CSS Browser Selector Plus: A JavaScript Library to Support Cross-browser Responsive Design

Richard Duchatsch Johansen
W3C Accessibility WG Member and Senior Front-end Developer at Eventials – Rua Itapaiúna, 2434 São Paulo – SP – Brazil, Zip Code 05707-001
+55 14 9711-7983
ridjohansen@gmail.com

Talita C. Pagani Britto
Assistant Coordinator of Educational Projects - MStech
Rua Joaquim Anacleto Bueno, 1-42 Bauru – SP – Brazil, Zip Code 17047-281
+55 14 3235-5500
talita.britto@mstech.com.br

Cesar Augusto Cusin
Professor at Faculdade Paraíso do Ceará and W3C Accessibility WG Member – Rua da Conceição, 1228 Juazeiro do Norte – CE – Brazil, Zip Code 63010-465
+55 15 8100-4466
cesar@cusin.com.br

ABSTRACT
Developing websites for multiples devices have been a rough task for the past ten years. Devices features change frequently and new devices emerge every day. Since W3C introduced media queries in CSS3, it's possible to developed tailored interfaces for multiple devices using a single HTML document. CSS3 media queries have been used to support adaptive and flexible layouts, however, it's not supported in legacy browsers. In this paper, we present CSS Browser Selector Plus, a cross-browser alternative method using JavaScript to support CSS3 media queries for developing responsive web considering older browsers.

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: Language Constructs and Features – *frameworks.*

H.5.2 [**Information Interfaces and Presentation (e.g., HCI)**]: User Interfaces – *standardization.*

H.5.4 [**Information Interfaces and Presentation (e.g., HCI)**]: Hypertext/Hypermedia – *architectures.*

Keywords
Responsive Web Design, cross-browser, web standards, JavaScript.

1. INTRODUCTION
Nowadays, we are facing a bias on web development to support different devices, platforms and web browsers that increases in variety every day. As media queries are not supported in older browsers, novel approaches, such as code scripts named polyfills or fallbacks, were developed to provide this functionality in browsers that don't support it. However, these polyfills and fallbacks present limitation in its usage and often lack some features. This paper presents a cross-browser JavaScript library named CSS Browser Selector Plus, a script that allow developers to work with responsive web design for older browsers.

2. CSS3 MEDIA QUERIES
CSS Media queries allow you to target CSS rules based on - for instance - screen size, device-orientation or display-density. This means you can use CSS Media Queries to tweak a CSS for a mobile devices, printer or create a responsive site.

Media queries is an extension to the `@media` (or `media=""` attribute, in `<link>` tag) specification on CSS, allowing declaration of conditional queries expressions to detect particular media features, such as viewport width, display color, orientation and resolution [1], as shown on Table 1.

Table 1. Example of CSS3 media queries expressions

Feature	Query
Maximum viewport width of 480px	`@media screen and (max-width:480px)`
Landscape orientation	`@media all and (orientation:landscape)`
Widescreen displays (16:9)	`@media screen and (device-aspect-ratio: 16/9)`
Display with minimum width of 400px and maximum of 700px	`@media screen and (min-width: 400px) and (max-width: 700px)`

Its basic syntax consists of:

```
@media <mediatype>[ <boolean_operator> (<condition>)]
```

Where the expression `[ <boolean_operator> (<condition>)]` can be repeated to accomplish multiple conditions.

2.1 Fallbacks and Polyfills
Once media queries were introduced in CSS3 specification, it's not supported by legacy browsers such as Internet Explorer 8 (and older). To emulate this functionality in browsers that don't support it natively, there are several polyfills – often called as fallback – available:

- **css3-mediaqueries-js:** the library css3-mediaqueries-js is one of the most used to empower the use of media queries in older browsers. It's usage consists in just add the JS file to the HTML document and use media queries as usual on CSS [2].
- **Respond.js:** Respond.js is a JS polyfill to support only the min/max-width of CSS3 Media Queries. Its focus is the support to legacy versions of Internet Explorer [3].
- **Mediatizr:** Mediatizr is a JS library to allow media queries support to older browsers. The features detected by Mediatizr

WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2038-2/13/05.

include min-width/max-width, min-device-width/max-device-width [4].

- **matchMedia:** The script matchMedia is a polyfill helper to test on JavaScript whether a CSS media type or media query applies, however, it doesn't empower the use of media queries in CSS documents. MatchMedia is used in Respond.js [5].
- **jQuery Media Helpers:** Similar to matchMedia, jQuery Media Helpers is a utility script to handle cross-browser media queries features and behavior on JavaScript, not being applicable to enable cross-browser media queries in CSS [6].

