Document and website structure

In addition to defining individual parts of your page (such as "a paragraph" or "an image"), HTML also boasts a number of block level elements used to define areas of your website (such as "the header", "the navigation menu", "the main content column"). This article looks into how to plan a basic website structure, and write the HTML to represent this structure.

Prerequisites:	Basic HTML familiarity, as covered in Getting started with HTML. HTML text formatting, as covered in HTML text fundamentals. How hyperlinks work, as covered in Creating hyperlinks.
Objective:	Learn how to structure your document using semantic tags, and how to work out the structure of a simple website.

Basic sections of a document

Webpages can and will look pretty different from one another, but they all tend to share similar standard components, unless the page is displaying a fullscreen video or game, is part of some kind of art project, or is just badly structured:

header:

Usually a big strip across the top with a big heading and/or logo. This is where the main common information about a website usually stays from one webpage to another.

navigation bar:

Links to the site's main sections; usually represented by menu buttons, links, or tabs. Like the header, this content usually remains consistent from one webpage to another — having inconsistent navigation on your website will just lead to confused, frustrated users. Many web designers consider the navigation bar to be part of the header rather than an individual component, but that's not a requirement; in fact, some also argue that having the two separate is better for accessibility, as screen readers can read the two features better if they are separate.

main content:

A big area in the center that contains most of the unique content of a given webpage, for example, the video you want to watch, or the main story you're reading, or the map you want to view, or the news headlines, etc. This is the one part of the website that definitely will vary from page to page!

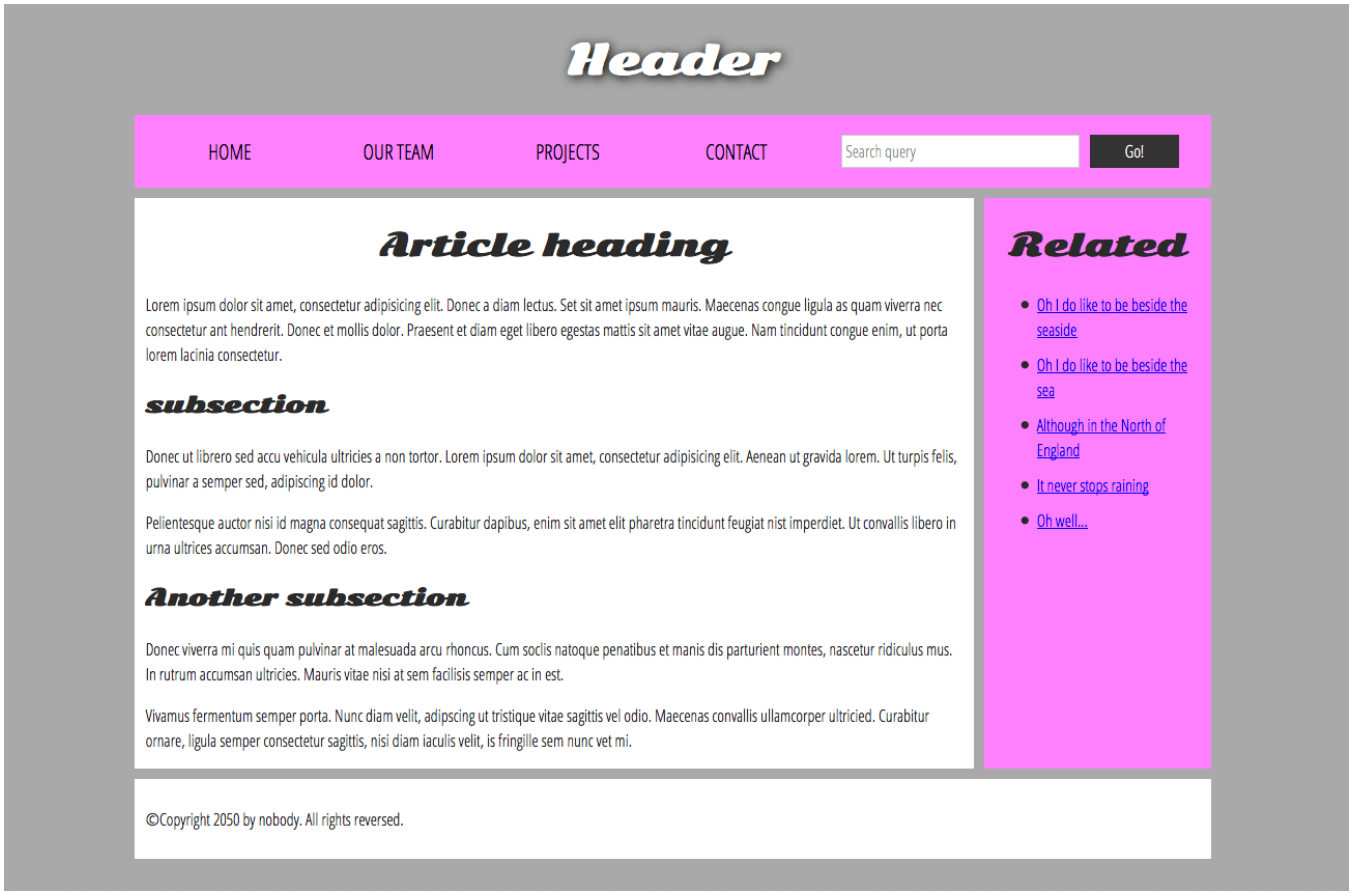
sidebar:

Some peripheral info, links, quotes, ads, etc. Usually, this is contextual to what is contained in the main content (for example on a news article page, the sidebar might contain the author's bio, or links to related articles) but there are also cases where you'll find some recurring elements like a secondary navigation system.

footer:

A strip across the bottom of the page that generally contains fine print, copyright notices, or contact info. It's a place to put common information (like the header) but usually, that information is not critical or secondary to the website itself. The footer is also sometimes used for SEO purposes, by providing links for quick access to popular content.

A "typical website" could be structured something like this:



HTML for structuring content

The simple example shown above isn't pretty, but it is perfectly fine for illustrating a typical website layout example. Some websites have more columns, some are a lot more complex, but you get the idea. With the right CSS, you could use pretty much any elements to wrap around the different sections and get it looking how you wanted, but as discussed before, we need to respect semantics and **use the right element for the right job**.

This is because visuals don't tell the whole story. We use color and font size to draw sighted users' attention to the most useful parts of the content, like the navigation menu and related links, but what about visually impaired people for example, who might not find concepts like "pink" and "large font" very useful?

> **Note**: Colorblind people represent around 4% of the world population or, to put it another way, approximately 1 in every 12 men and 1 in every 200 women are colorblind. Blind and visually impaired people represent roughly 4-5% of the world population (in 2012 there were 285 million such people in the world, while the total population was around 7 billion).

In your HTML code, you can mark up sections of content based on their *functionality* — you can use elements that represent the sections of content described above unambiguously, and assistive technologies like screenreaders can recognise those elements and help with tasks like "find the main navigation", or "find the main content." As we mentioned earlier in the course, there are a number of consequences of not using the right element structure and semantics for the right job.

To implement such semantic mark up, HTML provides dedicated tags that you can use to represent such sections, for example:

- **header:** `<header>`.
- **navigation bar:** `<nav>`.
- **main content:** `<main>`, with various content subsections represented by `<article>`, `<section>`, and `<div>` elements.
- **sidebar:** `<aside>`; often placed inside `<main>`.
- **footer:** `<footer>`.

