Chapter 3, Page Apperance and CSS

John M. Morrison

August 10, 2017

Contents

0 Introduction

So far, your pages have very little personality. They are black-and-white and all text is formatted with the default options. This is as it should be.

Now we will deal with a second language whose grammar is different from that of HTML, that of *Cascading Style Sheets* (CSS). The purpose of this language is to control the appearance of web pages. So a web page has two layers: HTML provides the structure layer and CSS provides the presentation layer.

Many of the references we saw in the last chapter will be of a great deal of use to you. This includes such things as [4] and [3]. The Mozilla Developer Network [1] also contains a ton of useful resources. A little poking around in it can pay off in big ways. You are also encouraged to become a member of [2]; this is an extremely useful forum and reference. It is highly searchable and it is prowled by some very smart people who are wonderfully generous with their knowledge.

1 Getting Started with CSS

We show a very simple CSS File here.

```css
h1, h2
{
    text-align:center;
}
body
{
    background-color:blue;
    color:red;
}
```

Here we see style rules that govern the headline tags `h1` and `h2`, and the `body` tag. We see statements of the following form.

```
selector
{
    property1:value1;
    property2:value2;
    .
    .
    .
    propertyN:valueN;
}
```

The selector is so-called because it selects page elements. The most basic type of selector consists of one or more tag types. If there are several tag types, use a comma-separated list as we did for h1 and h2 in our example. The purpose of the selector is to select the elements listed; in the style rule

```
h1, h2
{
    text-align:center;
}
```

all h1 and h2 elements are selected and the style rules listed are in force in those elements. When you begin to use CSS, you can see why it is very desirable that your document be well–formed; it is important for style rules to begin and end in the right places.

Curly braces bound the list of style rules belonging to each selector. Each style rule is ended with a semicolon. Omit this and experience pain. Every tag in HTML has a list of admissible properties. For example, the body tag has the properties color and background-color. The color property controls the color of text on the page and the background-color property controls the background color of the page. Elements that bound text, such as headline elements or paragraphs have the property text-align which has possible values left, right, and center.

You can check the validity of your CSS with the CSS Validator in WebDeveloper. It is under the Tools menu. The remarks made about reading error messages for web pages apply here, too. If you have syntax errors in your CSS, your style will not show properly on the page. Use the CSS validator to locate these and extirpate them.

1.1 How do I Make CSS Work on a Page?

There are three ways to use style rules on a page. They are as follows.

- **Local Style Sheet** You can impose style rules on any element on a page by using the style attribute. For example if you do this

  ```
  <p style="color:red;">  Some text</p>
  ```

  all of the text in the paragraph element will be red. The *scope*, or lifetime, of this style rule is confined to this one paragraph element. More generally, the scope of any attribute is confined to the element bounded by its tag. Note that if a rule applies to an element, it "cascades" down onto the elements inside of that element. We have seen a few examples of this already.
- **Page Style Sheet** In the head of your document, you can place a style sheet inside of a style element. The scope of these style rules is the one

page. A typical application looks like this

```
<style type="text/css">
h1, h2
{
    text-align:center;
}
</style>
```

On this page, all text in `h1` and `h2` headlines will be centered.

- **External Style Sheet** You can create an external style sheet in a file with a `.css` extension. Using the self-closing `link` tag in the head of the document, you can link the style rules from this file to your document. In this manner, one style sheet can control the appearance of many pages. If you wish to link the file `myStyle.css`, you would place this in the head of your document.

  ```
  <link rel="stylesheet", type="text/css"
      href="myStyle.css"/>
  ```

  You should place any `style` element after linking any external style sheets. You can link several sheets by using several `link` tags.

1.2 Precedence Rules

Let us learn about these rules by seeing an example. Begin by creating this page and naming it `bareBones.html`.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<link rel="stylesheet" type="text/css" href="myStyle.css"/>
<title>Page with CSS</title>
</head>
<body>
<h2>Page with CSS</h2>
</body>
</html>
```

Then create this file, `myStyle.css` in the same directory. Using a `.css` extension for style files causes `vi` or any other good text editor to give you productivity-enhancing syntax coloring.

