XHTML

The Div Tag and More CSS
The Mighty `<div>` Tag

The `<div>` element is used for formatting large blocks of text, images, and just about anything else that has an HTML tag around it.

Using CSS and the `<div>` tag, you can place elements exactly where you want them, without interrupting the flow of your document’s structure.
The <div> Tag

<div>
<h3>Hi, welcome to the div tag.</h3>
<p><img src="mydiv.gif"></p>
<p>All these elements are contained within a div</p>
</div>
Class and ID Selectors

- Previously we looked solely at HTML selectors - those that represent an HTML tag.

- You can also define your own selectors in the form of Class and ID selectors.

- The benefit of this is that you can have the same HTML element, but present it differently depending on its class or ID.

- In the CSS, a class selector is a name preceded by a full stop (.) and an ID selector is a name preceded by a hash character (#).
Class and ID Selectors

- So the CSS might look something like:

```css
#top {
  background-color: #ccc;
  padding: 1em
}
.intro {
  color: red;
  font-weight: bold;
}
```
Class and ID Selectors

- The HTML refers to the CSS by using the attributes `id` and `class`. It could look something like this:

```
<div id="top">
<h1>Chocolate curry</h1>
<p class="intro">This is my recipe for making curry purely with chocolate</p>
<p class="intro">Mmm mm mmmmm</p>
</div>
```
The difference between an ID and a class is that an ID can be used to identify one element, whereas a class can be used to identify more than one.

You can also apply a selector to a specific HTML element by simply stating the HTML selector first, so `p.jam { whatever }` will only be applied to paragraph elements that have the class 'jam'.
With this in mind, the div properties we are going to look at are:

<table>
<thead>
<tr>
<th>position</th>
<th>height</th>
</tr>
</thead>
<tbody>
<tr>
<td>left</td>
<td>z-index</td>
</tr>
<tr>
<td>top</td>
<td>visibility</td>
</tr>
<tr>
<td>width</td>
<td>clip</td>
</tr>
</tbody>
</table>
CSS Positioning

- **Position**

  - Can be *relative* or *absolute* (default is *relative*)

  - *relative*: works the way you'd expect a normal html page to flow. Things come in from the top and lay themselves out, with each element's position determined by the previous element's end point.

    - The *absolute* setting positions elements based on their distance from their parent's upper-left corner.
CSS Positioning

- **Position**

  - So if you have a paragraph that isn't contained within any elements, its position is determined by the upper left-hand corner of the body.

  - But if you've contained it within a `<div>` tag, the paragraph's position is determined by the upper-left corner of the `<div>` that contains it.
CSS Positioning

- `left` and `top`

```css
div{
    position:absolute;
    left:135px;
    top:100px;
    background:grey
}
```
CSS Positioning
CSS Positioning

- Absolutely positioned elements appear the number of units you specify from the left and top of their parent element.

- Units are in the standard CSS units: pixels - px, points - pt, inches – in, and percentage - %.

- So a `<div>` that's at the top level of the HTML hierarchy with an absolute positioning of left: 50px; top: 50px, would appear 50 pixels from the left, and 50 pixels from the top of the page.
CSS Positioning

**z-index**

- The *z-index* determines which elements are drawn over others.

- Eg., if you have two elements that inhabit the same space, you need to specify which gets drawn and which is hidden.

- The one with the highest z-index number gets placed on top, while the one with the lowest gets placed on the bottom.
CSS Positioning

- **z-index**

- This order is relative to the parent element.

- Even if an element has a *z-index* of a million, but its parent is at the bottom of the *z-index*, it can't rise above it in the overall scheme of things.
CSS Positioning

- z-index
CSS Positioning

- **visibility**
  - **visibility** controls whether or not the element is drawn on the screen.
  - values are **visible** and **hidden**, which are pretty much self-explanatory.

- **width and height**
  - **width** and **height** work pretty much the way they always have, but absolutely positioned objects can also have percentage widths and heights.
Now you can position things on the page, to the exact pixel.

Please remember that people still have monitors and browsers that are different sizes than the one you are currently using.
Grouping

- You can give the same properties to a number or selectors without having to repeat them by separating the selectors by **commas**.
- For example, if you have something like:

```css
h2 {
  color: red;
}
.cls1 {
  color: red;
}
.cls2 {
  color: red;
}
```
Grouping

- You could make it:

```css
h2, .thisOtherClass, .yetAnotherClass {
  color: red;
}
```
Nesting

If the CSS is structured well, there shouldn't be a need to use many class or ID selectors. This is because you can specify properties to selectors *within* other selectors. E.g.:

```css
#top {
  background-color: #ccc;
  padding: 1em
}
#top h1 {
  color: #ff0;
}
#top p {
  color: red;
  font-weight: bold;
}
```
Nesting

- This removes the need for classes or IDs if it is applied to HTML that looks something like this:

```html
<div id="top">
<h1>Chocolate curry</h1>
<p>This is my recipe for making curry purely with chocolate</p>
<p>Mmm mm mmmmm</p>
</div>
```
Nesting

- This is because, by separating selectors with spaces, we are saying 'h1 inside ID top is colour #ff0' and 'p inside ID top is red and bold'.

