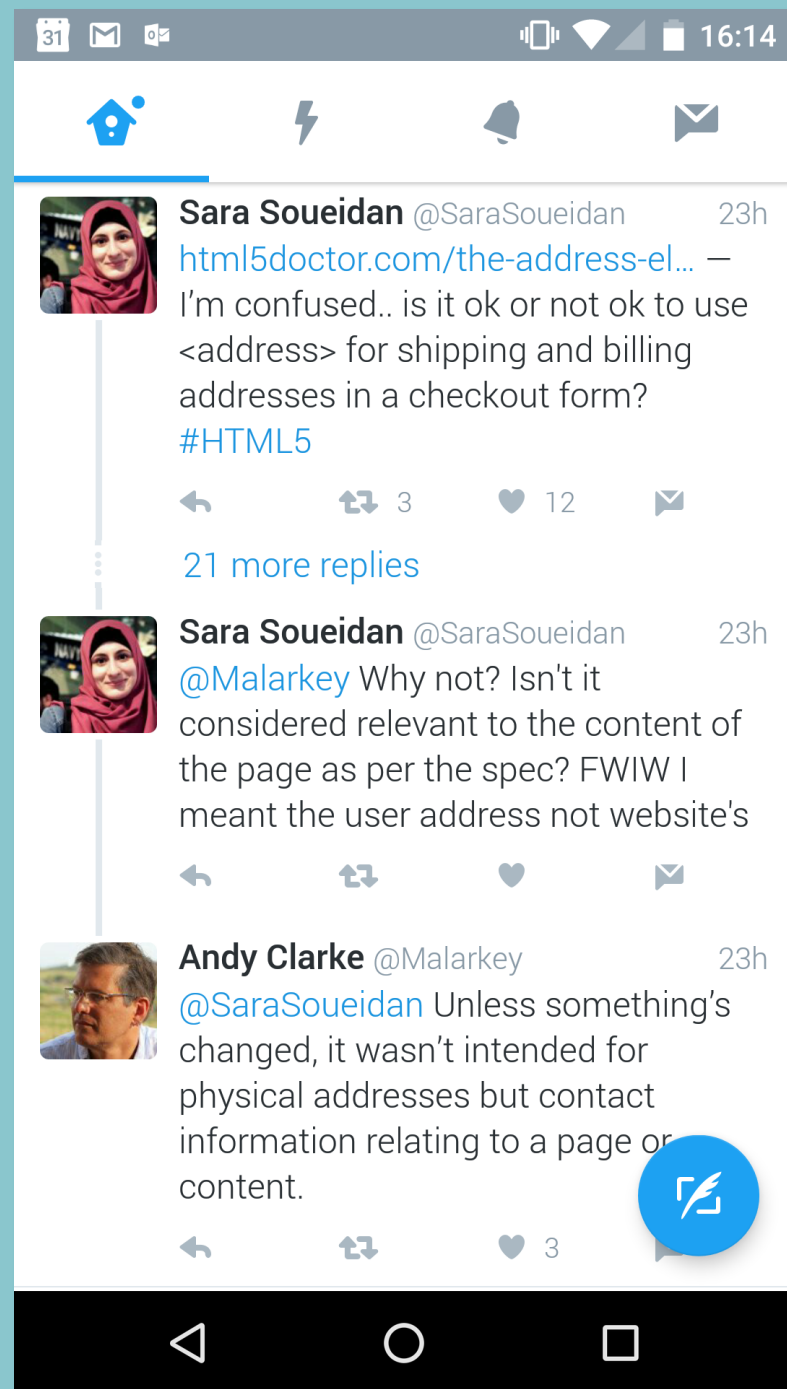


The Structure Layer (Hypertext Markup Language)

