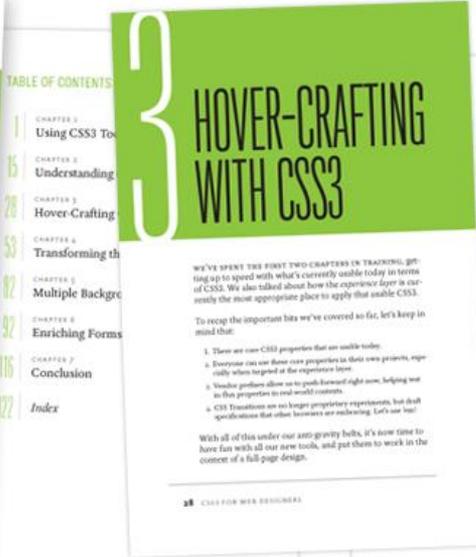
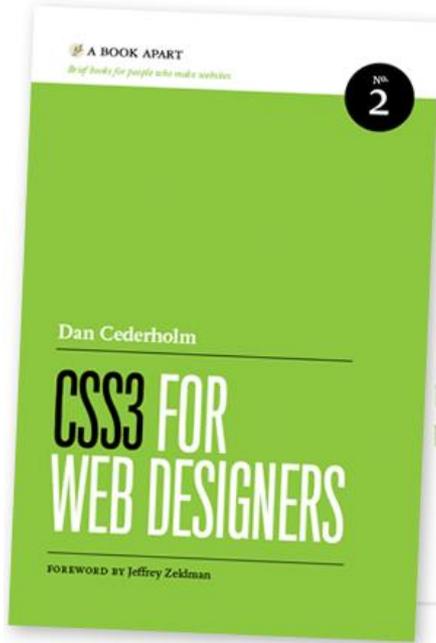


More CSS goodness with CSS3

Webpage Design



CSS is Evolving

- Currently we have been working with CSS 2.1
- This specification in its entirety is supported by all current browsers (there is some inconsistency in older browsers).
- CSS3 does not rewrite the rules of web design, it is a (relatively) gentle evolution and improvement of the methods we already use.
- CSS3 just adds some extra tools to our toolkit.

CSS3 is modular

- The CSS 2.1 specification was finally recommended on 7th June 2011...
- ...13 years after CSS2 was recommended back in May 1998.
- W3C do not want to make the same mistake with CSS3, so the specification has been split into “modules”.
- Each module is being developed independently

[It's official: W3C finalises CSS 2.1](#)

[Cascading Style Sheets Level 2 Revision 1 \(CSS 2.1\) Specification](#)

CSS3 is a work in progress

- Unlike CSS 2.1, we cannot assume that all CSS3 properties will work consistently in all current browsers.
- CSS3 is still in development and its use should be considered “experimental”.
- However, some (small) parts of the CSS3 specification have been adopted by the main browser vendors and can now be used for “modern” browsers.

What are the “modern” browsers?

 Internet Explorer 9 (but not earlier versions)

 Firefox 3.5+

 Opera 10+

 Chrome

 Safari 3+

- As there are still a large number of web users viewing pages with versions of Internet Explorer earlier than version 9, CSS3 cannot be used for “mission critical” design elements.

[Is IE9 a modern browser?](#)

Where can CSS3 be used?

- In his book, *CSS3 for Web Designers*, Dan Cederholm identifies non-critical design elements/functions where CSS3 can be used.

CRITICAL	NON-CRITICAL
Branding	Interaction
Usability	Visual Rewards
Accessibility	Feedback
Layout	Movement

A website's visual experience can be grouped into critical and non-critical categories. The latter are where CSS3 can be applied today.

What CSS3 properties can be used?

PROPERTY	SUPPORTED IN
border-radius	 3+  3+  1+  10.5+  9 beta
text-shadow	 1.1+  2+  3.1+  9.5+
box-shadow	 3+  3+  3.5+  10.5+  9 beta
Multiple background images	 1.3+  2+  3.6+  10.5+  9 beta
opacity	 1.2+  1+  1.5+  9+  9 beta
RGBA	 3.2+  3+  3+  10+  9 beta

TABLE 1.02: CSS3 properties and the browsers that support them.

