|  |  |
| --- | --- |
| 1 | Layout Techniques |
| 2 | Best Practices |

**1. Layout Techniques**

While creating any basic Html layout using CSS, we use CSS properties like **float,** **display: Inline block** or **flexbox.**

But we are confused whether to use **float, inline-block** or **flexbox.**Both **float** and **inline-block** work fine in different scenarios when you just want to place HTML elements side-by-side, but each has its limitations.  
  
As long as you use a **clearfix** hack on the parent container, **floats** work just fine, but you can’t center floats

Inline-block elements, however, can be centered fairly easily with text-align: center but you can’t easily align two elements to opposite sides of a parent container like you can with a simple **float:left** and **float:right**.  
  
Flexbox solves the shortcomings listed above and many more, We can use **flexbox** and **inline-block** (or **flexbox** and **floats**)  
  
**Flexbox** will ignore float and inline-block properties, which means it can be treated as an enhancement to our layouts. Therefore, older browsers will simply use the float and inline-block-style layouts, while modern browsers will use the flexbox goodness.

Here is one example in below url of layering flexbox on top of inline-block, and the enhancements flexbox brings along with it.  
  
<https://codepen.io/joeaugie/pen/KdaaBb>  
  
**What if we wanted to have each grid item equal height and position the button to always be at the bottom of the container, no matter the whitespace above it?**To accomplish this with the inline block layout, we may have to write some javascript to equal out the heights and do some funky absolute positioning in order to keep the buy button at the bottom of the container. Not fun and very time consuming.  
  
Here are some steps we can take:  
  
Next, use **flex-direction:column** to change each grid item’s axis to vertical. (By default, it's horizontal)

Then, since the axis is vertical, use **justify-content: space-between** to pin the top content (the image, title and paragraph all inside a div) to the top of the grid container, and the buy button to the bottom.

Finally, use **align-items: center** so the buy button doesn’t get stretched out, because by default, align-items is set to stretch.  
  
Here we can find result:  
<https://codepen.io/joeaugie/pen/WQRjwx>

Here you can find video url for same:  
<https://www.youtube.com/watch?v=Bp9ctfvk3ww>

For complete knowledge of flex-box you can go through with below url:  
<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

**HTML 5 Semantic Layout**This is in reference to a shift in how tags are named in the HTML markup. So in addition to things like div which could mean anything, there are new tags like:

**Header  
article  
aside  
nav  
footer** etc.

The use of these tags provides more information about the semantic content of the markup, as opposed to relying on specifications that are more about style than content (such as **<div class="header">**).

The idea here is that the content itself tells you "what" it is. A clear benefit is for things like screen readers for visually impaired users, which can more effectively present the content to the user in a more structured manner.  
  
Please go through with below url for more information:  
<https://codepen.io/mi-lee/post/an-overview-of-html5-semantics>

**2. Best Practices**

**HTML Coding Practices**

A lot of coding best practices emphasize keeping code lean and well organized. The general practices within HTML are no different. The goal is to write well-structured and standards-compliant markup. The guidelines described here provide a brief introduction to HTML coding practices; this is by no means an exhaustive list.

**Make Use of Class Attribute instead of Id**

**Bad Code**

|  |  |
| --- | --- |
| 1  2  3 | <p id="intro">New items on the menu today include <strong>caramel apple cider and breakfast crepes</p>.</strong>  <p id="intro">The caramel apple cider is delicious. |

**Good Code**

|  |  |
| --- | --- |
| 1  2  3 | <p class="intro">New items on the menu today include <strong>caramel apple cider and breakfast crepes</strong>.</p>  <p class="intro">The caramel apple cider is delicious.</p> |

**Make Use of Semantic Elements**

**Bad Code**

|  |  |
| --- | --- |
| 1  2  3  4  5 | <span class="heading"><strong>Welcome Back</span></strong>  <br><br>  It has been a while. What have you been up to lately?  <br><br> |

**Good Code**

|  |  |
| --- | --- |
| 1  2  3 | <h1>Welcome Back</h1>  <p>It has been a while. What have you been up to lately?</p> |

**Use the Proper Document Structure**

###### Bad Code

|  |  |
| --- | --- |
| 1  2  3  4  5 | <html>  <h1>Hello World</h1>  <p>This is a web page.</p>  </html> |

**Good Code**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <!DOCTYPE html>  <html>  <head>  <title>Hello World</title>  </head>  <body>  <h1>Hello World</h1>  <p>This is a web page.</p>  </body>  </html> |

