Chapter 1, Page Apperance and CSS

John M. Morrison

February 28, 2020

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. You got a preview of it when you learned about tables.

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 [**?**] and [**?**]. The Mozilla Developer Network [**?**] 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 [**?**]; 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.

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

Each line you see of the form *foo:baz;* is called a *style declaration*. Here we see style declarations 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;
}
```

These statements are called *style rules*. Each style declaration consists of a *property* and a *value*; the property and value are separated by a colon and each line is terminated with a semicolon.

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 declaration

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

all `h1` and `h2` elements are selected and the style declaration 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 declaration to begin and end in the right places.

Curly braces bound the list of style declarations belonging to each selector. Each style declaration is ended with a semicolon. Omit this and experience pain; your style declaration after this omission will just not work.

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, table cells, or paragraphs have the property `text-align` which has possible values `left`, `right`, `justify`, and `center`. The W3Schools site has a complete reference on this.

You can check the validity of your CSS with the CSS Validator at `https://jigsaw.w3.org/css-validator/`. You can enter a URL, upload a file, or paste text into a box; just click the appropriate tab to do so. Validating your CSS finds mistakes so you don't have to ferret them out on your own.

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 declaration 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. You can have several style declarations in a local style sheet but you must separate them with semicolons. 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>
  h1, h2
  {
      text-align:center;
  }
  </style>
  ```

On this page, all text in `h1` and `h2` headlines will be centered. You should use this mechanism only for style rules that are particular to a single page.

- **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" 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" 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 your 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 locally, 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 change, 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.

5. Open a page with Firefox and find Eyedropper in its development tools. Then do some exploring. It's a really cool tool you will find very useful.

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 inline 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 us|that from
these honored dead we take increased devotion to that cause for
which they gave the last full measure of devotion|that we here
highly resolve that these dead shall not have died in vain|that
this nation, under God, shall have a new birth of freedom|and
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 Miss Wormwood's dreaded 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 declarations 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. At this point in the development, it's best to stick to classes and not use IDs.

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> Donald Trump</li>
    <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> Donald Trump</li>
    <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?"

As we saw in the previous chapter, 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.

Here is a common way to center a table.

```
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 is the style sheet `table.css` we used in the previous chapter.

```
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 bounded 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 element containing the main content of the document.

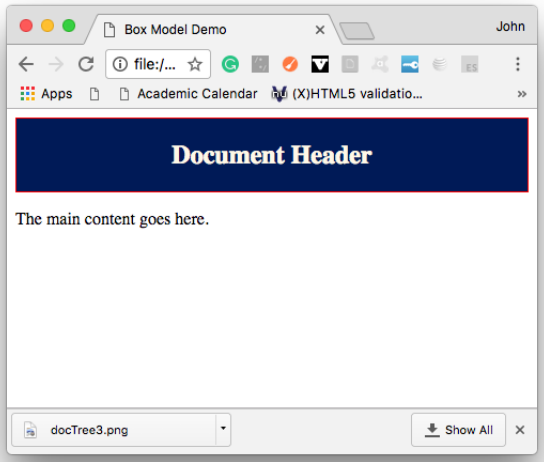
```
<!doctype html>
<html lang="en">
<head>
```

```
<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.



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' in the element's font size. 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?

5.1 CSS Units

Many properties carry a value that is a length. CSS units come in two species, absolute and relative. Let us begin by seeing the absolute units available in CSS.

`cm`	centimeters
`in`	inches
`mm`	millimeters
`px`	pixels (1/96 of an inch)
`pt`	points (1/72 of an inch)
`pc`	picas (1/12 of an inch)

Pixels seem to be the most popular unit. Absolute unit do not scale with screen size.

Relative units are relative to some other length property. By and large they tend to be more mobile-friendly than absolute units. However, you might use absolute units maintain the integrity of images or other page elements. You can set properties such as `min-width` and `min-height` in absolute units to accomplish this.

`em`	This is relative to the current font size. For example, 1.5 em makes a enlarges the font size by 50% in the selected element.
`rem`	This is relative to the current font size in the root element. For example, 1.5 rem makes a enlarges the font size by 50% from the size specified in the root element (usually the HTML element).
`vw`	This is relative to the size of the viewport. It is 1% of the viewport's width.
`vh`	This is relative to the size of the viewport. It is 1% of the viewport's height.
`vmin`	This is relative to the size of the viewport. It is 1% smaller of the viewport's height or width.
`vmax`	This is relative to the size of the viewport. It is 1% larger of the viewport's height or width.
`%`	This is relative to the parent element.

Programming Exercise Time for a little spelunking! Create these two files, `units.html` and `units.css`

```
<!doctype html>
<!--Author: Morrison-->
<!--Date: 2020-02-25-->
<html lang="en">
<head>
<title>units</title>
<meta charset="utf-8"/>
<link rel="stylesheet" href="units.css"/>
</head>
<body>
    <h1>CSS Unit Demonstration</h1>

    <div class="bigBox">
        <div class="littleBox">
        </div>
        <div class="littleBox">
        </div>
        <div class="littleBox">
        </div>
    </div>
</body>
</html>
```

```
/*Author: Morrison*/
/*Date: 2020-02-25*/
*
{
    box-sizing:border-box;
}
body
{
    margin:0;
    padding:0;
}
h1
{
    text-align:center;
    font-size:10vmin;
}
.bigBox
{
    width:80%;
```

```
    border:solid 1px black;
    height:40vh;
    margin-left:auto;
    margin-right:auto;
    min-width:400px;
}
.littleBox
{
    width:30%;
    border:solid 1px red;
    height:50%;
    display:inline-block;
    margin-left:1.3%;
    margin-right:1.3%;
}
```

Modify the units to be various relative and absolute units. Resize the browser window and observe what happens.

6 Creating New Selectors From Old

We can create new selectors from old using a means called *combinators*. We have seen that you "or" several selectors by using a comma-separated list of selectors. For example

```
p, .foo, #me
{
    /*style rules*/
}
```

will select all paragraphs, anything marked with `class="foo"`, or with id `me`. Hence, the comma combinaor acts as an or operator for CSS. There are quite a few ways of combining selectors; we will mention a few here.

The selector `ul li` will select all list items inside of an unordered list. Making a list of selectors with no comma such as this looks for all elements that are within all of the specified elements. For example

```
.hot ul li em
{
    /*style rules*/
}
```

will apply the listed style rules to any `em` element which is inside of a list item belonging to an unordered list that is, in turn, inside any element marked with

a class of `hot`. You can see that the space is acutally on operator. Think of it as "descendent of" in the document tree.

The cobinator `>` is the "child-of" cobinator. The selector `A > B` selects if an element of type `B` is an immediate child of an element of type `A`. Here is an example

```
body > ul
{
    color:red;
}
```

Suppose the page it is linked to contains this HTML.

