

# /CSS/create a navigation bar



Follow Craig Grannell through the process of generating a horizontal, two-tier navigation bar that you can use as the basis for navigation on as many websites as you like

**Knowledge needed** Intermediate CSS and HTML

**Requires** A text editor

**Project time** 20 minutes

The subject of this issue's tutorial is a horizontal, two-tier navigation bar, using HTML divs and lists for structure and CSS for styles. This can be used on various sites with just a few CSS rule changes and image swaps, all of which dramatically alter its look but not its actual structure. I'll also show you how to automate the active tab on a web page by way of class and id values, along with a couple of CSS rules (one for the default state and one for the hover state).

## Getting started

Copy the 'nav website' folder from the CD to your hard drive and check its contents. You should have a web page ('index.html'), three style sheets ('dot-net-nav-bar.css', 'ie-hacks.css' and 'ie5-hacks.css') and an 'assets' folder containing seven images.

In the 'head' section, the main style sheet – 'dot-net-nav-bar.css' – is imported via the 'style' element. The other two style sheets are linked via conditional comments – 'ie-hacks.css' is loaded by all versions of IE, while 'ie5-hacks.css' is loaded by versions of IE pre-version 6 ('lt IE6' literally means 'less than IE6').

The body section of the document has a few things that require mentioning. The first is the body start tag which, in the example page, has a class value of 'sectionBrowsers'. This is used in tandem with other defined values to highlight the relevant link on the navigation bar. Also, the first element on the page is a div with an id value of 'skipNav'. This will be a familiar device to most readers – it enables users viewing the page without CSS or on alternate devices (such as screen readers) to skip to the page's content rather than having to read through the navigation each time a page is accessed.

```
<div id="skipNav">
  <p><a href="#content">Skip to main content</a></p>
</div>
```

Structurally, the layout for this example page is simple. Within a wrapper div are a masthead div and a content div, the latter of which has a few headings and some paragraphs. Within the masthead div, there's a logo div (to house, unsurprisingly, the site's logo), and a searchArea div, within which is a simple dummy form for a site-wide search.

```
<div id="searchArea">
  <form action="http://somewhere" method="post">
  <p class="mainSearchLabel"><label for="mainSearchField">Search this website:</label></p>
  <p><input id="mainSearchField" type="text" size="25" tabindex="11" /></p>
  <p><input id="mainSearchSubmit" type="submit" value="GO" tabindex="21" /></p>
</form>
```

## Adding the navigation

Next, there's the navContainer div, which houses the navigation. Because this website has a two-tier navigation bar, it makes sense to use a separate div (with its own id value) to house each tier – this makes it much easier to style various elements in CSS. In the example, the first div's id is 'mainNavigation', and the second div's is 'subNavigation'.

In terms of style, each tier is distinct; because the subnavigation items show pages (or subsections of pages) relating to the chosen item in the main navigation, colour is used to show this relationship. The main navigation's background is dark grey, but the selected section link's background blends smoothly into the subnavigation's background. The subnavigation links don't need individual styling, hence the content of the subNavigation div being a

## Expert tip Logical navigation

When working on navigation, keep things logical. Categorise information and show your groupings to others, taking advice as necessary. Try to keep item names action-based ('buy now', 'contact us') or topic-based ('our shop', 'contact details'), and try to avoid mixing the two, especially in the same block of navigation. Finally, keep the names you use as succinct and obvious as possible. Navigation isn't the place to be poetic or flamboyant!



**Skip to the content** At the top-right, you can see the skip-navigation in its hover state. This is of little use to most users, but its value is clear when working with screen readers

# Modular thinking

## Save time by creating reusable elements

Many web designers have a kind of ground-up mentality when working on websites, in the same way that illustrators begin with a blank canvas. While that's perfectly fine from a graphic design standpoint, you can be far more efficient from a coding point of view if you start thinking in a modular way. Although the two-tier navigation in this issue's tutorial has a very specific look and feel, it's not so different from the navigation on the Snub Communications website ([www.snubcommunications.com](http://www.snubcommunications.com), pictured right). Both have a containing div with two divs nested within, each of which contains an unordered list, and both use a 'body' element start tag 'class' value and unique id for each link's containing list item, in order to highlight the current section/page.

