Home News Docs Sites

Documentation

This page contains the current documentation for Modernizr.

What is Modernizr?

Modernizr is a small JavaScript library that detects the availability of native implementations for next-generation web technologies. These technologies are new features that stem from the ongoing HTML 5 and CSS 3 specifications. Many of these features are already implemented in at least one major browser (most of them in two or more), and what Modernizr does is, very simply, tell you whether the current browser has this feature natively implemented or not.

With this knowledge that Modernizr gives you, you can take advantage of these new features in the browsers that can render or utilize them, and still have easy and reliable means of creating fallbacks—should you choose to—for the browsers that cannot.

How Modernizr works

Unlike with the traditional—but highly unreliable—method of doing "UA sniffing", which is detecting a browser by its (user-configurable) `navigator.userAgent` property, Modernizr does actual feature detection to reliably discern what the various browsers can and cannot do. After all, the same rendering engine may not necessarily support the same things*, and some users change their UserAgent string to get around poorly developed websites that don't let them through otherwise.

Modernizr aims to bring an end to the UA sniffing practice. Using feature detection is a much more reliable mechanic to establish what you can and cannot do in the current browser, and Modernizr makes it convenient for you in a variety of ways:

1. It tests for over 20 next-generation features, all in a matter of milliseconds;
2. It creates a JavaScript object (named `Modernizr`) that contains the results of these tests as boolean properties;
3. It adds classes to the `html` element that explain precisely what features are and are **not** natively supported

For a lot of the tests, Modernizr does its "magic" by creating an element, setting a specific style instruction on that element and then immediately retrieving it. Browsers that understand the instruction will return something sensible; browsers that don't understand it will return nothing or "undefined".

Many tests in the documentation come with a small code sample to illustrate how you could use that specific test in your web development workflow. Actual use cases come in many more varieties, though. The possible uses of Modernizr are limited only by your imagination.

HTML 5 elements in IE

Modernizr runs through a little loop in JavaScript to enable the various elements from HTML5 (as well as `abbr`) for styling in Internet Explorer. Note that this does not mean it suddenly makes IE support the Audio or Video element, it just means that you can use `section` instead of `div` and style them in CSS. As of Modernizr 1.5, this script is identical to what is used in the popular html5shim/html5shiv library. Both also enable printability of HTML5 elements in IE6-8, though you might want to try out the performance hit if you have over 100kb of css.

Table of Contents:

RSS

Modernizr on Twitter

SEE MORE TWEETS »

Modernizr & HTML5

HTML

Modernizr is the right micro-library to get you off and running with HTML5 & CSS3 today.

Features detected by Modernizr

@font-face

ˆ Table of contents

CSS classes: .fontface / .no-fontface

JavaScript property: Modernizr.fontface

Modernizr does a couple of things to detect the browser's @font-face support. In a highly contradictory—but still reliable—way, we start by detecting for Internet Explorer 5 and above. IE5+ supports @font-face via the proprietary EOT—Embedded OpenType—format. If you use web-licensed fonts, you can **create your own EOT** copy for use in IE.

After the IE check, Modernizr embeds a tiny font glyph using a `data:` URI. In this, the single "." character is created; it is then applied to a temporary element whose innerHTML is set to '........' and whose width is measured, first with a basic serif font and then with the custom font applied. If the width is different, we know the browser rendered the new font data we supplied.

Important notice: Unfortunately, WebKit and Gecko, the rendering engines of Safari, Chrome and FireFox (primarily), both load the font data asynchronously from their execution of the code—even the embedded data: URI. For this reason, Modernizr does a delayed check to re-verify itself. While this workaround is typically sufficient, the design of a page and the unreliability of the browsers' loading behavior can still cause inconsistencies. This delayed check is configured in a separate private property, `fontfaceCheckDelay`, found near the top of the script. You can customize it from 50 (milliseconds) to whatever is the optimal value for your specific design.

For the interested parties, here are the bugs for: **Gecko**, **WebKit**. Note that the WebKit bug report asks for an even better solution: being able to simply query which font formats the browser supports.

