HTML

Table of Contents

What is html?

html stands for *Hyper Text Markup Language*.

It is the language in which web pages are written.

It is used to say what the content of a web page is - the text and images and links.

The web page style - layout, colours, fonts and so on - not done by html. It is done using CSS, which is *Cascading Style Sheets*.

html is not a programming language. *JavaScript* is a programming language, and is used to allow web pages to run programs.

Study html, then CSS, then Javascript.

How does the World Wide Web work?

The world wide web is part of the Internet, which is a set of digital networks connected together. Other parts are things like ftp (file transfer protocol) and email (but a lot of email now is sent over the web).

A web page is a text file, containing html (and often, CSS and Javascript).

On the web, servers store web page files on their drives.

A URL (*Uniform Resource Locator*) is an address of a web server and a web page (or other type of file) on it.

Users run *web browsers* on their devices - like Firefox and Chrome and Opera and Safari.

When the user types a URL into a browser, the browser sends a message across the net to the server, asking for the web page. The server tries to find it, and if it does, it sends the file back, and the browser displays it. If it cannot find it, it sends back a web page reporting the 404 error - file not found.

Web pages also contain links, called *hyperlinks*. If the user clicks on a hyperlink, the browser again sends a request to the server, and it sends the new web page back.

Files are sent across the net using http, which is *Hyper Text Transfer Protocol*. For this reason web servers are sometimes called http servers. There are other types of servers such as email, database and application servers.

How to write html

An html file is just text. So to write it, we just need a text editor. That is software which processes pure text - no fonts, images, styles, underlining and so on. Just text only, so, not a word processor.

On Windows some options are notepad or textpad.

On Linux, nano or geany or gedit or others.

Web pages can be written on any device with a text editor, including a mobile phone.

To view the web page, we need to see it in a web browser, such as Internet Explorer or Edge or Firefox or Chrome or Safari.

One way to do that is to upload it to a *remote server*, and fetch it from there.

A simpler way is to get the web browser to open it as a *local file*. Many browsers do that by CTRL-O, then simply browse to where the file has been saved.

The first page

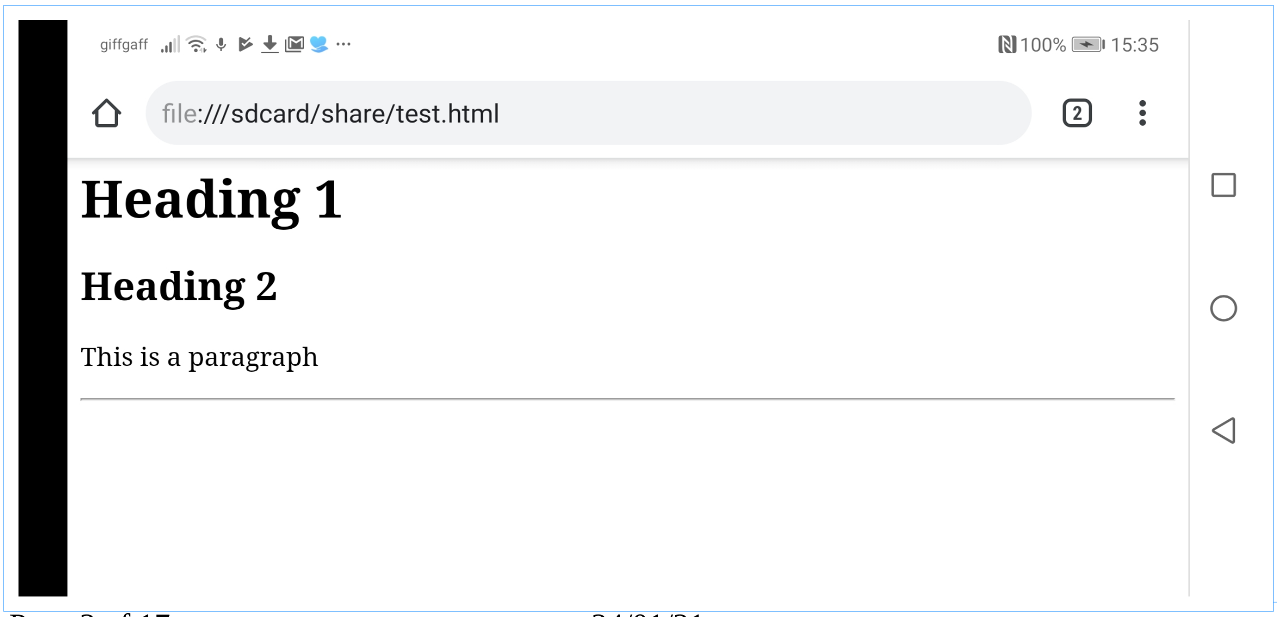
Type this into your text editor, save it, and view it in your web browser. Take care to copy it exactly. If possible, just copy and paste:

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

<head>
<title>Hello</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1">
</head>