What do those properties do?

RGBA Colors



Background color overlay on child element is using RGBA, semi-transparent.

Border Radius



This box has rounded corners using CSS, with no images or extra markup

Text Shadow



Example Text

Box Shadow



This box has a drop shadow

Opacity



This text is white, but it's set to 40% opacity

Multiple BG



More information:

[RGBA Colours](#)

[Border Radius](#)

[Text Shadow](#)

[Box Shadow](#)

[Opacity](#)

[Multiple Backgrounds](#)

How are they implemented?

- Most CSS3 properties can be considered to be in “beta” and browser vendors have their own implementation...
- ...so we need to cover all these different implementations for each property.
- Fortunately, each browser vendor has an unique “vendor prefix” we can use to feed a specific property to a specific browser.

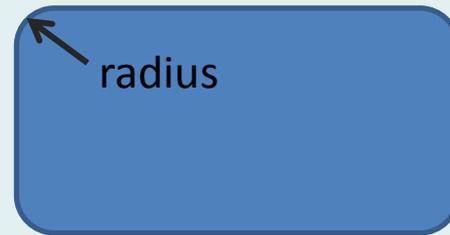
What are the vendor prefixes?

 WebKit	<code>-webkit-</code>
 Mozilla	<code>-moz-</code>
 Opera	<code>-o-</code>
 Konqueror	<code>-khtml-</code>
 Microsoft	<code>-ms-</code>
 Chrome	<code>-chrome-</code>

TABLE 1.03: The most widely-used vendors and their associated prefixes.

So, what does a CSS3 rule look like?

```
.foo {  
-webkit-border-radius: 10px;  
-moz-border-radius: 10px;  
border-radius: 10px;  
}
```