```
<ul>
    <li>one</li>
    <li>two</li>
    <li>three</li>
</ul>

<div>
    <ul>
        <li>four</li>
        <li>five</li>
        <li>six</li>
    </ul>
</div>
```

The list items in the first list will be red since the `ul` element resides *directly* in the body. The second list will not be red; it is not an immediate child of the body; it is actually a grand-child. You might find this useful.

```
ul > li
{
    /*style rules*/
}
```

This selects list items that are direct children of unordered lists. The W3Schools site has a complete list of combinators.

7 CSS Layout

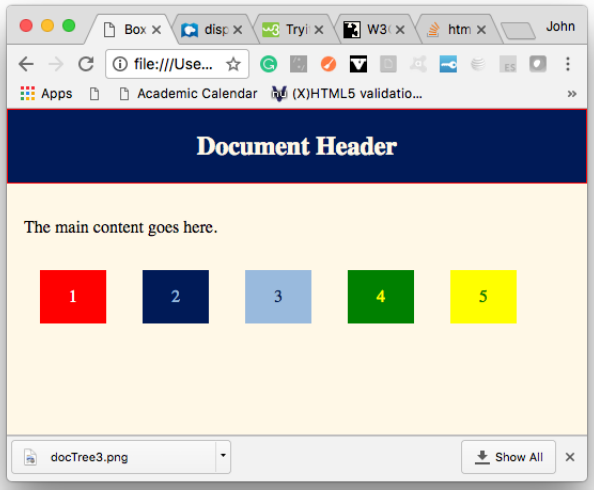
HTML5 presents several new tags that are basically divs, but which have names evocative of their purpose. You have met two, `header` and `main`. Here are a few some others.

`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. You will often use this in conjuction with JavaScript, which can dynamically change the display property to cause text to appear in response to an event such as clicking on a button.
`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 element 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.

```
<!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`.

```
.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;
}
```

7.1 Table-Style Display with CSS

Various hackish individuals decided that making an entire page a giant table was a convenient means of controlling layout on web pages. Unfortunately, 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.

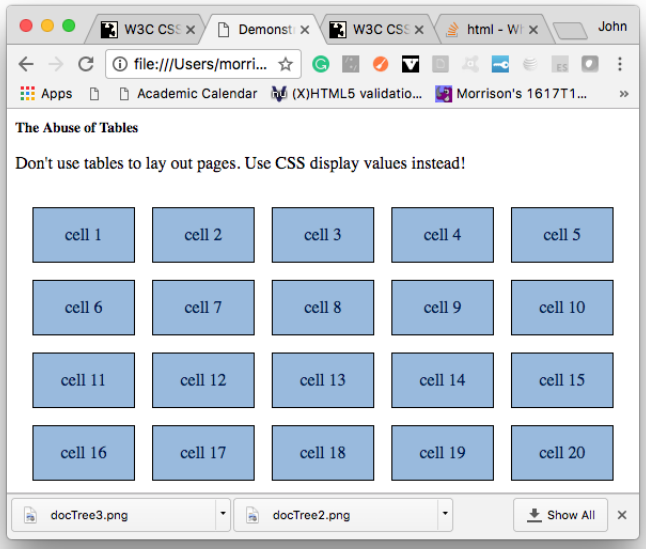
```
html, head
{
    margin:0;
    padding:0;
}
h1, h2, .display
{
    text-align:center;
}
```

```
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.



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.

8 Pseudoclasses

Let us begin with `hover`. You can apply this to any element or elements on your page and, when the user mouses over any of these elements, the attached list of style rules is applied. Here is an example.

```
p:hover
{
    background-color:yellow;
    font-weight:bold;
}
```

Place this in a stylesheet linked to a page, and whenver the user mouses over any paragraph, it gets highlighted with yellow and the text becomes bold. Consider this example.

```
ul li:hover
{
    color:green;
}
```

If the user hovers over a list item inside of an ordered list, the text in the item turns green.

Pseudo-classes are also useful for styling links. Here are four useful items. You should use them in this order in your CSS.

`a:link`	This allows you to specify a link's style prior to it being visited.
`a:visited`	This allows you to specify a visited link's style after it has been visited
`a:hover`	This allows you to specify a link's style prior when itt being moused over by the user.
`a:active`	This allows you to specify a link's style when to it being visited at the time it is clicked.

Common items that get styled are `color`, `background-color`, and `text-decoration`. The `text-decoration` property can be set to `none` to get rid of underlining of links.

9 A Brief Introduction to FlexBox

FlexBox is a powerful and useful layout tool. We will conduct a quick tour of it here and show some examples of things it can do well. What we offer is a "quick start;" you can find all sorts of information on the web about this powerful tool. You are encouraged to do some spelunking on your own.

In our example here, we begin by creating a `div` with a class called `container`.

```
h1, h2, .display
{
    text-align:center;
}
.container
{
    display:flex;
}
```

Here is our `div`.

```
<!doctype html>
<html lang="en">
<head>
<title>flexExample</title>
<meta charset="utf-8"/>
<link rel="stylesheet" href="flexExample.css"/>
<script src="flexExample.js"></script>
</head>
<body>
    <div class="container">
    </div>
</body>
</html>
```

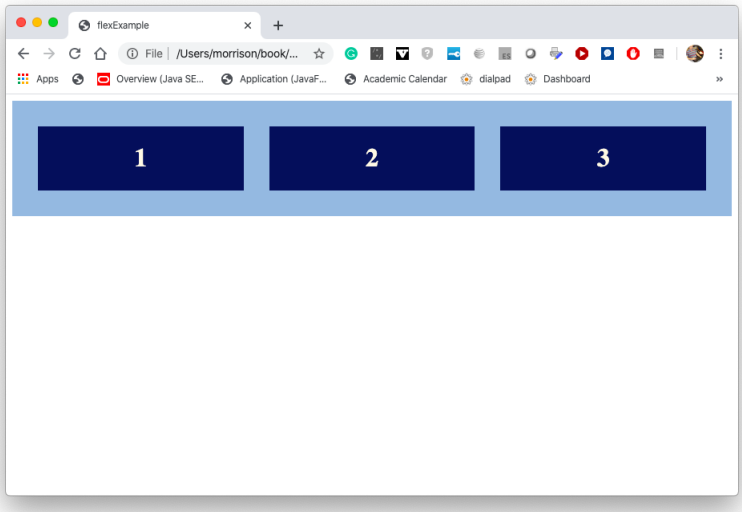
If you put a `div` inside of the container, it is automatically a `flex` container. Let's put three little `div`s inside. So far, it's dullsville.

Next, style the internal divs with the aid of the `>` combinator. and the container div.

