

/CSS/enhance internal page links



Craig Grannell shows you how to use a couple of nifty CSS3 tricks to improve the usability of internal page links on your websites

Knowledge needed Intermediate CSS and HTML

Requires A text editor

Project time 20 minutes

Back in issue 178, we spent a happy few pages looking at the very real benefits CSS3 is likely to bring in the not-too-distant future. Several of them were primarily visual – adding drop shadows, rounded corners and so on – but this issue’s tutorial checks out a pseudo-class that can provide some handy usability aids to your web pages, **:target**.

On target

If you’ve dabbled in CSS, you’ll almost certainly be used to working with pseudo-classes already. They provide styles relating to a selector’s state and, sometimes, to conceptual document components. The most obvious examples are links – **:link**, **:visited**, **:active** and **:hover** are all pseudo-classes that are used extremely frequently. The **:target** pseudo-class is also related to links, but instead of styling links, it’s used to style link targets.

You might wonder how such a thing can be useful, but with a little lateral thinking you should be able to find various uses for this nifty piece of CSS. In this tutorial, we’ll cover two examples that enhance internal page links in very different ways.

First, we’ll look at a fairly typical web page FAQ – the kind that has a bunch of questions with answers further down the page – and enable you, via links, to rapidly navigate back and forth between queries and solutions. While this kind of design can be efficient from a coding and maintenance standpoint, it typically leads to usability problems – especially for inexperienced users.

Using anchors to link to elements with specific **id** values is common in such circumstances, but the web page ‘snaps’ to its target when a link is clicked

Expert tip Background images

Some sites now make use of icons to denote links that lead off-site (usually a square with an arrow pointing to the top-right). This is a handy usability aid, preparing users for the fact that clicking a marked link will open a new website. Similarly, you can use backgrounds for internal page links, making it obvious that a link doesn’t target a different page on the same website. We suggest that a rectangle with a upwards or downwards-facing arrow (depending on the direction travelled when the link is clicked) is a good convention to adopt.

and it can be difficult to know where to focus. For newcomers, the jolt on such pages can be particularly fierce, and the problem is only compounded if there’s not much vertical space between the bottom target and the end of the page. If there’s plenty of space, the target appears at the top of the browser view area; if not, it could appear anywhere within the browser window.

Focus point

By way of example, open **internal-links-before.html** from the **tut_css** folder on the CD. This is a straightforward, unordered list with internal page links for accessing content elsewhere on the page. The design is a question and answer style FAQ page, with the basic structure for the list and two of the answers shown in the following code block:

```
<ul id="answerList">
<li><a href="#answerOne">Go to answer one</a></li>
<li><a href="#answerTwo">Go to answer two</a></li>
<li><a href="#answerThree">Go to answer three</a></li>
<li><a href="#answerFour">Go to answer four</a></li>
</ul>
<h2 id="answerOne">Answer one</h2>
<p>Lorem ipsum [...]</p>
<p><a href="#questionsList" class="goToQuestions">Go to questions</a></p>
<hr />
<h2 id="answerTwo">Answer two</h2>
<p>Fusce convallis [...]</p>
<p><a href="#questionsList" class="goToQuestions">Go to questions</a></p>
<hr />
```

Click on any of the first three links in this list and the focus point is relatively obvious: the corresponding heading appears at the top of the browser view area. However, click on **Go to answer four** and, unless your browser window is very short, the corresponding heading appears part-way down the view area.

This can be irritating for users, who now have to ‘find’ the content they’ve just clicked to access. Flicking back and forth between questions and answers also means some users may find themselves quickly confused.

