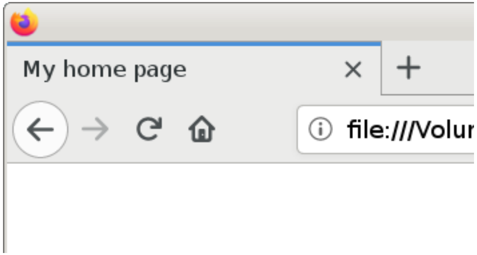
the `head` element contains metadata

metadata describes the document itself

it is not part of the page content

e.g., the title of the document (shown in the browser tab)

```
<head>
  <title>My home page</title>
</head>
```


My home page

file:///Volur

the body element contains content

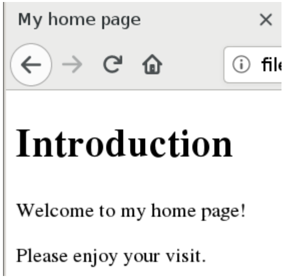
everything shown in the main part of the browser window

`h1` to `h6` make headings

`p` makes a paragraph

- all normal text should be enclosed in `p` elements

```
<body>
  <h1>Introduction</h1>
  <p>Welcome to my home page!</p>
  <p>
    Please enjoy your visit.
  </p>
</body>
```



Introduction

Welcome to my home page!

Please enjoy your visit.

inline content is anything that flows like text

small images also flow like text

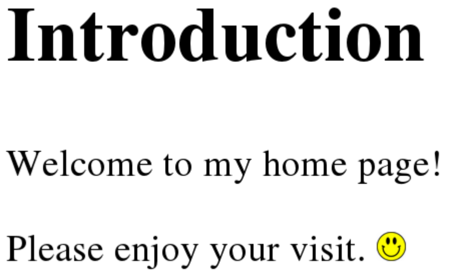
you can (and *should*) put `img` elements inside a `p` element

note: file URL of image is specified as `src` *attribute*

attributes add *non-content* information to elements

```
<p>
  Please enjoy your visit.
  <img src="smiley.jpg"
       height=12pt>
</p>
```

note: attribute values should always be quoted



Introduction

Welcome to my home page!

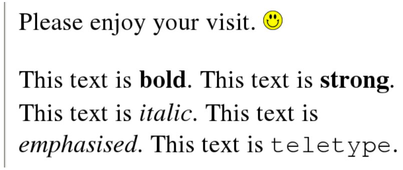
Please enjoy your visit.

simple style-changing elements appear inline too

place text in these elements to change its style

- bold: `b`, `strong` (usually bold)
- italic: `i`, `em` (emphasised, usually italic)
- teletype: `tt` (fixed-width font)

```
<p>
  This text is <b>bold</b>.
  This text is <strong>strong</strong>.
  This text is <i>italic</i>.
  This text is <em>emphasised</em>.
  This text is <tt>teletype</tt>.
</p>
```


Please enjoy your visit. ☺

This text is **bold**. This text is **strong**. This text is *italic*. This text is *emphasised*. This text is `teletype`.

hyperlinks (to other documents) are also inline

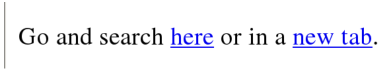
the `a` element (for ‘anchor’) creates hyperlinks

the children are the visible ‘label’ of the link

the destination URL is given by its `href` attribute

to open in a new window, add `target="_blank"`

```
<p>
  Go and search
  <a href="http://google.com">here</a>
  or in a
  <a href="http://google.com" target="_blank">new tab</a>.
</p>
```

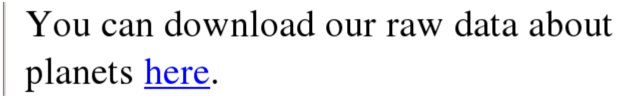

Go and search here or in a new tab.