```
.container
{
    display:flex;
    background-color:#99BADD;
    padding:1em;
```

```
}
.container > div
{
    background-color:#001A57;
    color:#FFF8E7;
    flex:1;
    margin:1em;
}
```

Here is the result.



Let's add some content to the `div`s. Also, add the `id`s for the three inner `div`s. We will use these shortly.

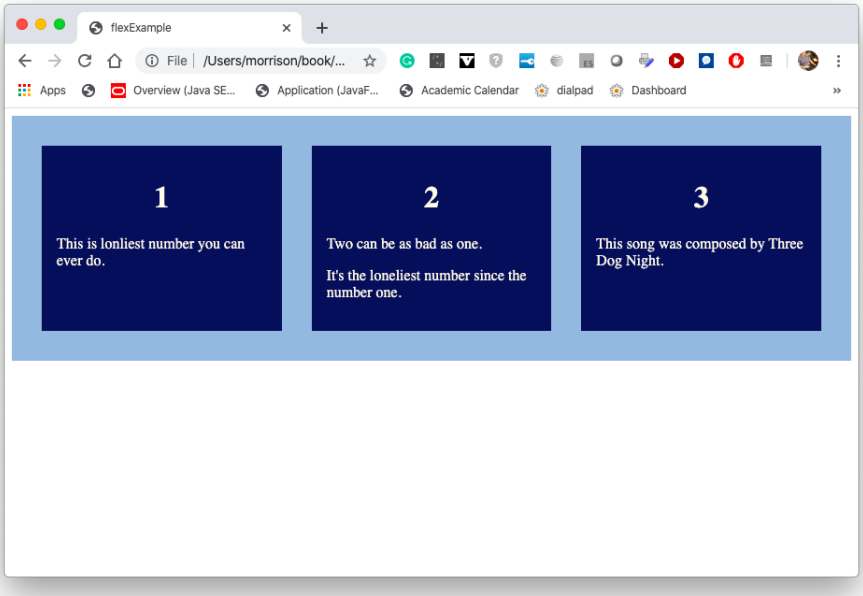
```
<div class="container">
    <div id="one">
        <h1>1</h1>
        <p>This is lonliest number you can ever do.</p>
    </div>
    <div id="two">
        <h1>2</h1>
        <p>Two can be as bad as one.</p>
        <p>It's the loneliest number since the number one.</p>
    </div>
    <div id="three">
        <h1>3</h1>
```

```
            <p>This song was composed by Three Dog Night.</p>
        </div>
    </div>
```

Add some padding as shown here, so we don't get claustrophobic.

```
.container > div
{
    background-color:#001A57;
    color:#FFF8E7;
    flex:1;
    margin:1em;
    padding:1em;
}
```

Look how pretty

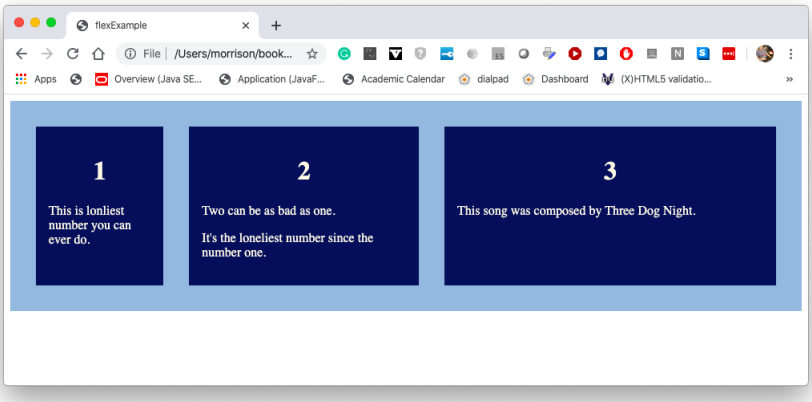


Finally, do this in the CSS file.

```
.container > div
{
    background-color:#001A57;
```

```
    color:#FFF8E7;
    /*flex:1;  comment me out */
    margin:1em;
    padding:1em;
}
#one
{
    flex:1;
}
#two
{
    flex:2;
}
#three
{
    flex:3;
}
```

You see this



Programming Exercises

1. Change the numbers used in the inner divs' `flex` property. What happens?
2. Add two more inner `divs`. How are they accomodated? Give them a `flex` property.
3. Add this style declaration of the container `div`'s rule. `flex-direction:column;` What happens?

4. Add this style declaration of the container `div`'s rule. `width:80%;` What happens? How would you center this? Notice how everyone stays the same height.
5. Add this to your CSS and try narrowing the screen.

```
@media screen and (width:500px)
{
    .container
    {
        flex-direction:column;
    }
}
```

This is called a *media query*.

10 Properties of Flex Containers

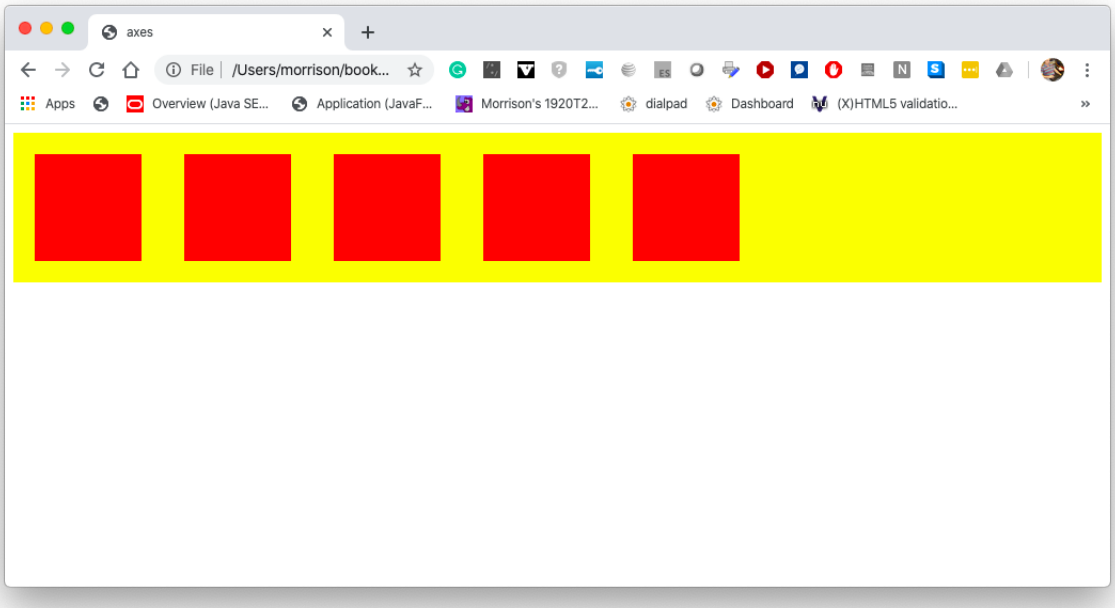
We will begin by looking at CSS rules that can be written for a flex container. In the next section, we will concern ourselves with rules that can be written for the items withing a flex container.

10.1 A Pain in the Axis

By default, items are added to a flex container horizontally; the *main axis* is the horizontal axis. As you add items to a flex container, they are placed in that container side-by-side. Let us begin with these examples