Design for web content

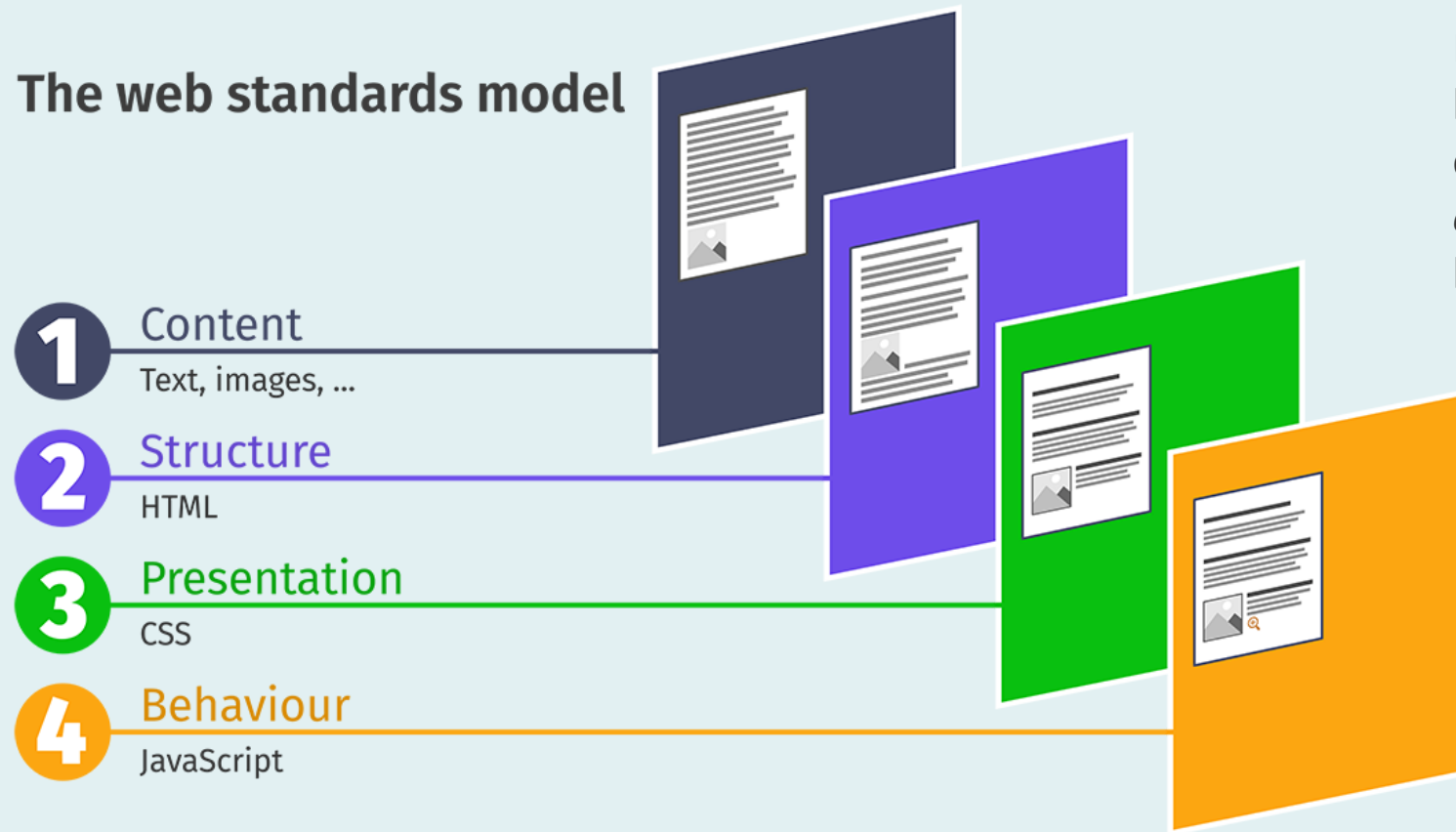


Even our heroes sometimes find the semantics of HTML confusing, so you're in good company!

Anatomy of a webpage

Webpages can be considered to consist of three layers of technology. This is sometimes referred to as the “Web Standards Model”. The aim is to separate structure, presentation and behaviour. Each layer is defined by a different technology or language; HTML for structure, CSS for presentation and JavaScript for behaviour.

The web standards model



Each layer, moving from bottom to top, could be considered a *progressive enhancement* of the one below it.

The structure layer

2 Structure

Structure is the first layer of enhancement that we apply to our content. The process involves “marking up” content with HTML tags. HTML defines the structure of our document – appropriate heading levels will define the document outline. HTML will also add meaning (semantics) to our content, allowing it to be more easily understood by browsers and bots.



Document **structure** and content **meaning** are defined using a *markup language* such as HTML

[The Web Standards Model](#)

The nature of HTML

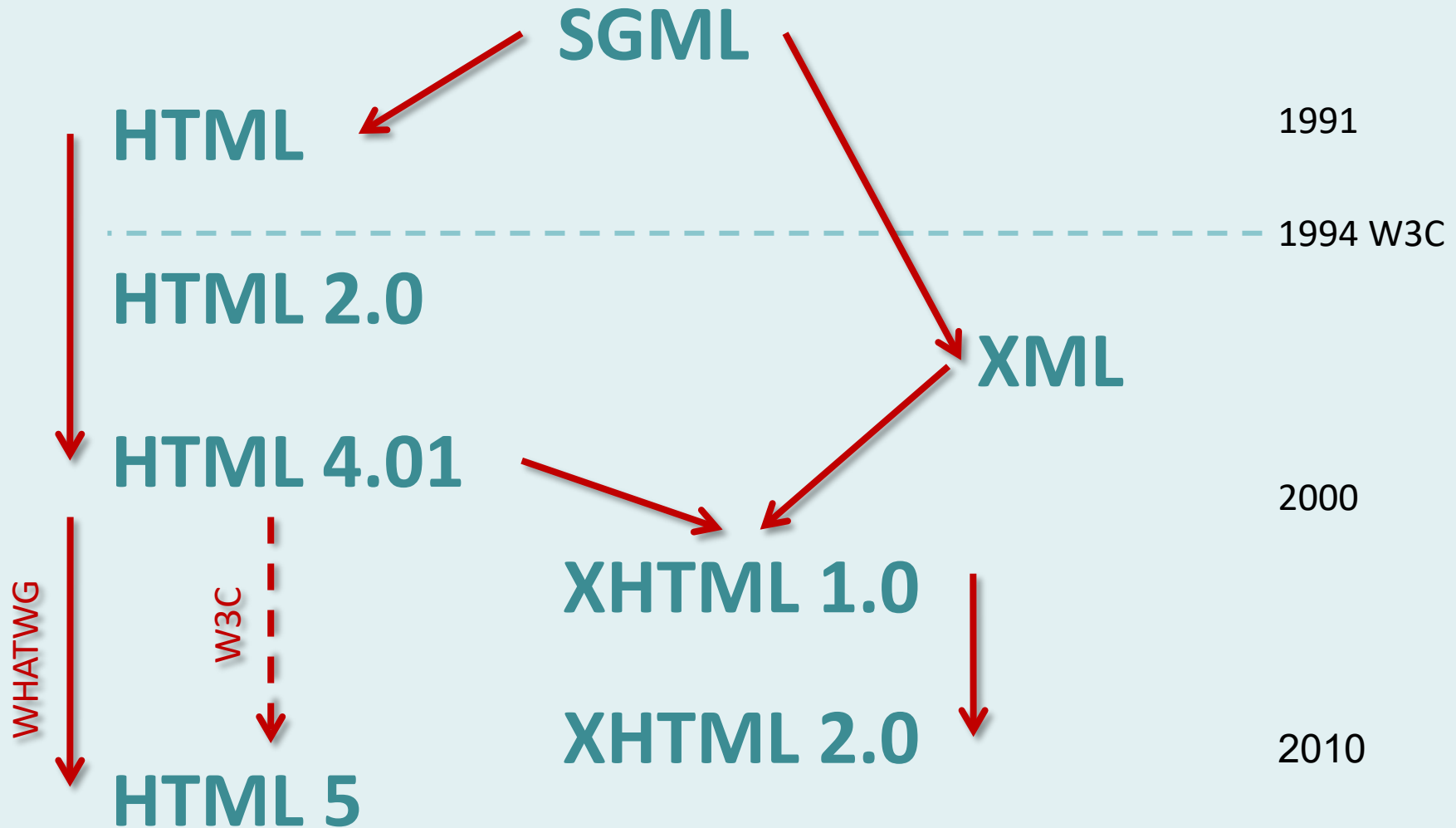
HTML is a *declarative* language. That means it declares what should be done but doesn't explain how to do it. For example, the HTML below tells the browser that an image should be displayed, but it doesn't tell the browser how to do that. We trust that the browser knows how to do it.

```