**Keep the Syntax Organized**

###### Bad Code

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <Aside>  <h3>Chicago</h3>  <H5 HIDDEN='HIDDEN'>City in Illinois</H5>  <img src=chicago.jpg alt="Chicago, the third most populous city in the United States" />  <ul>  <li>234 square miles</li>  <li>2.715 million residents</li>  </ul>  </ASIDE> |

**Good Code**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <aside>  <h3>Chicago</h3>  <h5 hidden>City in Illinois</h5>  <img src="chicago.jpg" alt="Chicago, the third most populous city in the United States">  <ul>  <li>234 square miles</li>  <li>2.715 million residents</li>  </ul>  </aside> |

**Use Practical ID & Class Values**

###### Bad Code

|  |  |
| --- | --- |
| 1  2 | <p class="red">Error! Please try again.</p> |

**Good Code**

|  |  |
| --- | --- |
| 1  2 | <p class="alert">Error! Please try again.</p> |

**Use the Alternative Text Attribute on Images**

**Bad Code**

|  |  |
| --- | --- |
| 1  2 | <img src="puppy.jpg"> |

**Good Code**

|  |  |
| --- | --- |
| 1  2 | <img src="puppy.jpg" alt="A beautiful, two-year-old hound mix puppy"> |

**Separate Content from Style**

**Bad Code**

|  |  |
| --- | --- |
| 1  2 | <p style="color: #393; font-size: 24px;">Thank you!</p> |

**Good Code**

|  |  |
| --- | --- |
| 1  2 | <p class="alert-success">Thank you!</p> |

**CSS Coding Practices**

Similar to those for HTML, the coding practices for CSS focus on keeping code lean and well organized. CSS also has some additional principles regarding how to work with some of the intricacies of the language.

**Organize Code with Comments**

**Bad Code**

|  |  |
| --- | --- |
| 1  2  3  4 | header { ... }  article { ... }  .btn { ... } |

**Good Code**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | /\* Primary header \*/  header { ... }  /\* Featured article \*/  article { ... }  /\* Buttons \*/  .btn { ... } |

**Write CSS Using Multiple Lines & Spaces**

**Bad Code**

|  |  |
| --- | --- |
| 1  2 | a,.btn{background:#aaa;color:#f60;font-size:18px;padding:6px;} |

**GOOD CODE**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | a,  .btn {  background: #aaa;  color: #f60;  font-size: 18px;  padding: 6px;  } |

**Use Proper Class Names**

###### Bad Code

|  |  |
| --- | --- |
| 1  2 | .Red\_Box { ... } |

**Good Code**

|  |  |
| --- | --- |
| 1  2 | .alert-message { ... } |

**Build Proficient Selectors**

**Bad Code**

|  |  |
| --- | --- |
| 1  2  3 | #aside #featured ul.news li a { ... }  #aside #featured ul.news li a em.special { ... } |

**Good Code**

|  |  |
| --- | --- |
| 1  2  3 | .news a { ... }  .news .special { ... } |

**Use Specific Classes When Necessary**

**Bad Code**

|  |  |
| --- | --- |
| 1  2 | section aside h1 em { ... } |

**Good Code**

|  |  |
| --- | --- |
| 1  2 | .text-offset { ... } |

**Use Shorthand Properties & Values**

**Bad Code**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | img {  margin-top: 5px;  margin-right: 10px;  margin-bottom: 5px;  margin-left: 10px;  }  button {  padding: 0 0 0 20px;  } |

**Good Code**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | img {  margin: 5px 10px;  }  button {  padding-left: 20px;  } |

**Use Shorthand Hexadecimal Color Values**

**Bad Code**

|  |  |
| --- | --- |
| 1  2  3  4  5 | .module {  background: #DDDDDD;  color: #FF6600;  } |

**Good Code**

|  |  |
| --- | --- |
| 1  2  3  4  5 | .module {  background: #ddd;  color: #f60;  } |

**Drop Units from Zero Values**

**Bad Code**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | div {  margin: 20px 0px;  letter-spacing: 0%;  padding: 0px 5px;  } |

**Good Code**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | div {  margin: 20px 0;  letter-spacing: 0;  padding: 0 5px;  } |

**Modularize Styles for Reuse**

###### Bad Code

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | .news {  background: #eee;  border: 1px solid #ccc;  border-radius: 6px;  }  .events {  background: #eee;  border: 1px solid #ccc;  border-radius: 6px;  } |

**Good Code**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | .feat-box {  background: #eee;  border: 1px solid #ccc;  border-radius: 6px;  } |

**CSS Properties in Alphabetic Order**

###### Bad Code

|  |  |
| --- | --- |
| 1  2  3  4  5 | .news {  background: #eee;  width: 100px;  float: left;  } |

**Good Code**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | .news {  background: #eee;  float: left;  width: 100px;  } |
|  |  |