```
<!doctype html>
<html lang="en">
<head>
<title>axes</title>
<meta charset="utf-8"/>
<link rel="stylesheet" href="axes.css"/>
<script src="axes.js"></script>
</head>
<body>
    <div class="container">
        <div class="contents">
        </div>
        <div class="contents">
        </div>
        <div class="contents">
        </div>
        <div class="contents">
```

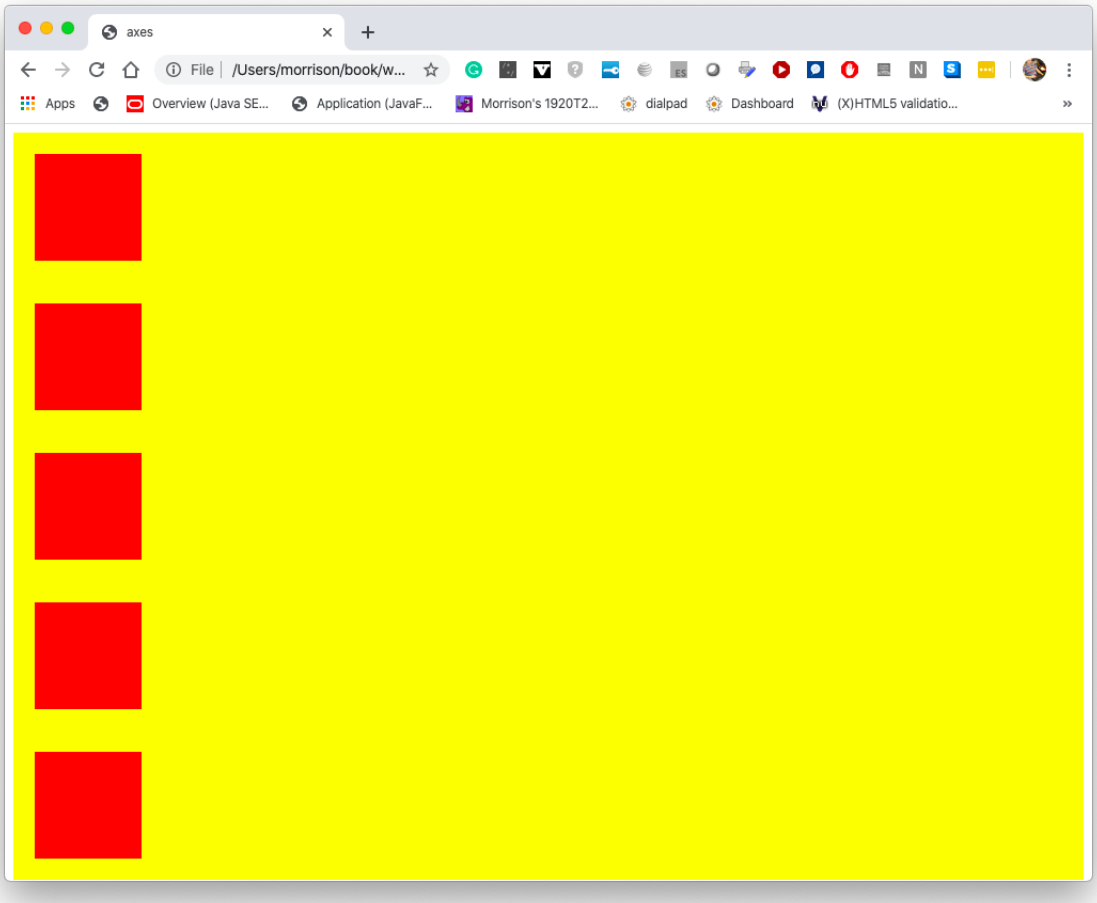
```
        </div>
        <div class="contents">
        </div>
    </div>

</body>
</html>

/*Author: Morrison*/
h1, h2, .display
{
    text-align:center;
}
.container
{
    display:flex;
    background-color:yellow;
}
.contents
{
    height:100px;
    width:100px;
    margin:20px;
    background-color:red;
}
```



Now let us change the main axis to vertical. Add this line to the `.container` declaration.

```
flex-direction:column;
```

Refresh and you will see this.



The flex-direction property for a flex container has these possible values.

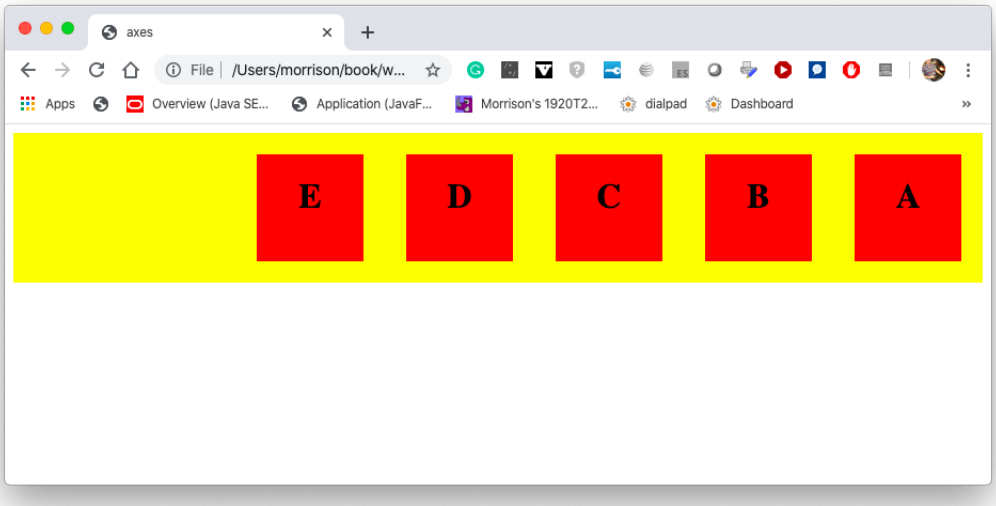
- `row` This makes the main axis horizontal; this is the default.
- `row-reverse` This makes the main axis horizontal but add the items into the container in the reverse order.
- `column` This makes the main axis vertical.
- `row-reverse` This makes the main axis vertical but add the items into the container in the reverse order.

Next, modify your inner divs to contain the letters A-E like so.

