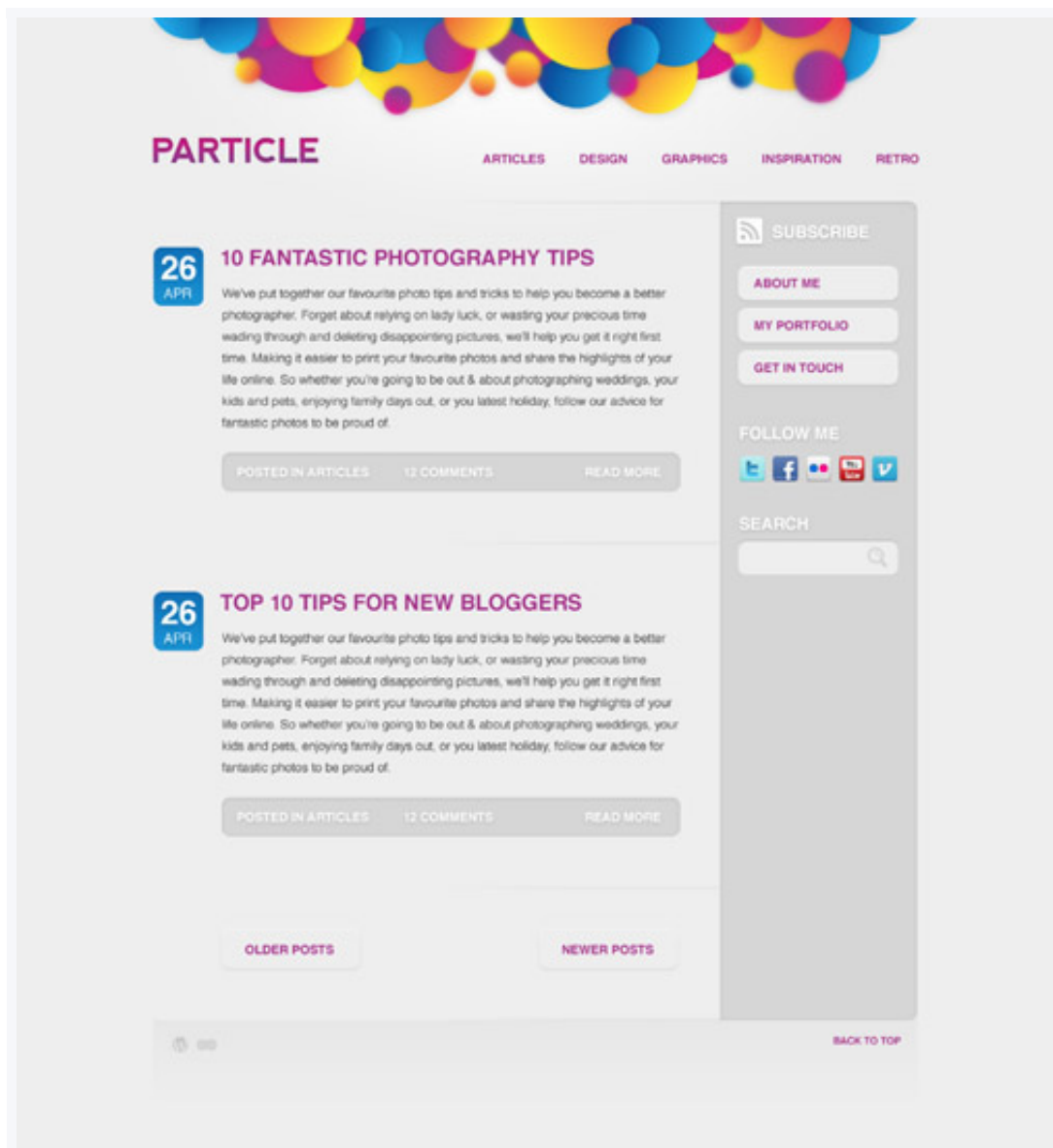


Coding Up a Web Design Concept into HTML & CSS

I've recently been working on a design concept for a WordPress theme as part of a personal project. In this walkthrough we'll go through the process of converting the design concept from PSD document right through to completed HTML and CSS mockup, complete with clean and valid code, a few touches of CSS3 and some quick fixes to help out old IE6.

The design concept

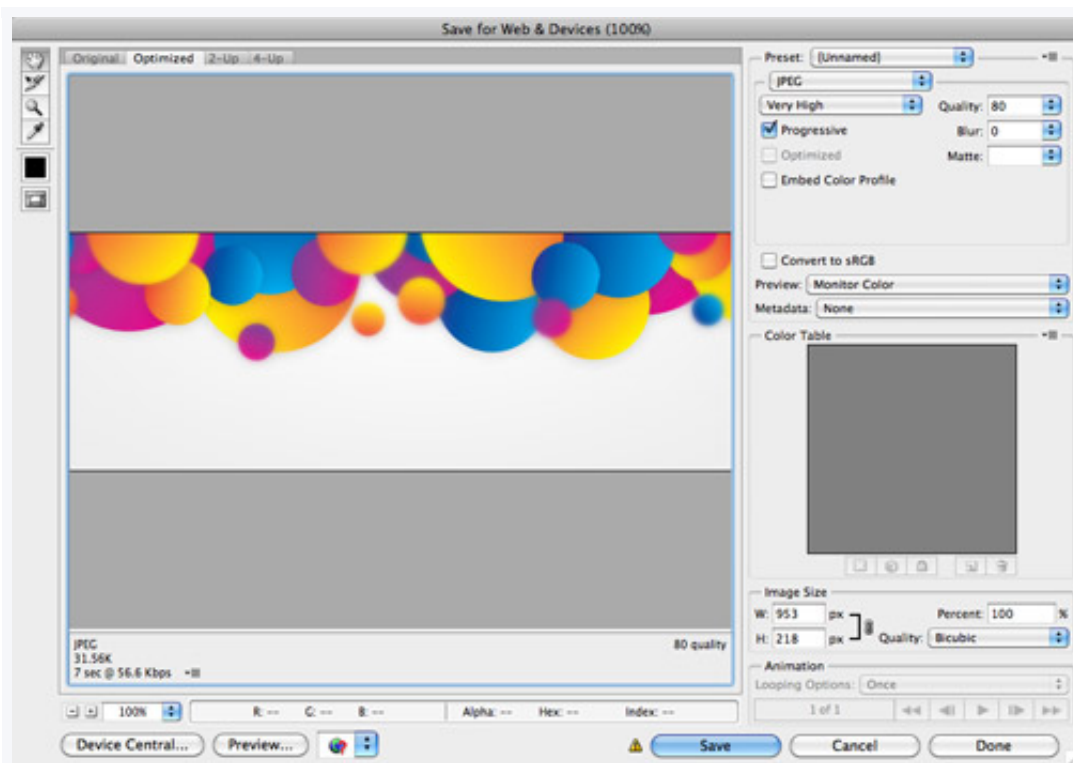


(<http://line25.com/wp-content/uploads/2010/coding-css-html/particle-concept.jpg>)