```
body
{
    background-color: #FFFFCC;
```

```
}
h1, h2
{
    text-align: center;
}
```

The style sheet `myStyle.css` is linked to our page `bareBones.html` so it is in force throughout the page. We can link this style sheet to as many pages as we would like. Make sure your style sheet has 644 permissions so it is visible to the world. Otherwise, the client browser cannot fetch it and use it to style your page.

Notice the use of a 24 bit hexadecimal integer (`FFFFCC`) to describe a color; CSS grammar requires a pound-sign (`#`) be placed as a prefix to this hexadecimal integer. We will take a brief detour in the next section to discuss color. For now, just know you can use any 6 digit hex number to specify a color, and that you can call some colors by name, (e.g. blue, red, green, and some others). See if you can fiddle with the hex code and try to figure out how it works before it explained to you. See what kind of color names you can get away with using.

You can view this page by typing

```
http://yourURL/bareBones.html
```

where you place your URL in lieu of `yourURL`, or if you are working offline, by opening it via the file menu in the browser. The style file `myStyle.css` controls the page's appearance. Change the hex code in the style file, reload your page and see the background color change! Experiment with some named colors as values for `background-color`. Type some stuff inside of an `h1` or `h2` tag. Watch it *automatically* center.

The main rule of precedence is simple: *The most local rule prevails.* Local rules have the highest precedence and the smallest scope. Page style sheets only are in force on a single page. You can use them to override style rules you get by linking in external style sheets. The scope of a page style sheet is just the page it is placed in. Finally, external style sheets can govern many pages on a website. They have the biggest scope and the lowest precedence.

Programming Exercise Do some Googling and see if you can find a list of named colors your browser will accept. Try making them appear on your page.

2 Color

One reason we discussed hex numbers is that they are used to specify color. The representation of color you learn about here is used in virtually all modern computing languages, so the usefulness of this section far transcends the purposes we are discussing here.

Color on web pages lies in the purview of CSS. If you are dealing with colors, you should be doing it in the context of a style sheet.

There are three primary colors of light: red, green and blue. This is different from the primary colors of pigment: red, yellow and blue. This is because pigment is a *subtractive* phenomenon, and the colors of light are an *additive* phenomenon; it is a physical phenomenon that involves the combination of wavelengths. The modern standard for rendering color on a computer is called *24-bit* color.

The 24 bits are organized into six hex digits; remember, each hex digit can be expanded to 4 bits. A typical color hex code looks like this: `0xFF0000`. Remember, the 0x prefix simply says, "This is a hex number." Colors are represented with the "RGB", or red-green-blue system. The first two hex digits tell how much red is in the color. The second pair controls green, and the last pair controls blue. The hex code we just showed, `0xFF0000` has `0xFF` parts of red, `0x0` (no) blue, and `0x0`(no) green. This hex code is the hex code for the color red. Now what does the `FF` mean? If you convert this to base 10, `0xFF = 255`.

Each of the three primary colors has 256 levels from 0 `(0x0)` up to 255 `(0xFF)`. You may use any hex code from `0x0` to `0xFF` for a color's level. As a result, you have a choice of 16,777,216 colors. Since the human eye only perceives about 3–5 million colors, you can see that 24-bit color does an excellent job of rendering color on a computer screen.

To see a color, change your minimum page by inserting a `style` attribute into the `body` tag as follows.

```
<body style="background-color:#FF0000;">
```

Save this in `vi` and then open your page with a browser. If the color is still white, the browser is using an old version of the page it has stored (caching). Hit the reload button and watch your page turn red. By changing the hex code in the body tag, you can see the color associated with any hex code. Note the usage for a hex code in the body tag: omit the `0x` and replace it with a `#`.

Exercises Try these things out so you get more comfortable with colors.

1. Change the hex code to `0x00FF00` and `0x0000FF` to see green and blue.
2. Change the hex code to `0xFFFF00`, `0xFF00FF`, `0x00FFFF, 0x000000,` and `0xFFFFFF`. What do you see? Do you know the names of all of these colors? Are you surprised by `0xFFFF00`?
3. Change your body tag to look like this

```
<body style="background-color:#FF0000; color:#0000FF;">
```

Here we are using two style rules for the `body` tag. Is the result aesthetically pleasing?

4. Experiment with mixing colors and seeing them on the screen. You can look on the web and see tables with colors and hex codes. Fool around with color and get used to it.

2.1 Named Colors

There are color names that are recognized by browsers. Different browsers recognize different names, but *all* browsers understand hex codes. This makes hex codes the preferred way to specify color. All browsers understand the colors red, green and blue. You can look on the web for more named colors. The usage for named colors looks like this.