<body>
<h1>Heading 1</h1>
<h2>Heading 2</h2>

<p>This is a paragraph</p>
<hr/>

</body>

</html>
```

Save it somewhere where it can be found again. Give it a file extension of .html - so name it something like test.html.

Load it into a web browser with Ctrl-O and browse to where it was saved.

In the browser you should see something like this:



Heading 1

Heading 2

This is a paragraph

The exact appearance depends on the browser.

Explaining the first page

An html page is made of some *elements*.

Most elements start and end with *tags.* A tag starts with <, or </ for an ending tag, and ends >

For example, the head element is <head>..</head>

The title element is <title>Hello</title>

<hr/> is a *self-closing element*. It means 'horizontal rule'

Some elements enclose other elements. Enclosing everything is the *html element* <html>..</html>

This contains the *head* element, then the *body* element.

The head element contains *meta-data*, which is information *about* the web page, such as what its title is.

The body element contains the actual web page content.

Elements like h1 and h2 are *headings*. h1 is most important, h2 less, and so on. A p is a *paragraph* element. Most text is in paragraphs.

Experiment by changing the web page with the text editor, saving it, and re-loading it into the web browser to see the results.

Versions and references

These are just brief outline notes, not full reference documents. Work through them and build basic web pages. Use the links below to see details of all aspects of html.

There have been several versions of html. The current version is html 5, which is what this text is about. All modern browsers support html 5.

The rules of html are agreed by a committee of the Worldwide Web Consortium, w3c.

Their website is

https://www.w3.org/

The most recent version is:

https://www.w3.org/TR/2017/REC-html52-20171214/

Most recent versions are not always a good idea, because older browsers may not understand them yet.

The W3C standards are technical documents. Easier versions are w3schools:

https://www.w3schools.com/html/

This is also reliable:

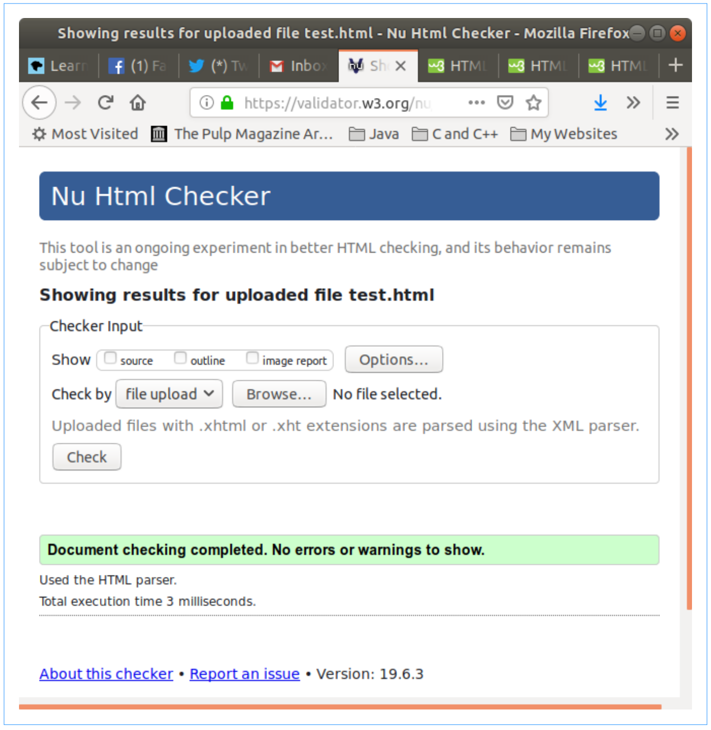
https://developer.mozilla.org/en-US/docs/Learn/HTML

Valid html

html has rules, and if the web page breaks those rules, it is *invalid*.

Even if it is invalid, browsers will still do their best to display it. But it is a good plan to make sure your html code is valid. The easiest way to do that is using a software validator, which checks code for validity.

If we run this on our web page (using file upload) we get:


Showing results for uploaded file test.html - Nu Html Checker - Mozilla Firefox
https://validator.w3.org/nu
Most Visited
The Pulp Magazine Ar...
Java
C and C++
My Websites

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for uploaded file test.html

Checker Input

Show ☐ source ☐ outline ☐ image report Options...

Check by file upload Browse... No file selected.

Uploaded files with .xhtml or .xht extensions are parsed using the XML parser.

Check

Document checking completed. No errors or warnings to show.

Used the HTML parser.

Total execution time 3 milliseconds.

About this checker • Report an issue • Version: 19.6.3

Accessibility

This means whether people with disabilities such as sight problems can use your page.

People with vision problems use *screen readers* - software which reads aloud what is on a web page. Badly written html makes screen readers go wrong.

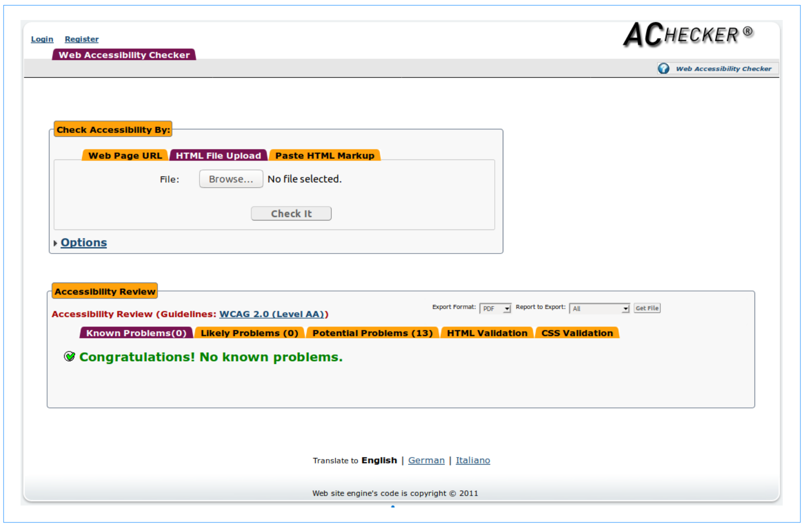
Problems are also caused by small text or low colour contrast.

In some countries websites which discriminate against groups of people, such as the disabled, are illegal.

Like html validation, there are sites which can test the accessibility of your web pages.

An example is https://achecker.ca/checker/index.php

If we run this on our page, we get:



No file selected.

Accessibility Review

Accessibility Review (Guidelines: WCAG 2.0 (Level AA))

Congratulations! No known problems.

Web site engine's code is copyright © 2011

Attributes

A web page contains elements, like a paragraph element.

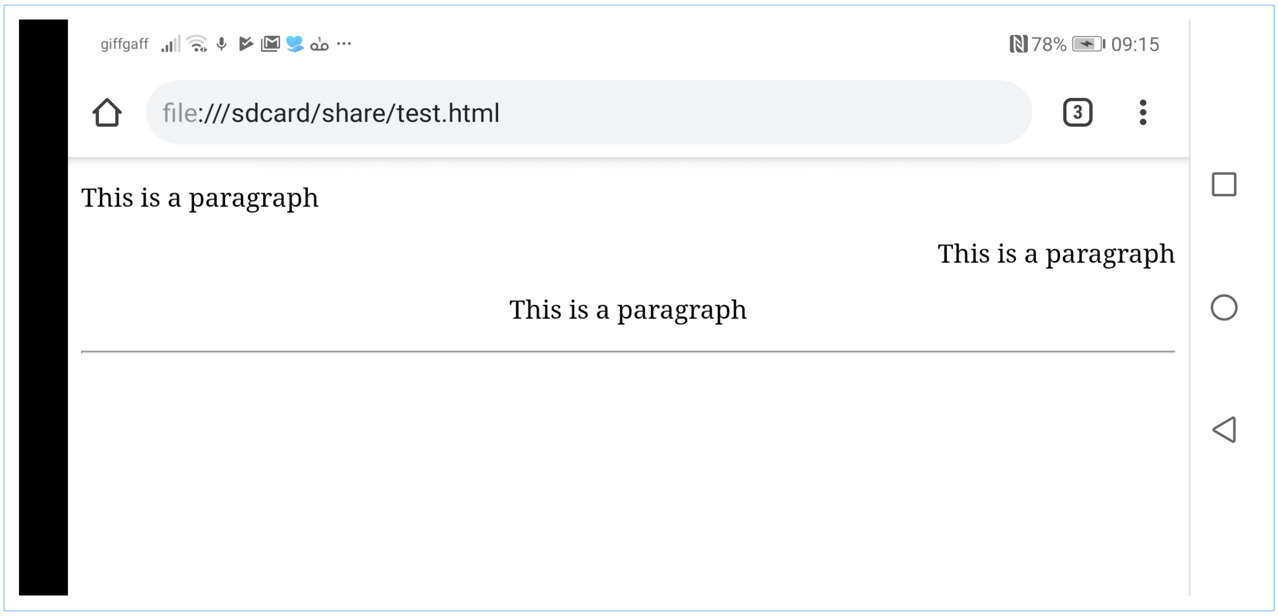
An element starts and ends with tags, like <p> and </p>

An element can also be given an *attribute* with a value. As an example, this page:

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

<head>
<title>Hello</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1">
</head>

<body>

<p align="left">This is a paragraph</p>
```

```
<p align="right">This is a paragraph</p>
<p align="center">This is a paragraph</p>
<hr/>

</body>

</html>
```