The design we'll be coding up is this WordPress theme concept I'm currently working on as part of a personal project. The design features a clean grey background, but with splashes of vibrant colour in the header, and throughout the page with links and buttons taking bright colour swatches from the main illustration. Overall the design has plenty of clean lines, and uses subtle shadows and inset text effects to add that touch of style.

View the demo (<http://line25.com/wp-content/uploads/2010/coding-css-html/demo/demo.html>)

Slice the PSD



The job is started by slicing up the various images that make up the design. The first being the bright header graphic. With this image being heavy on colours and gradients, the JPEG option would be the best fit.



Each subsequent element of the design is cut up. Layers are toggled off to allow access to background images, such as the date area and the button graphics.



Once every graphic has been exported, we're left with a collection of image files, some large, some small, some JPEG and some PNG. Certain images will be used as repeating background images, while others are combined as sprite graphics.

Build the HTML structure

A new HTML page is created and the basic structure of the document written out, including Doctype, links to stylesheets and a container div to hold the following markup.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.c
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Particle</title>

<link href="style.css" rel="stylesheet" type="text/css" media="screen" />

</head>

<body>
  <div id="container">

    </div>


</body>
</html>
```

The header

The first portion of design at the top of the page outlines a simple header. The logo can be recreated as a `<h1>`, seeing as there's no other title on the page, while the categories list can be formed with a good old unordered list.

```
<div id="header">
  <h1><a href="#">
```

```
        <li><a href="#">Articles</a></li>
        <li><a href="#">Design</a></li>
        <li><a href="#">Graphics</a></li>
        <li><a href="#">Inspiration</a></li>
        <li><a href="#">Retro</a></li>
    </ul>
</div>
```



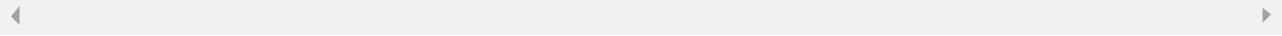
The content

Next we add a `<div>` with an ID of content to contain the main areas of the layout. This particular design uses two columns, one main content area and a sidebar. These are written out as individual `<div>` tags with corresponding IDs.

```
<div id="content">
    <div id="main"></div>

    <div id="side"></div>

</div>
```



Inside the main div sits the list of blog posts, each one contained within a `<div>` with a class of post. A class is used here rather than an ID because there will be multiple posts, whereas IDs may only appear once. Inside the post div the information is written out in the post relevant HTML elements, a `<p>` holds the number that will be generated for the date and a `` used to provide a target for the later CSS that will allow the month to appear at a different size. The next heading element available is a `<h2>`, so this is used along with an anchor to generate the post title.

A `` is then given a special class as post-meta, which then includes the information about the post in list format. The `` containing the read more link is given an extra class to allow this particular list item to be floated over the right, just like in the PSD concept.

```
<div id="main">
```

```

<div class="post">
  <div class="date">
    <p>26 <span>Apr</span></p>
  </div>

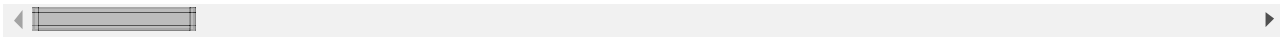
  <div class="post-content">
    <h2><a href="#">10 Fantastic Photography Tips</a>
    <p>We've put together our favourite photo tips ar

    <ul class="post-meta">
      <li>Posted in <a href="#">Articles</a></li>
      <li><a href="#comments">12 Comments</a></li>
      <li class="read-more"><a href="#">Read mc
    </ul>
  </div>
</div>

<div class="pagination">
  <p class="older"><a href="#">Older articles</a></p>
  <p class="newer"><a href="#">Newer articles</a></p>
</div>

</div>

```



Once the main div has been completely marked up, the next section of the design is the sidebar. The first item to appear in the div titled side, is the subscription option. This is written out as a plain old anchor, contained within a paragraph tag. A unique ID will allow us to target this exact anchor with CSS to add the relevant styling and RSS icon later.

The three links in the concept that look like buttons can simply be marked up as a `` and CSS can be used to give the button-like appearance.

```

<div id="side">
  <p id="subscribe"><a href="#">Subscribe</a></p>

  <ul id="pages">
    <li><a href="#">About me</a></li>
    <li><a href="#">My portfolio</a></li>
    <li><a href="#">Get in touch</a></li>
  </ul>

  <h3>Follow me</h3>
  <ul id="follow-me">
    <li><a href="#" class="twitter">Twitter</a></li>
    <li><a href="#" class="facebook">Facebook</a></li>
    <li><a href="#" class="flickr">Flickr</a></li>
  </ul>

```

```

        <li><a href="#" class="youtube">YouTube</a></li>
        <li><a href="#" class="vimeo">Vimeo</a></li>
    </ul>