By adding some simple CSS, the usability of the page can be improved massively. Add the following rule (inside the **head** section’s **style** element is fine) and then re-test the page:

A page with internal links

- [Go to answer one](#)
- [Go to answer two](#)
- [Go to answer three](#)
- [Go to answer four](#)

Answer one

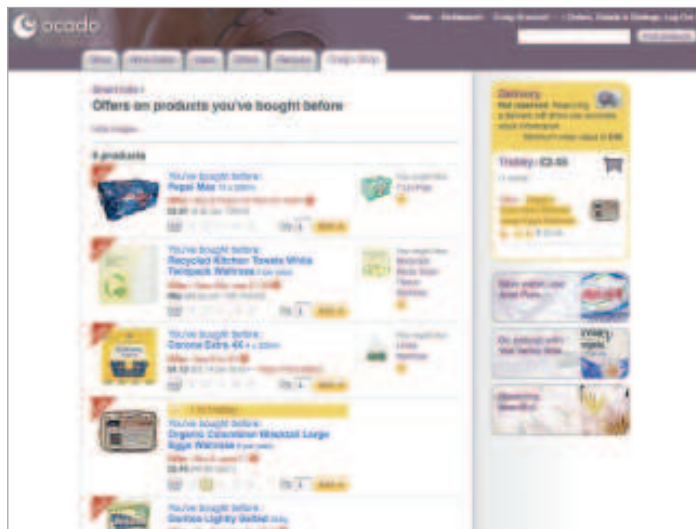
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean blandit varius risus. Maecenas pretium urna quis quam. Nulla tristique magna eu dolor. Sed eleifend commodo enim. Ut malesuada, nisi eget varius tincidunt, odio purus lacinia quam, nec vehicula lectus nibh id ipsum.

Etiam vulputate. Nullam faucibus nisi id nisi. Sed elit. Aenean porta. Nam et leo. In suscipit ullamcorper nisi. Aliquam vel pede. Suspendisse potenti. Aenean ac augue at erat rhoncus aliquam. Aliquam tempus dui at magna.

Sed ac magna. Aliquam auctor elit nec ipsum. Quisque justo arcu, semper in, dignissim et, rutrum nec, risus. Vestibulum fringilla. Vivamus semper nibh blandit est. Mauris dui pede, porta quis, sagittis id, accumsan at, diam.

Aliquam aliquet ornare nisi. Ut vestibulum. Aliquam vitae nisi at nunc dapibus porta. Proin euismod

Confused? When clicking on an internal link, it’s easy to get lost. Clicking on the **Go to answer four** link results in the targeted content appearing part way down the view area



Visual direction Items in your shopping basket are highlighted with a yellow background, providing visual confirmation of your actions. You can re-create this in CSS3

```

:target {
background: yellow;
}

```

If you're using a relatively modern browser (Firefox, Safari, Opera 9.5), you'll see that the target element now has a yellow background once you've clicked a link. In other words, click on **Go to answer four** and the element with the relevant **id** – which in this case is a level-two heading with the content **Answer four** – has a yellow background. This draws the eye and means there are no issues regarding where to focus. Even with the elements that appear at the top of the view area, this background helps users deal with the 'jolt' of a web page snapping from place to place.

If you're currently thinking, 'But I'm not using one of the browsers you mentioned and nor will a fair chunk of my users,' check out the legacy support box (right) for dealing with browsers that don't yet support **:target**.

Note that in this example a solid yellow background is used, because it's a common highlight colour that also happens to contrast with the design's white background. If you employ a technique such as this, you should always ensure you use a colour that provides enough contrast with your background, and that the colour fits in with your site's design as a whole. Alternatively, try experimenting with different property values. For example, instead of a background colour, use an image (such as an arrow) or a border. Some sites using this technique have an animated background, which 'flashes' a colour before fading away to nothing. Whatever you decide, ensure your highlight is obvious and is something that stays on screen long enough for users to clearly see what they're supposed to be focusing on.

Tab happy

Our second example looks at another way in which internal links are used, although it may not be apparent that they're used at all. Often news sites display a list of links to news articles or other features, ordered by some means, but provide an alternate listing within the same space when a link is clicked. For example, you might see a list of **most recent** news stories, which is replaced with **most popular** stories when a link is clicked. Typically some form of scripting is used to make this switch, but by using the **:target** pseudo-class, you can create the same effect using CSS alone.