```
<div class="contents">
```

```
    <h1>A</h1>
</div>
```

Now change your `flex-direction` property to `row-reverse`



You should also try out `column-reverse`.

10.2 Justify your existence, puny Earthling!

A flex container has a property called `justify-content`. Here are its possible values. This property controls the distribution of items along the main axis.

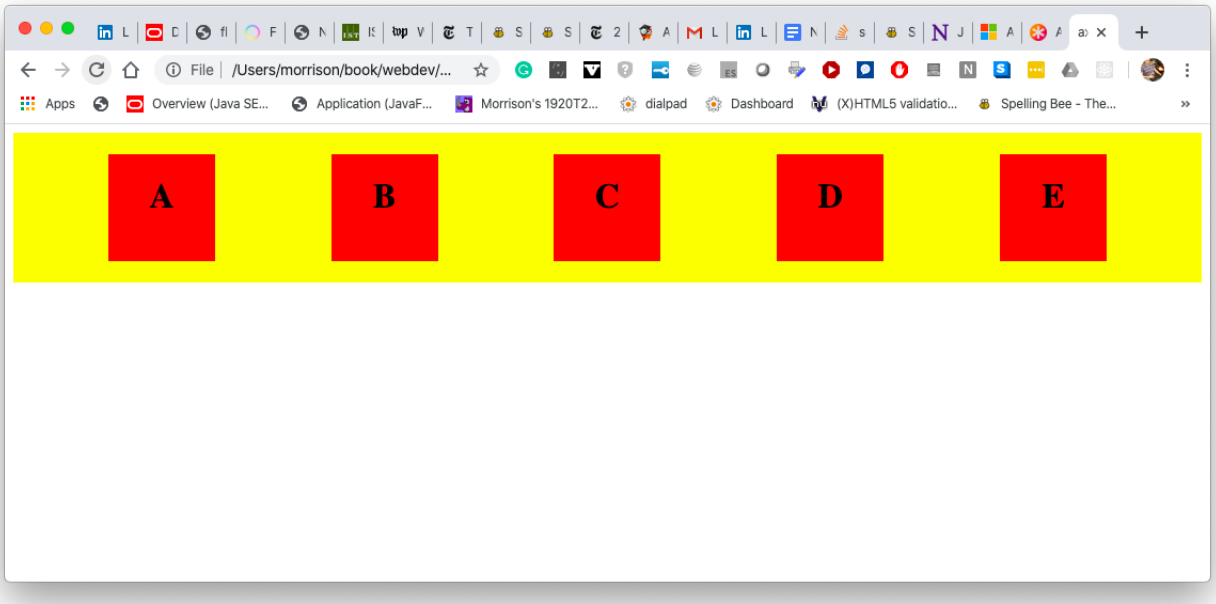
- `flex-start` This places the items in from left to right, starting at the left side of the container if the main axis is horizontal. (Think text: default is left-justified) If the main axis is horizontal, items are top-justified. This is the default
- `flex-end` This right-justifies the items inside of the container if the main axisis horizontal and bottom-justifies them it is vertical.
- `center` This centers the items in the container along the main axis.
- `space-between` This makes the main axis vertical.
- `space-around` This makes the main axis vertical.
- `space-evenly` This makes the main axis vertical.

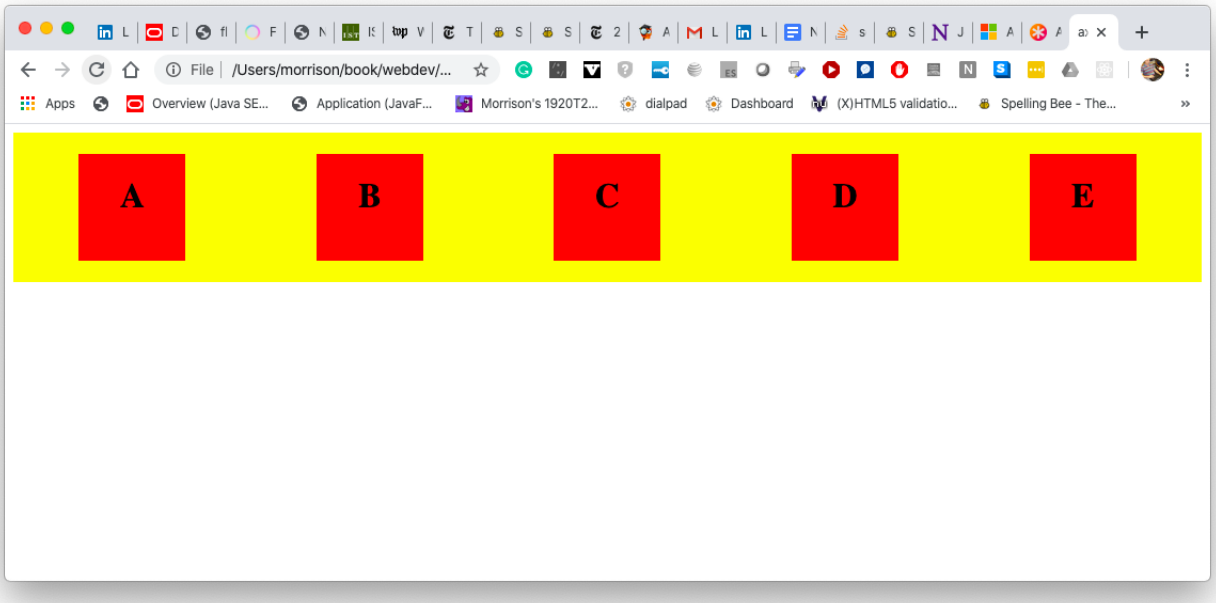
Modify your `.container` class as follows. The main axis will be horizontal.

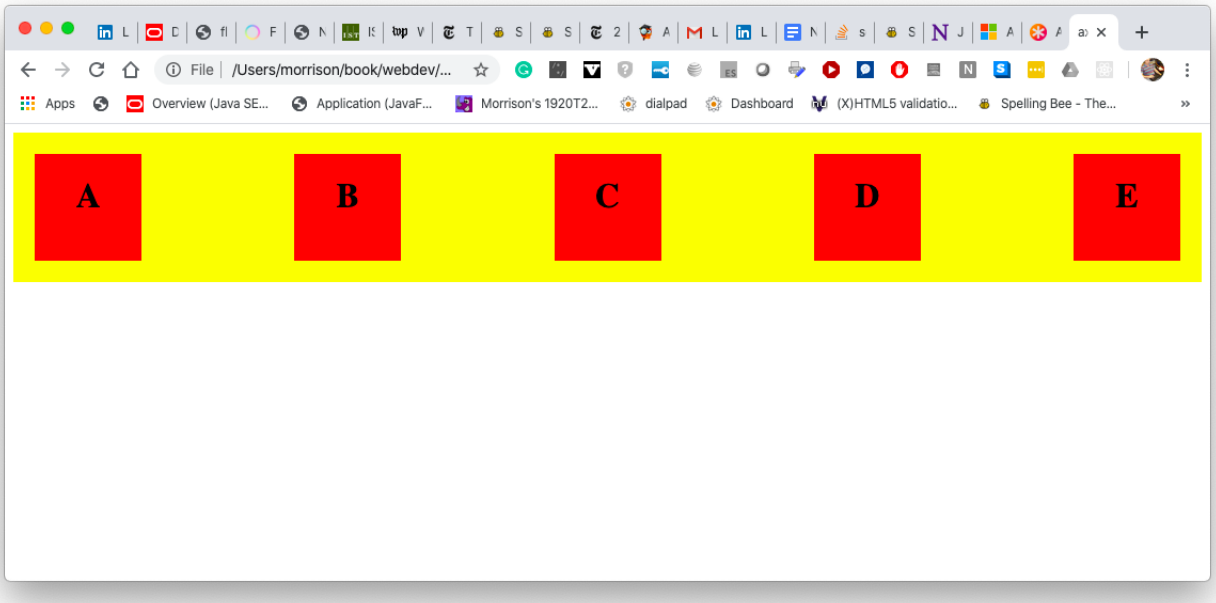
```
.container
{
    display:flex;
    background-color:yellow;
    justify-content:flex-end;
}
```

Now try cenering like this.

Programming Exercise Below we show three screenshots. Which is `space-around`? `space-between`? `space-evenly`?


File /Users/morrison/book/webdev/...
Apps
Overview (Java SE...
Application (JavaF...
Morrison's 1920T2...
dialpad
Dashboard
(X)HTML5 validatio...
Spelling Bee - The...
A
B
C
D
E


File /Users/morrison/book/webdev/...
Apps
Overview (Java SE...
Application (JavaF...
Morrison's 1920T2...
dialpad
Dashboard
(X)HTML5 validatio...
Spelling Bee - The...
A
B
C
D
E



10.3 Working at Cross Purposes: Aligning Items

The `align-items` property aligns things along the cross axis. Let us modify our HTML and CSS so our letter blocks have different sizes.