    <h3>Search</h3>
    <form id="search" method="get" action="">
        <fieldset>
            <input type="text" class="search-bar" />
            <input type="submit" value="Search" class="search" />
        </fieldset>
    </form>
</div>

```



The footer

The simple footer can then be finished off with a couple of links housed within the footer

`<div>` . Each of the two credits are given a separate class, so each link can be transformed into the relevant logo image later with the CSS. Finally the back to top link is placed as an anchor targeting the header. You can target an anchor to any internal ID by using the `#` symbol.

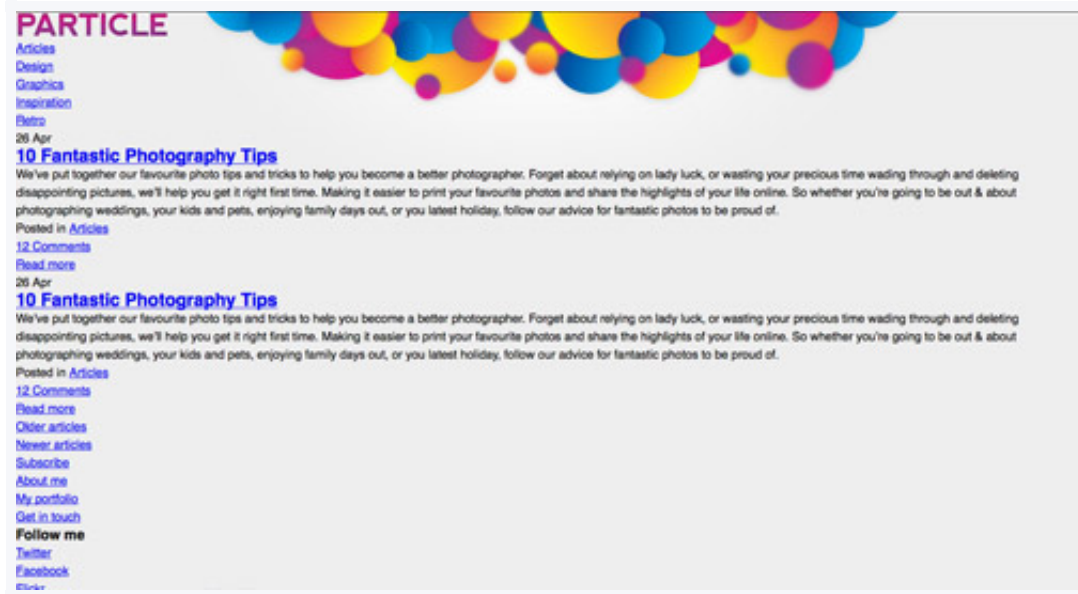
```

<div id="footer">
    <ul id="credits">
        <li><a href="http://wordpress.org" class="wordpress">Power
        <li><a href="http://www.blog.spoongraphics.co.uk" class="spoongraphics">Spoon
    </ul>
    <p id="back-top"><a href="#header">Back to top</a></p>
</div>

```



Write the CSS styling



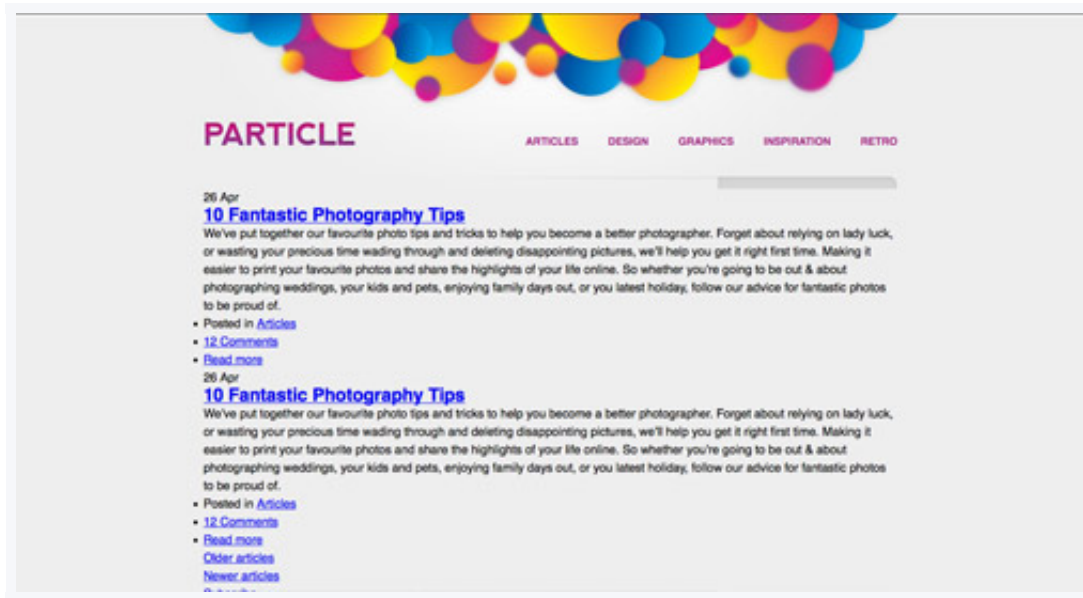
The first couple of lines in the CSS are used to reset any default browser styling, giving us a clean slate to start with, without the risk of inconsistencies between each browser. Next, the `body` is styled up with the basic font-family, and background. The colour `#eee` is specified, followed by the bright background image, which is positioned centrally on the page and told not to repeat. The container div provides the basic layout, allowing the design to appear centralised with the `margin: 0 auto;` declaration, and pushes the content down with a little padding so that it appears under the bright illustration at the top of the page.

```
body, div, h1, h2, h3, h4, h5, h6, p, ul, ol, li, dl, dt, dd, img, form,
    margin: 0; padding: 0; border: 0;
}

body {
    font-family: Helvetica, Arial, Sans-Serif; line-height: 24px;
    background: #eee url(images/body-bg.jpg) center top no-repeat;
}

#container {
    width: 925px; margin: 0 auto; padding: 143px 0 0 0;
}
```





We can then work down the HTML markup in order, adding the necessary CSS styling as we go. The header is the first section on the page, so this is given a background image forming the tip of the content area, and given some padding to match the PSD concept. The categories list is then floated off to the right, and each anchor given the various font sizing, and uppercase transformation to match the original design. A spot of CSS3 comes into play in the form of the `text-shadow` property to enable us to recreate the inset text effect. This extra styling will be visible to those with modern browsers, such as Safari, Firefox and Chrome, while IE will just show the flat version.

Colours are then sampled from the PSD and the hex codes pasted in place as the link and hover classes.