Sample Usage:

```
/* For an explanation of this cross-browser @font-face syntax, see:
    http://paulirish.com/2009/bulletproof-font-face-implementation-syntax/
*/
@font-face {
   font-family: MyWebLicensedFont;
   src: url(/fonts/my_web_licensed_font.eot);
   src: local('My Web Font'),
          url(/fonts/my_web_licensed_font.ttf) format("truetype");
}

.fontface #heading h1 {
   font: 16px/24px MyWebLicensedFont, Helvetica, sans-serif;
}
.no-fontface #heading h1 {/*
   Specify a background image as a fallback or prepare rules for sIFR/Cufon.
*/}
```

If you want to read the `Modernizr.fontface` property as the page loads, it's recommended to use the `Modernizr._fontfaceready(fn)` callback.

```
// The function you pass Modernizr._fontfaceready() that will execute after
//   the fontfaceCheckDelay has elapsed.
// The function will be passed the boolean value of Modernizr.fontface
Modernizr._fontfaceready(function(bool){

  // if @font-face isn't supported, you may want to employ Cufón
  if (!bool) getScript('cufon.withfont.min.js',function(){
    Cufon.now();
  });
});
```

Canvas

ˆ Table of contents

CSS CLASSES: .canvas / .no-canvas

JAVASCRIPT PROPERTY: Modernizr.canvas

The Canvas test is done by creating a `canvas` element and testing the presence of the `getContext` property.

Sample Usage:

```
<!-- In your HTML: -->
<div id="chartContainer">
   <p>You need to have JavaScript enabled to interact with this chart.</p>
   <img src="some_chart.png" alt="Chart description" />
</div>

/* In your CSS: */
.canvas #chartContainer p,
.canvas #chartContainer img {
   display: none;
}

// In your JavaScript:
if (Modernizr.canvas) {
   var c = document.createElement('canvas');
   var context = c.getContext('2d');
   //
   // Build your chart
   //
   document.getElementById('chartContainer').appendChild(c);
}
```

Canvas Text

ˆ Table of contents

CSS CLASSES: .canvastext / .no-canvastext

JAVASCRIPT PROPERTY: Modernizr.canvastext

To detect Canvas Text, Modernizr first checks that it got a positive result on the Canvas test (see above), then verifies that a 2D context on a newly created `canvas` element has a method called `fillText`. If so, Canvas Text is supported.

Sample Usage:

```
<!-- In your HTML: -->
<div id="canvasContainer">
   <canvas id="c"></canvas>
   <p>Much less interesting-looking fallback text</p>
</div>

/* In your CSS: */
.canvastext #canvasContainer p {
   position: absolute; visibility: hidden;
}
```

WebGL

ˆ Table of contents

WebGL brings a the OpenGL ES 2.0 API to browsers, natively. It is accessible via the `canvas` element.

Sample Usage:

```
if (Modernizr.webgl){
  loadAllWebGLAssets(); // webgl scripts can be > 100k
} else {
  var msg = 'You won't be able to see the WebGL experience without a great browser. \
            Try Chrome.';
  document.getElementById('#notice').innerHTML = msg;
}
```

HTML5 Audio

ˆ Table of contents

CSS classes: .audio / .no-audio

JavaScript property: Modernizr.audio

For HTML5 Audio, Modernizr detects whether the browser supports the `canPlayType` property on a `audio` element.

Sample Usage:

```
<!-- In your HTML: -->
<div id="audio">
   <audio>
      <source src="mySong.ogg" />
      <source src="mySong.mp3" />
   </audio>
   <button id="play">Play</button>
   <button id="pause">Pause</button>
</div>

/* In your CSS: */
.no-audio #audio {
   display: none; /* Don't show Audio options */
}
.audio #audio button {
   /* Style the Play and Pause buttons nicely */
}

// In your JavaScript:
if (Modernizr.audio) {
   // Hook up functionality to Play and Pause buttons
}
```

HTML5 Audio formats

ˆ Table of contents

If audio support is detected, Modernizr assesses which formats the current browser will play. Currently, Modernizr tests `ogg`, `mp3`, `wav` and `m4a`.

Important: The values of these properties are not true booleans. Instead, Modernizr **matches the HTML5 spec** in returning a string representing the browser's level of confidence that it can handle that codec. These return values are an empty string (negative response), "maybe" and "probably". The empty string is falsy, in other words: `Modernizr.audio.ogg == '' == false`

```
if (Modernizr.audio && Modernizr.audio.ogg){
          // preload ogg assets
```

```
        }
        else if (Modernizr.audio && Modernizr.audio.mp3){
          // preload mp3 assets
        }
```

HTML5 Video

ˆ Table of contents

CSS CLASSES: .video / .no-video

JAVASCRIPT PROPERTY: Modernizr.video

For HTML5 Video, Modernizr detects whether the browser supports the `canPlayType` property on a `video` element.