However, current polyfill libraries available detects only a subset of media queries features, usually **`[device-]<min|max>-width`**. Furthermore, css3-mediaqueries-js and Respond.js doesn't work with **`@imported`** style sheets and do not listen to media attribute of **`<link>`** and **`<style>`** elements, what make their usage restrict. Similarly, the library Mediatzr doesn't work with the **`<style>`** element. The matchMedia and jQuery Media Helpers libraries are useful for JavaScript-only use, as they are not intended to listen and parse media queries from CSS documents

3. PROPOSED SOLUTION

3.1 CSS Browser Selector

The CSS Browser Selector is an open-source library composed by a single JavaScript (JS) file, originally created to empower CSS selectors and write specific CSS code for different operating systems and web browsers without CSS hacks [7]. Similar to Modernizr [8], CSS Browser Selector adds classes to the **`<html>`** tag but, instead of feature detection, it detects the user's operating system (OS) and web browser. We proposed an extension called CSS Browser Selector Plus to support media queries features as an alternative method to the native CSS3 media queries. The generated classes on the **`<html>`** tag indicate characteristics such as device orientation, screen width and support to retina displays.

3.2 Syntax

The basic syntax to use the generated classes is:

```
.[feature1][.[feature2][…].[featureN]]
[.class|#ID|tag] { property: value }
```

Once the script adds CSS classes to the **`<html>`** tag, it's just needed to precede selectors with the desired feature. Table 2 presents some of the possible CSS classes generated:

Table 2. Example of CSS classes generated by the script

Feature	Generated code
Landscape orientation	`<html class="orientation_landscape">`
No support to HiPDI and support to DataURI	`<html class="no-hidpi datauri">`
Minimum screen width of 980px and maximum width of 1199px in portrait orientation	`<html class="orientation_portrait minw_980 maxw_1199">`

4. SAMPLE CODES

4.1 Data URI Selector

Data URI is a scheme defined in RFC 2397 [9] that allow to embed immediate data instead of usual URLs to load a certain resource. In CSS, this is used in **`background/background-image`** to specify binary codes for images encoded in **`base64 encoding`**, improving web page performance since the browser doesn't need to make an HTTP request to load a resource. Data URIs have been supported by Internet Explorer 8.0+ (partial support), Firefox 16.0+, Chrome 23.0+, Safari 5.1+, Opera 12.1 [10], considering desktop browsers. Figure 1 shows how it's possible to deal with browser that support Data URI and browsers that don't support it, presenting a regular file URL.

```
<style>
/* LOADING A DATA URI
 * IN SUPPORTED BROWSERS
 *  */
.datauri .example_bg {
    background-image:
url('
AABAAAAAQAQAMAAAAlPW0iAAAABlBMVEUAAAD///
+l2Z/dAAAAM0lEQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83N
DDeNGe4Ug9C9zwz3gVLMDA/A6P9/AFGGFyjOXZtQAAAAAElFTk
SuQmCC');
}
/* LOADING AN USUAL URL
 * FOR NON-SUPPORTED BROWSERS
 *  */
.no-datauri .example_bg {
    background-image: url('bg_default.png');
}
```

Figure 1. Filtering DataURI support and providing fallback for non-supported browsers

4.2 Screen Width

CSS Browser Selector Plus supports detection of viewport minimum and maximum width, updating on browser resizing. When there's no minimum width, the default value is zero (**`minw_0`**). Figure 2 presents examples of screen width filters comparing media queries and generated classes from the script.

```
/* SELECTING A VIEWPORT WITH MAX. WIDTH OF 767px
 * THROUGH MEDIA QUERIES */
@media (max-width: 767px) {
    .example {
        border: 2px solid purple !important;
    }
}
/* SELECTING A VIEWPORT WITH MAX. WIDTH OF 767px
 * USING CSS BROWSER SELECTOR + */
.minw_0 .example, .maxw_767 .example {
    border: 2px solid purple !important;
}

/* SELECTING A VIEWPORT WITH MIN. WIDTH OF 768px
 * AND MAX. WIDTH OF 979px
 * THROUGH MEDIA QUERIES */
@media (min-width: 768px) and (max-width: 979px) {
    .example {
        border: 2px solid green!important;
    }
}
/* SELECTING A VIEWPORT WITH MIN. WIDTH OF 768px
 * AND MAX. WIDTH OF 979px
 * USING CSS BROWSER SELECTOR + */
.minw_768.maxw_979 .example {
    border: 2px solid green!important;
}

/* SELECTING A VIEWPORT WITH MIN. WIDTH OF 1200px
 * USING CSS BROWSER SELECTOR + */
@media (min-width: 1200px) {
    .example {
        border: 2px solid orange!important;
    }
}
/* SELECTING A VIEWPORT WITH MIN. WIDTH OF 1200px
 * USING CSS BROWSER SELECTOR + */
.minw_1200 .example {
border: 2px solid orange!important;
}
```