Looks like this



This is a paragraph

This is a paragraph

This is a paragraph

We can use the 'align' attribute to control whether paragraphs are left aligned, right or centred.

Which attributes are possible is different for different elements.

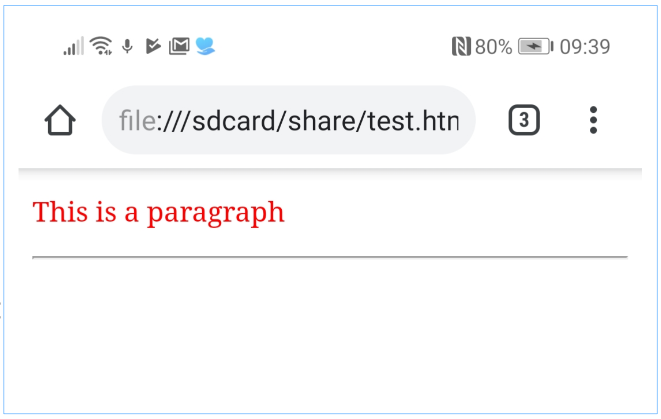
Style

One attribute value we can give to an element is called style:

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

which produces:

But setting styles one element at a time is not a good idea. If you want to change the appearance of your website this way, you must edit every element on every page, which is not practical. Instead we use CSS *style sheets*. But that is a different topic.



This is a paragraph

Styles control things like fonts, colours, text size, underline, borders, margins and so on.

If you do not set a size, the browser uses default values.