Sample Usage:

See **Video for Everybody** by Kroc Camen for a JavaScript-less way to use HTML5 `video` with graceful fallbacks for all browsers. With Modernizr's `video` detection, you can use CSS and JavaScript styling to further enhance the look and/or interactivity when the browser does support `video`.

HTML5 Video Formats

ˆ Table of contents

CSS CLASSES: (Classes for video formats are not applied)

JAVASCRIPT PROPERTY: Modernizr.video[format]

If video support is detected, Modernizr assesses which formats the current browser will play. Currently, Modernizr tests `ogg`, `webm` and `h264`.

Important: The values of these properties are not true booleans. Instead, Modernizr **matches the HTML5 spec** in returning a string representing the browser's level of confidence that it can handle that codec. These return values are an empty string (negative response), "maybe" and "probably". The empty string is falsy, in other words: `Modernizr.video.h264 == '' == false`

```
if (Modernizr.video && Modernizr.video.ogg){
            // preload ogg video assets
          }
          else if (Modernizr.video && Modernizr.video.h264){
            // preload h264 assets
          }
```

rgba()

ˆ Table of contents

Modernizr sets a background color with an `rgba` value, then attempts to retrieve it. If the browser returns the full `rgba` value, this feature is supported.

Sample Usage:

```
/* Remember: use the Cascade and default values whenever possible! */
.my_container {
   background-color: #ccc;
   color: #222;
}
.rgba .my_container {
   background-color: rgba(255,255,255, .5);
}
```

hsla()

˄ Table of contents

CSS CLASSES: .hsla / .no-hsla

JAVASCRIPT PROPERTY: Modernizr.hsla

Similar to **rgba()**, the `hsla()` test sets a background color. However, since browsers internally translate hsla to rgba, we retrieve the value expecting it as rgba—not as hsla.

Sample Usage:

```
/* Remember: use the Cascade and default values whenever possible! */
.my_container {
   background-color: #ccc;
   color: #222;
}
.hsla .my_container {
   background-color: hsla(120,40%,100%, .5);
}
```

border-image:

˄ Table of contents

CSS CLASSES: .borderimage / .no-borderimage

JAVASCRIPT PROPERTY: Modernizr.borderimage

Border Image is tested by specifying a `border-image:` (using all vendor prefixes) and testing the return value.

Sample Usage:

```
.my_elem {
   border: 1px inset #666;
}
.borderimage .my_elem {
   border: none;
   border-image: url(fancy-border.png) 5 5 5 5 round;
   -moz-border-image: url(fancy-border.png) 5 5 5 5 round;
   -webkit-border-image: url(fancy-border.png) 5 5 5 5 round;
}
```

border-radius:

˄ Table of contents

Border Radius is tested by specifying the shorthand `border-radius:` using all vendor prefixes and testing the return value.

A word of caution: while the shorthand seems to work in browsers that support border-radius, they don't have it fully implemented just yet. You can specify a single value to represent all corners, but that's it; not **all possible input values for the shorthand property** are available in all browsers. For custom corner radii, you'll have to specify each corner individually.

Sample Usage:

```
/* Remember: use the Cascade and default values whenever possible! */
nav a {
   background: #ccc url(not_adjustable_tab_image.png) bottom left no-repeat;
}
/* Note: Gecko uses non-compliant syntax! */
.borderradius nav a {
   background: #ccc;
   border-radius: 4px 4px 0 0;
   -moz-border-radius-topleft: 4px;
   -moz-border-radius-topright: 4px;
   -webkit-border-top-left-radius: 4px;
   -webkit-border-top-right-radius: 4px;
}
.borderradius nav a:focus,
.borderradius nav a:hover {
   background-color: #fff;
}
```

box-shadow

˄ Table of contents

CSS CLASSES: .boxshadow / .no-boxshadow

JAVASCRIPT PROPERTY: Modernizr.boxshadow

For Box Shadow, we test for it simply by setting it on an element and retrieving the value.