```
#header {
    background: url(images/header-bg.png) right bottom no-repeat; over
padding: 0 0 50px 0;
}

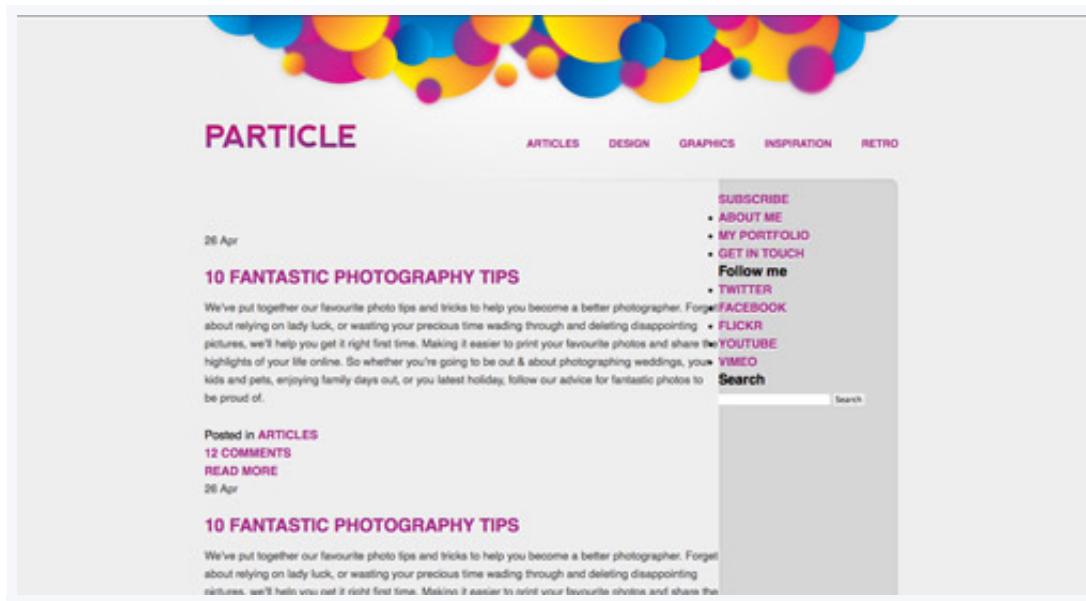
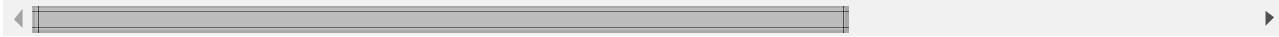
#header h1 {
    float: left;
}

#header ul#categories {
    float: right; list-style: none; margin: 16px 0 0 0;
}

#header ul#categories li {
    float: left; margin: 0 0 0 40px;
}
```

```
#header ul#categories li a {
    display: block;
    font-size: 14px; font-weight: bold; text-
    text-shadow: 0 2px 0px #fff; text-decorat
}

#header ul#categories li a:hover {
    color: #006ab1;
}
```



Next, the content div can be given a repeating background to create a faux column (http://line25.com/articles/create-sidebars-of-equal-height-with-faux-columns) effect. This image will generate the the darker grey sidebar, and will continue on from the background image added to the header. Because the main and side divs will be floated, the content div will need clearing. The easiest way to do this is to add `overflow: hidden; .`

The global anchor styling for the anchors are then added, so anything within the content div will be given this treatment, unless otherwise stated according to the more specific styling within the main or side divs.

```
#content {
    background: url(images/content-bg.png) right top repeat-y; overfl
}

#content a {
    font-weight: bold; text-transform: uppercase; color: #a42
    text-shadow: 0 2px 0px #fff; text-decoration: none;
```

```

}

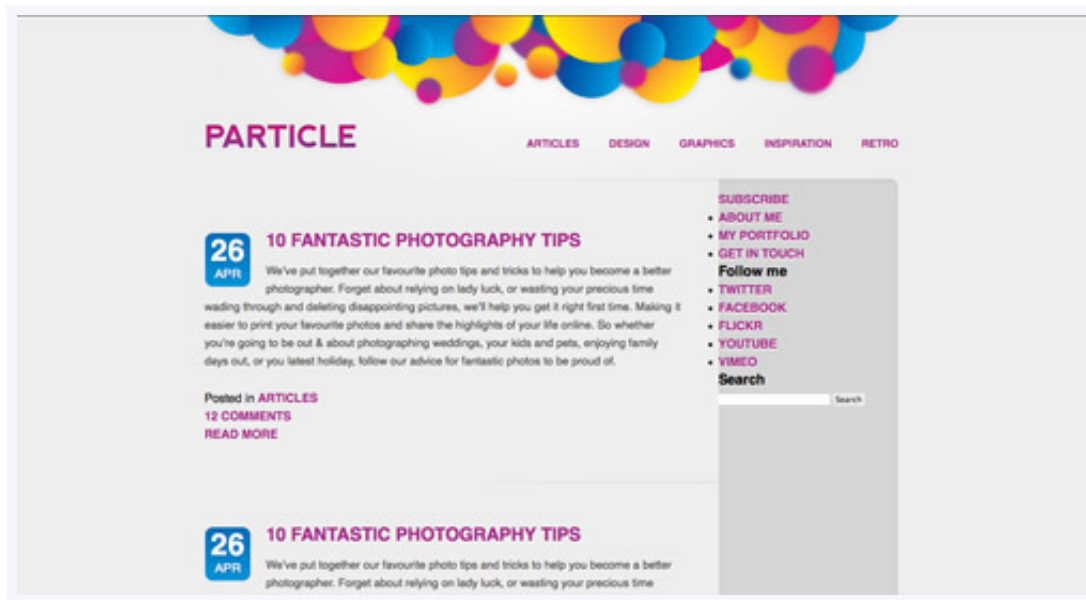
#content a:hover {
    color: #006ab1;
}

#content #main {
    width: 685px; float: left; padding: 55px 0 0 0;
}

#content #main h2 {
    margin: 0 0 16px 0;
}

#content #main p {
    margin: 0 0 25px 0; color: #474747; font-size: 15
}

```



Each individual post div is then given a background image to replicate the dividing lines between each entry. Some padding and margin helps recreate the same amount of white space as the concept, and the `overflow: hidden;` declaration is added to clear the post div after the floating the date and content side by side.

The date tab itself can be quickly styled up by specifying the exact dimensions and adding the blue background image. The `<p>` tag can then be given some typographic treatment to replicate the large white text in the concept, and the `` used to specifically target the month, making it smaller in size and block format so that it appears underneath the date number.