If you open **tabs.html** from this issue's CD, you'll see a bunch of headlines in a fetching green tab. Click the pink tab and its contents come to the front.

Scripting is typically used to switch tabs, but **:target** can re-create this in CSS alone

Legacy support

Dealing with unruly browsers

We've already mentioned that cutting-edge CSS technology often falls foul of poor browser support. The **:target** pseudo-class (which provides the bulk of the functionality and usability benefits explored in this issue's tutorial) is a case in point. It fails in any version of Opera prior to 9.5 and in all shipping versions of Internet Explorer.

As is often the case, you're left to decide whether to omit support for non-compliant browsers or create a fall-back using JavaScript. Although you could use a combination of **onclick** and **document.getElementById** to deal with targets on a case-by-case basis, a better method is to sort everything globally.

That's exactly what Peter Ryan's done with his handy script, which is available from blog.peter-ryan.co.uk/2008/03/14/using-css3-target-selectors and is free to use. Attach this to your page and duplicate your **:target** rules with **.target** equivalents and you're good to go – see the files on the CD for commented examples.



Time warp Peter Ryan's handy script enables you to support older browsers that won't work with the **:target** pseudo-class on a global scale

This is for CSS only – there's no JavaScript backup to fall back on. Code-wise, the structure is pretty simple. An abbreviated mark-up is shown in the following code block:

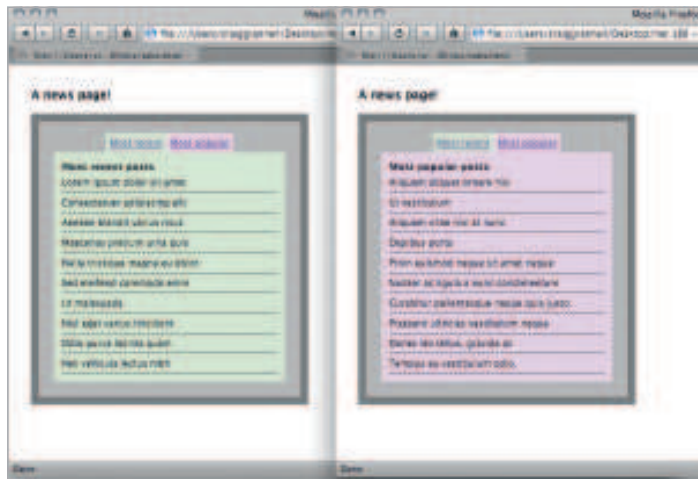
```

<div id="postSwitcherContainer">
  <p class="tabsController"><a href="#mostRecentPosts" class="mostRecentLink">Most recent</a><a href="#mostPopularPosts" class="mostPopularLink">Most popular</a></p>
  <div id="mostRecentPosts">
    <h2>Most recent posts</h2>
    <ul>
      [list items]
    </ul>
  </div>
  <div id="mostPopularPosts">
    <h2>Most popular posts</h2>
    <ul>
      [list items]
    </ul>
  </div>
</div>

```

As you can see, the entire system's wrapped in a div with an **id** value of **postSwitcherContainer**. The first item inside is a paragraph, with a **class** value of **tabsController**. Each of the links within targets the **id** value of one of the subsequent tab-block divs, and each of those has its own unique **id**. Contained inside each block is content comprising a level-two





Browse Tabs are often used on news sites for providing headlines, listed by various criteria, within a limited space. These tabs can be created using `:target` in CSS alone