Sample Usage:

```
/* Simulated box shadow using borders: */
div.somediv {
   border-bottom: 1px solid #666;
   border-right: 1px solid #777;
}
.boxshadow div.somediv {
   border: none;
   box-shadow: #666 1px 1px 1px;
   -moz-box-shadow: #666 1px 1px 1px;
   -webkit-box-shadow: #666 1px 1px 1px;
}
```

text-shadow

˄ Table of contents

For text-shadow, we ask the browser if it recognizes the text-shadow style property. Firefox 3.0 false-positives this test, but there is no known fix to that. All IEs, including IE9, do not support text-shadow and thus fail to **deliver delight**.

Sample Usage:

```
/* ghosted letters */
.glowy {
   color: transparent;
   text-shadow: 0 0 10px black;
}
.no-textshadow {
   color: black;
}
```

Multiple backgrounds

˄ Table of contents

CSS CLASSES: .multiplebgs / .no-multiplebgs

JAVASCRIPT PROPERTY: Modernizr.multiplebgs

Multiple backgrounds are tested by specifying three background images on an element with a `background-color` set on the last specification (the only one that allows the color to be set). Retrieving the value and counting the number of occurrences of "url(" in the return value gives us an accurate detection.

Sample Usage:

```
div.container {
   background: #ccc url(fancy_bg.jpg) bottom left no-repeat;
   width: 500px; /* Limited due to background image */
}
.multiplebgs div.container {
   background: url(fancier_bl.jpg) bottom left no-repeat,
               url(fancier_br.jpg) bottom right no-repeat,
          #ccc url(fancier_bm.jpg) bottom center repeat-x;
}
```

background-size

˄ Table of contents

CSS CLASSES: .backgroundsize / .no-backgroundsize

JAVASCRIPT PROPERTY: Modernizr.backgroundsize

Background size enables you to scale the size of a background to new dimensions, based on the size of its container. (**MDC background-size docs**)

opacity

˄ Table of contents

CSS CLASSES: .opacity / .no-opacity

JAVASCRIPT PROPERTY: Modernizr.opacity

Opacity is tested by setting it with a value of ".5" and retrieving it. Browsers that have `opacity` properly implemented will return a value of "0.5".

CSS Animations

˄ Table of contents

CSS Animations are tested by setting an animation instruction and retrieving the value. See the specification for detailed instructions on using Animations.

Sample Usage:

```
/*
   Only WebKit supports CSS Animations at this time. Unfortunately, the
   -webkit- prefix must also be used for the @keyframes rule. Once more
   browsers support it, you'll need to duplicate the keyframes for each
   prefix, as using commas to separate them in the selector won't work.
*/
@-webkit-keyframes rainbow {
     0% { background: #f00; }
     9% { background: #f90; }
    18% { background: #ff0; }
    27% { background: #9f0; }
    36% { background: #0f0; }
    45% { background: #0f9; }
    54% { background: #0ff; }
    63% { background: #09f; }
    72% { background: #00f; }
    81% { background: #90f; }
    90% { background: #f0f; }
   100% { background: #f09; }
}
div.rainbow {
   background: url(image_of_rainbow.png) 0 0 no-repeat;
   height: 100px;
   width: 100px;
}
.cssanimations div.rainbow {
   -webkit-animation: rainbow 5s linear infinite;
}
```

CSS Columns

˄ Table of contents

CSS CLASSES: `.csscolumns` / `.no-csscolumns`

JAVASCRIPT PROPERTY: `Modernizr.csscolumns`

CSS Columns are detected by setting a `column-count:` property with all vendor prefixes. Note that CSS Columns are supported in some browsers, but their behavior is still known to be relatively buggy and their implementations incomplete.

Sample Usage:

```
nav ol li {
   float: left;
}
.csscolumns nav ol {
   -moz-column-count: 4;
   -webkit-column-count: 4;
   column-count: 4;
}
.csscolumns nav ol li {
   float: none;
}
```

CSS Gradients

˄ Table of contents

A gradient is set as a background image using vendor prefixes and then retrieved. Please note that Safari and Mozilla/Firefox use different syntax at this time, so you'll have to write your CSS for each browser individually.