```

Is HTML a programming language? Strictly speaking, it is not, but [some people have argued](#) that, in a narrow sense, it could be considered a declarative programming language. Other languages that we will meet in the future (e.g. JavaScript) are considered *imperative* programming languages because they are used to tell the browser how to do something.

A *very* brief history of markup



A simple question

`<p>`What is the
``purpose`` of
HTML?`</p>`

What is the **purpose** of HTML?

Answer

`<p>`To add `structure` to documents and `meaning` to content.`</p>`

To add *structure* to documents and *meaning* to content.

What are tags for?

`<p>content</p>`

Tags are used to describe the type of content they contain (semantics). They are **not** used to describe how that content should look. They tell *user agents* (e.g. browsers) where an element begins and where it ends.

What is an element?

`<p>content</p>`

An HTML *element* usually consists of an opening tag, some content, and a closing tag. The example above is a paragraph element.

What is an element?

```

```

Some HTML *elements* consists only of an opening tag (considered to be self-closing), plus some extra information in the form of *attributes*. The example above is an image element.

Another question

`<p>`If HTML5 is the current version of markup, why do we need to know anything about XHTML?`</p>`

If HTML 5 is the current version of markup, why do we need to know anything about XHTML?

Answer

`<p>`Because current best practice is to use HTML5 elements with XHTML syntax.`</p>`

Because current best practice is to use HTML5 elements with XHTML syntax.

Note: XHTML has a strict syntax, HTML5 does not.

HTML and XHTML

Elements, attributes and values

Elements

`<p>`This is a paragraph. It contains some text.`</p>`

``

Most HTML elements are defined by opening and closing tag pairs; a few (like the image element) are defined using a single, self-closing tag.

Note: there is a difference in the way XHTML and HTML5 treat self-closing elements (such as images). In XHTML, the closing slash character (/) is mandatory but in HTML 5 it is optional (typically it is omitted).

Attribute names and values

```

```

Attributes give the browser more information about an element. The attributes in the example above are telling the browser how big the image is and where to find it. Like many elements, the image has a number of valid attributes (e.g. width and height). Each attribute must have a **name** and a quoted **value** in the form: **name**="value".

In HTML, some attributes are mandatory. For example, all image elements **must** have an **alt** attribute (text alternative) for accessibility and also a **src** (source) so the browser knows where to find the image file . Some elements are optional (e.g. **height** and **width**).

Note: there is no unit of measurement for the height and width attributes. The values are pixels, but we don't need to say so because there are no other options.

Structure and relationships

(talking the right language)

Block-level elements

`<p>`This is a paragraph. It contains some
``important`` text. The
paragraph is a block-level element.`</p>`

The paragraph `<p>` is a *block-level* element. Effectively, this means that it begins and ends with a line-break and forms a distinctive “block” of content within the page.

Inline elements

`<p>`This is a paragraph. It contains some ``important`` text. The important text is an inline element.`</p>`

`` is an *inline* element. It does not begin and end with a line-break, it runs inline with any surrounding text. Images are also inline elements (this is sometimes a cause of confusion).

Parent elements

`<p>`This is a paragraph. It contains some
``important`` text. The paragraph is
the parent of the emphasised text.`</p>`

The paragraph is said to be the *parent* of the `` element because the emphasis element is “nested” inside the paragraph.

Child elements

`<p>`This is a paragraph. It contains some
``important`` text. The emphasised
text is a child of the paragraph element.`</p>`

The `` element is said to be the *child* of the paragraph. Elements may have many children but only one direct parent.

