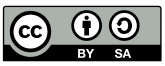
Tags and style



2019-05-06 22:54
1a0f14a

Contents

Tag inventory

We can divide tags into whether they are typically used for block display or inline display.

Block tags

Block display means that the boundaries extend as far as possible in the horizontal dimension, and these blocks flow vertically.

- `<h1>`, `<h2>`, `<h3>`, … for headings
- `<p>` for paragraph
- `<ul>`, `<ol>` for unordered, ordered lists, and `<li>` for list items
- `<figure>` indicates an illustration or embedded image, possibly with a caption
- `<nav>` no particular style, but indicates some navigation element such as a menu
- `<header>`, `<footer>` no particular style, but indicate introductory or closing content
- `<section>`, `<article>` represents a section of a document, or a post an a blog/news site
- `<div>` totally generic block content, usually used with an `id` and/or `class` attribute

Inline tags

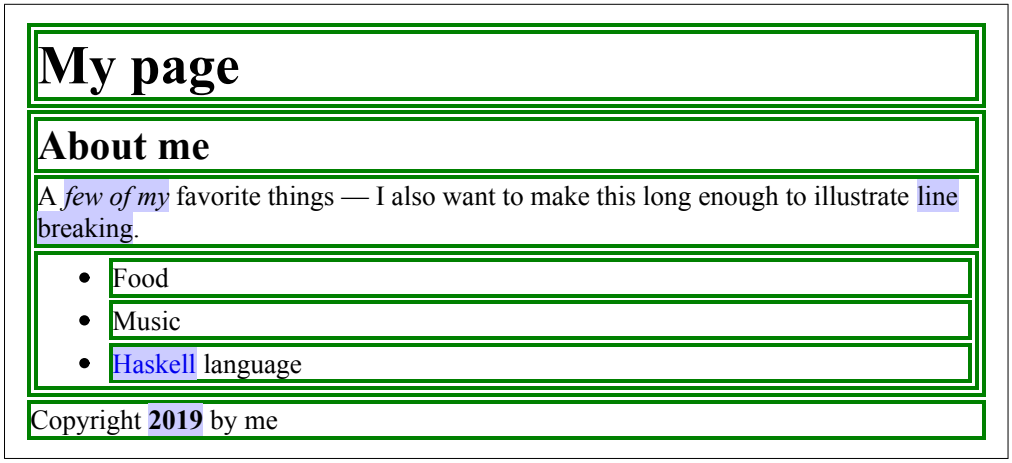
Inline display means that the boundary extends only as far as the content, and the elements flow horizontally and are subject to line breaks.

- `<b>` or `<strong>` for bold-face (visual) or important (semantic)
- `<i>` or `<em>` for italic (visual) or emphasis (semantic)
- `<a>` for hyperlinks
- `<img>` for embedded images, which do flow horizontally unless wrapped in a block
- `<span>` totally generic inline content,

Illustrate block vs inline

We can specify styles to show the difference between block and inline elements. We'll use green solid lines for blocks and a light blue background for inline elements.

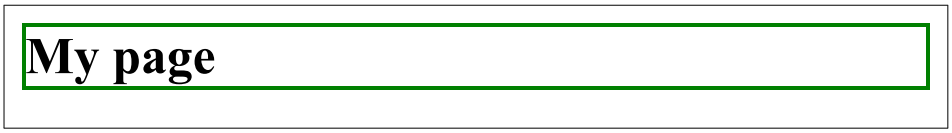
```
<style>
  header, section, footer, h1, h2, p, ul, li, div {
    border: 2px solid green;
    margin: 2px;
  }
  a, i, b, span {
    background: #ccf;
  }
</style>
<header>
  <h1>My page</h1>
</header>
<section>
  <h2>About me</h2>
  <p>A <i>few of my</i> favorite things &mdash; I also want to make
    this long enough to illustrate <span>line breaking</span>.
  </p>
  <ul>
    <li>Food</li>
    <li>Music</li>
    <li><a href="https://haskell.org/">Haskell</a> language</li>
  </ul>
</section>
<footer>
  Copyright <b>2019</b> by me
</footer>
```