Active learning: exploring the code for our example

Our example seen above is represented by the following code (you can also find the example in our GitHub repository). We'd like you to look at the example above, and then look over the listing below to see what parts make up what section of the visual.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">

    <title>My page title</title>
    <link href="https://fonts.googleapis.com/css?family=Open+Sans+Condense
    <link rel="stylesheet" href="style.css">

    <!-- the below three lines are a fix to get HTML5 semantic elements wo
    <!--[if lt IE 9]>
      <script src="https://cdnjs.cloudflare.com/ajax/libs/html5shiv/3.7.3/
    <![endif]-->
```

```
14    </head>
15
16    <body>
17      <!-- Here is our main header that is used across all the pages of our w
18
19      <header>
20        <h1>Header</h1>
21      </header>
22
23      <nav>
24        <ul>
25          <li><a href="#">Home</a></li>
26          <li><a href="#">Our team</a></li>
27          <li><a href="#">Projects</a></li>
28          <li><a href="#">Contact</a></li>
29        </ul>
30
31         <!-- A Search form is another commmon non-linear way to navigate thr
32
33         <form>
34           <input type="search" name="q" placeholder="Search query">
35           <input type="submit" value="Go!">
36         </form>
37       </nav>
38
39      <!-- Here is our page's main content -->
40      <main>
41
42        <!-- It contains an article -->
43        <article>
44          <h2>Article heading</h2>
45
46          <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Donec
47
48          <h3>Subsection</h3>
49
50          <p>Donec ut librero sed accu vehicula ultricies a non tortor. Lorer
51
52          <p>Pelientesque auctor nisi id magna consequat sagittis. Curabitur
53
```

```
        <h3>Another subsection</h3>

        <p>Donec viverra mi quis quam pulvinar at malesuada arcu rhoncus. C

        <p>Vivamus fermentum semper porta. Nunc diam velit, adipscing ut tr
      </article>

      <!-- the aside content can also be nested within the main content --
      <aside>
        <h2>Related</h2>

        <ul>
          <li><a href="#">Oh I do like to be beside the seaside</a></li>
          <li><a href="#">Oh I do like to be beside the sea</a></li>
          <li><a href="#">Although in the North of England</a></li>
          <li><a href="#">It never stops raining</a></li>
          <li><a href="#">Oh well...</a></li>
        </ul>
      </aside>

    </main>

    <!-- And here is our main footer that is used across all the pages of c

    <footer>
      <p>©Copyright 2050 by nobody. All rights reversed.</p>
    </footer>

  </body>
</html>
```

Take some time to look over the code and understand it — the comments inside the code should also help you to understand it. We aren't asking you to do much else in this article, because the key to understanding document layout is writing a sound HTML structure, and then laying it out with CSS. We'll wait for this until you start to study CSS layout as part of the CSS topic.

HTML layout elements in more detail

It's good to understand the overall meaning of all the HTML sectioning elements in detail — this is something you'll work on gradually as you start to get more experience with web development. You can find a lot of detail by reading our HTML element reference. For now, these are the main definitions that you should try to understand:

- `<main>` is for content *unique to this page.* Use `<main>` only *once* per page, and put it directly inside `<body>`. Ideally this shouldn't be nested within other elements.
- `<article>` encloses a block of related content that makes sense on its own without the rest of the page (e.g., a single blog post).
- `<section>` is similar to `<article>`, but it is more for grouping together a single part of the page that constitutes one single piece of functionality (e.g., a mini map, or a set of article headlines and summaries). It's considered best practice to begin each section with a heading; also note that you can break `<article>`s up into different `<section>`s, or `<section>`s up into different `<article>`s, depending on the context.
- `<aside>` contains content that is not directly related to the main content but can provide additional information indirectly related to it (glossary entries, author biography, related links, etc.).
- `<header>` represents a group of introductory content. If it is a child of `<body>` it defines the global header of a webpage, but if it's a child of an `<article>` or `<section>` it defines a specific header for that section (try not to confuse this with titles and headings).
- `<nav>` contains the main navigation functionality for the page. Secondary links, etc., would not go in the navigation.
- `<footer>` represents a group of end content for a page.

Non-semantic wrappers

Sometimes you'll come across a situation where you can't find an ideal semantic element to group some items together or wrap some content. Sometimes you might want to just group a set of elements together to affect them all as a single entity with some CSS or JavaScript. For cases like these, HTML provides the `<div>` and `<span>` elements. You should use these preferably with a suitable `class` attribute, to provide some kind of label for them so they can be easily targeted.

`<span>` is an inline non-semantic element, which you should only use if you can't think of a better semantic text element to wrap your content, or don't want to add any specific meaning. For example:

```
<p>The King walked drunkenly back to his room at 01:00, the beer doing notl
him as he staggered through the door <span class="editor-note">[Editor's no
play, the lights should be down low]</span>.</p>
```

In this case, the editor's note is supposed to merely provide extra direction for the director of the play; it is not supposed to have extra semantic meaning. For sighted users, CSS would perhaps be used to distance the note slightly from the main text.

`<div>` is a block level non-semantic element, which you should only use if you can't think of a better semantic block element to use, or don't want to add any specific meaning. For example, imagine a shopping cart widget that you could choose to pull up at any point during your time on an e-commerce site:

```
<div class="shopping-cart">
  <h2>Shopping cart</h2>
  <ul>
    <li>
      <p><a href=""><strong>Silver earrings</strong></a>: $99.95.</p>
      <img src="../products/3333-0985/thumb.png" alt="Silver earrings">
    </li>
    <li>
      ...
    </li>
  </ul>
  <p>Total cost: $237.89</p>
</div>
```

This isn't really an `<aside>`, as it doesn't necessarily relate to the main content of the page (you want it viewable from anywhere). It doesn't even particularly warrant using a `<section>`, as it isn't part of the main content of the page. So a `<div>` is fine in this case. We've included a heading as a signpost to aid screenreader users in finding it.

Warning: Divs are so convenient to use that it's easy to use them too much. As they carry no semantic value, they just clutter your HTML code. Take care to use them only when there is no better semantic solution and try to reduce their usage to the minimum otherwise you'll have a hard time updating and maintaining your documents.

Line breaks and horizontal rules

Two elements that you'll use occasionally and will want to know about are `<br>` and `<hr>`:

`<br>` creates a line break in a paragraph; it is the only way to force a rigid structure in a situation where you want a series of fixed short lines, such as in a postal address or a poem. For example:

```
<p>There once was a man named O'Dell<br>
Who loved to write HTML<br>
But his structure was bad, his semantics were sad<br>
and his markup didn't read very well.</p>
```

Without the `<br>` elements, the paragraph would just be rendered in one long line (as we said earlier in the course, HTML ignores most whitespace); with `<br>` elements in the code, the markup renders like this:

There once was a man named O'Dell
Who loved to write HTML
But his structure was bad, his semantics were sad
and his markup didn't read very well.

`<hr>` elements create a horizontal rule in the document that denotes a thematic change in the text (such as a change in topic or scene). Visually it just looks like a horizontal line. As an example:

```
<p>Ron was backed into a corner by the marauding netherbeasts. Scared, but
<hr>
<p>Meanwhile, Harry was sitting at home, staring at his royalty statement a
```

Would render like this:

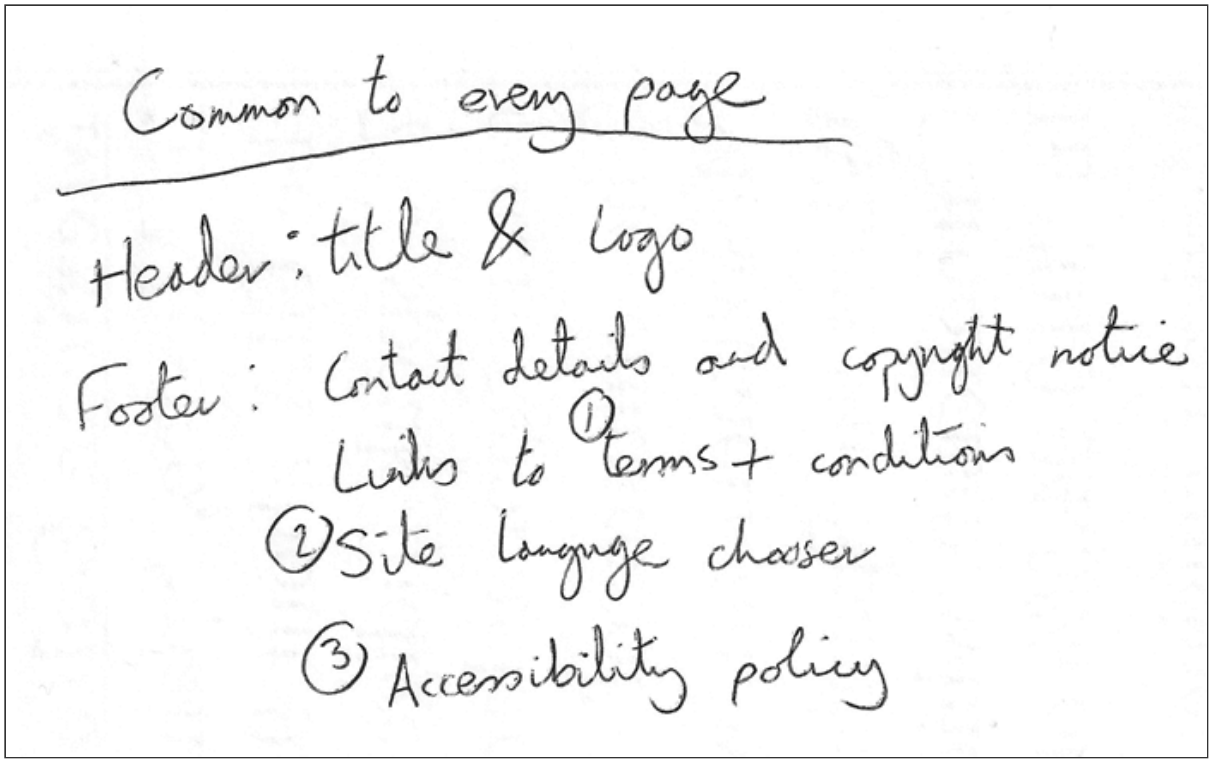
Ron was backed into a corner by the marauding netherbeasts. Scared, but determined to protect his friends, he raised his wand and prepared to do battle, hoping that his distress call had made it through.

Meanwhile, Harry was sitting at home, staring at his royalty statement and pondering when the next spin off series would come out, when an enchanted distress letter flew through his window and landed in his lap. He read it hazily and sighed; "better get back to work then", he mused.

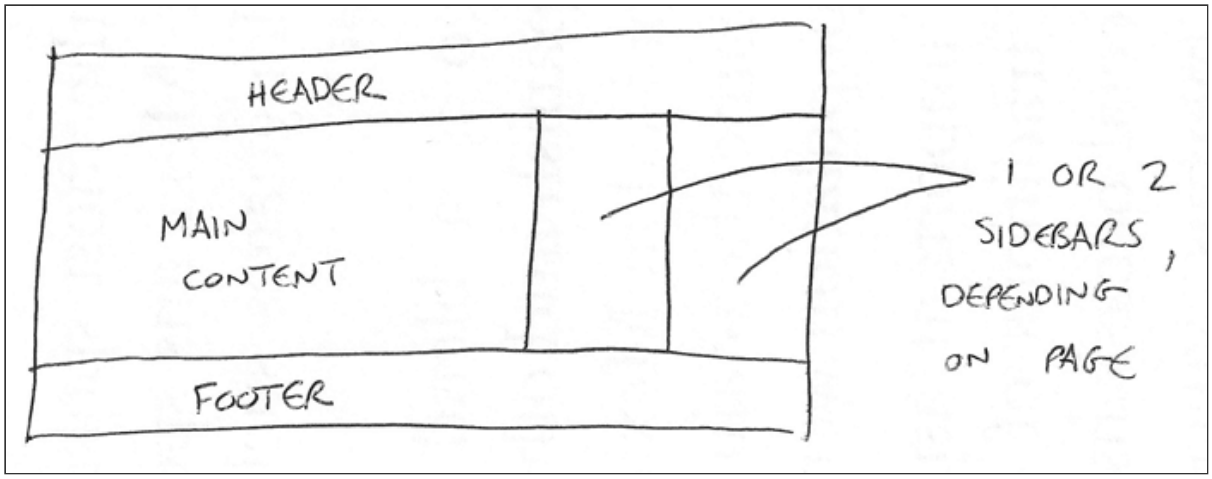
Planning a simple website

Once you've planned out the structure of a simple webpage, the next logical step is to try to work out what content you want to put on a whole website, what pages you need, and how they should be arranged and link to one another for the best possible user experience. This is called Information architecture. In a large, complex website, a lot of planning can go into this process, but for a simple website of a few pages, this can be fairly simple, and fun!

1. Bear in mind that you'll have a few elements common to most (if not all) pages — such as the navigation menu, and the footer content. If your site is for a business, for example, it's a good idea to have your contact information available in the footer on each page. Note down what you want to have common to every page.