Sibling elements

This is a list item.

This is an adjacent sibling.

The markup above defines an unordered list . Each list item is the child of the unordered list. Each list item is also the *sibling* of the other list items contained within the same unordered list. The unordered list is the parent of all the list items it contains.

Semantic markup

(the important, controversial stuff)

Incorrect use of markup

`<h1>Page Heading</h1>`

`<p>Some introductory text</p>`

`<p><big>A sub-heading</big></p>`

`<p>This is a paragraph. It contains some
important text and content relating
to the sub-heading above.</p>`

There are two things wrong with the markup above. A sub-heading should not be marked up as a paragraph because it is **not** a paragraph, it is a heading. Although `<big>` is part of the XHTML specification, it is *deprecated*, and *obsolete* in HTML5 because it is used to control presentation (i.e. it makes text bigger). Presentational markup is wrong.

Semantically correct markup

`<h1>Page Heading</h1>`

`<p>Some introductory text</p>`

`<h2>A sub-heading</h2>`

`<p>This is a paragraph. It contains some
important text and content relating
to the sub-heading above.</p>`

The markup above is correct – each tag is used to describe the content it contains and forms a logical document structure. None of the elements are deprecated or obsolete.

Why is this important?

`<h2>`A sub-heading`</h2>`

`<p><big>`A sub-heading`</big></p>`

These two lines of HTML may *look* the same when rendered in a browser but they have very different meanings. Remember, the purpose of HTML is to add *meaning* to our content, not to control the way it looks (the presentation). That job is for CSS, not HTML.

When designing websites, we need to be aware that our content is often read by non-visual agents (e.g. bots and screen readers). So the way our content looks is not the best way to convey meaning.

Although presentational elements such as `<big>` are still part of the XHTML specification (although deprecated), they have been removed from HTML5 (obsolete).

Incorrect use of markup

`<h2>`This is a list of colours`</h2>`

`<p>`Red`</p>`

`<p>`Green`</p>`

`<p>`Blue`</p>`

The markup above is non-semantic. The content is clearly a list but it is marked up as three paragraphs. Visually, it may look like a list, but semantically, it is not a list.

Semantically correct markup

`<h2>`This is a list of colours`</h2>`

``

``Red``

``Green``

``Blue``

``

Correctly marked up as an unordered list. Although this code is more *verbose*, we have added a lot more meaning to the content and we have made relationships between the different elements (parent, child etc.). Non-visual agents will understand these relationships.

Why is this important?

Red

Green

Blue

In addition to telling non-sighted agents that this is a list, making the correct relationships between different elements will help us a great deal when we come to style those elements with CSS. For example, we can easily select all the child elements of a specific list, by targeting their parent element, and style them consistently.

Hypertext links

Making links

```
<a href="file.html">Text Link</a>
```

The opening and closing tags of the anchor element are used to form a hyperlink. Any content between the tags will be clickable. The anchor element has one mandatory attribute, href (hypertext reference) which points to the linked file.

```
<a href="file.html" target="_blank">Text Link</a> (don't do this)
```

You will often see the target attribute being used to automatically open a link in a new browser tab. However, this is a usability issue and considered bad practice. In general, we should allow the user to control link actions.

Making image links

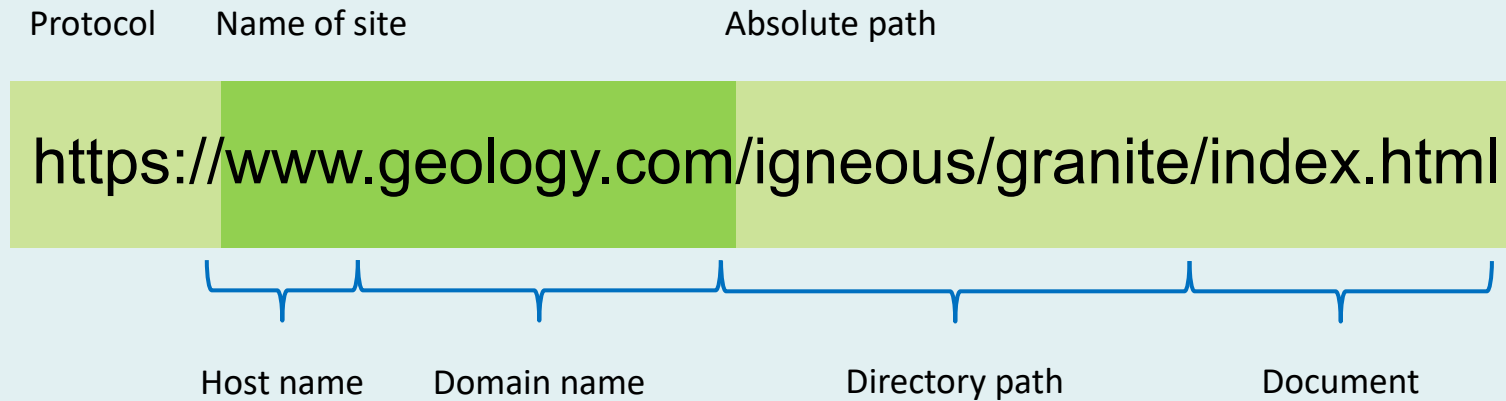
```
<a href="eagle.html"></a>
```

Image links are made in exactly the same way as text links. In the example above, the image element is nested inside the anchor element. The result is that the whole image is clickable.