```

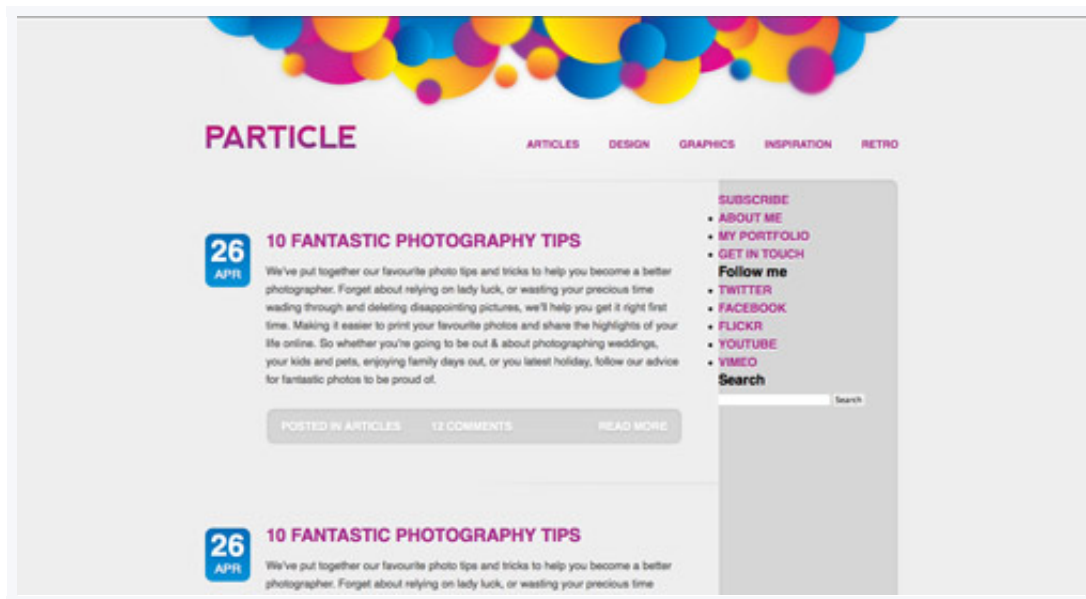
#content #main .post {
    background: url(images/post-divider.png) right bottom no-repeat;
    padding: 0 47px 55px 0; margin: 0 0 55px 0;
    overflow: hidden;
}

#content #main .post .date {
    width: 62px; height: 57px; float: left; padding: 15px 0 0 0;
    background: #0085cc url(images/date-bg.jpg);
    text-align: center;
}

#content #main .post .date p {
    font-size: 40px; font-weight: bold; color: #fff;
    text-shadow: 0 2px 3px #006ab1;
}

#content #main .post .date p span {
    margin: 5px 0 0 0;
    display: block; font-size: 17px; font-weight: normal;
}

```



With the date tab in place, the main content panel can also be floated alongside it. Then the styling can move on to fleshing out the meta information list with background image, and text styling. It's important to remember to add a background colour along with background images, just in case the user is viewing without images being displayed. Otherwise the white text will become unreadable on the light grey background.

The `` tags inside the meta list are floated to the left, with the exception of the read-more link, which is floated off to the right instead. That extra class allows this `` to be targeted

individually.

```
#content #main .post .post-content {
    width: 556px; float: left;
}

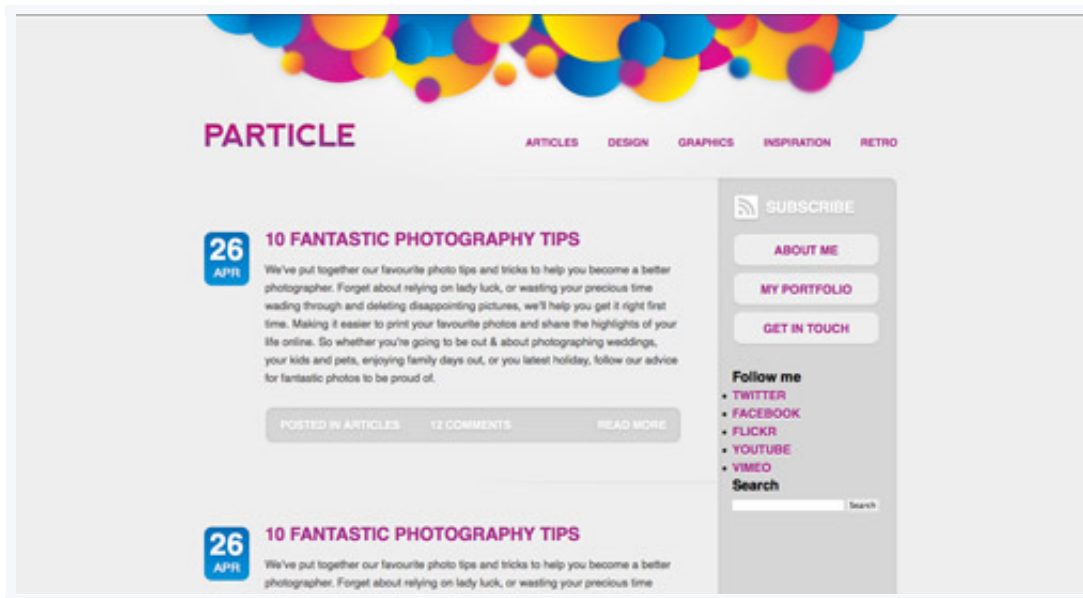
#content #main .post .post-content ul.post-meta {
    width: 515px; height: 21px; list-style: none; padding: 14px;
    background: #d6d6d6 url(images/meta-bg.jpg);
}

#content #main .post .post-content ul.post-meta li {
    float: left; margin: 0 40px 0 0;
    font-size: 15px; font-weight: bold; text-transform: uppercase;
}

#content #main .post .post-content ul.post-meta li:last-child {
    float: right; margin: 0 0 0 0;
}

#content #main .post .post-content ul.post-meta li a {
    color: #fff; text-shadow: 0px 2px 0px #000;
}

#content #main .post .post-content .post-meta {
    color: #eee;
}
```



The main div is now all styled up, so the next section is the sidebar. Its width is firstly calculated by taking away the width of the main div from the overall width of the container, taking into consideration any extra padding and margins. If this is calculated exactly, they'll both float side by side. A miscalculation of just 1 pixel will send the sidebar underneath the content, so here's

where that box model knowledge comes into play.

The first element within the sidebar is the subscribe button. The anchor is targeted directly through the unique ID, and is given the background image to generate the RSS icon. Some text-shadow soon styles up the text to replicate that of the original concept.

The button links in the sidebar are also styled up from basic anchors. First they need converting from inline to block elements, this then allows specific widths and heights to be added, along with the background image and necessary padding to push the text into place centrally. The actual colour and styling of the anchor text will be inherited from the previous styling to

#content a {} .

