Grouping

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

```css
h2 {
  color: red;
}
.classthisisOtherClass {
  color: red;
}
.classtheynothereeklass {
  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>
```
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.

- The most common pseudo classes are those 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

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 **spaces**.

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

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

Can be summed up as:
```css
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.
Font-related properties can also be gathered together with the font property:

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

(Where the '/1.5' is the line-height)
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.

```
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:
Validate your Pages

- It's important to validate your Web pages because most browsers rely on correct HTML to display the pages as you expect.

- Once you know your HTML is correct, you can move on to other steps in the troubleshooting path to solve other problems you might have.
Common Validation Problems and How to Fix Them

- Using a validator can be tricky. They give errors that you might not expect and can be hard to read and interpret.

- When you've submitted a page to the validator, after a few minutes you'll get a window with results explaining what errors you have.

- **You probably won’t get a completely valid page the first time they try.** So the most important thing to remember is that you're not a failure if your page fails validation.
Common Validation Problems and How to Fix Them

1. **Read the text of each error fully.**

   - Many validators, including the W3C validator, include a lot of information in the errors.
   
   - This can be overwhelming, but if you take the time to read the error, you usually find that the answer to fixing it is written right in the error itself.
Common Validation Problems and How to Fix Them

2. **Fix each error in order.**
   - HTML is read by browsers and validators sequentially, so if you try to start at the bottom of the validation list, you'll end up spending more time correcting errors than if you just start with error #1, fix it and move on.

3. **Re-validate after every fix.**
   - Because HTML is read sequentially, a single problem at the very top of your document could be generating 20+ errors. If you fix error #1 and re-validate, you may find that you've fixed 17 other errors as well.
Common Validation Problems and How to Fix Them

4. **Go to the line number of the error and read up.**
   - This may seem counter-intuitive, but because the HTML is read from top to bottom, most errors are going to be found on the exact line of the error message or **before** that line in the HTML.

5. **Don't assume that the line and column numbers are accurate.**
   - The W3C validator gives you it's best guess as to where the problem starts, but it's not always right. Use that information as a guide to show you approximately where the problem may be in your code. And as with the previous step, work up from there to find the problem.
Some of the Common W3C Validator Reports and What to Do About Them

“Start tag seen without seeing a doctype first.”
- Put `<!DOCTYPE html>` at the start of your document.

“The character encoding was not declared.”
- Include `<meta charset="utf-8" />` within the head element.
Some of the Common W3C Validator Reports and What to Do About Them

“End tag for “DIV" omitted, but its declaration does not permit this.”

- Any time you see that an end tag is omitted, make sure that a) it's there and b) it's not overlapping another tag.
- Incorrect nesting is the most common reasons for missing end tags.
- Remember that valid HTML doesn't allow tags to overlap one another. If you open a tag, you must close it before opening another one. E.g.

  `<div><p>text</p></div>`  - WRONG
  `<div><p>text</p></div>`  - RIGHT
Some of the Common W3C Validator Reports and What to Do About Them

“Bad Character”
- You haven’t encoded your entities.
- You must encode all ampersands (&), less-than (<), and greater-than (>) signs etc. to validate.
- This includes inside URLs.
Now test in Multiple Browsers

- It may be that the problem you're seeing is a result of the Web browser you're viewing it in.

Oct 2017 StatCounter Global Stats
Internet Explorer Testing Tools

- IE is well-known for bugs both with design layout and development scripts. There are several ways to attack these issues.

- Newer versions of IE provide debugging tools directly in the browser. Simply hit F12 when viewing a page and the developer tools appear in a new window.

- These F12 developer tools allow you to change the browser and document mode to view later versions, such as IE7 and IE8. They also allow you to inspect specific HTML and alter CSS styles.

- Since layout issues can be addressed directly in the browser, this should be the first method used to find a solution.
Free Tools

- **IETester**: provides the ability to replicate issues and bugs in the proper version of Internet Explorer.

- **Browser Shots**: one of the most effective free tools available. There is a comprehensive list of browsers and platforms available. Includes both recent browser versions along with legacy versions.

- **Browsera** includes a free version with limited browsers and low-resolution screenshots. However, does provide a detailed list of script errors and other potential problems.

- **Lunascape** is a free triple engine browser. That means that you can run and test a new website in Trident (IE), Gecko (Firefox) and Webkit (Chrome and Safari) and compare rendering engines side-by-side.
Simplify the Page

- If validating the HTML and CSS doesn't help, then you should narrow down the page to find the problem.

- The easiest way to do this is to comment out portions of the page until all that is left is the portion with the problem. You should also cut the CSS down in a similar fashion.

- The idea behind simplifying is not that you'll leave the page with only the fixed element, but rather that you'll determine what is causing the problem and then fix it.
Simplify the Page

- HTML Comments:
  <!--This won’t now display-->

- CSS comment:
  /*
   Write the text or information you want hidden
  Close the comment: */
Simplify the Page

CSS Comment Tips:

- Comments can span multiple lines.
- Comments can include CSS elements that you don't want in the document, but don't want to delete completely.
- Use comments whenever you write complicated CSS to add clarification.
- Comments can also include meta information like:
  - author
  - date created
  - copyright information
Subtract and then Add Back

- Once you have narrowed down the problem area of your site, begin subtracting elements out of the design until the problem goes away.