The image is the child of the anchor, the anchor is its parent.



Universal resource locator (URL)



A URL is a sub-class of a URI (Universal Resource Identifier), just as the Web could be said to be a sub-class of the Internet. The two should not be confused – they are not the same thing.

The protocol is the HyperText Transport Protocol (*http*). You may also see *https* where the “s” stands for *secure* and is the new standard on the Web. Such connections are encrypted.

Absolute links

```
<a href="https://www.mysite.com/design/file.html">Text</a>
```

Absolute links are always used for links to *external* pages (i.e. those on other websites). The absolute path **must** begin with “http” or “https”. They will also work for local files but are not usually used because they are verbose and not portable – there’s a neater way to make local links...

Relative links

```
<a href=" ../design/file.html">Text Link</a>
```

Relative links are used for links to local pages (i.e. those on the same website). In the example above, the two little dots mean “go up one level in the folder structure”. From there the link is to a file called *file.html* in the folder called *design*. Relative links are useful because they will work when developing sites locally (on a PC or Mac) **and** when they are uploaded to the server.

Example relative links

```
<a href="file.html">Text Link</a>
```

A link to a file in the same folder.

```
<a href="folder/file.html">Text Link</a>
```

A link to a file in a folder one level below.

```
<a href="../file.html">Text Link</a>
```

A link to a file in a folder one level above.

```
<a href="../folder/file.html">Text Link</a>
```

A link to a file in a different folder at the same level.

Note: You may sometimes see relative URLs begin with a single dot and slash `./file.html` where the link points to a resource in the same folder. This is functionally equivalent to not using the dot and slash, so it is usually omitted. In the example on the left, `file.html` is the same as `./file.html`.

Special characters



Character escaping

Some characters have a special meaning in HTML and **must** be *escaped* in order to display correctly in the browser and to validate.

For example "<" and ">" will be interpreted as the start and end of HTML tags unless they are escaped.

< = < less than

> = > greater than

& = & ampersand

" = " plain quote

<p>MA Web Design & Content Planning</p>

Character escaping

One of the benefits of this is that we can display characters that do not exist on our keyboard. For example:

`<p>© David Watson</p>`

`<p>© David Watson</p>`

Both of the paragraph elements above would render as:

© David Watson

Common structural elements

Headings

`<h1>`Heading level 1`</h1>`

`<h2>`Heading level 2`</h2>` =

`<h3>`Heading level 3`</h3>`

Heading level 1
Heading level 2
Heading level 3

Headings are used to define *hierarchy* within a document, **not** to specify the visual emphasis (text height).

Note: the rendering above uses the default browser styling.

The unordered list

List item 1

List item 2

List item 3

=

- List item 1
- List item 2
- List item 3

Unordered lists are very useful structural elements and are commonly used to define navigation on a web page (amongst other things).

The ordered list

List item 1

List item 2

List item 3

=

1. List item
2. List item
3. List item

Ordered lists have a specific use case and are used where the list items have a logical order. The definition list is a third list type in HTML, it also has a specific use case.

Tables

<table>

<tr>

<td>Row 1, cell 1</td>

<td>Row 1, cell 2</td>

</tr>

<tr>

<td>Row 2, cell 1</td>

<td>Row 2, cell 2</td>

</tr>

</table>

=

Row 1, cell 1	Row 1, cell 2
Row 2, cell 1	Row 2, cell 2

The table has been badly misused in the past (for page layout) but these days it is correctly used only for its specific purpose i.e. as a container for tabular data.

All the HTML5 elements



There are well over 100 elements in the HTML5 specification. Some are used all the time (e.g. headings and paragraphs) and some are used very rarely. You don't need to learn all of them by heart but you should familiarise yourself with the list so that you know what's available to you when marking up content.

[HTML5 Element Reference](#)

Writing HTML

HTML file structure

<html>

<head>

<title>Webpage Design</title>

</head>

<body>

<p>My first web page!</p>

</body>

</html>

Not displayed in the
browser window =
meta information

Displayed in the
browser window =
document content

XHTML doctype

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml" dir="ltr" lang="en">
```

```
<head>  
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />  
<title>Webpage Design</title>  
</head>
```

```
<body>...
```

Although the basic file structure is simple, we must add some gobbledygook in order to tell user agents what type of markup is being used. You're not expected to remember this stuff – just copy it from a reliable source. Fortunately, no one should be using XHTML for new documents.

<http://www.w3.org/TR/html401/struct/global.html>

How HTML5 differs