These notes are about html, not CSS. As a result they look pretty dull and boring. But html is the basic toolkit. After html, learn CSS, then Javascript.

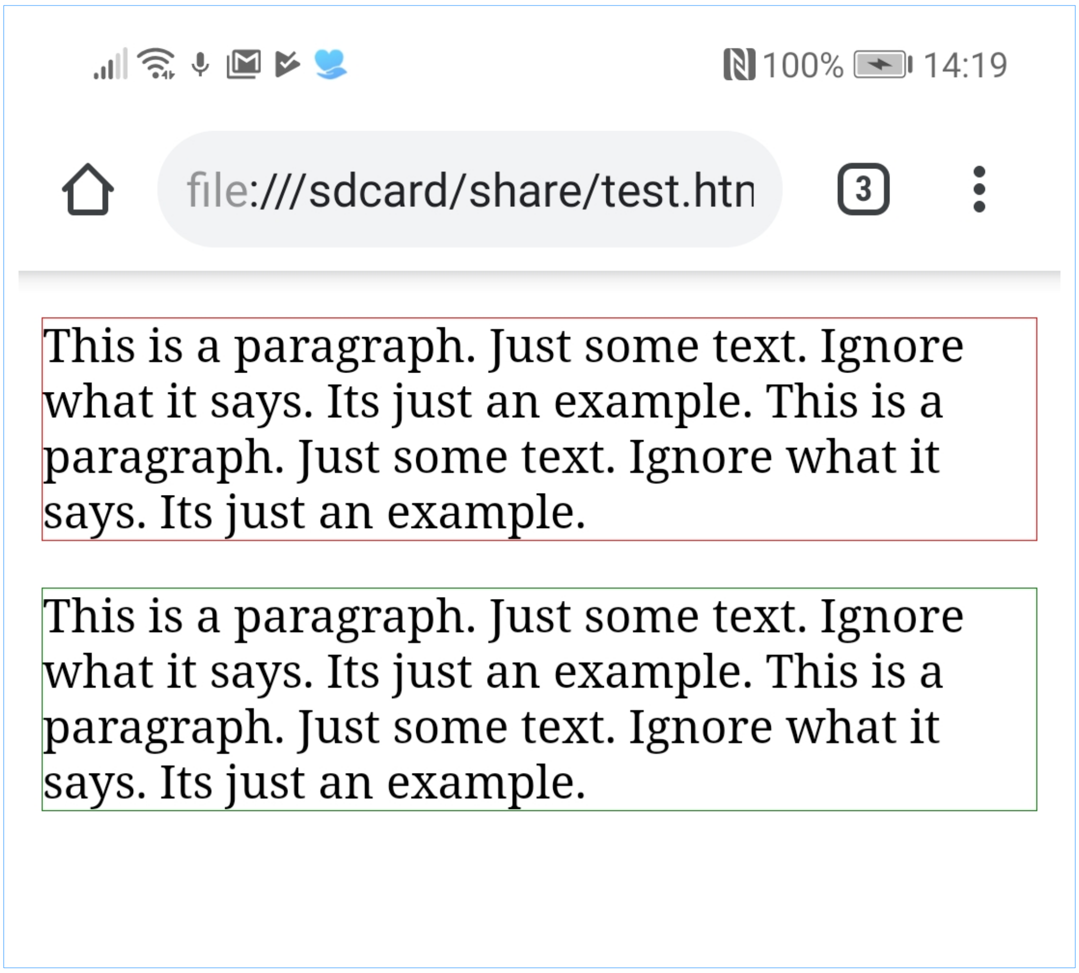
Text

The basic text element is a paragraph.

Whitespace means space characters, tabs and new lines. html strips out all of these and replaces them with a single space. For example:

```
<p style="border:thin red solid">This is a paragraph. Just some 
text.


Ignore what it says.  Its just an example. This is a paragraph.
       Just some text. Ignore what it says.  Its just an 
example.</p>

<p style="border:thin green solid">This is a paragraph. Just some 
text.
Ignore what it says.  Its just an example. This is a paragraph.
Just some text. Ignore what it says.  Its just an example.</p>
```

display as:



This is a paragraph. Just some text. Ignore what it says. Its just an example. This is a paragraph. Just some text. Ignore what it says. Its just an example.

This is a paragraph. Just some text. Ignore what it says. Its just an example. This is a paragraph. Just some text. Ignore what it says. Its just an example.