For syntax documentation on each implementation, please see the following resources:

- http://webkit.org/blog/175/introducing-css-gradients/
- https://developer.mozilla.org/en/CSS/-moz-linear-gradient
- https://developer.mozilla.org/en/CSS/-moz-radial-gradient
- http://dev.w3.org/csswg/css3-images/#gradients-

Sample Usage:

```
button.glossy {
   background: #ccc url(gloss.png) 50% 50% repeat-x;
}
.cssgradients button.glossy {
   background: #ccc -webkit-gradient(linear, left top, left bottom,
         from(rgba(255,255,255, .4)),
         color-stop(0.5, rgba(255,255,255, .7)),
         color-stop(0.5, rgba(0,0,0, .2)),
         to(rgba(0,0,0, .1)));
}
.cssgradients button.glossy:hover {
   background-color: #fff;
}
```

CSS Reflections

˄ Table of contents

CSS CLASSES: .cssreflections / .no-cssreflections

JAVASCRIPT PROPERTY: Modernizr.cssreflections

CSS Reflections are currently a WebKit-only extension, but since they can be useful as visual enhancers they are included in Modernizr.

Sample Usage:

```
h2 {
   margin-bottom: 8px; /* No extra space needed */
}
/* Reflections use gradients as a mask, so check both values */
.cssgradients.cssreflections h2 {
        margin-bottom: 16px; /* Add some extra space! */
        -webkit-box-reflect: below -1px -webkit-gradient(linear,
              left top, left bottom,
              from(transparent),
              color-stop(0.55, rgba(255,255,255, .01)),
              to(rgba(255,255,255, .15)));
}
```

CSS 2D Transforms

˄ Table of contents

CSS 2D Transforms allow elements to be transformed in 2D space. We test for the `transform` property (with all vendor prefixes) to see if the browser supports it.

Sample Usage:

```
.csstransforms img.banner {
        transform: rotate(5deg);
        -moz-transform: rotate(5deg);
        -webkit-transform: rotate(5deg);
}
/* This will rotate it by a couple of degrees, like the 1.0 on our Home page */
```

CSS 3D Transforms

ˆ Table of contents

CSS CLASSES: .csstransforms3d / .no-csstransforms3d

JAVASCRIPT PROPERTY: Modernizr.csstransforms3d

CSS 3D Transforms are a three-dimensional version of the 2D Transforms spec. Using it, you can transform elements in a 3D space across your page, transforming them in myriad ways. Elements transformed in 3D space remain fully interactive.

For Modernizr, we test the browser's support for the `perspective` property.

Sample Usage:

```
/* Only WebKit supports 3D Transforms, on iPhone 2.0+ and Leopard and later. */
#container {
   -webkit-perspective: 500;
}
#container div {
   -webkit-transform: rotateY(20deg);
}
```

Flexible Box Model

ˆ Table of contents

CSS CLASSES: .flexbox / .no-flexbox

JAVASCRIPT PROPERTY: Modernizr.flexbox

The flexible box model (aka flexbox) offers a different way for positioning elements, that addresses some of the shortcomings of float-based layouts.

CSS Transitions

ˆ Table of contents

CSS CLASSES: .csstransitions / .no-csstransitions

JAVASCRIPT PROPERTY: Modernizr.csstransitions

CSS Transitions are an incredibly useful new part of CSS3. Using them, you can let the browser animate—or rather, transition—from one state to the other. You only have to specify a start and end and the browser takes care of the rest.

In Modernizr we test for CSS Transitions using the `transition` property with all vendor prefixes.

Transitions can typically be used without using Modernizr's specific CSS class or JavaScript property, but for those occasions you want parts of your site to look and/or behave differently they are available. A good example use case is to build Modernizr into an animation engine, which uses native CSS Transitions in the browsers that have it, and relies on JavaScript for the animation in browsers that don't.

Sample Usage:

```
a {
   color: #090;
   -webkit-transition: color .2s ease-out;
}
a:focus,
a:hover {
   color: #9f9;
}
```

Geolocation API

ˆ Table of contents

CSS CLASSES: `.geolocation` / `.no-geolocation`

JAVASCRIPT PROPERTY: `Modernizr.geolocation`

The **Geolocation API** allows a user to permissively disclose their location, thereby enabling location-aware functionality or content within a site or application.

In Modernizr, we test the `navigator.geolocation` object.