The important point here is that although the designs are very different, the code is not. So, when you're working on sites, you should get used to storing generic, modular elements that can be reused. Often, entire web pages can be mostly constructed from a handful of such files, leaving you with more time to get on with implementing the styles and working on the design side of things.



**Recycle your code** Two-tier navigation systems use the same code and structure, but they can vary wildly in terms of design, and this site is a great example

straightforward unordered list. This isn't the case for the mainNavigation div, however, as shown in the following code block:

```
<div id="mainNavigation">
<ul>
<li id="linkHome"><a href="#">Home</a></li>
<li id="linkNews"><a href="#">News</a></li>
<li id="linkDesign"><a href="#">Design</a></li>
<li id="linkStandards"><a href="#">Standards</a></li>
<li id="linkBrowsers"><a href="#">Browsers</a></li>
<li id="linkStatistics"><a href="#">Statistics</a></li>
<li id="linkCommunity"><a href="#">Community</a></li>
</ul>
</div>
```

As you can see, each list item has a unique id value that relates to the link nested within it. This, in combination with the class value defined for the body start tag, means the selected tab's background can be automated via a single CSS rule.

### Styling the page

Open 'dot-net-nav-bar.css'. This is the main CSS document that styles the web page. The first rule uses the universal selector ("\*") to remove margins and padding from all elements; the body rule defines a background colour ('#404040' – dark grey) and background image (a black-to-grey gradient, tiled horizontally), and it adds padding at the bottom of the page so that page content doesn't touch the bottom of the browser window. The third rule, '#skipNav', styles the div that has the skip navigation link within it.

```
#skipNav {
position: absolute;
top: 2px;
right: 5px;
width: 60px;
text-align: center;
}
```

In this case, it's positioned in an absolute manner at the top-right of the design. Styles later in the document ('#skipNav a' and '#skipNav a:hover') make the content invisible by default against the page's background image, only making it visible on the hover state. Depending on your site's requirements, you could change these rules to make the link permanently visible, or replace 'right: 5px' with something like 'left: -100px' to position the div offscreen, making it only really useful to screen readers.

In the CSS file's 'structure' section, the divs controlling the page's layout are styled. The '#wrapper' rule styles the site's container, defining a width, centring the

## You could replace 'right: 5px' with something like 'left: -100px' to position the div offscreen

layout, defining a background colour for behind the content, and setting a border. Note the 'border-top' override – since the site sits at the top of the browser window, the top border is redundant, hence removing it.

```
#wrapper {
width: 860px;
margin: 0 auto;
background-color: #ffffff;
border: 1px solid #555555;
border-top: 0px;
}
```

The '#masthead' rule adds a horizontally tiled background to the masthead div, along with a solid bottom border (which helps to visually separate the masthead from the main content area). The following two rules, '#logo' >>



**Without the hacks** If you don't add the rule in the 'ie-hacks.css' style sheet, IE fails to draw some elements. The fix? Define a set height for specific elements in an IE-only style sheet

>> and #searchArea, style the divs that contain the logo and the search box. The slightly awkward-looking height and width values for #logo (132px and 437px, respectively) are required to move the logo into place, so that the gradient behind it matches with the masthead's (see also #logo img' later on). Because both of these divs are floated left, the following div, #navContainer, which houses the navigation bar, is set to clear floated content.

```
#navContainer {
clear: left;
}
```

The final rule in the section is #content, which sets padding around the content of the content div.