2. Next, draw a rough sketch of what you might want the structure of each page to look like (it might look like our simple website above). Note what each block is going to be.



3. Now, brainstorm all the other (not common to every page) content you want to have on your website — write a big list down.

Search for Flights

Hotels / other accommodation

Transport

Things to do

Special offers

Popular holiday packages

e.g. Winter sun

Disneyworld

Skiing

Search results

Country - specific info

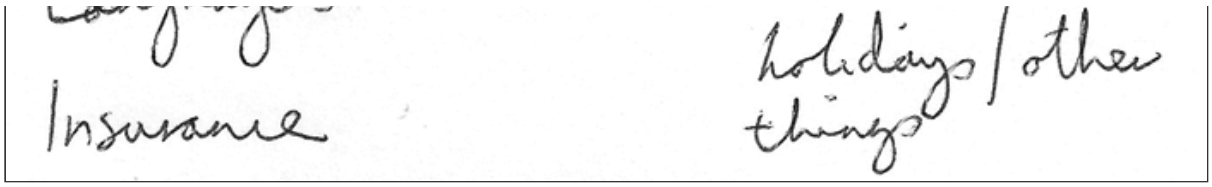
Accommodation / attraction reviews

Visa / entry requirements

Money / Currency

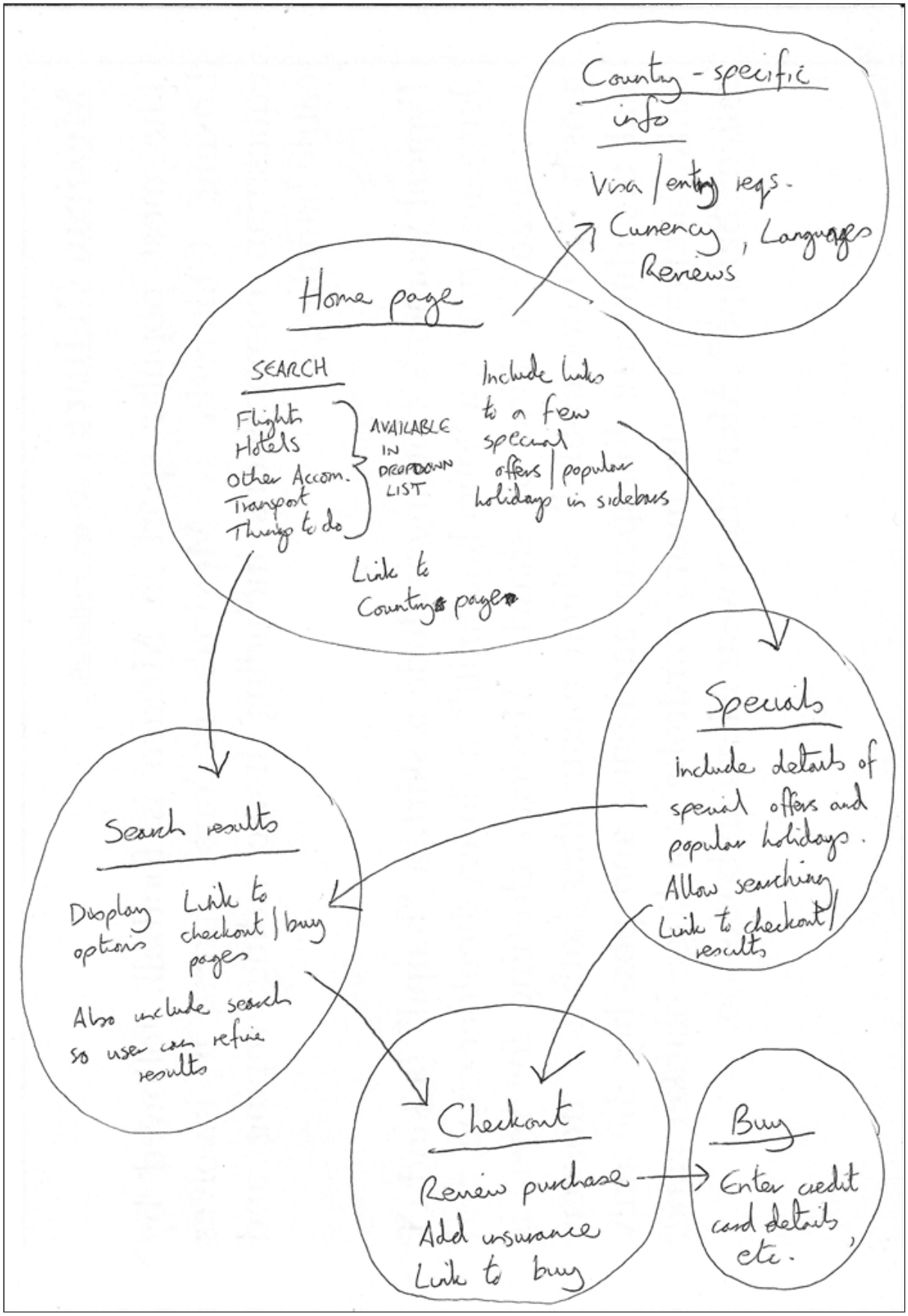
Buy

Language



4. Next, try to sort all these content items into groups, to give you an idea of what parts might live together on different pages. This is very similar to a technique called Card sorting.

5. Now try to sketch a rough sitemap — have a bubble for each page on your site, and draw lines to show the typical workflow between pages. The homepage will probably be in the center, and link to most if not all of the others; most of the pages in a small site should be available from the main navigation, although there are exceptions. You might also want to include notes about how things might be presented.



Active learning: create your own sitemap

Try carrying out the above exercise for a website of your own creation. What would you like to make a site about?

> **Note**: Save your work somewhere; you might need it later on.

Test your skills!

You've reached the end of this article, but can you remember the most important information? You can find a detailed assessment that tests these skills at the end of the module; see Structuring a page of content. We'd advise going through the next article in the series first and not just skipping to it though!