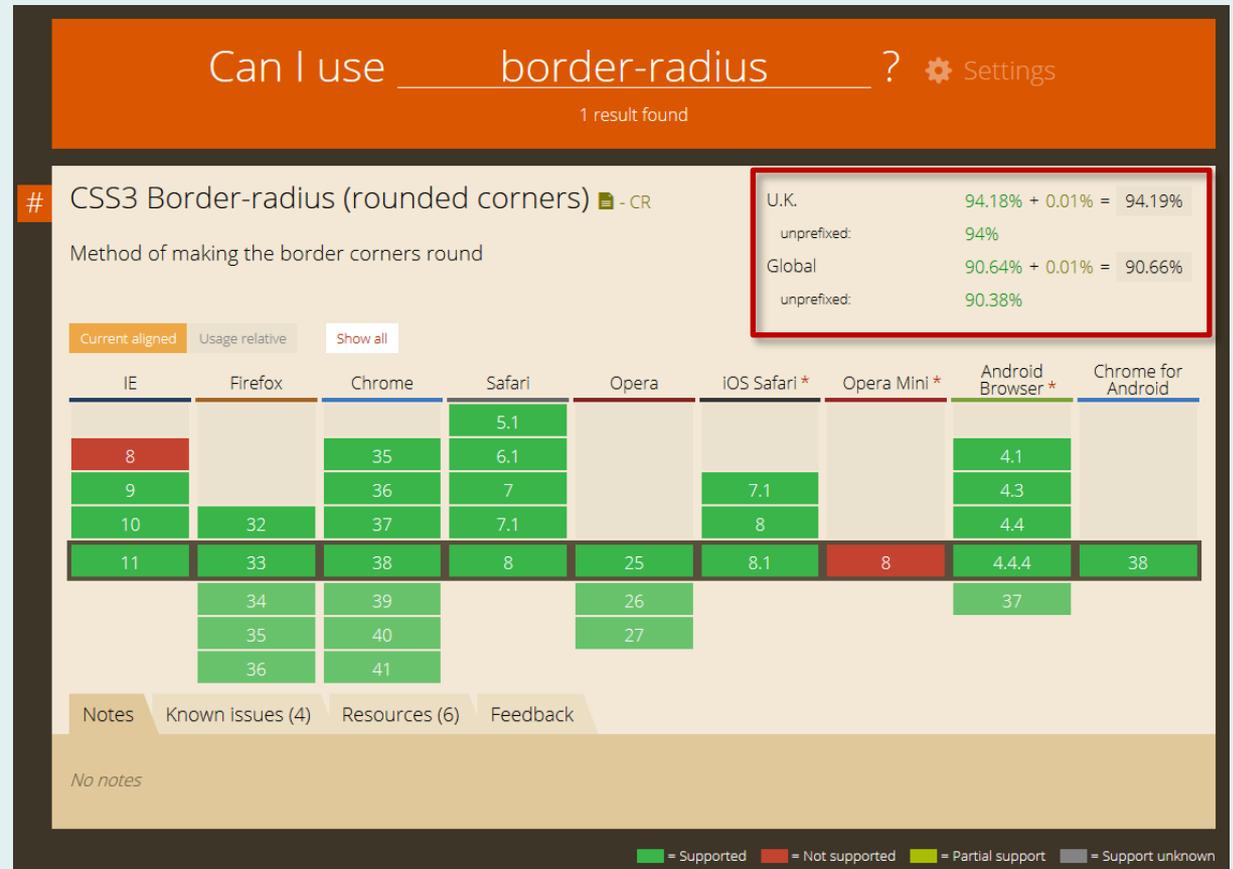


In the above rule, the vendor prefixes for Safari and Firefox are used, followed by the standard W3C version of the property. In the case of border-radius we don't need to specify vendor prefixes for Opera or Internet Explorer because they both recognise the standard property name. Chrome will recognise the -webkit- prefix.

The standard property name should always be last in the stack so that when browsers finally recognise it, they will use it in preference to any previous vendor-specific version.

Can I use unprefixed?

Currently, just over 94% of browsers in the UK can understand the unprefixed version of *border-radius*. If you consider *border-radius* to be a progressive enhancement or non-critical, you can now drop the prefixed versions.



Can we mix CSS 2.1 and CSS3?

```
#nav li a {  
padding: 5px 15px;  
font-weight: bold;  
color: rgba(255, 255, 255, 0.7);  
text-shadow: 0 1px 1px rgba(0, 0, 0, 0.5);  
-webkit-border-radius: 14px;  
-moz-border-radius: 14px;  
border-radius: 14px;  
}
```

Yes, CSS3 properties work just fine alongside CSS 2.1 properties. Just remember to add those vendor prefixes when they are required.

Do all properties need prefixes?

- Of the 6 properties we've covered, 3 have never needed vendor prefixes. They are:
 - background:** (for multiple images)
 - opacity:**
 - text-shadow:**
- In addition, most current browser versions will also recognise:
 - box-shadow:**
 - border-radius:**
- However, they are still required for backwards compatibility.

[How to deal with vendor prefixes](#)

Another bad idea

- There have been many bad ideas in web design over the years; frames is a good example. Things that seemed like a good idea at the time but ended up causing more problems than they set out to cure.
- Vendor prefixes are now considered a mistake, but one we have to live with.
- Most browser developers are now avoiding the use of prefixes and supporting the standard property names from the start.
- At some point in the future, we'll be able to stop using them but for now we have to deal with them.

Keeping on top of things

- The most difficult aspect of working with vendor prefixes is knowing when to use them.
- Some CSS3 properties need them and others don't.
- As browsers evolve, some prefixes get dropped and are no longer needed except for backwards compatibility.
- How do you keep on top of all this?

Help is at hand



Only
2KB
gzipped

Click to download →



-prefix-free

Break free from CSS prefix hell!

-prefix-free lets you use only unprefixed CSS properties everywhere. It works behind the scenes, adding the current browser's prefix to any CSS code, only when it's needed.

The most important reference is the [Can I Use](#) database. This resource is constantly updated with the latest vendor prefix information and gives you the statistics that help you decide whether you need a prefix or not.

There are also plenty of new tools that automate the process of adding prefixes such as [Pleeese](#) and [Autoprefixer](#). There is even a JavaScript application you can use to automatically add them on the client side: [-prefix-free](#) by Lea Verou

CSS Transitions

- CSS3 also introduces transitions
- Transitions allow the designer to specify how one property value (say, colour) will transition to another.
- For example, a colour change on a:hover can be transitioned over a period rather than just changed...