Style rules

- CSS = Cascading Style Sheets
- The CSS rules can be embedded in the HTML, usually within the `<head>` (but it's not required) and within a `<style>` block (which is required). Example:

```
<style>
h1 { border: 2px solid green; }
</style>
<h1>My page</h1>
```

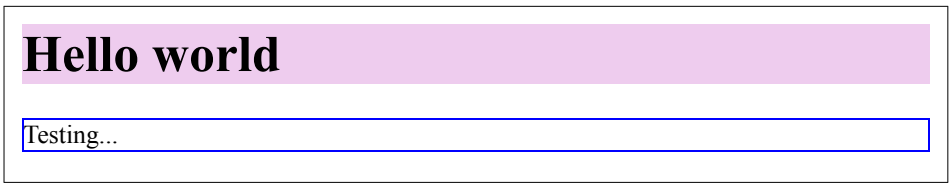


- CSS rules can also be loaded from an external stylesheet, specified using a filename or URL. These `<link>` elements also usually appear within `<head>`.

```
<head>
  <link rel="stylesheet" href="mystyle.css">
  <link rel="stylesheet"
        href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.5/
</head>
```

- Finally, you can also apply style properties to a *particular* element by placing them in a `style` attribute:

```
<h1 style="background:#ece">Hello world</h1>
<p style="border:1px solid blue">Testing...</p>
```



`id` and `class` attributes

The CSS rules we've seen so far just match an element by its tag name, such as:

```
h1 { border: 2px solid green; }
span { background: #ccf; }
```

We can distinguish elements from each other using `class` and `id` attributes. These contain identifiers (alphanumeric, including hyphen or underscore) that can be used to distinguish roles and styles.

For example, you could distinguish:

```
<h2 class="optional">Section title</h2>
<div id="sidebar">
...
</div>
<p class="big warning">
...
</p>
```

In the case of the <p> tag above, the identifiers `big` and `warning` are *two separate* classes (because identifiers cannot have spaces).

We can then designate some rules that match based on the identifier and class, with or without tag name. We use the # (hash) character to specify an `id`, or `.` (dot) character to specify a `class`. For example:

```
h2.optional { background: #ffc; }
p.big { font-size: 140%; }
.warning { font-weight: bold; color: red; }
#sidebar { ... }
```

In the first two rules, we require a match of both tag name and class name. In the third rule, it's just the class name and can be applied to any tag. In the final rule, it's just the `id`.

The main difference between `id` and `class` is that the former are meant to be **unique** across the page. There should be *only one* element that matches `#sidebar`, but there could be many elements matching `.warning`.

Demo page

Here is the extended example that we ended up with in class. First I'll put the contents of `style.css`:

```
div#sidebar {
  width: 25%;
  float: left;
  margin-right: 20px;
  margin-bottom: 20px;
}
div#content {
  margin-left: 25%;
}
h1 {
  border: 1px solid green;
```

```
}
div#footer {
  margin-top: 50px;
  border-top: 5px solid #aaa;
}
div.severe {
  font-size: 125%;
}
div.warning {
  font-weight: bold;
  color: red;
  background: pink;
  padding: 10px;
}
```

And then the `index.html`:

```
<!DOCTYPE HTML>
<html lang="en">
  <head>
    <title>My new page</title>
    <link rel="stylesheet"
          href="./data/33/ff506f-7e4e-4d58-9000-718280ab057f/style.css">
  </head>
  <body>
    <div id="sidebar">
      We can put sidebar content here. It could be navigation or
      social media integration, etc.
    </div>
    <div id="content">
      <h1>My new page</h1>
      <p>This is a tale of <b>my travels</b> over the <i>past
          year.</i></p>
      <ul>
        <li>Norway</li>
        <li>Finland</li>
        <li>Japan</li>
      </ul>
      <p>Prioritized list of foods to try:</p>
      <ol type="i">
        <li>Ramen</li>
        <li>Fresh fish from the Helsinki market.</li>
      </ol>
```

```
      <p>Some tags represent block content, other tags represent
        inline content.
      </p>
      <div class="severe warning">
        Use this advice at your own risk.
      </div>
    </div>
    <div id="footer">
      "div" is a generic container for block content.
      <span>"span" is a generic container for inline content.</span>
    </div>
  </body>
</html>
```

✅ valid HTML