HTML and page layout

John R Hudson

18 July 2021

1 Introduction

1.1 Continuous flow

The second word-processing program for microcomputers, WordStar, adopted the continuous flow approach to layout. This was in part because it was aimed at typists and secretaries used to typing continuously without lifting their fingers from the keyboard.

Interruptions to the continuous flow such as:

- absolute page boundaries

- conditional page boundaries — declaring the page boundary if there were less than a specified number of lines left in the page

- child documents

- data — as when individualised story books were produced for children with their names in the story

- images and

- columns

were all treated as interruptions to the continuous flow after which the continuous flow resumed.

WordStar even allowed you to script the insertion of child documents and data or to create user prompts as to whether to insert particular child documents or data; unless specifically directed not to do so, WordStar gave child documents and data the same formatting as the main document. So there was no need to format child documents or bother about the variable length of data; WordStar would work out how to integrate them seamlessly into the main document.

This approach also reflected the approach of traditional typesetters when typesetting books and was adopted by TeX, the typesetting program written by Donald Knuth which sought to bring traditional typesetting to computers.

Figure 1: Page based layout

1.2 Page based

However, there was an equally long-standing approach in the newspaper industry: page based, in which material is typeset to fit on a single page, which was adopted by desktop publishing programs such as QuarkXpress, Adobe InDesign and Scribus. Everything is held in separate containers for the different content types such as text, images and tables. Provision is made for text to flow from one page to another but only from one container on one page to another container on another page.

Figure 1 offers an example of a page based layout following traditional lines created in Scribus. Note that there are headlines at the top of each page and sub-headers throughout to draw the reader in; also there are no horizontal or vertical breaks across the main body of each page.

2 HTML

Initially HTML presented itself as continuous flow but, perhaps under the influence of desktop publishing programs, by the early years of the 21st century, using tables and `<div>` elements to divide up the page became the norm. This appeared to work well but using tables and `<div>` elements almost inevitably violated the long established principle that there should be no horizontal or vertical breaks across the main body of each page.

So even before the arrival of the iPhone in 2007, Apple, Opera and Mozilla had decided that
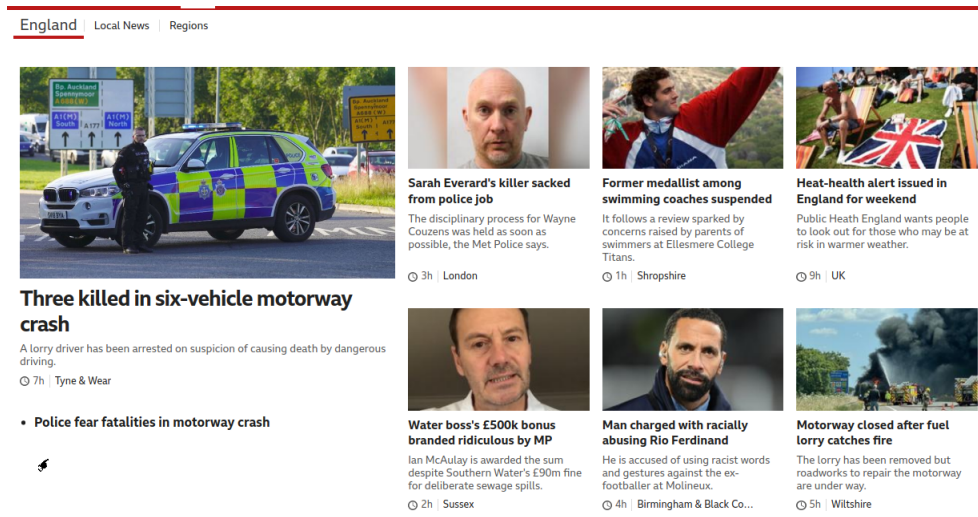
Figure 2: BBC website on 2007 computer

this was not the best way to take HTML forward and begun to work on what was to become HTML5. This went back to the 1990s principle that content and styling should be separate; so using tables and `<div>` elements for styling was deprecated. They also recognised that the geometry of devices had changed dramatically and so table layouts which worked well on the almost square devices of the early part of century no longer worked on the more oblong devices of the second decade of the century. I have a computer from 2007 with an almost square screen on which the BBC website fits perfectly (figure 2) whereas on my later computers there is lots of white space on either side of most pages because the BBC website does not expand to fill the available space but assumes it is still displaying on a device from the early 21st century.