Transition background colour

```
a.foo {  
padding: 5px 10px;  
background: #9c3;  
-webkit-transition: background 0.3s ease;  
-moz-transition: background 0.3s ease;  
-o-transition: background 0.3s ease;  
transition: background 0.3s ease;  
}  
a.foo:hover {  
background: #690;  
}
```

The rules above cause a transition of the background colour on an anchor element when the mouse passes over it. The transition takes 0.3 seconds and uses the ease transition type. Only the -webkit- prefix is still required.

[CSS Fundamentals: CSS 3 Transitions](#)

Can I transition anything?

- Lots of CSS properties can be transitioned, including padding, margin and border.
- Any element can be transitioned if it has a :hover state.
- Did I mention that elements other than anchor can use :hover?
- This means you could add a transition effect to a paragraph (see example).



Can I use CSS3 now?

- Yes, go ahead.
- Add them to your designs as and when you need them.
- **Warning!** Use CSS transitions with caution, they can seriously damage your designs...

COOL WEBSITE

Progressive Enhancement

- For a long time, web designers have used the concept of “graceful degradation” in order to design rich, modern websites that don’t break in older browsers – rather, they degrade gracefully.
- Since CSS3, this idea has evolved into a new concept called “progressive enhancement”.
- Rather than caring for older browsers, this approach rewards modern browsers with a more beautiful experience.

What's that again?

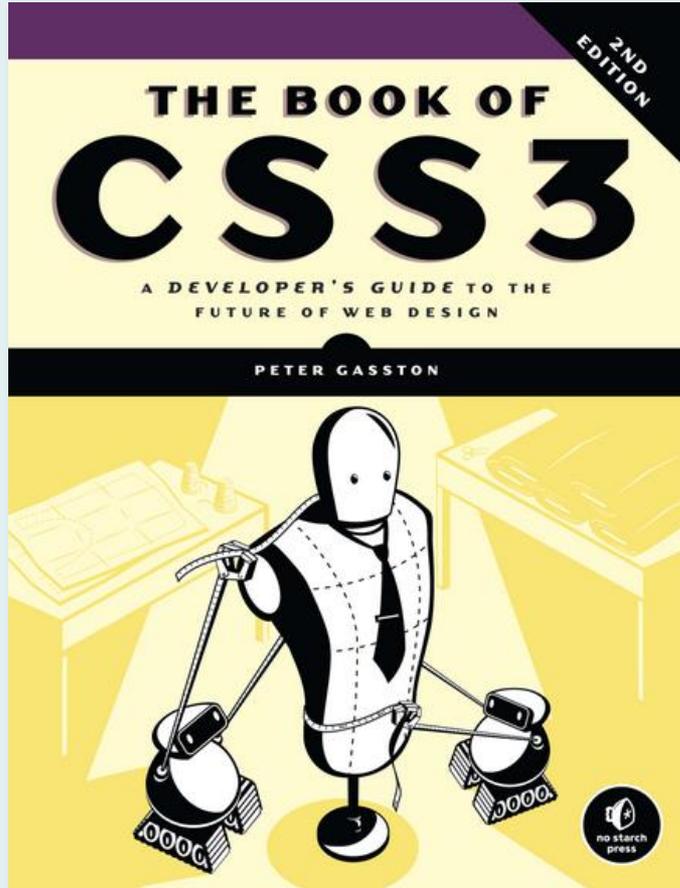
So, **graceful degradation** is the practice of building your web functionality so that it provides a certain level of user experience in more modern browsers, but it will also *degrade gracefully* to a lower level of user experience in older browsers. This lower level is not as nice to use for your site visitors, but it does still provide them with the basic functionality that they came to your site to use; things do not break for them.

Progressive enhancement is similar, but it does things the other way round. You start by establishing a basic level of user experience that all browsers will be able to provide when rendering your web site, but you also build in more advanced functionality that will automatically be available to browsers that can use it.

[Christian Heilmann](#)

[Graceful degradation versus progressive enhancement](#)

You'd like to know more about CSS3?



Peter Gasston's *The Book of CSS3* (2nd Ed.) is currently the most up-to-date reference book on the use of CSS3. Obviously, the CSS3 modules are constantly evolving but this book, published in November 2014 sets out the current state of play.

End