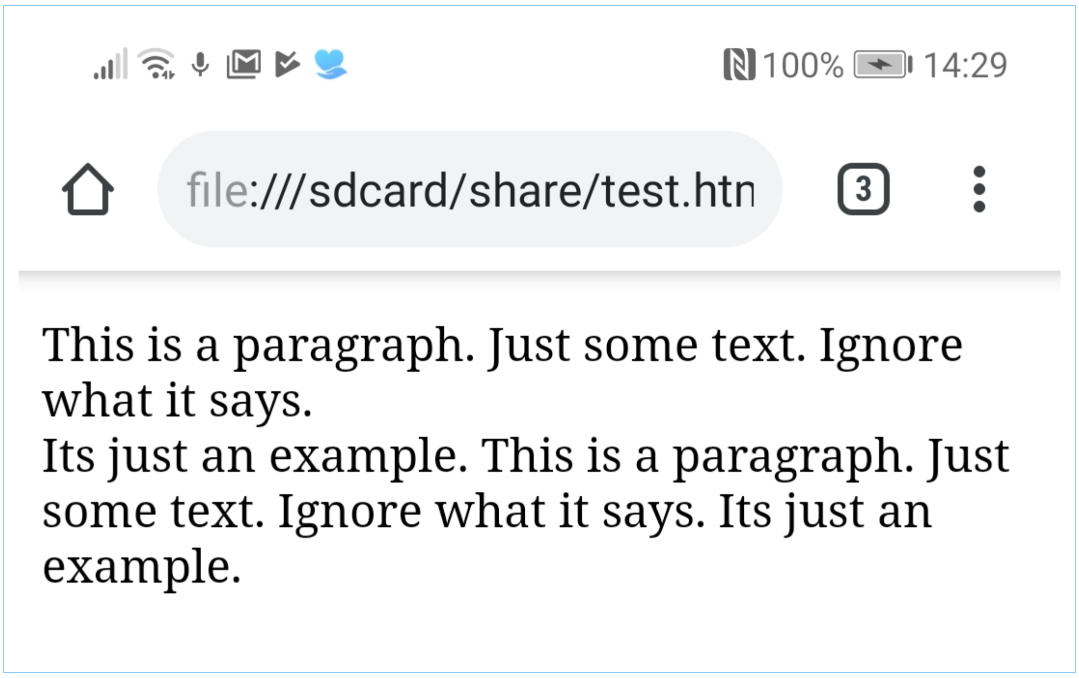
We have set the border style so that we can 'see' where the two paragraphs are. There is a space between the paragraph borders and the surrounding, which is a *margin*. We can control the size of the margin as a style setting.

Paragraphs default to displaying as *block* elements. This means a paragraph is displayed, then the browser moves down the page to the next element. This is in contrast with a *span* display, such as a link, which is displayed in-line.

As well as paragraphs there are header elements, h1 being biggest, then h2 and so on.

The browser will display text across the screen and go onto the next line when it must. We can force a new line using a *break* element, <br>

```
<p>This is a paragraph. Just some text.
Ignore what it says.<br>Its just an example. This is a paragraph.
Just some text. Ignore what it says.  Its just an example.</p>
```



100% 14:29

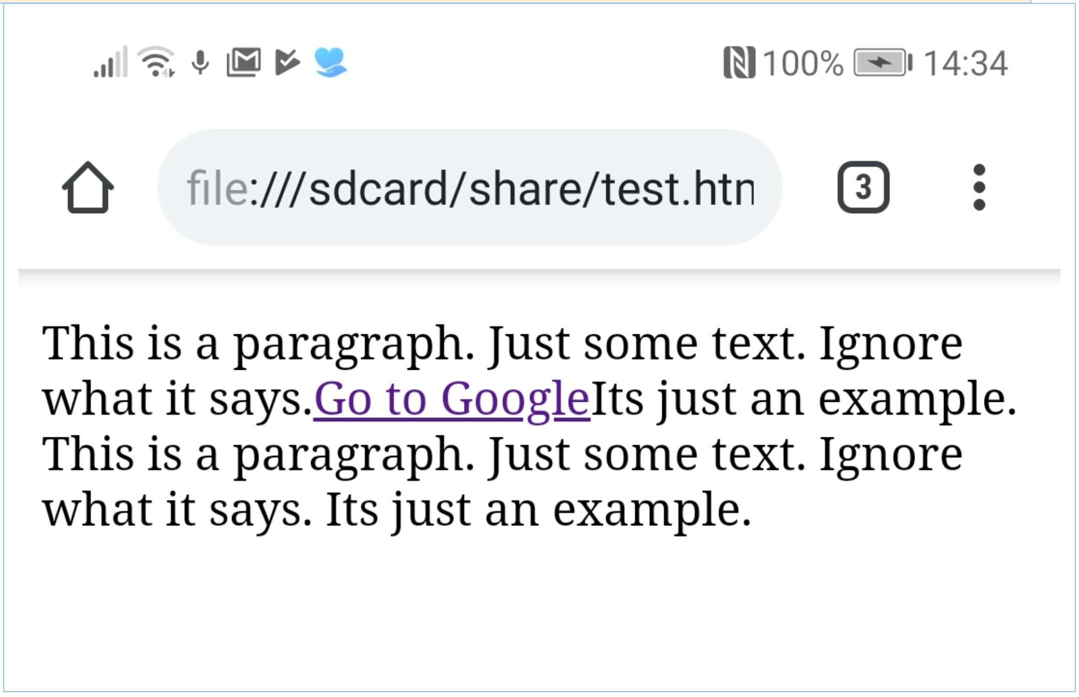
file:///sdcard/share/test.htn

This is a paragraph. Just some text. Ignore what it says.
Its just an example. This is a paragraph. Just some text. Ignore what it says. Its just an example.

Links

A basic hypertext link is like this example:

```
<p>This is a paragraph. Just some text.
Ignore what it says.<a href="http://google.com">Go to
Google</a>Its just an
example. This is a paragraph. Just some text. Ignore what it says.
Its just an example.</p>
```



100% 14:34

file:///sdcard/share/test.htn

This is a paragraph. Just some text. Ignore what it says.Go to GoogleIts just an example. This is a paragraph. Just some text. Ignore what it says. Its just an example.