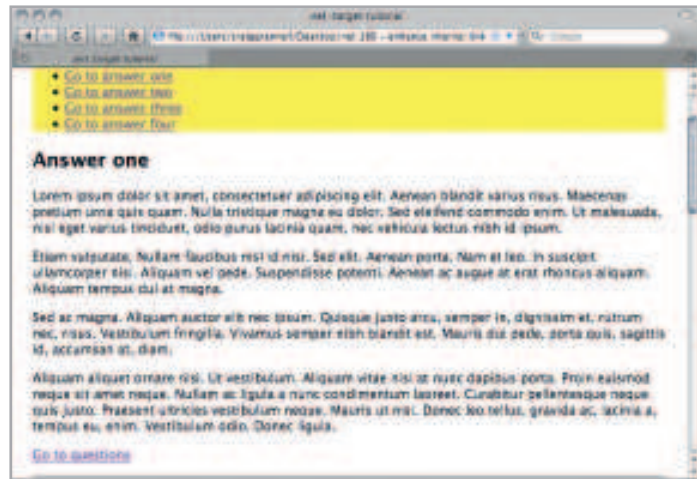
>> heading and an unordered list. Although the level-two headings aren't necessary visually, they're handy from a semantic standpoint, ensuring each list has a heading associated with it. If the headings annoy you, they're easily hidden away.

Most of the CSS for the page, which is in a `style` element within the document's `head` section, deals with visual styles. So if you've got a rudimentary understanding of CSS, you should be fine trawling through the various font, width, border and list settings. (As an aside, we know that whopping a load of CSS in the `head` section of a web page is naughty, but this is a quick example. On an actual website, you should always put your CSS in an external document.)

The most important rules are the ones that deal with the positioning and stacking of the tab bodies. The tabs themselves are dealt with via the `.tabsController` rule, which centres the content of the relevant paragraph and adds some bottom padding, and `.tabsController a`, which adds some padding around the links that make up the tabs. The rule `div#postSwitcherContainer div` defines styles for the tab bodies by floating divs within the `postSwitcherContainer` div left, setting a width and padding for them.

The next four rules define background colours for the tab bodies and links. The `#mostPopularPosts` rule then defines a negative left margin equal to the amount of horizontal space the **most popular** tab's body takes up – see the following code block.

This positions the **most popular** tab body in front of the **most recent** tab body. By then adding the `position: relative` and `z-index: -1 !important` property/value pairs, the **most popular** tab body is moved behind the **most recent** one:



Stay focused By making use of the CSS `:target` pseudo-class the targeted element is highlighted, making it more obvious where the user's focus should lie

```
.mostRecentLink {
background: #c9edcb;
}
#mostRecentPosts {
background: #c9edcb;
}
.mostPopularLink {
background: #edcaeb;
}
#mostPopularPosts {
position: relative;
margin-left: -300px;
background: #edcaeb;
z-index: -1 !important;
}
```

The final rule uses the `:target` property again, but instead of changing a background as in our first example, the `z-index` value is amended when a link is clicked. By defining the `z-index` value as `1` for the targeted element – a tab body, in this example – it appears in front of the other tab's content. Because we've used fairly basic CSS positioning, the system's layout works well in all fairly recent browsers. Other methods for creating this kind of system exist, but they rely on absolute positioning, which takes the tab bodies outside of the document flow and makes the container's height the same as the tabs. This screws up the subsequent positioning of elements unless you define an exact height for the container, which may be an unknown:

```
#postSwitcherContainer :target {
z-index: 1 !important;
}
```

The final link

CSS3 isn't some exciting technology in the distant future, it's something you can start using today to benefit your sites. Workarounds are sometimes required (the legacy support box is your friend), but these will become less necessary within a relatively short space of time as browsers are updated to support new standards. Like many other CSS-oriented hacks, solutions are often an 'apply once, then forget about it' deal anyway. For the examples shown here, that's most definitely the case. ●

Resources Where to find out more



World Wide Web Consortium
Get the skinny on what's what in the world of web standards, and what's coming soon to a browser near you, at the World Wide Web Consortium website. Invaluable for forward-looking designers.
www.w3.org



CSS3.info
This excellent resource provides a CSS3 selector compatibility table (go to www.css3.info/modules/selecter-compat) as well as a handy test suite to check the compatibility of current browsers (www.css3.info/selectors-test).



About the author

Name | Craig Grannell
Site | www.snubcommunications.com
Areas of expertise | Information architecture, site concepts, graphics, interface and front-end design
Clients | Swim~, Rebellion, IDG
Favourite crisp flavour | Bacon & Brown Sauce Seabrook