```
#content #side {
    width: 200px; float: left; padding: 5px 20px;
}

#content #side p#subscribe a {
    display: block; height: 30px; font-size: 20px; color: #fff;
    text-shadow: 0px 2px 3px #b8b8b8;
    background: url(images/rss-icon.png) left no-repeat;
}

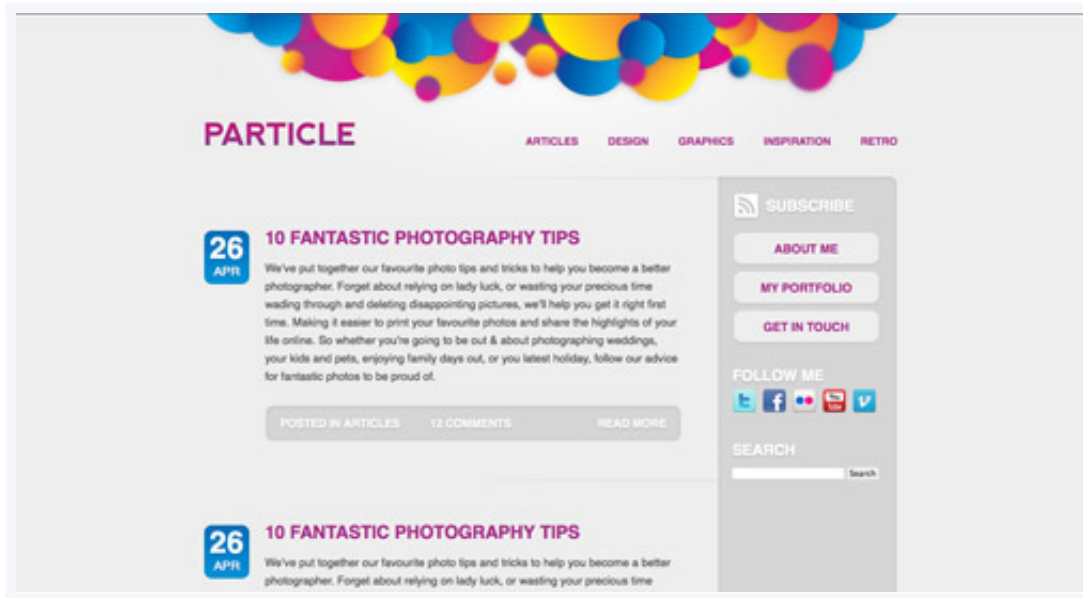
#content #side p#subscribe a:hover {
    color: #eee;
}

#content #side ul#pages {
    list-style: none; margin: 0 0 30px 0;
}

#content #side ul#pages li {
    margin: 0 0 5px 0;
}

#content #side ul#pages li a {
    display: block; width: 157px; height: 23px;
    background: #eee url(images/sidebar-btn.png) no-repeat;
}
```





The next feature in the design is the list of social icons (<http://www.komodomedia.com/download/#social-network-icon-pack>) (courtesy of Komodo Media (<http://www.komodomedia.com/download/#social-network-icon-pack>)). The basic markup is made up as a simple list and anchors, but they can be manipulated with CSS to give the desired appearance. First the list bullet points are removed with the `list-style: none;` declaration, and because the icons are floated, the `` will need clearing with `overflow: hidden;`. Each anchor is given specific dimensions, then each individual anchor can be targeted with the relevant class to add the correct icon as a background image.

```
#content #side h3 {
    font-size: 20px; text-transform: uppercase; color: #fff; margin:
    text-shadow: 0px 2px 3px #b8b8b8;
}

#content #side ul#follow-me {
    overflow: hidden; list-style: none; margin: 0 0 30px 0;
}

#content #side ul#follow-me li {
    float: left; margin: 0 8px 8px 0;
}

#content #side ul#follow-me li a {
    display: block; width: 32px; height: 32px;
    text-indent: -9999px;
}

#content #side ul#follow-me li a.twitter
    background: url(images/social-icc
}

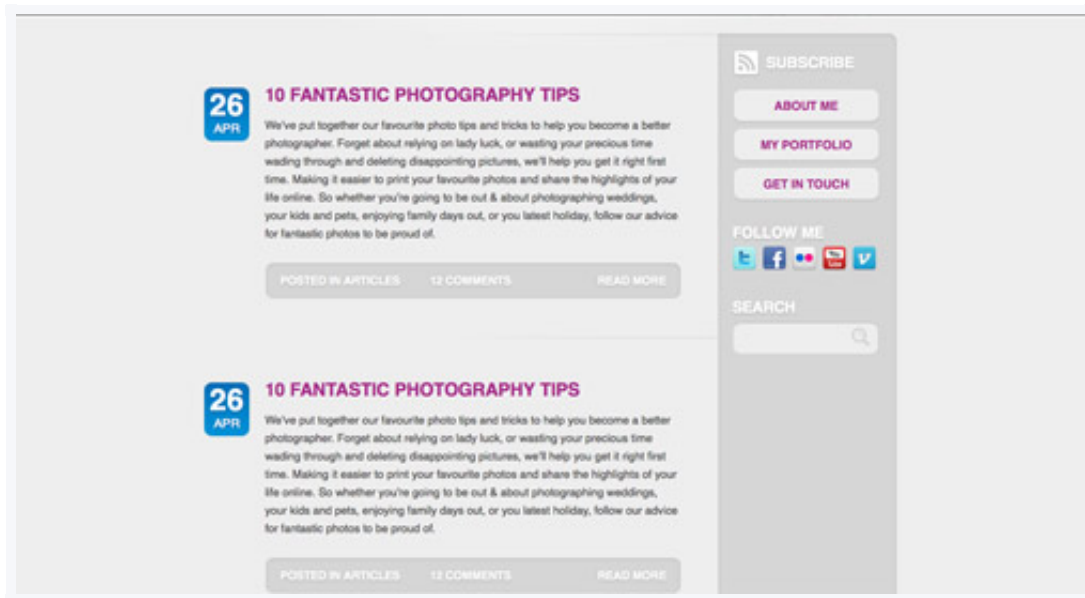
#content #side ul#follow-me li a.facebook
```



```

        background: url(images/social-icc
    }
    #content #side ul#follow-me li a.flickr {
        background: url(images/social-icc
    }
    #content #side ul#follow-me li a.youtube
        background: url(images/social-icc
    }
    #content #side ul#follow-me li a.vimeo {
        background: url(images/social-icc
    }

```



The search area then needs a couple of lines of styling to match it up to the concept. The basic structure is there from the HTML elements, but we can quickly spruce them up with the relevant background images. The search bar is given exact dimensions so the background image appears as expected. Some padding and font styling also tweaks how the input text will display. The search button itself appears as a small icon within the search bar. This is replicated with CSS by adding the appropriate background image, then moving it into place with floats and negative margins. Extra padding on the right side of the search bar then ensures the text won't overlap the button.