### Styling links and navigation

The 'links and navigation' section of the style sheet begins with a rule to set the default link colour ('a'), and continues with rules to style the skip-navigation. The #skipNav a' rule defines a font, displays the link as a block element (thereby making the hit-area expand to the size of the link's container), and adds padding at the link's bottom edge. This enables a background image to be shown on the hover state, further emphasising the nature of what the link is for, should it be visible on the page (rather than being offscreen). The background image is a downward-facing arrow.

```
#skipNav a {
font: 1.0em Arial, sans-serif;
display: block;
color: #000000;
width: 60px;
padding-bottom: 10px;
}
```

```
#skipNav a:hover {
color: #898989;
background: url(assets/skip-nav.gif) 50% 100% no-repeat;
}
```

The main navigation is styled next. The containing div is styled via the #mainNavigation' rule, which defines a background colour (dark grey, which contrasts nicely with the bright navigation text colours), adds a top border (to help define the navigation area), and sets vertical padding, so that the navigation isn't squished to the height of the text.

```
#mainNavigation {
background: #111111;
```



Expanding elements Because of how the text for this design is sized, it can be enlarged in Internet Explorer. The layout also caters for this, despite using fixed-width elements

## The horizontal padding provides spacing between each item, enabling users to tell them apart

```
border-top: 1px solid #555555;
padding: 6px 0;
}
```

The #mainNavigation ul' rule centres the unordered list's content within the mainNavigation div, while #mainNavigation li' displays list items inline, stacking them horizontally.

```
#mainNavigation ul {
text-align: center;
}
```

```
#mainNavigation li {
display: inline;
}
```

The following two rules, #mainNavigation a' and #mainNavigation a:hover', style the links within the main navigation bar.

```
#mainNavigation a {
font: bold 1.2em Arial, sans-serif;
color: #ffffff;
text-decoration: none;
padding: 6px 10px;
}
```

```
#mainNavigation a:hover {
color: #5ac88b;
}
```

Note the padding values in the #mainNavigation a' rule, which provide a larger clickable area around each link's text. For the active link – the one representing the section of the current page – this also provides an area for the background image. The rule required to style this link is the following one, which has a large grouped selector. What this is essentially doing is looking for similar section/link combinations – '.sectionHome #mainNavigation #linkHome a' refers to the link

### Resources Find out more online



The excellent 'On having layout' article discusses the nature of Internet Explorer's 'hasLayout' concept, which determines how web elements are drawn and how they react to each other. This concept is often to blame for portions of standards-compliant layouts suddenly becoming invisible in IE. [www.satzansatz.de/cssd/onhavinglayout.html](http://www.satzansatz.de/cssd/onhavinglayout.html)



There are various websites that outline the options for creating conditional comments, some of which make more sense than others! A succinct and useful one to bookmark is on Marko Dugonjic's site, and the article also links to Microsoft's thorough overview page. [www.maratz.com/blog/archives/2005/06/16/essentials-of-css-hacking-for-internet-explorer](http://www.maratz.com/blog/archives/2005/06/16/essentials-of-css-hacking-for-internet-explorer)

## Expert tip Visual hierarchies

Unless you have a small site with only a few pages, there'll be some kind of hierarchical structure – that is, sections with subsections. When these are available via a navigation bar, the tiers need to be made distinct or the user will get confused. The most common method for this is to vary the type size (making it larger for the main category list and smaller for the subsections), along with using colour, and perhaps tabs, to show the relationship between a link and the subsection links that belong to it.

with an id of 'linkHome' within the mainNavigation div within any element with a class of 'sectionHome'. Following that? Phew. In our example web page, the 'section' class for the body start tag is 'sectionBrowsers'.

In the selector below, you'll see '.sectionBrowsers #mainNavigation #linkBrowsers a', hence the link with the id value 'linkBrowsers' being highlighted in the completed page. This logic follows through to all of the other top-tier links (on a page with a body class value of 'sectionCommunity', the link within the list item with an id value of 'linkCommunity' will be highlighted, and so on).

```
.sectionHome #mainNavigation #linkHome a, .sectionNews #mainNavigation
#linkNews a, .sectionDesign #mainNavigation #linkDesign a, .sectionBrowsers
#mainNavigation #linkBrowsers a, .sectionStatistics #mainNavigation #linkStatistics
a, .sectionCommunity #mainNavigation #linkCommunity a {
background: #28b767 url(assets/active-tab-background.jpg) 0 100% repeat-x;
border-top: 1px solid #5cc98d;
}
```