```
<!doctype html>
<html lang="en">
<head>
<title>axes</title>
<meta charset="utf-8"/>
<link rel="stylesheet" href="axes.css"/>
<script src="axes.js"></script>
</head>
<body>
    <div class="container">
        <div class="first">
            <h1>A</h1>
        </div>
        <div class="second">
            <h1>B</h1>
        </div>
```

```
        <div class="third">
            <h1>C</h1>
        </div>
        <div class="fourth">
            <h1>D</h1>
        </div>
        <div class="fifth">
            <h1>E</h1>
        </div>
    </div>

</body>
</html>
```

```
h1, h2, .display
{
    text-align:center;
}
.container
{
    display:flex;
    background-color:yellow;
    justify-content:space-between;
}
.first, .second, .third, .fourth, .fifth
{
    margin:20px;
    background-color:red;
}
.first
{
    height:100px;
    width:100px;
}
.second
{
    height:200px;
    width:100px;
}
.third
{
    height:300px;
    width:100px;
}
.fourth
{
```

```
    height:400px;
    width:100px;
}
.fifth
{
    height:500px;
    width:100px;
}
```

Let us now see the result.

Now let us look at the `align-items` property. This is, again, a property of the flex container. The default is `flex-start`.

- `flex-start` Items are aligned at the start of the cross-axis; this is the default.
- `flex-end` Items are aligned at the end of the cross-axis.

- `center` Items are centered along the cross-axis.
- `stretch` Items stretch along cross-axis.

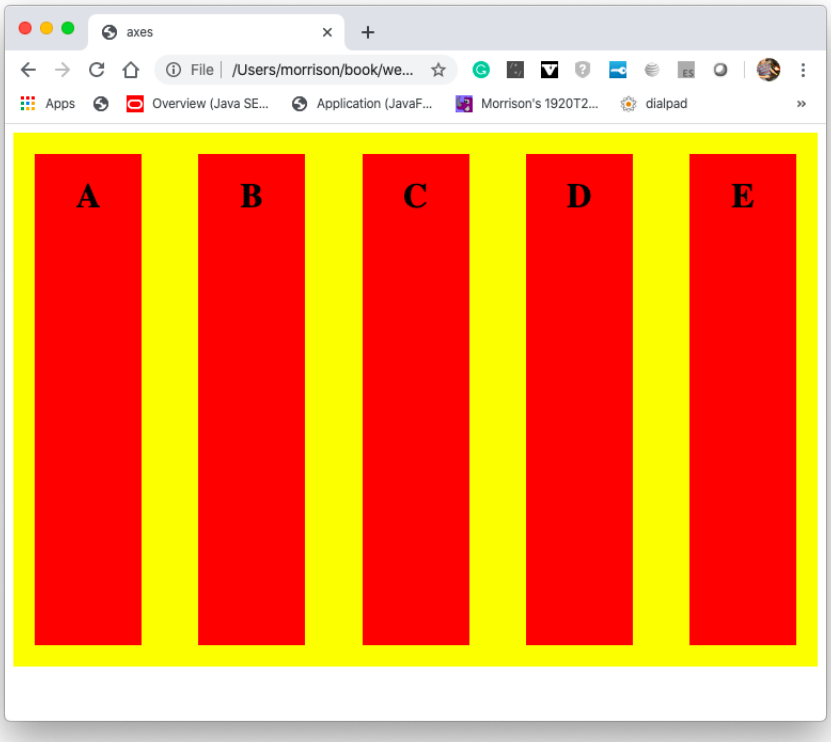
Here we show `flex-end` at work.

Next, we check out `center`.

Now we are entering the **stretch**. To fully appreciate its powers, we give a height to the yellow container and take away the heights of the items. as follows

```
/*Author: Morrison*/
h1, h2, .display
{
    text-align:center;
}
.container
{
    display:flex;
    background-color:yellow;
    justify-content:space-between;
    align-items:stretch;
    height:500px;
}
.first, .second, .third, .fourth, .fifth
{
    margin:20px;
```

```
    background-color:red;
}
.first
{
    width:100px;
}
.second
{
    width:100px;
}
.third
{
    width:100px;
}
.fourth
{
    width:100px;
}
.fifth
{
    width:100px;
}
```



10.4 It's a wrap!

Consider this HTML file. We introduce a little "crowding" into our example.

```
<!doctype html>
<html lang="en">
<head>
<title>axes</title>
<meta charset="utf-8"/>
<link rel="stylesheet" href="axes.css"/>
<script src="axes.js"></script>
</head>
<body>
    <div class="container">
        <div class="square">
            <h1>A</h1>
        </div>
        <div class="square">
```

```
            <h1>B</h1>
        </div>
        <div class="square">
            <h1>C</h1>
        </div>
        <div class="square">
            <h1>D</h1>
        </div>
        <div class="square">
            <h1>E</h1>
        </div>
        <div class="square">
            <h1>F</h1>
        </div>
        <div class="square">
            <h1>G</h1>
        </div>
        <div class="square">
            <h1>H</h1>
        </div>
        <div class="square">
            <h1>I</h1>
        </div>
        <div class="square">
            <h1>J</h1>
        </div>
    </div>

</body>
</html>
```

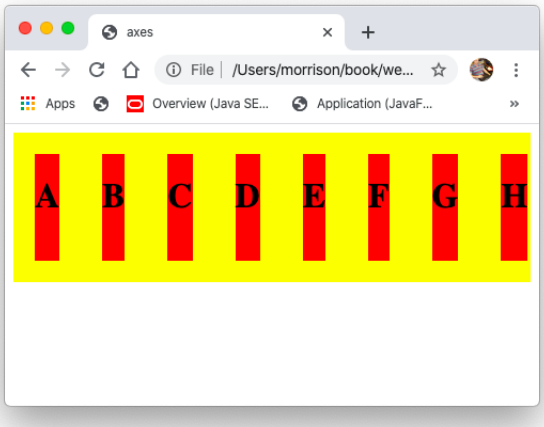
Here is our basic style sheet.