```

#content #side form#search {
    margin: 0 0 30px 0;
}

#content #side form#search input.search-bar {

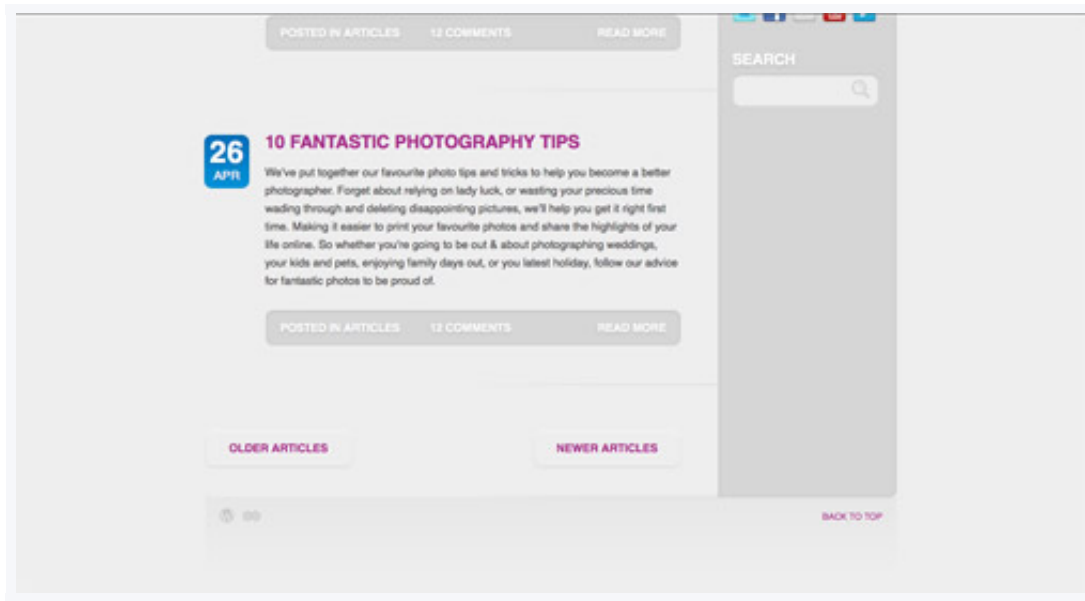
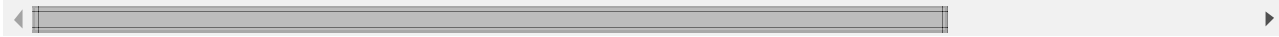
```



```

width: 135px; height: 43px; padding: 1px 45px 0 15px; float: right;
background: #eee url(images/search-bar.png);
font-size: 15px; color: #474747;
}
#content #side form#search input.search-btn {
width: 27px; height: 27px; float: right; margin: -36px 15px;
background: url(images/search-btn.png); text-indent: -99px;
}

```



Everything can then be finished off with some styling to the footer div. The footer background image includes the bottom portion of the content panel, giving the rounded corners and soft shadow, so this is positioned to the top. Inside the footer are two credit links, which are styled up using the image replacement technique just like the social icons. Finally there's the back to top link which is sent over to the right, and given the same link styling as the rest of the document.

```

#footer {
background: url(images/footer-bg.png) center top no-repeat; overflow: hidden;
min-height: 139px;
}

#footer ul#credits {
overflow: hidden; list-style: none; float: left; margin: 0;
}

#footer ul#credits li {
float: left; margin: 0 10px 0 0;
}

```

```

#footer ul#credits li a {
    display: block; height: 21px; background:
}

#footer ul#credits li a.wordpress { width:
#footer ul#credits li a.spoongraphics { v

#footer p#back-top {
    float: right; margin: 25px 20px 0 0;
}

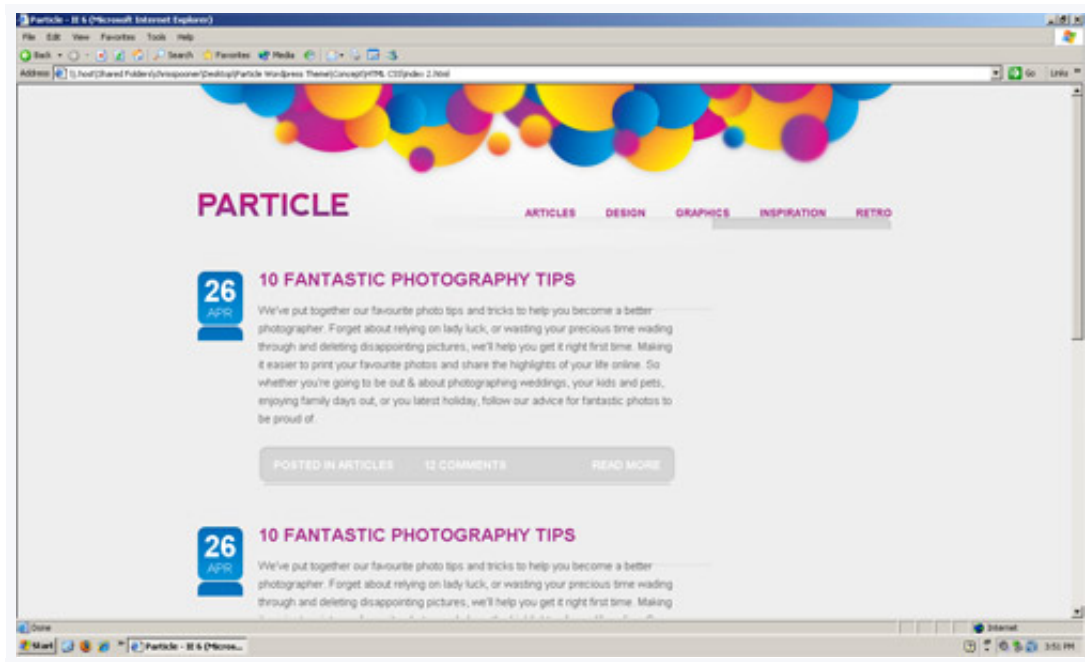
#footer p#back-top a {
    font-size: 12px; text-transform: uppercase; text-
    color: #a42988; text-shadow: 0 2px 0px #fff;
}