When the user clicks or touches the link, the browser will switch to Google.

This uses an *anchor element* <a..>...</a>

The important attribute is the href: <a href="http://google.com">

http://google.com is a URL.

There are different kinds of URL.

It starts with http: which is a *protocol*, saying how to get this - using http.

An alternative protocol is *mailto*. If we said

```
<a href="mailto:michael@mouse.com">Email me</a>
```

then when the user touches the link, the default email client on the device will start set up to send an email to michael@mouse.com

But, you should not put email address in web pages, because bots scrape email addresses and send spam to them. Instead you use a form and a server-side code - see later.

Another protocol is file:. This goes to a local file directly, instead of a file on a remote server using http.

google.com is a *domain*. A URL like http://google.com directs the browser to the server at that domain, and fetches the default page there, which will be index.htm or index.html or index.php or similar.

As well as the domain, we can say a folder and file on that domain, not just the home page. For example:

```
<a href="http://waltermilner.com/text.pdf">Read this</a>
```

This links to the file text.pdf on domain waltermilner.com. This file is a pdf, not a web page. What the browser does with this depends on the browser settings. It might display the pdf in the browser, display it with the default pdf viewer, or download and save it.

If there is no protocol, the browser will go try to load another web page from the current domain. So for example

<a href="page2.htm">Read this</a>

will fetch the web page named page2.htm from the current domain. There might also be a folder, such as

<a href="/morepages/page2.htm">Read this</a>

looks in folder morepages.

We can also link to one place in a document. For example:

```
  <p id="here">here</p>

<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
r><br>

  <p>This is a paragraph. Just some text. Ignore what it says.<a
  href="#here"> Go to top </a>Its just an example. This is a
paragraph.
  Just some text. Ignore what it says. Its just an example.</p>
```

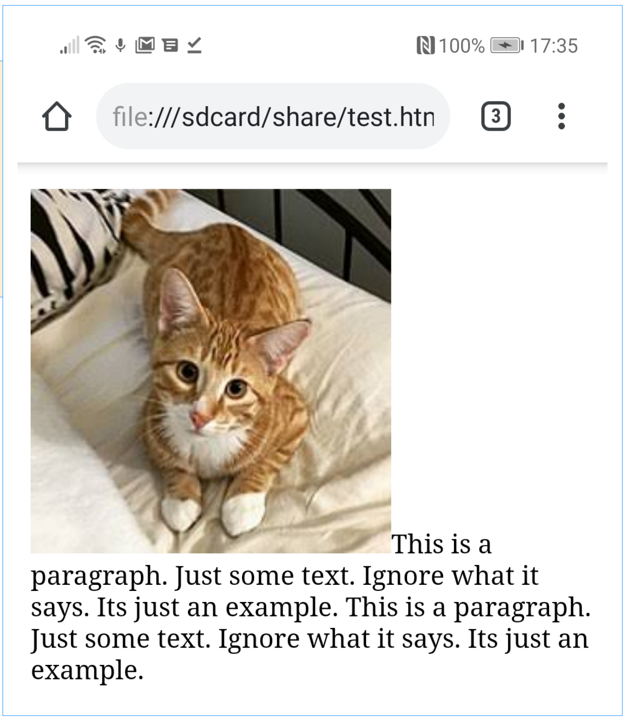
We have a lot of breaks, so when we scroll down to the second paragraph the first will be not visible. Touching the <a href="#here"> Go to top </a> link will make the browser scroll back to the first paragraph. This uses the *id attribute*.

The destination is an *anchor*, which is why a link element is named <a>

Images

An image in a web page is actually a separate file. We use an *image* element - for example

```
<p><img src="cat.jpeg">This is a
paragraph. Just some text.
Ignore what it says. Its just an
example. This is a paragraph.
Just some text. Ignore what it
says.
Its just an example.</p>
```



This is a paragraph. Just some text. Ignore what it says. Its just an example. This is a paragraph. Just some text. Ignore what it says. Its just an example.