```
<body style="background-color:blue; color:green">
```

3 divs, spans and Classes

Sometimes you will find you want an portion of a page to have a set of style rules applied to it that are different from the rest of the page. This brings us to two tags that exist to control scope for style rules: `div` and `span`. A `div` tag controls a vertical segment of a page; it is a block-level element. A `span` tag controls an in-line segment of a page; it is an inline element.

Suppose you want some text to appear in red on a page. You might do something like this.

```
This is a <span style="color:red">really hot topic</span>.
```

The text within the `span` element will be displayed in red.

Suppose you have a run of text you wish to have centered. Then you might do something like this.

```
<h2>The Gettysburg Address by A. Lincoln</h2>
<div style="text-align:center">
<p>Fourscore and seven years ago our fathers brought forth on
this continent a new nation, conceived in liberty, and
dedicated to the proposition that all men are created
equal.</p>

<p>Now we are engaged in a great civil war, testing whether
```

```
that nation, or any nation, so conceived and so dedicated, can
long endure. We are met on a great battle-field of that war. We
have come to dedicate a portion of that field, as a final
resting place for those who here gave their lives that that
nation might live. It is altogether fitting and proper that we
should do this.</p>

<p>But, in a larger sense, we can not dedicate, we can not
consecrate, we can not hallow this ground. The brave men,
living and dead, who struggled here, have consecrated it, far
above our poor power to add or detract. The world will little
note, nor long remember what we say here, but it can never
forget what they did here. It is for us the living, rather, to
be dedicated here to the unfinished work which they who fought
here have thus far so nobly advanced. It is rather for us to be
here dedicated to the great task remaining before usthat from
these honored dead we take increased devotion to that cause for
which they gave the last full measure of devotionthat we here
highly resolve that these dead shall not have died in vainthat
this nation, under God, shall have a new birth of freedomand
that government of the people, by the people, for the people,
shall not perish from the earth.</p>
</div>
```

3.1 Classes

Suppose you are creating a style sheet for a collection of pages and you want important text to be colored red. You could, for each segment of important text, do this.

```
This is a <span style="color:red">really hot topic</span>.
```

Next week your boss comes along and says, “Our focus groups find that red text judgmental. It reminds them of Mrs. Wormwood’s red grading pen. You have to make it green instead.” Ugh. You now must go through and change all of those local style sheets so the important text is green. What if there are hundreds of them? You have a tedious, time consuming, error-prone task in front of you. This brings us to an important issue.

The Eleventh Commandment *Thou shalt not maintain duplicate code.*

How do we avoid this kind of duplicate code horror? How do we escape the mind-numbing Kafkaesque world of unending search-and-replace? In our CSS we can do the following.

```
.important
{
    color:red;
}
```

We have created a *class* called `important`. To use it you can do this. The `.important` selector will select all elements with the attribute `class="important"`, and apply the style rules inside to them.

```
This is a <span class="important">really hot topic</span>.
```

With this mechanism, you can fulfill your boss's request by making a single change in the style sheet. Goodbye duplicate code. Hello easier page maintenance.

3.2 Giving Elements IDs

You can give a name to any element in your document by using an *id*. Here we will give a paragraph and id of `cows`.

```
<p id = "cows">  We bovines choose to ruminate at length on our
meals. It affords us the opportunity to bring up a pleasing
subject again.</p>
```

Selecting by ID is simple. In any style sheet, you can do this

```
#cows
{
    //style rules
}
```

The # sign triggers selection by ID and the style rules listed to be applied in that element. One caveat applies here: *An ID can only be used once on a page.* The beauty of the ID will really show when we study JavaScript and use the ID to grab and change an element on a page specified by ID.

4 CSS and Tables

There are three basic ways to present data in HTML: ordered lists, unordered lists, and tables. You can use styles to customize the appearance of all of these.

Ordered lists are, by default, enumerated with Arabic Numerals. They are delimited by the `ol` tag. Items in unordered lists are set off with a bullet character, •, by default; they are delimited by the `ul` tag. List items are each delimited by the `li` tag. Text may be placed directly inside of a `li` tag.

Below we show some code for each. An unordered list of presidents listed backwards looks like this.