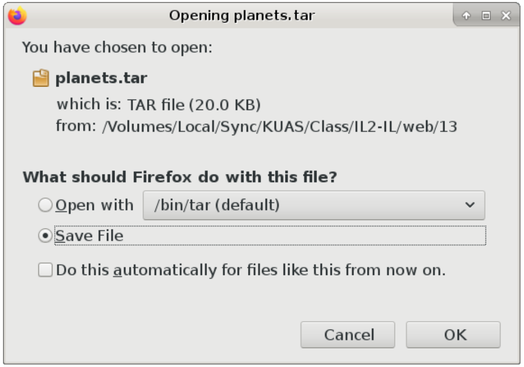
a Web page is great for explaining and distributing data

use an `a` element to make a hyperlink to a local file

```
<p>
  You can download our raw data about planets
  <a href="planets.tar">here</a>.
</p>
```

click link → download file


You can download our raw data about planets here.


Opening planets.tar

You have chosen to open:

planets.tar

which is: TAR file (20.0 KB)
from: /Volumes/Local/Sync/KUAS/Class/IL2-IL/web/13

What should Firefox do with this file?

Open with /bin/tar (default)

[✓ Save File]

Do this automatically for files like this from now on.

explore making a download for yourself in one minute

create a file of data to be downloaded (the content is irrelevant)

```
echo "irrelevant" > data-download.txt
```

add this to your `index.html`:

```
<html>
  <body>
    <p>
      Please download this fascinating
      <a href="data-download.txt">data</a>!
    </p>
  </body>
</html>
```

browse!

audio and video content are easy to add

dedicated `audio` and `video` elements

one or more child `source` elements specify potential sources

a mini-player will be displayed if possible

otherwise the element content is displayed

```
<video width="320" height="240" controls>
  <source src="planets-animation.mp4" type="video/mp4">
  <source src="planets-animation.ogg" type="video/ogg">
  Sorry, your browser does not support video.
</video>
```

explore these elements yourself if you want to share media

block content goes outside paragraphs

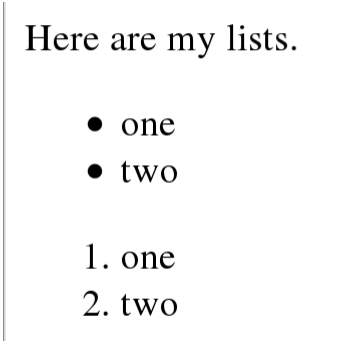
we already saw `h1` and `p` which go outside paragraphs

lists are also block-level elements

`ul` uses bullets, `ol` is numbered

each list item is a `li` element

```
<p>Here are my lists.</p>
<ul>
  <li>one</li>
  <li>two</li>
</ul>
<ol>
  <li>one</li>
  <li>two</li>
</ol>
```



Here are my lists.

- one
- two

1. one
2. two

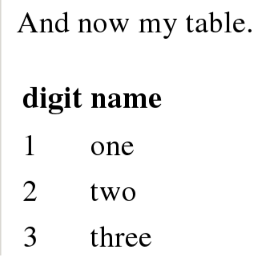
tables are also block-level elements

the `table` element contains `tr` row elements

each `tr` element contains `td` or `th` elements

- `td` are data: actual table cells
- `th` are column headers (usually bold)

```
<p>And now my table.</p>
<table>
  <tr><th>digit</th><th>name </th></tr>
  <tr><td>1    </td><td>one  </td></tr>
  <tr><td>2    </td><td>two  </td></tr>
  <tr><td>3    </td><td>three</td></tr>
</table>
```



And now my table.

digit	**name**
1	one
2	two
3	three

style (CSS) changes the appearance of elements

`style` elements can be placed anywhere

- inside the `head` is a good place

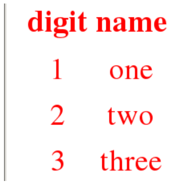
the `style` element contains declaration blocks

`selector` `{` *`property`* `:` *`value`* `; }`

the simplest *selector* is the name of an element type

```
<head>
  <style>
    table { color : red; }
    td    { text-align : center; }
  </style>
</head>
```

↓ *make all `tables` red and all `tds` centred*



digit	name
1	one
2	two
3	three

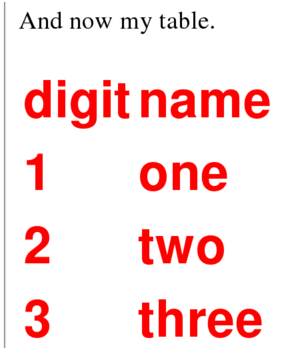
you can put multiple style declarations in a block

each declaration inside `{ ... }` looks like this:

```
property : value ;
```

for example:

```
table {
   color:       red;
   font:        sans-serif;
   font-weight: bold;
   font-size:   200%;
}
```



And now my table.

