RecipeCssForms

Work in progress...

Forms - an OS feature

The forms elements of a web page (input, textarea, button, radio, checkbox, etc) are considered part of the operating system, not a part of the web page. Generally, it is frowned upon to style such elements - mainly for user identification and accessibility reasons.

However, most theme designers know that it is possilbe to lightly style the forms elements, including turning on/off 3D bevel/ridge border effects, changing the background colour, adding a background or list image, changing the text and border colours and with browsers using the Gecko engine (used by Mozilla, Navigator, Firefox and others), rounding the corners. The Tiki themes are good examples of what is possible using standard CSS. The damian.css theme shows off the rounded corners feature of Gecko.

What is less widely known is that by using the internal features of IE and Gecko, more radical changes are possible.

Whether one uses the light styling of standard CSS or the heavier styling that is the subject of this recipe, designers should think about the accessibility implications of what they are doing. Arguably, the tikineat theme shouldn't change the borders, colors and fonts of the forms elements at all, seeing as it is the default theme for 1.9.

More importantly, the button2 div class (used for the edit, lock, history, etc buttons on this page) should be <input or <button, especially for 508.css

Heavy styling in Gecko

For the moment, this recipe will only deal with browsers that use Mozilla's (Gecko's) CSS engine at its heart.

Gecko generates its widgets as a composite of simpler HTML (actually XUL) elements.

{

http://docs.mandragor.org/files/Misc/Mozilla_applications_en/mozilla-chp-4-sect-5.html#mozilla-CHP-4-EX-13

Here is the first problem: you need to know the element names for each part of the composite you want to change. Unfortionately, documantation for both the UI components used to build the applications (Firefox, Thunderbird, etc) and the toolbox widgets, are sparse. Google may help, but one is pretty much on their own.

So let's take a look at the checkbox widget for Mozilla 1.6 classic theme:

./skin/classic/global/checkbox.css

```
checkbox { -moz-appearance: checkbox-container; -moz-box-align: center; margin: 2px 4px; padding: 0px 2px; border: 1px solid transparent; } ... /* ::::: checkmark image ::::: */ .checkbox-check { -moz-appearance: checkbox; -moz-box-align: center; border: 2px solid; margin: 5px 0px 4px 0px; -moz-border-top-colors: ThreeDShadow ThreeDDarkShadow; -moz-border-right-colors: ThreeDHighlight ThreeDLightShadow; -moz-border-bottom-colors: ThreeDHighlight ThreeDLightShadow; -moz-border-left-colors: ThreeDShadow ThreeDDarkShadow; min-width: 13px; min-height: 13px; background: -moz-Field no-repeat 50% 50%; } ... /* ..... checked state ..... */ checkbox[checked="true"] > .checkbox-check { background-image: url("chrome://global/skin/checkbox/cbox-check.gif"); }
```

That is only a part of the file - take a look at your own copy for the details. Now, that is the styling for the

XUL checkbox - changing it doesn't affect HTML checkboxes at all (at least in Mozilla 1.6 and Firefox 1.0). However, it does provide an insight into how the checkboxes are built. Furthermore, some of the pitfalls we are about to experience with HTML also applies to anyone trying to override the XUL style.

(...to be continued...)