```
<!DOCTYPE html>
```

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

```
<head>
```

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

```
  <title>Webpage Design</title>
```

```
</head>
```

```
<body>...
```

HTML5 is a much less verbose markup language. Much of the unnecessary baggage has been stripped away. Whereas in XHTML, every setting must be made explicitly, in HTML5, many settings assume a common default value. Notice that the doctype doesn't even include the number "5". Why do you think that is?

HTML is evolving

The HTML specification is hosted with WHATWG but is a collaboration between them and W3C. Note that it is a “living standard”, meaning no more iterations but a process of evolution.

HTML

Living Standard — Last Updated 1 October 2021



[One-Page Version](#)

html.spec.whatwg.org

[Multipage Version](#)

/multipage

[Version for Web Devs](#)

/dev

[PDF Version](#)

/print.pdf

[Translations](#)

日本語・简体中文

[FAQ](#)

on GitHub

[Chat](#)

on Matrix

[Contribute on GitHub](#)

whatwg/html repository

[Commits](#)

on GitHub

[Snapshot](#)

as of this commit

[Twitter Updates](#)

@htmlstandard

[Open Issues](#)

filed on GitHub

[Open an Issue](#)

whatwg.org/newbug

[Tests](#)

web-platform-tests html/

[Issues for Tests](#)

ongoing work

Table of contents

- 1 Introduction
- 2 Common infrastructure
- 3 Semantics, structure, and APIs of HTML documents
- 4 The elements of HTML
- 5 Microdata
- 6 User interaction
- 7 Loading web pages
- 8 Web application APIs
- 9 Communication
- 10 Web workers
- 11 Worklets
- 12 Web storage
- 13 The HTML syntax
- 14 The XML syntax
- 15 Rendering
- 16 Obsolete features
- 17 IANA considerations
- Index
- References
- Acknowledgments
- Intellectual property rights

[File an issue about the selected text](#)

Deprecated tags and attributes

```
<p><font size="5">big text</font></p>
```

Deprecated tags are those that should not be used in the current version of the markup language. However, they are still recognised by browsers and will be rendered accordingly, irrespective of the doctype you are using. The use of deprecated tags and/or attributes will invalidate your markup. You should not use deprecated tags. Obsolete tags may not be supported by modern browsers.

In XHTML, most of the deprecated tags are those that were used in HTML4 to describe presentation. Examples include ``, `<center>` and `<u>` (underline). Tags *deprecated* in XHTML are considered *obsolete* in HTML5.

HTML5 is backwards-compatible, so tags that are not part of the specification are considered “obsolete” (not deprecated) and any markup containing them is “non-conforming”.

Which tags can I use?

The screenshot shows the MDN Web Docs interface. At the top, there's a navigation bar with 'mdn web docs', 'References', 'Guides', and 'MDN Plus'. On the right, there are links for 'Theme', a search icon, 'Already a subscriber?', and a 'Get MDN Plus' button. Below this is a breadcrumb trail: 'References > HTML > Elements'. The main content area is titled 'HTML elements reference'. It includes a paragraph stating that the page lists all HTML elements created using tags, grouped by function. A blue note box contains a tip to see the 'Introduction to HTML article' for more basics. Below this is a section titled 'Main root' with a table. The table has two columns: 'Element' and 'Description'. The first row shows the '<html>' element, describing it as the root (top-level element) of an HTML document. To the left of the main content is a sidebar with 'Related Topics' including 'HTML', 'Tutorials' (HTML basics, Introduction to HTML, Multimedia and embedding, HTML tables), 'References' (HTML elements, Global attributes, <input> types), and 'Documentation' (Useful lists, Contribute). To the right is a 'In this article' sidebar listing various topics like 'Main root', 'Document metadata', 'Sectioning root', 'Content sectioning', 'Text content', 'Inline text semantics', 'Image and multimedia', 'Embedded content', 'SVG and MathML', 'Scripting', 'Demarcating edits', 'Table content', 'Forms', and 'Interactive elements'.

mdn web docs References Guides MDN Plus Theme Search Already a subscriber? Get MDN Plus

References > HTML > Elements English (US)

HTML elements reference

This page lists all the [HTML elements](#), which are created using [tags](#).

They are grouped by function to help you find what you have in mind easily. An alphabetical list of all elements is provided in the sidebar on every element's page as well as this one.

Note: For more information about the basics of HTML elements and attributes, see [the section on elements in the Introduction to HTML article](#).

Main root

Element	Description
<html>	The <html> HTML element represents the root (top-level element) of an HTML document, so it is also referred to as the <i>root element</i> . All other elements must be descendants of this element.

Related Topics

HTML

Tutorials:

- HTML basics
- ▶ Introduction to HTML
- ▶ Multimedia and embedding
- ▶ HTML tables

References:

- ▶ HTML elements
- ▶ Global attributes
- ▶ <input> types

Documentation:

- ▶ Useful lists
- ▶ Contribute

In this article

- Main root
- Document metadata
- Sectioning root
- Content sectioning
- Text content
- Inline text semantics
- Image and multimedia
- Embedded content
- SVG and MathML
- Scripting
- Demarcating edits
- Table content
- Forms
- Interactive elements

Use a reference. The excellent online reference at Mozilla is a good option. You can even use a book if you prefer that medium but things change quickly on the Web ;)

[HTML Elements](#)

Indentation

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" href="style.css">
  <title>A descriptive title</title>
</head>

<body>
  <main>
    <h1>A first order heading</h1>
    <p>Some introductory text in a paragraph.</p>
    <p>Another paragraph.</p>
    <p>A third paragraph.</p>
  </main>
</body>
</html>
```

The use of indentation is **very important** when writing HTML. Indentation is used to indicate the nesting level of elements (parents and children). This makes your code much easier to read and understand than if it were all flush left. In the example above, it's easy to see that all paragraphs are children of main.

Syntax highlighting

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8">
5      <link rel="stylesheet" href="style.css">
6      <title>A decriptive title</title>
7  </head>
8
9  <body>
10     <main>
11         <h1>A first order heading</h1>
12         <p>Some introductory text in a paragraph.</p>
13         <p>Another paragraph.</p>
14         <p>A third paragraph.</p>
15     </main>
16 </body>
17 </html>
```

Text editors with syntax highlighting (colour coding) are really useful for spotting errors in your code. The combination of indentation and syntax highlighting makes your code much easier to read and to work with.

Comments in HTML

It's **always** a good idea to comment your markup in order to remind yourself (or to let someone else know) what your code is intended to do or to clarify the document structure. Comments will not display in the browser window.