```
<ul>
<li> Barack Obama</li>
<li> George W. Bush</li>
<li> William Clinton</li>
<li> George H. Bush</li>
<ul>
```

An ordered list of the same presidents looks like this.

```
<ol>
<li> Barack Obama</li>
<li> George W. Bush</li>
<li> William Clinton</li>
<li> George H. Bush</li>
<ol>
```

Place these in an HTML file and view them with your browser. Note their appearance. Go to `http://www.htmldog.com` and now look up styles for lists, and create a style sheet to change the appearances of both lists. Can you enumerate alphabetically? With roman numerals? Can you use an image as a "bullet?"

Tables are used for presenting data in HTML. You will see them sometimes used for page layout; this is not cricket. Use CSS for page layout.

Without CSS, tables are very plain. They just show their data in a very plain rectangular array. For your ready reference, here is a list of the pertinent tags.

- `<table>` This tag bounds the table element. You may use the `width` attribute to control the table's width. It is best to use a percentage to set width, as in `width = "50%"`, rather than using pixels or other units. Remember, you have no control over the size of the client's browser window.
- `<thead>` This can delimit an element for the table's header and you can attach style rules to it.
- `<tbody>` This can delimit an element for the table's body and you can attach style rules to it.
- `<tfoot>` This can delimit an element for the table's footer and you can attach style rules to it.

- `<tr>` This tag bounds a table row element. Tables are set row-by-row. Table datum and table header elements go inside of table row elements.
- `<th>` This tag bounds a table header element. By default, text is centered and boldface in a table header.
- `<td>` This tag bounds a table datum element. By default, text is plain and left-justified. For both table header and table row elements, the attribute `colspan` can be used to make a cell span more than one colum and `rowspan` can be use to make a table cell span more than one row.

The general structure of a table looks like this. Any table cell can be a header or a regular datum cell.

```
<table>
<tr><th>First Header</th><th>Second Header</th> .... </tr>.
<tr><td>First Datum</th><th>Second Datum</th> .... </tr>.

.
.
.

</table>
```

It is up to you to make sure you have the correct number of table cells in each row or the right-hand side of your table will be "ragged" and aesthetically unappealing.

To center a table, use this CSS Here is another useful example for table making.

```
table.center
{
    margin-left:auto;
    margin-right:auto;
}
```

When making an HTML table you type

```
<table class="center">
.
.
</table>
```

and your table will be centered on the page. By using the `table.center` notation, you restrict the use of this class to tables.

This CSS can be used as a starting point for styling tables. You should make a table, then experiment with the CSS to see what the various statements do.

```
table
{
    border-collapse:collapse;
    border: solid #000000 1px;
}
td, th
{
    border-collapse:collapse;
    border: solid #000000 1px;
}
th
{
    background-color:#FFFF00;
}
```

Exercise Create a web page that tells about you or one of your interests. Use good color and tasteful design. Validate it with the W3C validator.

Here are some free resources on the web for learning about web pages.

- `http://www.w3schools.com` This site has a huge array of tutorials on CSS, HTML5, JavaScript and other web site technologies. It features the "Try it" editor that allows you to easily enter HTML and preview it. Avail yourself of this and do lots of experimenting.
- `http://www.webmonkey.com` This site offers useful tutorials on an array of subjects.
- `http://www.w3c.org` This is the World Wide Web Consortium site.
- `https://html5.validator.nu/` This is the html5 validator.

5 The Box Model

Elements defined by self-closing tags are empty elements. Elements bouned by a matching open and close tag within the body form rectangles on the screen. CSS uses the *box model* to define rules about spacing around and with page elements. The following are important

content	This is the “good stuff” inside of the element.
`margin`	This is the spacing around the outside of an element. You can individually control the margin on the four sides with `margin-left`, `margin-right`, `margin-top`, and `margin-bottom`.
`padding`	This is spacing around the content that is inside of the element. You can individually control the padding on the four sides with `padding-left`, `padding-right`, `padding-top`, and `padding-bottom` e element.
`border`	This is the border that goes around the element. You can specify the border for the four sides using `border-left`, `border-right`, `border-top`, and `border-bottom`.

When you specify a with and a height, you specify the width and height of the content; the margin, border, and padding all add to this.

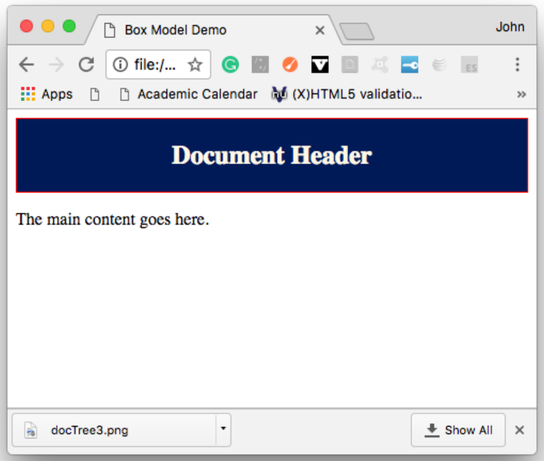
Create this document. YOu will see a `header` tag; this is just a named div that will go at the top of your document. The `main` tag bounds an elmeent containing the main content of the document.