- This can get quite complicated (because it can go for more than two levels, such as this inside this inside this inside this etc.) and may take a bit of practice.
Pseudo Classes

- **Pseudo classes** are bolted on to selectors to specify a state or relation to the selector. They take the form of `selector:pseudo class {property: value;}`, simply with a **colon** in between the selector and the pseudo class.

- Many CSS proposals are not supported by all browsers, but there are four pseudo classes that can be used safely when applied to links.
  - **link** is for an unvisited link.
  - **visited** is for a link to a page that has already been visited.
  - **active** is for a link when it is gains focus (for example, when it is clicked on).
  - **hover** is for a link when the cursor is held over it.
Pseudo Classes

```css
a.snowman:link {
  color: blue;
}
a.snowman:visited {
  color: purple;
}
a.snowman:active {
  color: red;
}
a.snowman:hover {
  text-decoration: none;
  color: blue;
  background-color: yellow;
}
```
**Pseudo Classes**

- **Note:** Although CSS gives you control to bypass it, maintaining different colours for visited links is good practice as many users still expect this. Traditionally, text links were **blue** if not visited and **purple** if visited, and there is still reason to believe that these are the most effective colours to use.

- **Note 2:** You should also be able to use the hover pseudo class with elements other than links.
Shorthand Properties

- Some CSS properties allow a string of values, replacing the need for a number of properties. These are represented by values separated by \textit{spaces}.

- \textit{margin}, \textit{padding} and \textit{border-width} allow you to amalgamate \textit{margin-top-width}, \textit{margin-right-width}, \textit{margin-bottom-width} etc. in the form of \texttt{property: top right bottom left};
Shorthand Properties

- So:
  ```
  p {
  border-top-width: 1px;
  border-right-width: 5px;
  border-bottom-width: 10px;
  border-left-width: 20px;
  }
  ```

- Can be summed up as:
  ```
  p {
    border-width: 1px 5px 10px 20px;
  }
  ```
Shorthand Properties

- `border-width`, `border-color` and `border-style` can also be summed up as, for example:

  ```css
  p {
    border: 1px red solid;
  }
  ```

  (This can also be applied to `border-top`, `border-right` etc.)

- By stating just two values (such as `margin: 1em 10em;`), the first value will be the top and bottom and the second value will be the right and left.
Shorthand Properties

- Font-related properties can also be gathered together with the font property:

```css
p {
  font: italic bold 1em/1.5 courier;
}
```

(Where the '/1.5' is the line-height)
Shorthand Properties

- So, to put it all together, try this code:

```css
p {
  font: 1em/1.5 "Times New Roman", times, serif;
  padding: 3em 1em;
  border: 1px black solid;
  border-width: 1px 5px 5px 1px;
  border-color: red green blue yellow;
  margin: 1em 5em;
}
```
Background Images

- There are a number of properties involved in the manipulation of background images.

- Luckily, the property background can deal with them all.

```css
body {
    background: white url(kitty.jpg) no-repeat top right;
}
```
This amalgamates these properties:

- **background-color**, which we came across before.
- **background-image**, which is the location of the image itself.
- **background-repeat**, which is how the image repeats itself. This can be **repeat** (equivalent to a 'tile' effect across the whole background), **repeat-y** (repeating on the 'y-axis', above and below), **repeat-x** (repeating on the 'x-axis', side-by-side) or **no-repeat** (which shows just one instance of the image).
- **background-position**, which can be **top**, **center**, **bottom**, **left**, **right** or any sensible combination, such as above.
Background Images

- Background-images can be used in most HTML elements - not just for the whole page (body) and can be used for simple but effective results, such as shaped corners.

**Note:** It is easy to get carried away with background images and plaster them all over your web pages. Some visually hyperactive people might believe it looks good to have a full-on brightly coloured photograph tiled across the background of a page, giving the user a serious challenge in deciphering the foreground text. This is an extreme example, but the fact is that the most user-friendly, readable text is black on a plain white background or white on a plain black background (there is also a suggestion that a slightly off-white or off-black background is better as this reduces glare).

So, the best use of background images is either to use them where there will be no content over the top or making the background image very light, which would also reduce the file size of the image, because there should be less colours involved (supposing you are using an indexed-colour format, such as GIF).
Useful Links

For generating hex values:

- [http://www.w3schools.com/tags/ref_colorpicker.asp](http://www.w3schools.com/tags/ref_colorpicker.asp)