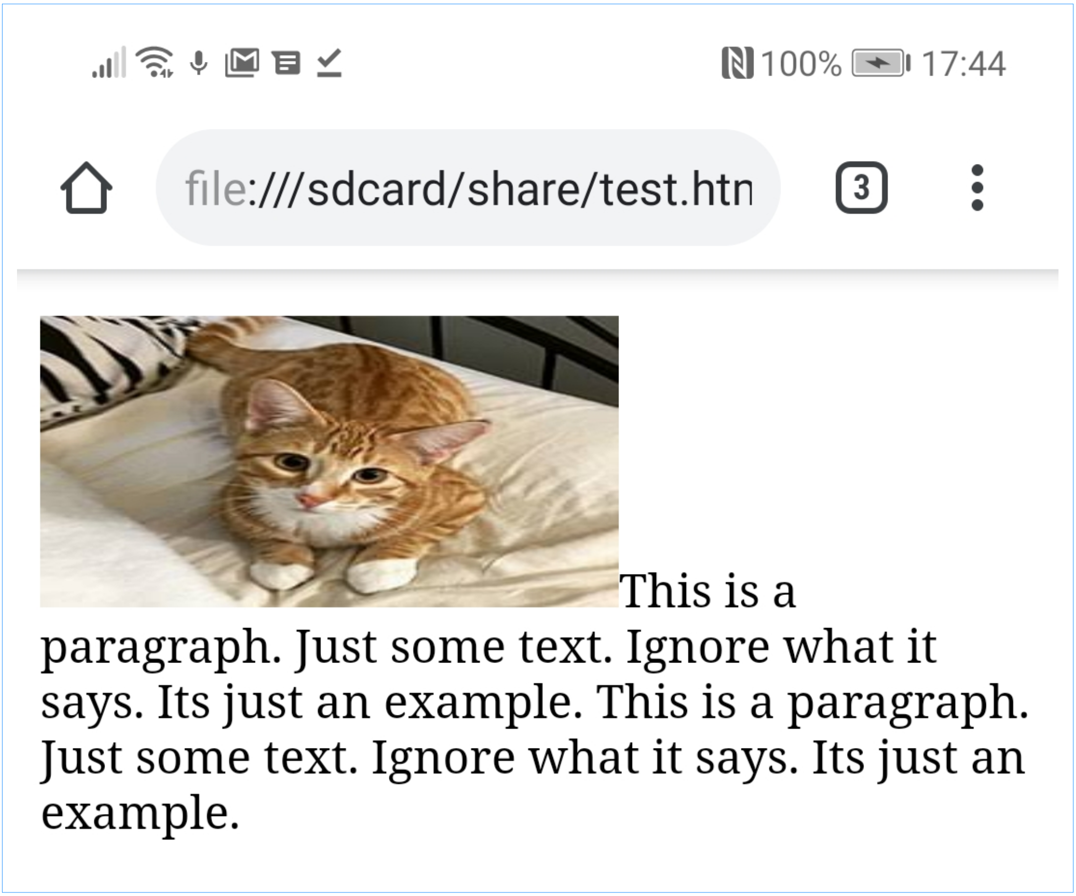
This is displaying the image file with the name cat.jpeg, which is in the same folder as the webpage. We could put a complete URL here, with a domain - so we can display an image from another site.

Image elements by default are displayed inline, and need to be inside block elements such as a paragraph.

We can do better positioning than this, but this is best done using styles.

We can specify the height and width (in pixels) like

```
<img src="cat.jpeg" height="100" width="200">
```



This is a paragraph. Just some text. Ignore what it says. Its just an example. This is a paragraph. Just some text. Ignore what it says. Its just an example.

But -

1. Keep the width and height proportion - the *aspect ratio* - same as the original, or you get a distorted image, as here.

2. If you make a small image bigger, it will look blurred and pixellated.

3. If you make a large image smaller, you are wasting bandwidth, fetching a big image and displaying a small one.

If possible you should size the actual image file as you want it, using a graphics program maybe.

Browsers know how to display jpeg, gif and png files, and maybe other formats, but probably not.

If your users are likely to have a slow connection, do not use a lot of big images.

Tables

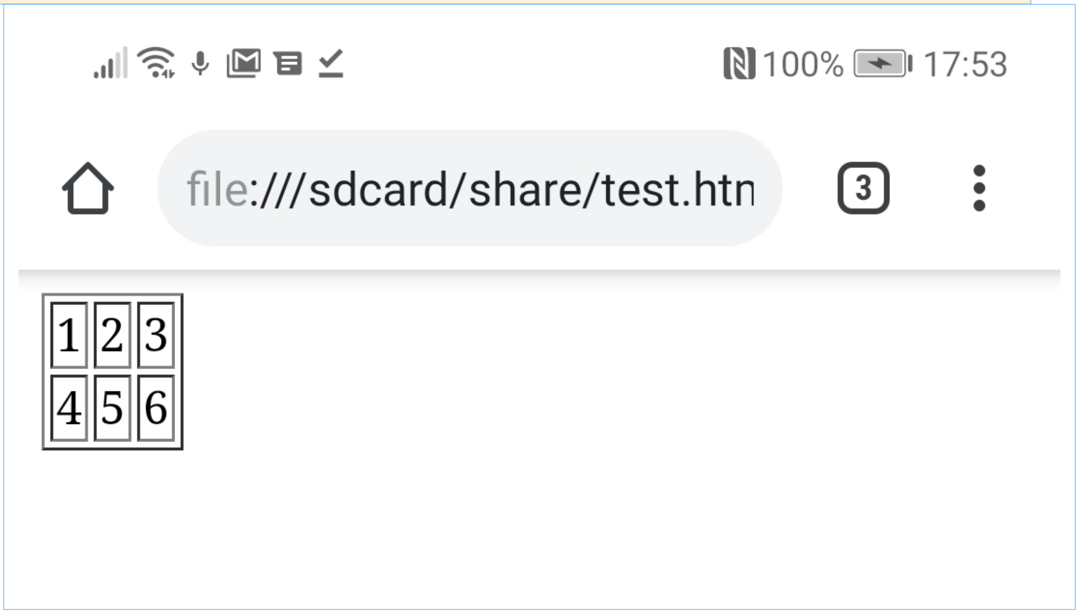
A *table* element consists of a pair of tags <table> and </table>.

In that you have a set of *rows* <tr>..</tr>

And in each row a sequence of *table cells* <td>..</td> - the td is for 'table data'.

For example

```
<table border=1>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
</table>
```



1	2	3
4	5	6

We set a border attribute on the table so we can see the cell boundaries.