```
<!-- start of main page content -->
```

```
<h2>This week's book review</h2>
```

```
<p>The Return of the Native by...</p>
```

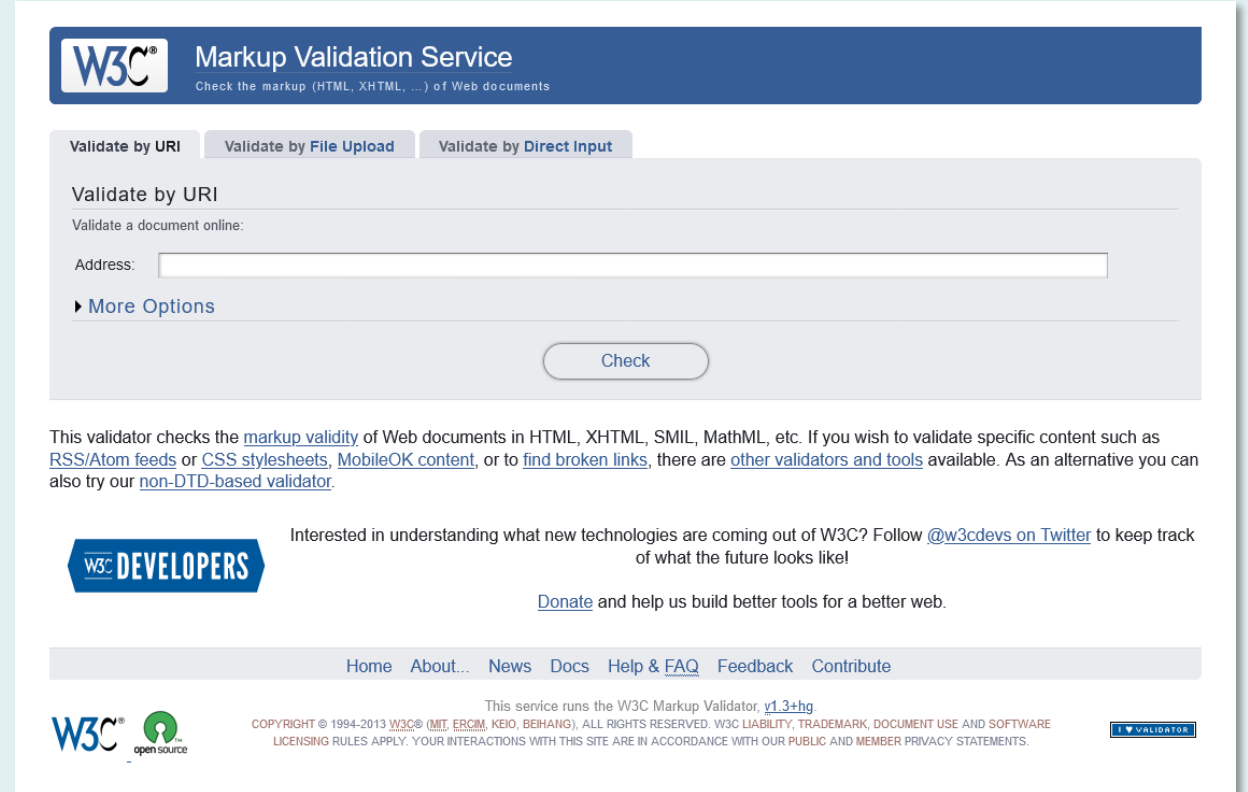
Comments are useful