The City of Bradford website similarly displays with lots of white space and its HTML code is full of `<div>`s (figure 3); there are at least seven different `<div>`s defined in this short snippet including one for 'accessibility' which is redundant at the website does not display in a screenreader!

Since 2011 most HTML elements have been defined by their content and styling is left up to CSS. There are a few exceptions where using an HTML tag changes the styling; for example,

- the content of an `<i>` tag normally displays in italics as it is intended to hold foreign words with, for example, `<i la="fr">` indicating that the content of the tag is in French — this is useful for warning screenreaders about a change in language;

- the content of a `<cite>` tag normally displays in italics as it is the title of a book and

- the content of a `<code>` tag normally displays in `monospace` because it is code.

The designers of HTML5 found that most websites displayed five elements:

- a header,

- a footer,

- a navigation bar or pane,

```
<body>
    <header aria-label="BMDC Print logo">
        <div class="BMDClogo visible-print">
            <!--<img src="~/css/images/logoBMDCMob.png" alt="City of Bradford Metropolitan District Council logo" />-->
        </div>
    </header>
    <div class="navbar navbar">

        <div class="container">


            <div class="topHeader">
                <div id="skiptocontent" role="navigation" aria-labelledby="skiptocontent"><a href="#maincontent">skip to main content</a></div>

                <header aria-label="BMDC Logo">
                    <a href="/">
                        <div class="BMDClogo">
                            <!--<img src="~/css/images/logoBMDCMob.png" alt="City of Bradford Metropolitan District Council logo" />-->
                        </div>
                    </a>
                </header>

                <div class="browseAloud">
                    <div id="__ba_launchpad"></div>
                </div>
                <div class="topRightHeader">

                    <div class="accessibility">

                        <div role="complementary" aria-label="Language selection">
                            <div id="google_translate_element"></div>
```
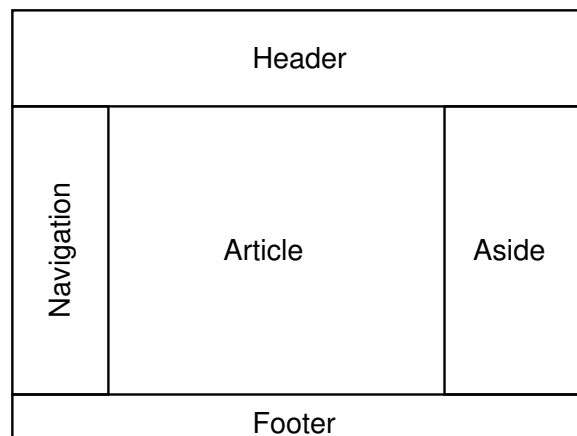
Figure 3: HTML code from City of Bradford website

- an element with related material which was not part of the main content and

- the main content

and they gave these five elements the names: `<header>`, `<footer>`, `<nav>`, `<aside>` and `<article>`.

Header		
Navigation	Article	Aside
Footer		

Placing the `<nav>` and `<aside>` elements on either side of the main content helped to keep the line length of the main content down on more oblong devices, albeit at the expense of white space below the `<nav>` and `<aside>` elements where the main content was lengthy.

Subsequently, a `<main>` element was added partly to point users with disabilities to the main part of the web page and partly to open up a range of more sophisticated layouts using CSS containers than had ever been possible using tables and `<div>`.