Sample Usage:

```
if (Modernizr.geolocation){
  navigator.geolocation.getCurrentPosition(function(position) {
    // pass the lat and long values to an application
    // e.g. a setUserLatandLong() function may find the closest bodega
    setUserLatandLong(position.coords.latitude,position.coords.longitude);
  });
}
```

Input Types

˄ Table of contents

CSS CLASSES: (Classes for input types are not applied)

JAVASCRIPT PROPERTY: `Modernizr.inputtypes[type]`

HTML5 introduces **thirteen new values** for the `<input>`'s type attribute. They are as follows: `search, tel, url, email, datetime, date, month, week, time, datetime-local, number, range, color`.

These types can enable native datepickers, colorpickers, URL validation, and so on. If a browser doesn't support a given type, it will be rendered as a text field. Modernizr cannot detect that date inputs create a datepicker, the color input create a colorpicker, and so on—it will detect that the input values are sanitized based on the spec.

Sample Usage:

```
<!-- In your HTML: -->
<input type="date" name="birthday" id="birthday">

// In your JavaScript:
if (!Modernizr.inputtypes.date){
  // if no native support, use a datepicker script
  createDatepicker(document.getElementById('birthday'));
}
```

Input Attributes

˄ Table of contents

HTML5 introduces **many new attributes for input elements**. Modernizr tests for these: `autocomplete, autofocus, list, placeholder, max, min, multiple, pattern, required, step`.

These new attributes can do things such as: enable native autocomplete, mimic elem.focus() at page load, do form field hinting, and do client-side validation.

View the **HTML5 input attribute support** page from Mike Taylor to see them in action.

Sample Usage:

```
    // if placeholder isn't supported:
    if (!Modernizr.input.placeholder){
      // use a input hint script
      setInputHint(document.getElementById('username'),'Enter Username');
    }
```

localStorage

˄ Table of contents

CSS CLASSES: .localstorage / .no-localstorage

JAVASCRIPT PROPERTY: Modernizr.localstorage

Local Storage is part of the **new HTML5 Web Storage spec**. See that specification for implementation details.

In Modernizr, we test for `localStorage in window`.

Sample Usage:

```
if (Modernizr.localstorage){
  // Take advantage of local storage that is persistent
  // between tabs on the same site and can store MBs of data
} else {
  // resort to cookies as best you can
}
```

sessionStorage

˄ Table of contents

CSS CLASSES: .sessionstorage / .no-sessionstorage

JAVASCRIPT PROPERTY: Modernizr.sessionstorage

Session Storage is part of the **new HTML5 Web Storage spec**. See that specification for implementation details.

In Modernizr, we test for `sessionStorage in window`.

Sample Usage:

```
if (Modernizr.sessionstorage){
  // Take advantage of session storage in the browser that
  // won't conflict between tabs on the same site
} else {
  // resort to cookies as best you can
}
```

Web Workers

˄ Table of contents

Web Workers is a new API for running scripts in the background independently of any user interface scripts. Please see the Web Workers specification for implementation details and tutorials.

In Modernizr, we test the `window.Worker` object.

applicationCache

˄ Table of contents

The `applicationCache` is a crucial interface part of the **HTML5 Offline Web Applications API**. Using it, you can turn your web application into one that'll continue functioning even when the user goes offline. With Modernizr, you can easily determine whether the user is capable of using your web application in offline mode, thanks to applicationCache.

In Modernizr, we test the `window.applicationCache` object.

Sample Usage:

```
if (Modernizr.applicationcache){
  // We have offline web app support! Continue operation,
  // indicating to the user that the app will sync up once they get back online
} else {
  // No offline support, show errors if the user goes offline
}
```

SVG

˄ Table of contents

CSS CLASSES: `.svg` / `.no-svg`

JAVASCRIPT PROPERTY: `Modernizr.svg`

Sample Usage:

```
if (Modernizr.svg){
    // SVG is supported natively
    } else {
    // kick off flash fallback
    }
```

Inline SVG

˄ Table of contents

CSS CLASSES: `.inlinesvg` / `.no-inlinesvg`

JAVASCRIPT PROPERTY: `Modernizr.inlinesvg`

In HTML5 you can now mix SVG elements in with HTML without namespacing or other tricks.

SVG Clip Paths

˄ Table of contents

With clip paths, you can clip based on SVG shapes. This especially powerful when using SVG along with HTML content. **More detail on MDC**

SMIL

˄ Table of contents

SMIL is declarative animation used along with SVG.