- E.g., if you've narrowed down the problem to a specific `<div>` and the CSS that styles it, begin by removing one line of CSS at a time.

- Test after every removal. If what you've removed fixes or completely removes the problem, then you know what you need to fix.
Subtract and then Add Back

- Once you know exactly what is causing the problem begin adding it back with items changed.

- **Be sure to test after every change.**
  - When you're doing Web development, it's surprising how often little things can make a difference.
  - But if you don't test how the page looks after every change, even seemingly minor ones, you may not determine where the problem is.
Design for Standards Compliant Browsers First

- The most common problems that Web designers face revolve around getting pages looking the same in most browsers.

- While we've discussed that it can be very difficult, if not impossible, to get Web pages to look the same in all browsers, it is still a goal of most designers.

- So you should start by designing for browsers like Firefox or Chrome, which are standards compliant. Once you have them working, you can play with the other browsers to get them working.
Keep Your Code Simple

- Once you've found and fixed your problems, you should stay vigilant to keep them from cropping up again later.

- The easiest way to avoid problems is to keep your HTML and CSS as simple as possible.

- Note that I'm not saying you should avoid doing something like creating rounded corners simply because the HTML or CSS is complicated. Only that you should avoid doing complex things when a simpler solution presents itself.
Inappropriate Title

This is specified in the `<title>` tag located in the `<head>` section of your web page.

It identifies a Web page when a user adds the page to his/her bookmarks.

Always specify a title and always use one that best describes your page.

E.g., do not use the title “Our Home Page” for the first page of a Web site for a college sports club. Use something more descriptive like “Dublin University Knights of the Campanile.”

Descriptive titles are also useful for SEO.
Common HTML Mistakes

- **Missing End Tags**
  - This is one of the easiest mistakes to make.
  - html5 is very forgiving but by always closing your tags you avoid potential nesting problems.
  - Use proper indentation and spacing – makes it easier to spot.

- **Forgetting to Spellcheck your Pages**
  - Always spellcheck your content.
  - Play close attention to buttons and icons that contain text.
  - A simple spelling error can make your whole site seem unprofessional.
Common HTML Mistakes

- Not Viewing Your Pages after They Are Uploaded
  - Just because your pages are fine and functioning locally, do not assume they will be fine when transferred to the web server.
Common CSS Problems

- Using incorrect Measurement Units.
  - Think about what units are appropriate
  - Specify them correctly.

- Creating Conflicting Rules
  - This can happen if you are using a mixture of inline, internal, and external style sheets.
  - Try not to!
Common CSS Problems

- **Using Fixed Font Sizes**
  - Not all users have the same type, style, and size of fonts installed on their computer.
  - Try and design your page so that flexible font sizes work.
  - Give multiple options for font styles.

- Forgetting to include semicolons

- **Using HTML instead of CSS for Formatting**
Common CSS Problems

- **Not inserting comments**
  - Most stylesheet selectors are abbreviations such as `.bhdr` for a bold header or `.lhm` for left-hand margin.
  - If you don’t clearly comment this can make them difficult for someone else (or yourself at a later stage) to update.
Avoid Common Design Mistakes

- Not thinking about Screen Resolution and/or colour depth.
  - Remember that users have different resolutions and colour depths to you and test, test, test...

- Not allowing users to skip presentations
  - Splash pages must die
  - Introductory movies – skip option.

- Pop-up windows
  - Many people block them by default.
  - Don’t use unless you really, really have to.
Common CSS Problems

- ‘Under Construction’
  - If it’s not ready, don’t put it live!

- Not Proof-Reading your site.
  - Grammar, spelling...
  - Remember your buttons and icons.
Check for Image Problems

- **Not providing enough contrast for backgrounds**
  - When using images as backgrounds always specify the colour of the text on the page.
  - Make sure there is enough of a contrast between the background colour/image and the text.

- **Not including the Proper images for printing.**
  - Remember people can and will want to print your page.
  - Provide a separate stylesheet for printing

```html
<link rel="stylesheet" href="print.css" type="text/css" media="print" />
Check for Image Problems

- **Using images for text**
  - Frustrated with CSS typography control? Want to create something that looks nicer in Photoshop instead?
  - Don’t!
  - Not searchable
  - Larger file size

- **Using too many animated images**
  - Ouch...
Navigation Problems

- All web navigation should answer these questions:
  - Where am I?
  - Where have I been?
  - Where can I go next?
  - Where's the Home Page?

- Navigation must be simple and consistent.
Navigation Problems

- Common mistakes include
  - different types of navigation on the same site,
  - poorly worded links so visitors don't know where they'll end up if they click,
  - no links back to the home page
  - confusing links to the home page.
Mystical Belief in the Power of Web Standards, Usability, Tableless CSS and HTML5

- These are simply tools.

- Remember, nobody gets excited about the tools used to build a house ("Please tell me what brand of hammers you used!"). People get excited about how the house looks and performs.

- The priority is compelling content.