Summary

At this point you should have a better idea about how to structure a web page/site. In the last article of this module, we'll study how to debug HTML.

See also

- Using HTML sections and outlines: Advanced guide to HTML5 semantic elements and the HTML5 outline algorithm.

Previous | Overview: Introduction to HTML | Next

In this module

Last modified: Feb 5, 2020, by MDN contributors

Related Topics

Complete beginners start here!

- Getting started with the Web

HTML — Structuring the Web

- ▼ Introduction to HTML
  - Introduction to HTML overview
  - Getting started with HTML
  - What's in the head? Metadata in HTML
  - HTML text fundamentals
  - Creating hyperlinks
  - Advanced text formatting
  - Document and website structure
  - Debugging HTML
  - Assessment: Marking up a letter
  - Assessment: Structuring a page of content
- ▶ Multimedia and embedding
- ▶ HTML tables
- ▶ HTML forms

CSS — Styling the Web

- ▶ CSS first steps
- ▶ CSS building blocks
- ▶ Styling text
- ▶ CSS layout

JavaScript — Dynamic client-side scripting

- ▶ JavaScript first steps
- ▶ JavaScript building blocks
- ▶ Introducing JavaScript objects
- ▶ Asynchronous JavaScript

- Client-side web APIs

Accessibility — Make the web usable by everyone

- Accessibility guides
- Accessibility assessment

Tools and testing

- Cross browser testing

Server-side website programming

- First steps
- Django web framework (Python)
- Express Web Framework (node.js/JavaScript)

Further resources

- Common questions

How to contribute