

Web Design

Lesson 2

Development Perspective: DIV/CSS

Why tables have been tabled

Tables are a cell based layout tool used in HTML development. Traditionally they have been the primary tool used by web developers to control the position of images, navigation and general web content. While they can be useful, their rigid structure and bloated mark up can make development a tedious and confusing task. As well as leading to a larger page size their nested code structure can be dizzying, making updating very difficult. Furthermore, controlling structure between separate HTML pages is not controlled globally requiring the same HTML code to be used in multiple instances, making consistency very difficult.

Keeping to the underlying goal of creating simple, seamless and light code structures, developers sought a new way of controlling layout. The answer; split up structure and style by coding HTML pages with DIV tags rather than tables and controlling their display using a Cascading Style Sheets (CSS).

DIVTags <div>

The <div> tag defines logical divisions within the content of a page. When used properly they can create a simple definition for a page's content layout. Structurally, a <div> tag is only useful when given a class name from a CSS file.

```
<div class="header">
```

What does it look like?

Visually a <div> tag is a box that can be proportioned any way you choose. When you start a DIV tag using <div> you are creating the opening top-left corner of the box. The box is not complete (or programmatically created) until you close it using the </div> tag which closes off the bottom-right corner.

You can place <div> tags inside of one another to create boxes inside of boxes until you reach your desired structure. Nearly any HTML tag or content can be placed inside of <div> tags.

```
<div class="content">  
  <div class="header">  
    My Site is Smart.  
  </div>  
</div>
```

Controlling style

The <div> tag has a tremendous amount of controllable characteristics. Unlike its <table> counterpart, using nested <div> tags allows a developer quick and decisive control of layout while adjusting for margin, border, padding, typography and much more under a uniform set of rules defined in a CSS file.

The CSS file (Cascading Style Sheet) is a separate document that simply contains the 'styles' for your HTML document. On its own it is useless and does not get rendered by a browser until it is linked within the <head> tag of an HTML page.

HTML page (index.html)

```
<head>
  <link rel="stylesheet" href="screen.css" type="text/css" />
</head>
```

Styles that are used in the HTML page are created within the CSS file itself. These styles can overwrite the default settings of any HTML tag (<p>, <body>, <h1> etc.) or define style rules for any class name (class="my_class_name") or id (id="my_id_name") you wish. When defining styles it is important to note that *you* create the name of the style and can name it whatever you wish. Take careful note that your CSS rule is named *exactly* as it is referenced in the HTML.

HTML file (index.html)

```
<div class="mY_Class_NAME">
  ...
</div>
```

CSS file (screen.css)

```
.mY_Class_NAME {
  ...
}
```

Special care must be taken when defining rules in a CSS file. A rule is simply the definition of a style structure for a given HTML element. A rule can be applied to native HTML tags, classes or IDs. The discretion as to which entity the rule will be applied too is called the selector and is defined at the time the rule is created in the CSS file.

Defining a tag

```
selector {  
    declaration;  
}
```

An HTML tag definition
<body>

HTML tag selectors are defined by the tag name.

```
body {  
    ...  
}
```

A class definition
<div class="my_class_name">

Class selectors are defined by the class name with a leading period (.)

```
.my_class_name {  
    ...  
}
```

An id definition
<div id="my_id_name">

ID selectors are defined by an ID name with a leading number symbol (#)

```
#my_id_name {  
    ....  
}
```

Declarations

Inside each selector rule is the declaration of what your style does. This is articulated by a series of property definitions. Each property you wish to control must be declared to the left of the desired value, separated by a colon (:), and ending with a semi-colon (;).

Declaration syntax

```
body {  
    property: value;  
}
```

Example

```
body {  
    font-family: "Arial";  
    font-size: 12px;  
}
```

Using Classes and IDs

A class is a global element that is used more than once during the display of an HTML page. An ID is an element that is used only once.

Class usage

When defining the layout of a page you may use a class to dictate a general style of system text.

CSS (screen.css)

```
.big_font {  
    font-size: 2em;  
}
```

HTML (index.html)

```
<div class="big_font">  
    This a is a heading  
</div>  
<p>This text gets <span class="big_font">REALLY BIG</span></p>
```

ID usage

An ID on the other hand would be used for an element that appears once, such as a navigation block.

CSS (screen.css)

```
#my_navigation {  
    padding-top: 10px;  
}
```

HTML (index.html)

```
<div id="my_navigation">  
    ...  
</div>
```

Preparation is vital

Before moving into the development of a page the appropriate forethought must be considered. A good game plan will aid in creating the right code for you web site. Take the time to perform the following tasks before moving into code.