```
<!doctype html>
<html lang="en">
<head>
<title>Box Model Demo</title>
<link rel="stylesheet" href="box.css">
<meta charset="utf=8"/>
</head>
<body>
<header>
<h2>Document Header</h2>
</header>
<main>
<p>The main content goes here.</p>
</main>
</body>
</html>
```

Now create this style sheet and name it `box.css`.

```
h1, h2, .display
{
    text-align:center;
}
header
{
    border: solid 1px red;
    background-color:#001A57;
    color:#FFF8E7;
}
```

This is the result.



Document Header

The main content goes here.

You will notice an irritating white rim around your header. You are doing battle with browser defaults. We can get rid of this annoyance by adding this code to our CSS file.

```
html, body
{
    margin:0;
    padding:0;
}
```

You should now change margin, padding, and border on the style rules for the `header` selector and observe the effects.

Programming Exercises

1. Add this to your CSS file. The unit `em` is the size of the letter 'm.' It is very mobile friendly and it is a recommended way of specifying margin and padding.

   ```
   main
   {
       background-color:#FFF8E7;
       padding:1em;
   }
   ```

2. Create a class for divs that puts text inside of a yellow box that is 80% of the width of the page. Put a solid black 1 pixel border around it. Choose padding so it looks nice.
3. How can you specify margin for a paragraph to control the size of the line skip between paragraphs?

6 CSS Layout

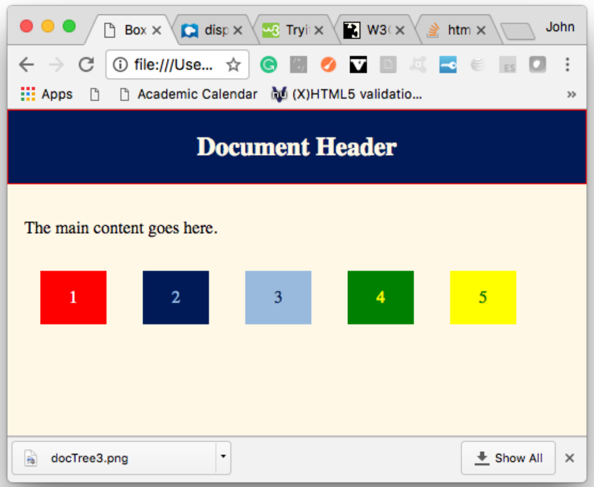
HTML5 presents several new tags that are basically divs, but which have names evocative of their purpose. You have met two, `header` and `main`. Here are a few some others.

`nav`	This is for a navigation area, which you can configure to be vertical or horizontal.
`aside`	This is for side notes to main content.
`footer`	This is for content at the bottom of the page.
`figure`	This is for self-contained content such as images, code listings, or diagrams.
`figcaption`	This is for a caption for a figure.

To achieve layout effects, you will want to use the `display` property. This table gives some common, very useful values for this property.

`none`	This suppresses the display of the element.
`inline`	This makes an element an inline element.
`block`	This makes an element a block element.
`inline-block`	The inside of the element behaves like a block, but the entire elment is treated as inline element.
`table`	This makes an element a block-level element and displays it like a table. If you use this, you will put items inside of it whose `display` property is marked `table-cell`. inside of it.
`table-cell`	This makes an element a block-level element inside of the element and displays it like a table cell inside of an element whose `display` is marked `table`. it like a table. If you use this, you will put `table-cell` items inside of it.

Now we are going to create this, using our shiny new toys.



To to this, we will create a div that goes all the way across and we will stick the five divs for each of the colored rectangles inside of it. We will give each little square an `id`, which is a unique identifier for that element. So here is what happens in the HTML.