```
/*Author: Morrison*/
h1, h2, .display
{
    text-align:center;
}
.container
{
    display:flex;
    background-color:yellow;
    justify-content:space-between;
    align-items:baseline;
}
.square
```

```
{
    height:100px;
    width:100px;
    background-color:red;
    margin:20px;
}
```

This is what we have.

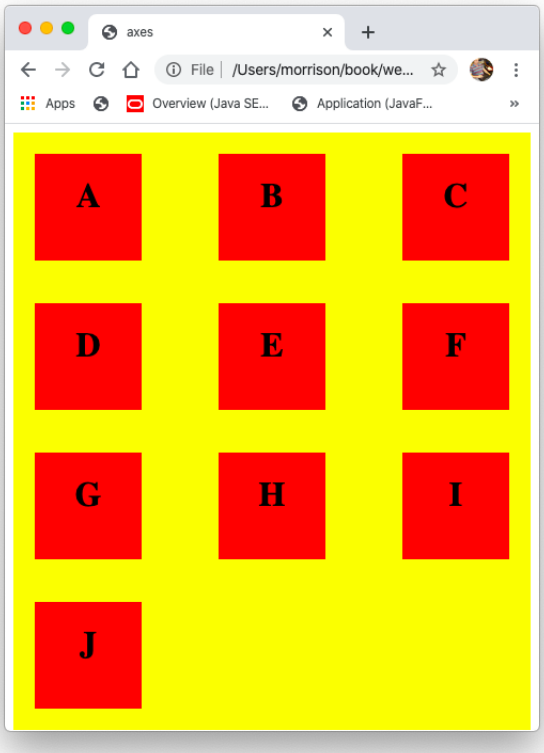
Now try to narrow the window as much as possible. This is the denouement.



The items in the flex container do not "wrap". However you can change this with the container property `flex-wrap` by setting it to the value `wrap`, as shown here.

```
.container
{
    display:flex;
    background-color:yellow;
    justify-content:space-between;
    flex-wrap:wrap;
}
```

This is the result.

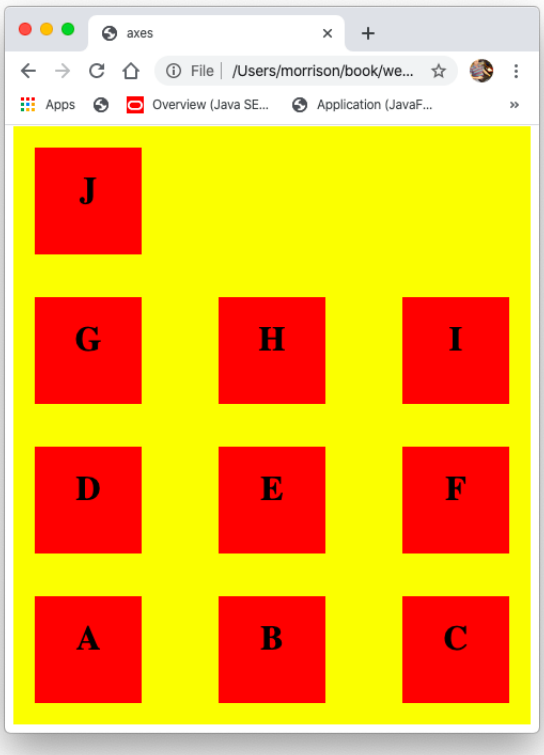


These are the possible value for the property `flex-wrap`.

- `nowrap` All items will appear on one line. This is the default.

- `wrap` Items will wrap texticographically.
- `wrap-referse` Items will wrap starting at the bottom, going up reverse-texticographically.

Here we show `reverse-wrap` at work.



The `flex-wrap` property can cause items to be displayed across several rows. We can control their positioning along the main axis with `align-items`. We can control their cross-axis alignment with `align-content`.

- `flex-start` Items are aligned at the start of the cross-axis; this is the default.
- `flex-end` Items are aligned at the end of the cross-axis.
- `center` Items are centered along the cross-axis.
- `stretch` Items stretch along cross-axis.

11 Properties of Items in a Flex Container

Let us begin with `flex`, which tells an item how much its share is relaive to others. By default, this value is 1.

12 CSS Grid: A Learning Lab

CSS has a new feature called `grid`, which gives fine-grained control over the layout of pages. You will be guided through this new tool via exercises and demonstrations. By the time you are done, you will have good basic facility with `grid`.

13 Setting up a Website on a Server

You will need access to a Linux server capable of hosting a website to have your materials display on the web. However, you can build web pages offline and see one on your local machine; this is all that is required to master the ideas in this book.

13.1 Step 0, Obtain a Server Account

To get started, make sure you know the following. Your system administrator who creates your account will know all of this information; so don't be shy and ask about it.

1. Your username
2. Your password

3. The server's name. It will look something like this: `cs.ncssm.edu`.
4. Ask the administrator what name to give your www directory; this is usually `public_html`.
5. Ask the administrator what name to give index files on your website. This is usually `index.html`.

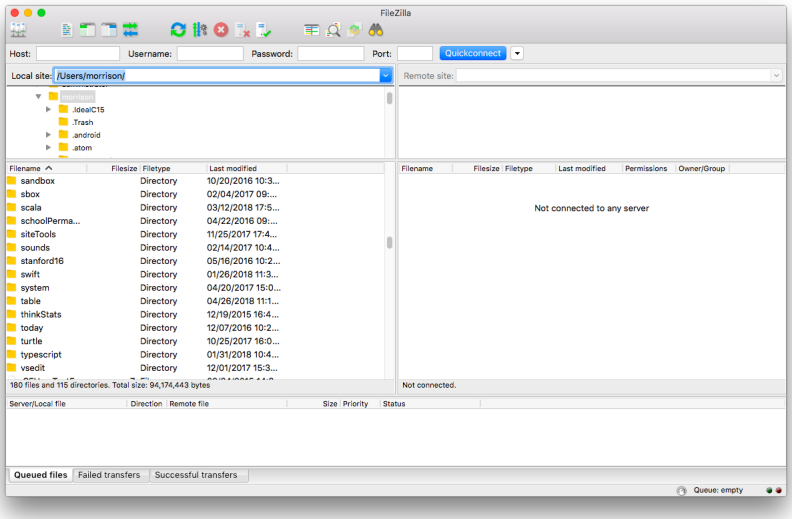
13.2 Step 1, For Windoze Users Only

If you use Windoze, download the `PuTTy` application. To find this, just Google "putty;" it will be the first link you see. Obtain the MSI file for your machine. Double-click on it and install it. This little piece of software will allow you to establish a session with your server over the network. If you are using MacOSX or Linux, no action is required for this step.

13.3 Step 2: Obtaining and Using FileZilla

This application will alllow you to transfer files between your machine and the server you are using. Happily, its interface is extremely intuitive and simple. Begin by going to this website `https://filezilla-project.org/`. Download the client, not the server. Then, install it on your machine. This application works for Macs, PCs and Linux boxes.

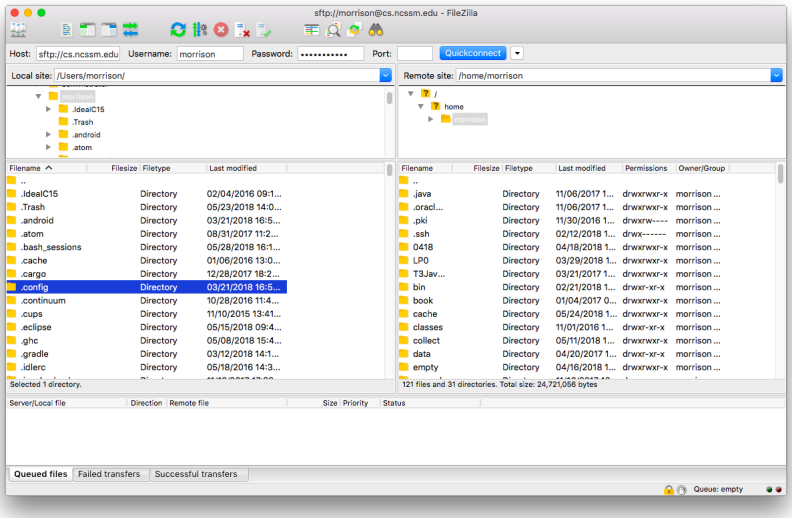
Next, open the application. It looks like this.



You now do the following

1. In hostname, put your server's name. In the example shown here, we use `cs.ncssm.edu`.
2. Under Username, enter your username.
3. Enter your password in the password box.
4. Under Port, enter 22.
5. Hit ENTER or the Quickconnect button.

You will now see this.



On the left, is your LOCAL machine. On the right is the REMOTE server. To open a folder, double-click on it. To send a file to the server, just open the folder you want it in, drag it across, *et voila...* the file is transferred. This works both ways, so you can easily send files back and forth. The top windows makes it easy to navigate among your directories.

13.4 Step 3, Prepare the Server

You will need to log into your server. The process varies a little by OS.

Windoze Users If you use windows, start PuTTy. It will open a window. In the slot for hostname, put the name of your server. In the slot for Port, enter 22. For connection type, choose SSH (this is the default). Put the word "website" under Saved Sessions and click the save button. Once you do this, on

successive logins you can just double-click on the word website, and putty will launch.

You will see a window pop up asking for your login name. Enter you user name. Then enter your password; note that the characters you type when entering your password do not appear on the screen. When you are done entering your password, hit enter.

Mac/Linux Open a terminal; on a Mac this is located in your `Applications/Utilites` folder. If you run Linux, it is an icon on your desktop. Enter this at the command prompt:

```
ssh username@hostname
```

where `hostname` is the name of your machine. A typical person might enter something like this.

```
ssh morrison@cs.ncssm.edu
```

You will then be asked for your password. Enter this; they keystrokes will not appear on your screen. Then hit the ENTER key.

For Everyone You will see a window that looks like this. Yours might vary in appearance

The text you see in the window is called a *prompt*; it tells you the terminal is ready to accept commands.

13.5 Step 4, Configure your server

You will need to make some basic preparations so your site is visible.

Begin by making a the www directory to hold your site. The most common name for this directory (folder) is `public_html`; refer to the information you asked for in Step 0. Almost all sites use `public_html`.

To get started, log into the server and enter these commands. Note that the $ sign just stands for the server's system prompt.

```
$ cd
$ chmod 711 .
$ mkdir public_html
$ chmod 755 public_html
$ cd public_html
$
```

Here, in a nutshell, is the purpose of what you did. You enter your home directory. You then give permission for Apache to see *through*, but not *into* your home directory. The general public will not be able to see those files. You then make the contents of your `public_html` directory readable by Apache. Apache on a UNIX server is just another user and it needs proper permission so it can fetch your files to be served. This allows your content to be served on the web. This directory is the root of your *public subtree*. You can create directories inside of `public_html`; in fact this is a good way to organize a site containing several parts.

Keep your server session open; you will need it again. Now use FileZila to place your index file in your `public_html` directory. Name it `index.html` prior to transferring it.

It should cause the other pages of the directory to be linked, so they can be viewed by visitors to your site. We will see how to do this soon.

Now go back to your server session. Make sure that the all HTML pages have permission level 644 so they can be read by the world. This can be done quickly using

```
$ cd
$ cd public_html
$ chmod 644 *.html
$
```

This command will make all files in your current working directory with extension `.html` visible to the world.

Now enter this in the address window of your browser.

`http://faculty.ncssm.edu/~morrison`

Use your server's name instead of `faculty.ncssm.edu` and your user name instead of `morrison`. Do not omit the tilde `~`. This gives you the *base URL* for your page. You will see a white page with the word "Hello" on it. In the top of the tab or the title bar of the browser you will see the title "My First Page."

You can place an index page in any subdirectory of `public_html`. The URL of this page can be found by appending the relative path from `public_html` to your directory to the base URL of your page. This index page should link the contents of the directory so they can be viewed.

Where should you be now? You should be able to see your web page.

13.6 Subfolders for `public_html`

You can create subfolders of `public_html` to organize your site as it grows. Each subfolder should have 755 permissions so Apache can see into it contents. Each subfolder should have an index file named `index.html`. All files you want visible on the web should have 664 or 644 permissions.

14 Terminology Roundup

24-bit color This refers to the standard scheme of color for modern computers. The first eight bits are for red, the next are for green, and the last are for blue.

additive Light blends according to an additive color scheme; the result of mixing colors is determined by adding waveforms.

border This is the decoration around the boundary of an element. You can specify the border for the four sides using `border-left`, `border-right`, `border-top`, and `border-bottom`.

box model This is how CSS controls spacing in elements. Each element consists of content, padding, border, and margin.

cascading style sheets (CSS) This is the language of page appearance on the web.

class You can apply a set of style rules to an arbitrary collection of page elements by marking them with the samle class. The usage is

```
<element class="classname">..... </element>
```

combinator These combine the actions of selectors. Examples include the space operator, "descendant of", the child operator `>` and the comma operator, "or".

content This is the actual stuff (text, images, etc) inside of an element.

id. This is an attribute that can be given to any element on a page. It should be a unique identifier of an element on a page; do not have two elements with the same id on page or page behavior might be unpredicatable.

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`.

property This is part of a style rule. Every element type has a set of admissible properties. Examples include such things as `color`, `background-color`, or `text-align`.

pseudoclass A pseudoclass is identifiable by the presence of a colon in HTML. For example, `p:hover` applies style rules when a paragraph is moused over.

scope This refers to the lifetime of a style rule.

style rule This is a property-value statement in CSS. For example the style rule

```
color:red;
```

makes text be rendered in red. **subtractive** This is the system of colors that is imposed by pigments, instead of light **value** This is a valid value for a style property. For example in the declaration `color:red`, `red` is the value for the property `color`. This rule causes all text in the element to be red.