Web SQL Database

ˆ Table of contents

CSS CLASSES: .websqldatabase / .no-websqldatabase

JAVASCRIPT PROPERTY: Modernizr.websqldatabase

Web SQL Database is a client-side SQLLite database.

IndexedDB

ˆ Table of contents

CSS CLASSES: .indexeddb / .no-indexeddb

JAVASCRIPT PROPERTY: Modernizr.indexeddb

IndexedDB is a client side storage database. Vendors have inconsistent prefixing with the experimental Indexed DB: Firefox is shipping indexedDB in FF4 as `moz_indexedDB`, and Webkit's implementation is accessible through `webkitIndexedDB`. We test both styles.

Web Sockets

ˆ Table of contents

CSS CLASSES: .websockets / .no-websockets

JAVASCRIPT PROPERTY: Modernizr.websockets

WebSockets provides bi-directional, full-duplex communications channels, over a single TCP socket.

Sample Usage:

```
if (Modernizr.websockets){
        // Use Web Sockets
        } else {
        // revert to a basic comet-based exchange
        }
```

Hashchange Event

ˆ Table of contents

The hashchange event fires when a window's hash (`location.hash`) changes.

Sample Usage:

```
if (Modernizr.hashchange){
        // bind to window.onhashchange
        } else {
        // set up a polling loop to watch for changes
        }
```

History Management

ˆ Table of contents

We test two objects to detect presence of the HTML5 history API: `window.history` and `history.pushState`. In Modernizr 1.5 this was available via the `historymanagement` name, but it was simplified to `history` as of 1.6.

Drag and Drop

ˆ Table of contents

CSS classes: .draganddrop / .no-draganddrop

JavaScript property: Modernizr.draganddrop

In Modernizr 1.5, we test for the following drag events:

- drag
- dragstart
- dragenter
- dragover
- dragleave
- dragend
- drop

Cross-window Messaging

ˆ Table of contents

CSS classes: .postmessage / .no-postmessage

JavaScript property: Modernizr.postmessage

We test availability of the `window.postMessage` method. In Modernizr 1.5 this was available via the `crosswindowmessaging` name, but it was simplified to `postmessage` as of 1.6.

Touch

ˆ Table of contents

CSS classes: .touch / .no-touch

JavaScript property: Modernizr.touch

The Modernizr.touch test only indicates if the browser supports touch events, which does not necessarily reflect a touchscreen device. For example, Palm Pre / WebOS (touch) phones do not support touch events and thus fail this test. Additionally, Chrome (desktop) used to lie about its support on this, but **that has since been rectified**. Modernizr also tests for Multitouch Support via a media query, which is how Firefox 4 exposes that for Windows 7 tablets. For more info, see: **http://modernizr.github.com/Modernizr/touch.html**. Added in 1.6

Sample Usage:

```
if (Modernizr.touch){
  // bind to touchstart, touchmove, etc and watch `event.streamId`
} else {
  // bind to normal click, mousemove, etc
}
```

Extensibility

addTest() Plugin API

ˆ Table of contents

You may want to test additional features that Modernizr currently does not support. For that, you can use the addTest function. For example, some users have requested tests for IE's float double margin bug, and support for `position:fixed`. Using addTest, you can add these yourself and get the exact same API as the fully supported tests.

Sample Usage:

```
// Test for position:fixed support

Modernizr.addTest('positionfixed', function () {
    var test    = document.createElement('div'),
        control = test.cloneNode(false),
        fake = false,
        root = document.body || (function () {
            fake = true;
            return document.documentElement.appendChild(document.createElement('body'));
        }());

    var oldCssText = root.style.cssText;
    root.style.cssText = 'padding:0;margin:0';
    test.style.cssText = 'position:fixed;top:42px';
    root.appendChild(test);
    root.appendChild(control);

    var ret = test.offsetTop !== control.offsetTop;

    root.removeChild(test);
    root.removeChild(control);
    root.style.cssText = oldCssText;

    if (fake) {
        document.documentElement.removeChild(root);
    }

    return ret;
});
```

Assuming the above test passes, there will now be a `.positionfixed` class on the HTML element and `Modernizr.positionfixed` will be true. IE6, of course, will now have a `.no-positionfixed` class.