```html
<!doctype html>
<html lang="en">
<head>
<title>Box Model Demo</title>
<link rel="stylesheet" href="inARow.css">
<meta charset="utf=8"/>
</head>
<body>
<header>
<h2>Document Header</h2>
</header>
<main>
<p>The main content goes here.</p>
<div class="boxRow">

    <div id = "box1">
    <p>1</p>
    </div>
    <div id = "box2">
    <p>2</p>
    </div>
    <div id = "box3">
    <p>3</p>
    </div>
    <div id = "box4">
    <p>4</p>
    </div>
    <div id = "box5">
    <p>5</p>
    </div>

</div>
</main>
</body>
</html>
```

In CSS, selecting by ID is simple. To select `box1`, just use the selector `#box1`. Note the use of the pound sign. *Warning: only use a given* `id` *once on a page!*

Now for the CSS. Let us begin by making the `boxRow` div have a width of `100%` and let us cause its `display` property to have value `block`.

```css
.boxRow
{
    display:block;
    width:100%
}
```

Notice that the five little boxes are all inside of the `boxRow div`. We now give them a common width, margin and make their display property be `inline-block`. We will also make them center their number.

```
#box1, #box2, #box3, #box4, #box5
{
    display:inline-block;
    text-align:center;
    width:12%;
    margin:3%;
}
```

Finally, we give each a color and background color.

```
#box1
{
    background-color:red;
    color:white;
}
#box2
{
    background-color:#001A57;
    color:#99badd;
}
#box3
{
    background-color:#99BADD;
    color:#001A57;
}
#box4
{
    background-color:green;
    color:yellow;
}
#box5
{
    background-color:yellow;
    color:green;
}
```

6.1 Table-Style Display with CSS

Various hackish indiduals decided that making an entre page a giant table was a convenient means of controlling layout on web pages. Unfortuantely, this breaks the separation between the presentational and structural layers in the HTML/CSS/JavaScript stack. It also had the lamentable consequence of code embedded inside of tables that was basically unreadable and very difficult to maintain.

However, this hack often proved to be the poison apple that appealed because it did the job. Now we have CSS controls that enable table-style layout but which offer far greater flexibility and control over layout.

Suppose we want to lay out portions of a page in an rectangular array. Here is a basic CSS file that outlines what is to be done.

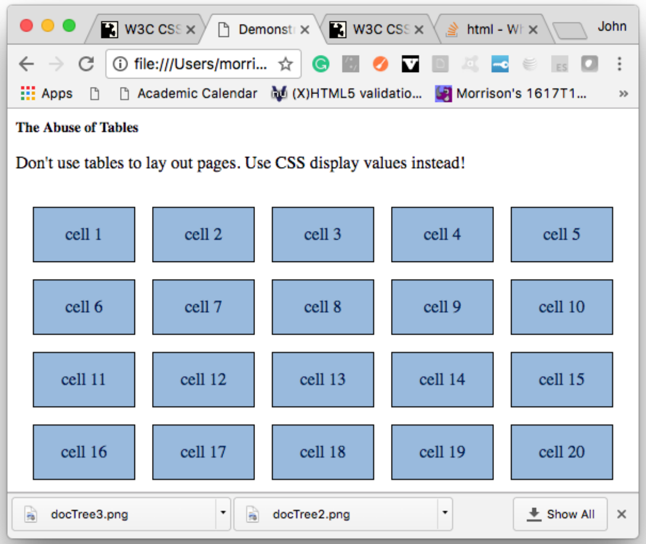
```
html, head
{
    margin:0;
    padding:0;
}
h1, h2, .display
{
    text-align:center;
}
h5
{
    display:inline;
}
.tableDisplay
{
    display:table;
}
.tableRow
{
    display:table-row;
}
.tableCell
{
    display:table-cell;
}
```

It is tied to this page.