We can style a table a lot.

Forms

A form takes data input from a user. Forms can be on paper, or on a screen. The data collected goes into an information system.

An html form on a web page is used to take input data from a user. It has a 'submit' button. When the user clicks it, the browser usually sends the data to a remote server for some processing.

Forms are used for logins, updating databases, sending messages in a 'contact' form and so on.

get and post

The form sends data to the server as a set of name-value pairs. For example a contact form might send names 'to' 'from' 'subject' 'body' The corresponding values might be 'michael@mouse.com', 'w.w.milner@gmail.com','Greetings' and 'Hello'

There are two ways the name-value pairs can be sent - *get* and *post*.

In get, they are sent in the URL, at the end, after a ?, like this.. http://somedomain.com/contact.php?to=michael@mouse.com&body=hello&..

This sends the data to the server at somedomain.com, and a script on it called contact.php (PHP is a common simple server-side language). Then after the ? we have the data as name=value pairs. PHP can easily get these back and use them in the script code.

This is simple, but not secure. The name value pairs are sent like this across the net, and are easily picked up - so passwords for example must not be sent like this.

The alternative is the post method. This sends a request packet to the server-side script, but the name-value pairs are sent in a data block at the end of the packet.

html forms

A form element is enclosed by tags <form>..</form>. It will have an 'action' attribute, which is the URL to the script which will process it, and a 'method', get or post.

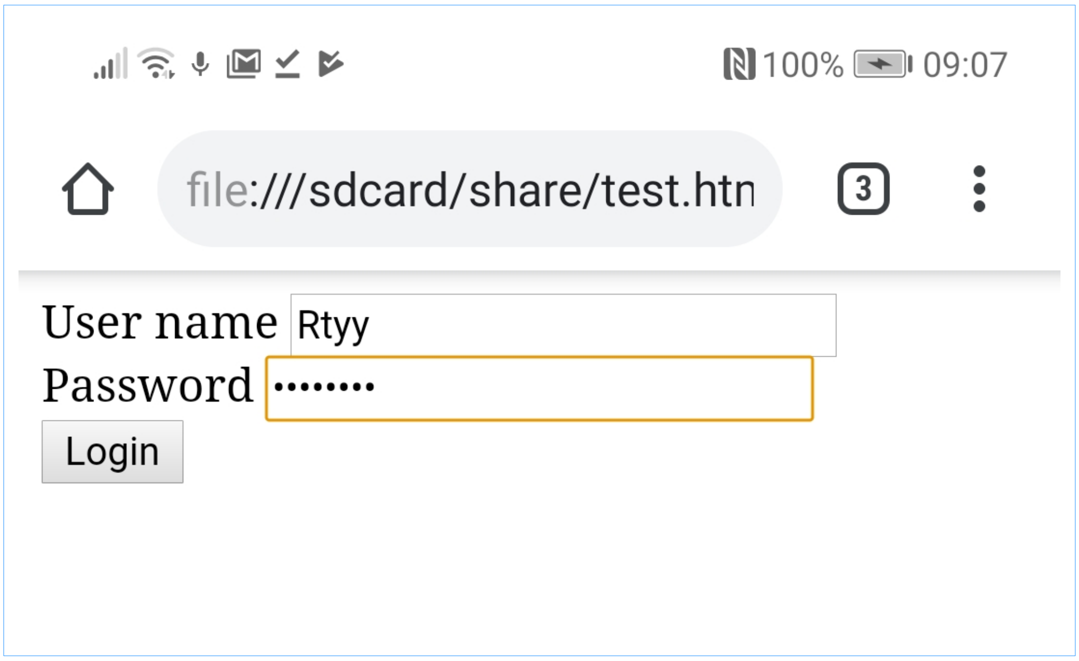
Inside the form there will be some input elements. These have different types, like text, date, radio, checkbox, password and so on.

For example:

```
<form name="myform" action="http://somewhere.com/login.php"
method="post">
User name <input type="text" name="name"><br>
Password <input type="password" name="password"><br>
<input type="submit" value="Login">
```

```
</form>
```

which looks like:



User name Rtyy
Password ••••••••
Login

This sends the data to a script named login.php. on server somewhere.com.

It sends two name-value pairs.

One pair has name 'name', and value 'Rtyy', or whatever they type in.

Th eother pair has name 'password', and value what they type in for a password.

The login.php script would check the password against a database, and send back confirming or rejecting the login.

New semantic elements

In html 5 some new elements were introduced. These include

<main> - the main content of a page

<header> a header - a title, logo, banner image and so on. It might head the page, or a section, or an article

<footer> a footer

<nav> a set of links - a navigation bar

<section> a section - a set of related articles

<article> an article - a topic which can stand by itself. An article iteslef might have a header, sections and so on

<aside> content not directly about an article - a sidebar

These are called *semantic elements* because semantic means to do with *meaning*. A <nav> element means it is a set of links for navigation.

Why semantic elements?

In html 4, developers would use divs, and give them CSS classes (see text on CSS). But:

1. Visually impaired users (4 or 5% of all users) use screen readers to read out web pages. These readers do not understand CSS classes

2. Neither do search engines. But they probably do understand <main> and <section>, and can offer direct links in search results to those places.

3. If someone is looking at the html of a web page, it is easier to make sense of it with semantic elements rather than a set of divs.