#footer p#back-top a:hover {
    color: #006ab1;
}

```



Fixing IE6



A quick test in the popular browsers shows no problems at all, with the design looking identical between the CSS3 supporting browsers of Safari, Firefox and Chrome, and just slightly less visually appealing in IE7. IE6 however, throws its usual panic attack at the CSS code and struggles to interpret it correctly.

At first impressions the site looks a complete mess, but under close inspection there's not too much to fix. Just a couple of extra CSS values soon brings everything into place. The two declarations of `overflow: hidden;` and `height: 100%;` were all that were needed on the problem elements. These are added using an IE6 only stylesheet link in the HTML.

```
#content #side, #content #main .post .date, #content #main .post .post-cc
    overflow: hidden;
}

#content, #footer, #header {
    height: 100%;
}
```

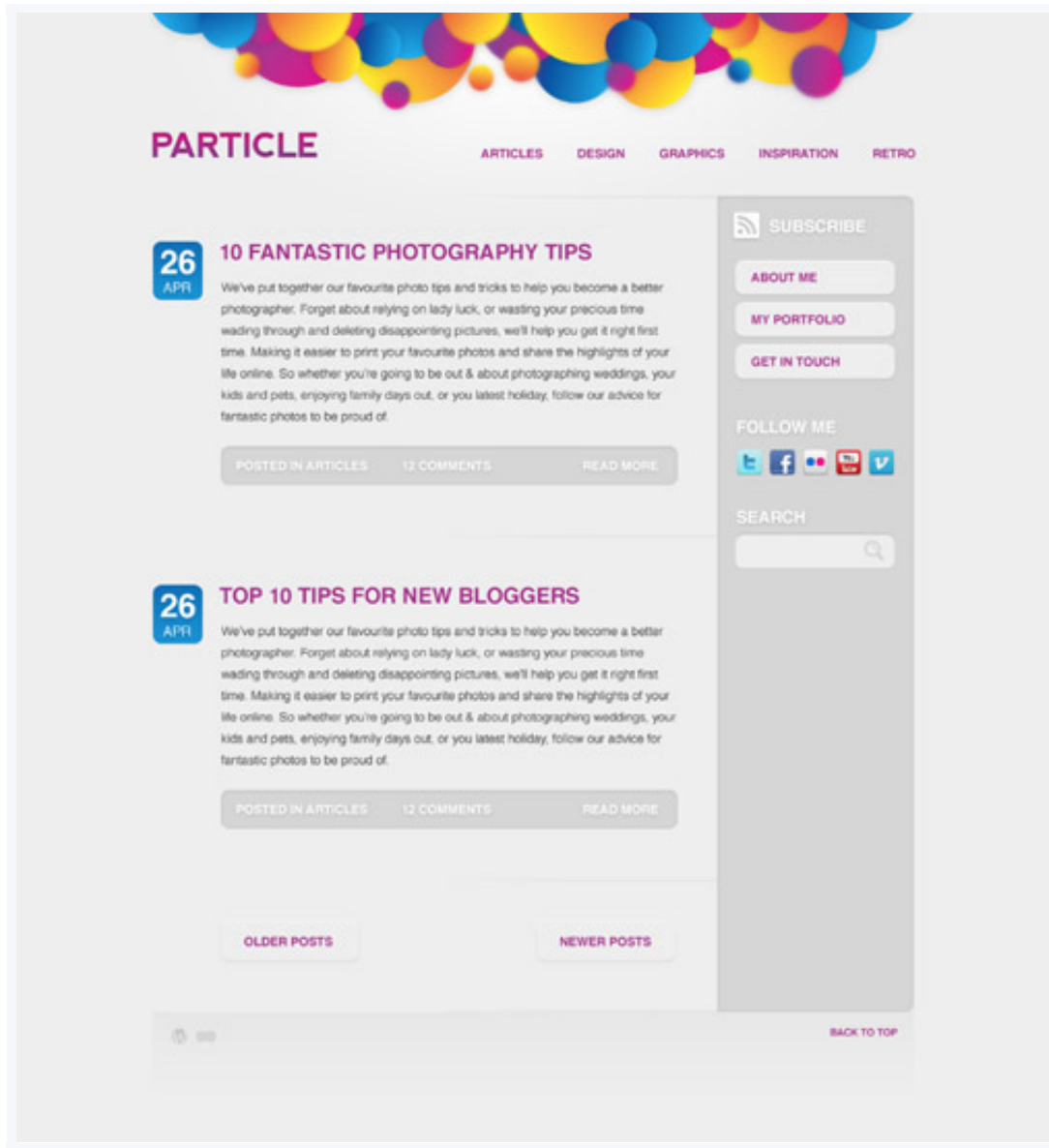


```
<!--[if IE 6]>
    <link href="css/ie6.css" rel="stylesheet" type="text/css" media="
<![endif]-->
```



The final design

With IE6 neatly tidied up, the site design is complete. Next week we'll be taking the design from this HTML/CSS mockup and transform it into a fully loaded WordPress theme. Stay tuned (<http://feeds2.feedburner.com/Line25>) !



(<http://line25.com/wp-content/uploads/2010/coding-css-html/particle-concept.jpg>)

View the demo (<http://line25.com/wp-content/uploads/2010/coding-css-html/demo/demo.html>)



Join the mailing list to have new content delivered straight to your email inbox. Every

subscriber gets a free pack of realistic web shadows (/subscribe) .

Subscribe



Written by Chris Spooner

Chris Spooner (/about) is a designer who loves experimenting with new web design techniques collating creative website designs. Check out Chris' design tutorials and articles at [Blog.SpoonGraphics](http://blog.spoongraphics.co.uk) (<http://blog.spoongraphics.co.uk>) or follow his daily findings on Twitter (<http://twitter.com/chris Spooner>) .