```
<!doctype html>
<html>
```

```
<head>
<title>Demonstrating Table Style Layouts</title>
<meta charset="utf-8"/>
<link rel="stylesheet" href="inATable.css"/>
</head>
<body>
<h5>The Abuse of Tables</h5>
<p>Don't use tables to lay out pages.  Use CSS display values instead!</p>
</body>
</html>
```

We are going to create this.



The Abuse of Tables

Don't use tables to lay out pages. Use CSS display values instead!

cell 1	cell 2	cell 3	cell 4	cell 5
cell 6	cell 7	cell 8	cell 9	cell 10
cell 11	cell 12	cell 13	cell 14	cell 15
cell 16	cell 17	cell 18	cell 19	cell 20

First we add the shell for our table. As we go, we will create more style rules to make everything work. We shall make our overall table have width 100%. Add this HTML. It creates four table rows.

```
<div class="tableDisplay">

    <div class="tableRow">
    </div>

    <div class="tableRow">
    </div>
    <div class="tableRow">
    </div>

    <div class="tableRow">
    </div>
</div>
```

Then add this style rule.

```
.tableDisplay, .tableRow
{
    width:100%;
}
```

Now let us add the cells. We will put text in each cell.

```
<!doctype html>
<html>
<head>
<title>Demonstrating Table Style Layouts</title>
<meta charset="utf-8"/>
<link rel="stylesheet" href="inATable.css"/>
</script>
</head>
<body>
<h5>The Abuse of Tables</h5>
<p>Don't use tables to lay out pages.  Use CSS display values instead!</p>

<div class="tableDisplay">

    <div class="tableRow">
        <div class="tableCell">
            <p>cell 1</p>
        </div>
        <div class="tableCell">
            <p>cell 2</p>
        </div>
        <div class="tableCell">
            <p>cell 3</p>
        </div>
        <div class="tableCell">
            <p>cell 4</p>
        </div>
        <div class="tableCell">
            <p>cell 5</p>
        </div>
    </div>

    <div class="tableRow">
        <div class="tableCell">
            <p>cell 6</p>
        </div>
        <div class="tableCell">
            <p>cell 7</p>
        </div>
        <div class="tableCell">
            <p>cell 8</p>
        </div>
        <div class="tableCell">
            <p>cell 9</p>
        </div>
        <div class="tableCell">
            <p>cell 10</p>
        </div>
    </div>

    <div class="tableRow">
        <div class="tableCell">
            <p>cell 11</p>
        </div>
        <div class="tableCell">
            <p>cell 12</p>
        </div>
        <div class="tableCell">
            <p>cell 13</p>
        </div>
        <div class="tableCell">
            <p>cell 14</p>
        </div>
        <div class="tableCell">
            <p>cell 15</p>
        </div>
    </div>

    <div class="tableRow">
        <div class="tableCell">
            <p>cell 16</p>
        </div>
        <div class="tableCell">
            <p>cell 17</p>
        </div>
```

```
        <div class="tableCell">
            <p>cell 18</p>
        </div>
        <div class="tableCell">
            <p>cell 19</p>
        </div>
        <div class="tableCell">
            <p>cell 20</p>
        </div>
    </div>
</div>
</body>
</html>
```

Finally, we put in the style rules for the table layout. Here is the CSS file.

```
html, head
{
    margin:0;
    padding:0;
}
h1, h2, .display
{
    text-align:center;
}
h5
{
    display:inline;
}

.tableDisplay
{
    display:table;
    border-spacing:1em; /*control spacing between divs*/
    /* Use this to collapse borders
    border-collapse:collapse;*/
}
.tableRow
{
    display:table-row;
}
.tableDisplay, .tableRow
{
    width:100%;
}
.tableCell
```

```
{
    display:table-cell;
    border:solid 1px black;
    background-color:#99BADD;
    color:#001A57;
    margin:0;
    padding:0;
    text-align:center;
}
```

You should fiddle with the border spacing and try the `border-colllapse` property to produce a tight grid of cells.

Programming Exercises

1. Use `border-collapse` to tightly pack the cells.
2.

7 A Mèlange of Goodies

In this section we will do a little exploration of the world of pseudo-classes and we will show how to make a footer and have it actualy stick to the boottom of the page.

A `footer` is a newfangled HTML5 tag that is meant to go at the bottom of a page. However, it won't go there unless CSS tells it to.

Link coloring

hover

References

[1] Mozilla developer network. `https://developer.mozilla.org/en-US`.

[2] Stack Overflow Community. Stack overflow. `http://www.stackoverflow.com`.

[3] Patrick Griffiths. Html dog. `https://htmldog.com`.

[4] Inc. Refsnes Data. Firefox web browser. `http://www.w3schools.com`.