```
1 <!DOCTYPE html>
2 <html lang="en">
3 ▼ <head>
4     <meta charset="utf-8">
5     <link rel="stylesheet" href="style.css">
6     <title>A decriptive title</title>
7 </head>
8
9 ▼ <body>
10 ▼ <main> <!-- The start of the main content -->
11     <h1>A first order heading</h1>
12     <p>Some introductory text in a paragraph.</p>
13     <p>Another paragraph.</p>
14     <!--<p>A third paragraph.</p>-->
15 </main>
16 </body>
17 </html>
```

Comments can be used as waymarkers in your code. They can be used to write notes for yourself or others in your team and they can be used to “comment out” sections of code.

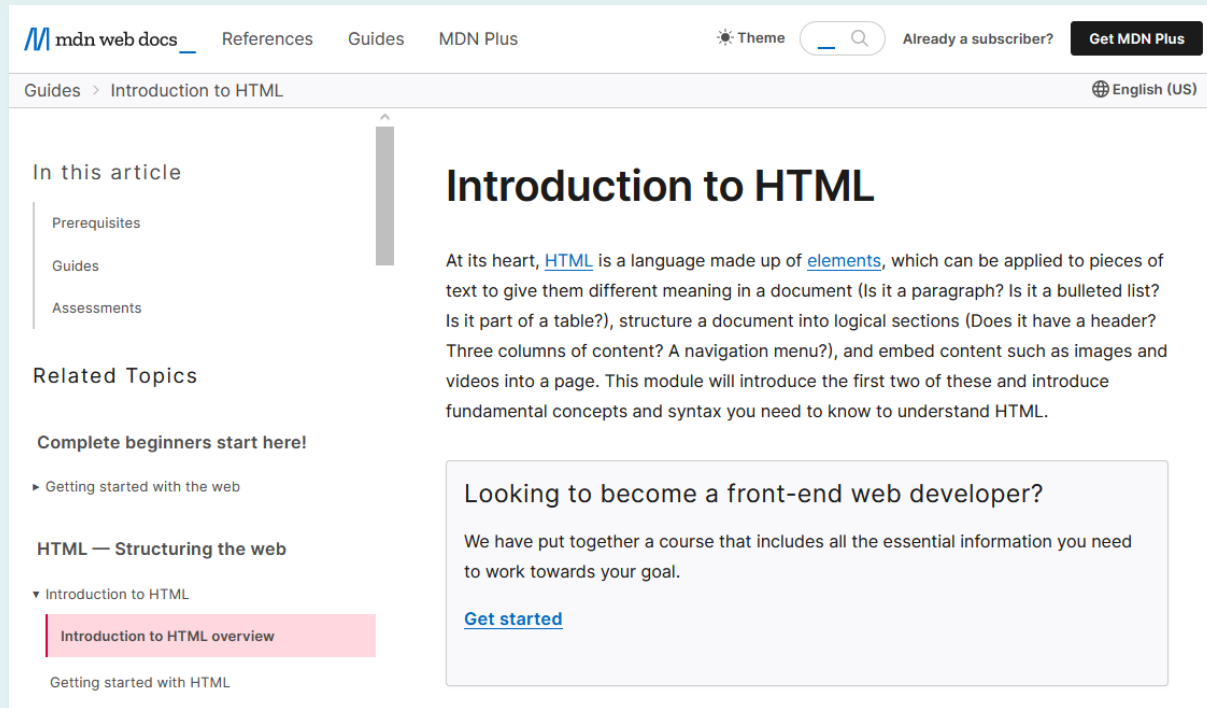
Code Validation

- **Always** validate your HTML using the W3C Markup Validation Service
- The service uses the doctype to determine what type of markup you are using
- Correct any errors or warnings in your markup



The screenshot shows the W3C Markup Validation Service interface. At the top, there's a blue header with the W3C logo and the text "Markup Validation Service" and "Check the markup (HTML, XHTML, ...) of Web documents". Below this, there are three tabs: "Validate by URI", "Validate by File Upload", and "Validate by Direct Input". The "Validate by URI" tab is selected. Under this tab, there's a section titled "Validate by URI" with the text "Validate a document online:". Below this, there's a text input field labeled "Address:". To the right of the input field is a "Check" button. Below the input field, there's a link "More Options". Below the input field and the "Check" button, there's a paragraph of text: "This validator checks the [markup validity](#) of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as [RSS/Atom feeds](#) or [CSS stylesheets](#), [MobileOK content](#), or to [find broken links](#), there are [other validators and tools](#) available. As an alternative you can also try our [non-DTD-based validator](#)." Below this paragraph, there's a blue button with the text "W3C DEVELOPERS". To the right of the button, there's a paragraph of text: "Interested in understanding what new technologies are coming out of W3C? Follow [@w3cdevs on Twitter](#) to keep track of what the future looks like!" Below this paragraph, there's a link "Donate" and the text "and help us build better tools for a better web." Below the "Donate" link, there's a horizontal navigation bar with links: "Home", "About...", "News", "Docs", "Help & FAQ", "Feedback", and "Contribute". Below the navigation bar, there's a footer section. On the left, there's the W3C logo and the text "open source". In the center, there's a paragraph of text: "This service runs the W3C Markup Validator, [v1.3+hg](#). COPYRIGHT © 1994-2013 W3C® (MIT, ERCIM, KEIO, BEIHANG), ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND MEMBER PRIVACY STATEMENTS." On the right, there's a small blue button with the text "VALIDATOR".

Learning HTML



The screenshot shows the Mozilla MDN website's 'Introduction to HTML' article. The top navigation bar includes 'mdn web docs', 'References', 'Guides', and 'MDN Plus'. A search bar and a 'Theme' selector are also present. The article title 'Introduction to HTML' is prominently displayed. Below the title, a paragraph explains that HTML is a language made of elements used to structure documents. A sidebar on the left lists 'In this article' sections: Prerequisites, Guides, and Assessments. Below that, 'Related Topics' are listed, including 'Complete beginners start here!' and 'Getting started with the web'. A 'Get started' link is highlighted in a pink box. A callout box at the bottom right asks 'Looking to become a front-end web developer?' and provides a link to 'Get started'.



The screenshot shows the 'Learn to Code HTML & CSS' website. The header includes the title 'Learn to Code HTML & CSS' and the subtitle 'Develop & Style Websites'. A list of lessons is provided, with some marked as 'New'. A 'Start Learning HTML & CSS' button is prominently displayed. Below the button, social media icons for Twitter, Facebook, and Google+ are shown. A section titled 'Learn to Code HTML & CSS the Book' is also visible, featuring a book cover image.

Learning to use HTML and to build websites isn't particularly difficult and there are plenty of helpful online resources but not all of them are up-to-date or accurate. The Mozilla [Introduction to HTML](#) is good, as is Shay Howe's [Learn to Code HTML & CSS](#). W3Schools should be avoided.

```
<end type="slideshow" />
```