Similarly, the following rule provides an override 'color' value for the hover state of these links, because the standard hover-state colour for the top-tier navigation is the same as the tab background colour.

```
.sectionHome #mainNavigation #linkHome a:hover, .sectionNews #mainNavigation
#linkNews a:hover, .sectionDesign #mainNavigation #linkDesign a:hover,
.sectionBrowsers #mainNavigation #linkBrowsers a:hover, .sectionStatistics
#mainNavigation #linkStatistics a:hover, .sectionCommunity #mainNavigation
#linkCommunity a:hover {
color: #0e5424;
}
```

The sub-navigation rules are somewhat simpler. Of the first three rules, '#subNavigation ul' and '#subNavigation li' centre the list's content and display the list items inline (as per the main links bar), and '#subNavigation' defines a horizontally repeating background image for the second tier of navigation, along with a bottom border (providing a solid-looking base) and some padding. There's also a negative margin value, used to pull the second tier up by one pixel. Not doing this sometimes results in a gap between the two navigation tiers when the layout is zoomed.

```
#subNavigation {
margin-top: -1px;
background: #27b767 url(assets/sub-navigation-background.jpg) 0 100% repeat-x;
border-bottom: 1px solid #6b6b6b;
padding: 6px 0;
}
```

The final two rules in the section, '#subNavigation a' and '#subNavigation a:hover', style the links within the second tier of navigation. Again, the first rule's vertical padding matches that of the containing div, and the horizontal padding provides spacing between each item, enabling users to easily tell them apart.

```
#subNavigation a {
font: bold 1.1em Arial, sans-serif;
color: #ffffff;
text-decoration: none;
padding: 6px 10px;
}
```

```
#subNavigation a:hover {
```



**Don't lose CSS!** Remove CSS and you see how vital the skip-navigation/skip-to-content link is. Without it, those links would be read out every time you accessed a new page

```
color: #0e5424;
}
```

## Styling other elements

The 'fonts' section of the style sheet, unsurprisingly, deals with font styles. The html and body rules set the default font size to 62.5 per cent, enabling text to be sized using em values that are a tenth of the target pixel size (so a value of 1em equates to 10px). This provides the accuracy of pixel sizing combined with the flexibility of using percentages, enabling text zooming in IE. The other rules in this section set font families and sizes for text elements, along with margins.

The 'images' section has two rules: 'a img' removes borders from linked images, and '#logo img' sets margins to the top and left of the image within the logo div, ensuring that the logo doesn't sit at the top-left of the design.

The 'forms stuff' section deals with styling the various elements within the searchArea div. There's nothing particularly complex here – the lead paragraph is set to display as a block element, while the others are floated left, stacking them horizontally. Margins, borders, background colours and padding have been applied to each element, making them look as good as possible across browsers. Each browser has its own idea about how to display certain form styles though, and some, like Safari, ignore properties such as borders and colours completely.

As usual, there are problems with Internet Explorer. Randomly, some versions of the browser fail to display aspects of the masthead, such as the background. This is down to 'hasLayout' (see Resources on page 72), and it can be fixed by defining a set height for the masthead div in the 'ie-hacks.css' style sheet.

```
#masthead {
height: 1%;
}
```

Internet Explorer automatically stretches a div's height to fit its content, so adding this rule doesn't cut off the navigation bars – it just fixes the 'hasLayout' issue.

The other problems are with version 5 of IE, and are dealt with in 'ie5-hacks.css'. The body and #wrapper rules centre the site, because IE5 doesn't correctly deal with the 'auto' margin values in the original #wrapper rule. Finally, '#searchArea form' removes some padding from the searchArea div, which otherwise displays incorrectly in IE5 due to the browser's box-model bugs. ●



## About the author

Name Craig Grannell  
Site [www.snuobcommunications.com](http://www.snuobcommunications.com)  
Areas of expertise Information architecture, site concepts, graphics, interface and front-end design  
Clients Swim~, Rebellion, IDG  
Favourite gig Sigur Rós, Hammersmith Apollo, 2006