The contrasts between the City of Bradford website and the Heath Old Boys Association website could hardly be starker (figure 4):

Not a single `<div>` in sight; instead, after the `<header>`, there is a `<main>` element serving two purposes — pointing screenreaders to the main content of the page and providing a CSS container for the `<article>`, `<aside>` and `<nav>` elements. It is self-evident what each element

```
<html lang="en"/>
    <head>
        <meta charset="UTF-8"/>
        <title>Heath Old Boys Association Welcome</title>
        <base href="http://heatholdboys.org.uk/">
        <link rel="stylesheet" type="text/css" href="hoba.css"/>
        <meta name="author" content="John Robert Hudson" >
    </head>
    <body role="document">
        <header role="banner">
            <h1>Heath Old Boys Association</h1>
            <br>
            <h2>Welcome</h2>
            <br>
        </header>
        <main role="main">
            <article role="article">
                <h3>Welcome to the Heath Old Boys Association website</h3>
                <figure class="left" aria-label="Heath Grammar School Crest">
                    <img src="graphics/crest.png" width="100" height="128" alt="" role="presentation">
                </figure>
                <p class="first">The Association exists to keep old boys of the former Heath Grammar School in touch
                <p>If you are a Heath Old Boy but not a member of the Association, please <a href="content/Join.ht
                <p>When the site was originally set up in 2003 there were only 820 names on our newsletter database
                <p>In the meantime the Association does all it can to help and support the new Crossley Heath Scho
                <p>In 2014 in was agreed that Savile Park Primary School would begin to move into the old Heath Gr
                <p>We make periodic donations to a variety of local good causes, for example, sport for less privi
                <p>The Association has an annual dinner (usually Autumn/Winter), an annual Bowls competition, an a
                <p>Do get in touch and let us know about yourself, your experiences and your memories as well as t
                <p>Our privacy statement is on our <a href="content/Contact.html" type="text/html" role="link">Con
            </article>
            <aside role="complementary" aria-labelledby="recent">
                <h3><a href="content/Searching.html" type="text/html" role="link">Searching the website</a></h3>
                <hr>
                <h3 id="recent">Recent additions</h3>
                <ul role="list">
```

Figure 4: HTML code from Heath Old Boys Association website

does because they all carry out the functions defined for them in post-2011 HTML. Figure 5 shows what this page looks like to a screenreader.
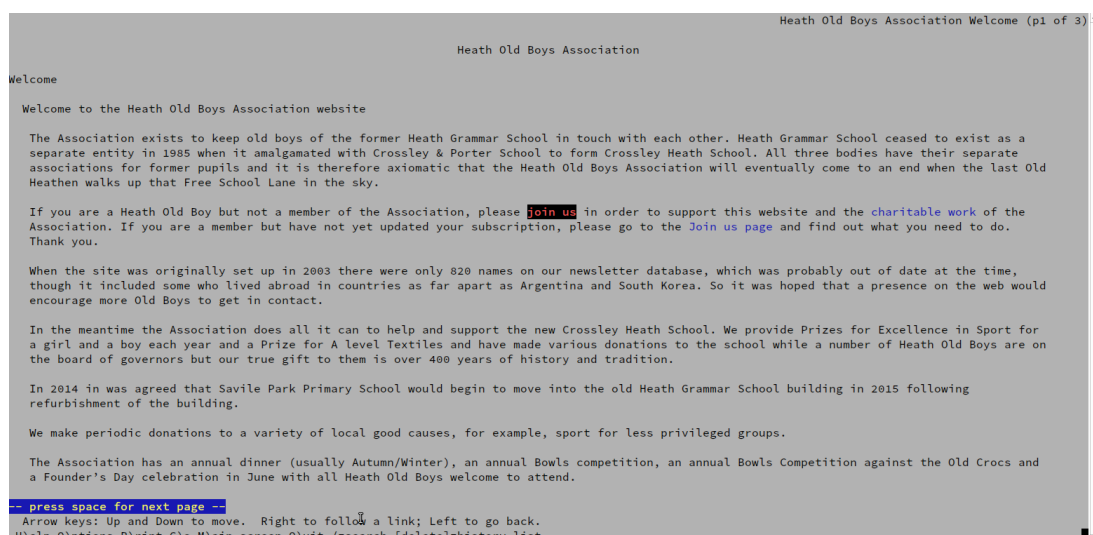
5

Figure 5: HOBA website as seen by a screenreader

Since 2011 an enormous range of styling features has been added in CSS (see, for example, CSS Snapshot 2020) which makes using tables and `<div>`s look primitive by comparison. However, these follow the principle of continuous flow; so someone coming to CSS containers, for example, from a desktop publishing perspective may be perplexed by their behaviour whereas someone coming to them from a continuous flow perspective will understand what is going on instinctively.