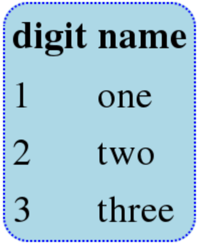
digit	name
1	one
2	two
3	three

some style properties need multiple values

the `border` property value has several parts:

width	`2px` (pixels)
style	`solid`, `dotted`, `dashed`, etc.
colour	`black`, `red`, etc.

```
table {
  border: 1px dotted blue;
  border-radius: 10px;
  background: lightblue;
}
```



digit	name
1	one
2	two
3	three

you can apply style to a single element

one way is to set the `style` attribute

value is a declaration (as within a `style` element)

```
<p style="color: red;">This paragraph is red!</p>
```

another way is to give the element an *identifier* (name)

```
<style> ↓ this declaration applies to the element with identifier "special"
  #special { color: red; }
</style>

<p id="special">This paragraph is red!</p>
    ↑ this attribute sets the identifier of this element to "special"
```

you can apply style to multiple elements

an element can have only one name, but...

an element can belong to multiple *classes*

```
<style> ↓ these declarations create classes to which elements can belong
   .coloured { color: blue; }
   .slanted  { font-style: italic; }
   .large    { font-size: 120%; }
</style>

<p class="coloured slanted">Italic blue!</p>
```

↑ this attribute specifies one or more classes to which the element belongs; style from all these classes are applied to the element

there are many other style properties for you to explore

`margin`	adds space outside an element
`padding`	adds space inside an element
`background`	background colours or images
`float`	makes text flow around the element

a lot of information is available online

```
https://www.w3schools.com/tags/default.asp
https://www.w3schools.com/cssref/default.asp
```

you can put comments in HTML and CSS

a comment in HTML looks like this

```
<!-- your comment goes here -->
```

your comment can include newlines

a comment in CSS (inside a `style` element) looks like this

```
/* your comment goes here */
```

your comment can include newlines

if you know SGML then the HTML syntax will be familiar, and C programmers will recognise the CSS syntax

assignment: overview

(1) create a mini ‘Web site’ about any topic you are interested in

(2) include several specified elements (at least once) in the main page

(3) create a .zip file containing the site

(4) turn in your zipped Web site in Teams

assignment (1): create a mini Web site

make a directory whose name is your student ID

e.g., if you are 2020m999 then

```
mkdir 2020m999
```
$\leftarrow$ *use your real ID here*

create your Web site documents inside the new directory

```
cd 2020m999
nano index.html
```

your Web site can present a topic that you are interested in

assignment (2): include some required elements

include each of the following required elements at least once

- paragraphs (`p`) containing your text
- headings (`h1`, etc.) appropriate to your content
- font style change (`b` or `i` or `u` or `tt`)
- an image (`img`)
- a list (`ul` or `ol`, `li`)
- a table (`table`, `tr`, `td`)
- a hyperlink (`a`) to an *external* URL (Web navigation)
- a hyperlink (`a`) to a *local* file (a download)
- a coloured border (CSS `border` property) around any block level element

bonus point: add a table of contents (e.g., as `ul`) at the top of your page with hyperlinks that take the reader to each section when clicked (search online to learn how to do this using `a` and `id="#..."`)

assignment (3): create a .zip file of your site

make sure all data (images, downloads, etc.) are inside your Web site directory

use your browser (with or without a local http server) to verify your site is working

make a .zip archive of the entire site; either:

```
cd ..
explorer.exe .
```
$\leftarrow$ *right click on your site and create the archive in Explorer*

or:

```
cd ..
rm 2020m999.zip ..
zip -r 2020m999.zip 2020m999
```

assignment (4): turn in your Web site in Teams

we will check your Web site and award you up to 10 points

note: you can make your site as plain or as fancy as you like
all 10 points can be earned by including the required elements in a very plain site
the objective is only to create a page that shares information and data

next week

The ‘Cloud’

14 Web applications and cloud services – risks, benefits

15 Safety and security – protecting yourself and your data