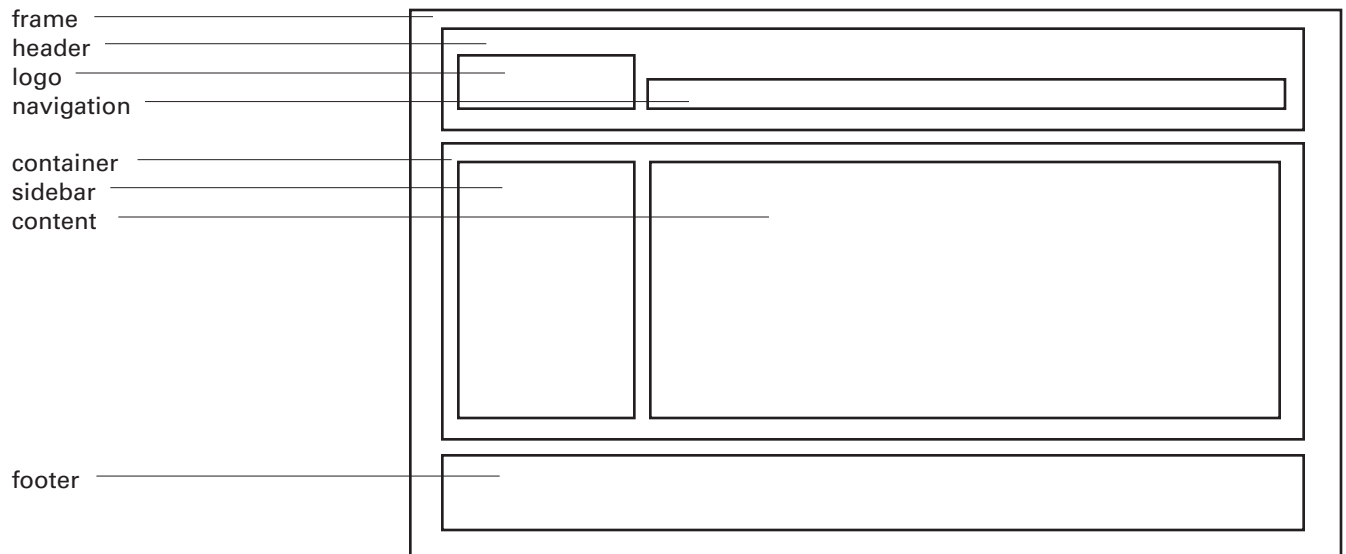
1. Setup Design Correctly

Ensure design items line up in your original document. Using guides and rulers will help you understand the spacing and placement of your DIV structure.

2. Explore structure early

Analyze your design to ensure you understand how the layout can be reflected by a <div> structure. If necessary, print out your design and draw boxes around all content items. A simple rule; if two or more elements sit side-by-side horizontally, contain those elements in a <div> block.

Possible <div> class setup



3. Create template files

Once you have determined the underlying structure, create template files that will contain the common layouts of your site. At a minimum this would mean creating a template HTML file and your global CSS file. When your templates are complete they can be used as production files for your site.

Example folder structure

```
> Sites
  > MyWebsite
    - index.html
    > styles
      - screen.css
```

Tying it all together

With solid preparation you can now begin development. To begin, open your template HTML file and write a <div> tag structure based on your layout analysis. In our example it might look like this.

index.html code

```
<html>
  <head>
    <title>ACAD Website</title>
    <link rel="stylesheet" href="styles/screen.css" media="screen" type="text/
css" />
  </head>
  <body>
    <div class="frame">
      <div class="header">
        <div class="logo">
          
        </div>
        <div class="navigation">
          <div class="nav"><a href="index.html">Home</a></div>
          <div class="nav"><a href="work.html">Work</a></div>
          <div class="nav"><a href="about.html">About</a></div>
          <div class="nav"><a href="location.html">Location</a></div>
        </div>
      </div>
      <div class="container">
        <div class="sidebar">
          
          <br/>
          
        </div>
        <div class="content">
          <h1>My Website</h1>
          <p>My website is great!</p>
        </div>
      </div>
      <div class="footer">
        Copyright &copy; Paul Bazay
      </div>
    </div>
  </body>
</html>
```

Style using CSS

Previewing this HTML file in the browser will initially show unexpected results. The content is not anywhere near what you are after. This is because the structure is in place but the styles for each element have not been delivered to the page. In order to control the proportions, positions, margin, padding, typography and colors of your elements you must create them in your CSS file. Our sample CSS file might look like this.

styles/screen.css code

```
body{
    background-color: "#FFFFFF";
    margin-top: 100px;
    margin-left: 46px;
}

h1 {
    font-size:18px;
    font-family: Arial;
    font-weight:bold;
}

p {
    font-size:12px;
    font-family: Arial;
    line-height: 18px;
}

.container {
    width:950px;
}

.header {
    width: 860px;
    height: 100px;
}

.logo {
    width: 206px;
    float: left;
}

.navigation {
    width:591px;
    float: right;
    font-size:12px;
    font-family: Arial;
    font-weight: bold;
    line-height: 18px;
}

.navigation a {
    text-decoration: none;
    color: #000000;
}

.navigation a:hover {
    color: #FF0000;
}

.navigation a:active {
    color: #333333;
}

.navigation a:visited {
    color: #666666;
}

.nav {
    margin-right:25px;
    float: left;
}

.content {
    width:860;
    float: left;
}

.sidebar {
    width: 206px;
    float: left;
}

.main_content {
    width:591px;
    margin-top:-14px;
    float: right;
}

.footer {
    margin-top: 52px;
    padding-top: 16px;
    border-top: 1px solid #3a94ae;
    font-size:10px;
    width:860px;
    font-family: Arial;
    float: left;
}
```



Project 2	Sketchsite
Discussion	DIV/CSS HTML development
Objective	Using the coding principles you have learned you must create a functioning website to house web-content that you will create in future assignments throughout this class.
Project	<p>Using HTML text editor only</p> <p>Similar to a traditional sketchbook, your sketchsite will function as a place for you to explore new concepts and techniques as learned. Although the pages of your sketchsite will initially be blank (with the exception of interface design) you will need to consider the following items:</p> <p>How will your visitors know it is your sketchsite? How will your visitors successfully navigate the pages of your sketchsite? How will you manage your sketchsite as it grows?</p>
Information Architecture	Your sketchsite must contain 5 pages (min) properly coded using a DIV structure with styles controlled via a CSS file.
Navigation	The pages of your sketchsite must be able to access all pages from any page.
Interface	Design. Your sketchsite must clearly communicate who owns the site and be reflective of the website goal.
Applications	Coda, TextEdit, TextMate (Mac), Komodo Edit, Notepad (PC) Photoshop, Illustrator
Online Examples	http://ffffound.com http://www.destroyrockcity.com http://www.presstube.com
Due date	Monday September 29th, 2008