Figure 2. Example of width filters with media queries and CSS Browser Selector Plus

4.3 HiDPI and Pixel-Ratio Selectors

HiDPI, often referred as "Retina", is the name given to displays with high pixel density, presenting a better quality on screen view [11]. The detection of this feature is done through the **`device-pixel-ratio`** and **`min-resolution`**. Non-retina displays present **`device-pixel-ratio`** of **`1.0`**, while displays with high definition present **`device-pixel-ratio`** of **`1.5`** and full HiDPI and retina

displays have usually the value of **`2.0`** [12]. Figure 3 shows how **`device-pixel-ratio`** can be used in media queries to detect retina displays. Notice that, as it's an experimental feature, it need to be have browser's prefix. On Figure 4, we present how this can be done with a friendly syntax by using the classes **`.hidpi`** and **`.retina_1x/.retina_2x`**.

5. FINAL THOUGHTS

Legacy browsers and cellphones do not have support to media queries and current polyfill libraries to enable CSS3 media queries support in this scenario don't contemplate all features of media queries and have limited usage. Our proposed solution, CSS Browser Selector Plus, is a lightweight script to allow web developers work with responsive web design for a wide variety of web browsers. It provides a friendly syntax to ensure a progressive enhanced code, ensuring basic functionalities for browsers that don't support specific features and provide advanced functionalities for newer browsers. While current polyfills and fallbacks attempt to emulate the support to media queries expressions, our proposed solution can be used as a substitute for media queries. As an open-source work, CSS Browser Selector Plus is available at GitHub (https://github.com/ridjohansen/css_browser_selector) for contribution from community.

6. ACKNOWLEDGMENTS

Thanks to Rafael Lima for authorizing the diffusion of the library described in this paper.

```
<style>
/* SELECTING RETINA DISPLAYS WITH
 * MINIMUM DEVICE PIXEL RATIO OF 1.5 */
@media
only screen and (min--moz-device-pixel-ratio: 1.5),
only screen and (-o-min-device-pixel-ratio: 3/2),
only screen and (-webkit-min-device-pixel-ratio: 1.5),
only screen and (min-device-pixel-ratio: 1.5) {
    .example_bg {
        background-image: url('bg_hidpi_1x.png');
    }
    .img_default, .x2 {
        display: none;
    }
}

/* SELECTING RETINA DISPLAYS WITH
 * MINIMUM DEVICE PIXEL RATIO OF 2.0 */
@media
only screen and (-webkit-min-device-pixel-ratio: 2),
only screen and (    min--moz-device-pixel-ratio: 2),
only screen and (      -o-min-device-pixel-ratio: 2/1),
only screen and (         min-device-pixel-ratio: 2),
only screen and (                 min-resolution: 192dpi),
only screen and (                 min-resolution: 2dppx) {
    .example_bg {
        background-image: url('bg_hidpi_2x.png');
    }
    .img_default, .x1 {
        display: none;
    }
}

.img_hidpi {
    display: none;
}
```

Figure 3. HiDPI detection through media queries using device-pixel-ratio and min-resolution

7. REFERENCES

[1] W3C. Media Queries. *Standards*. Retrieved February 12, 2013 from: http://www.w3.org/TR/css3-mediaqueries/

[2] Graaf, W. css3-mediaqueries-js. Retrieved February 13, 2013, from Google Code: http://code.google.com/p/css3-mediaqueries-js/

[3] Jehl, S. Respond. Retrieved February 13, 2013, from GitHub: https://github.com/scottjehl/Respond

[4] Delogu, A. mediatizr. Retrieved February 13, 2013, from GitHub: https://github.com/pyrsmk/mediatizr

[5] Jehl, S., Irish, P., Zakas, N. matchMedia.js. Retrieved February 13, 2013, from GitHub: https://github.com/paulirish/matchMedia.js/

[6] Jehl, S. jQuery Media Helpers. Retrieved February 13, 2013, from GitHub: https://github.com/scottjehl/jQuery-Media-Helpers

[7] Lima, R. CSS Browser Selector. Retrieved February 13, 2013, from: http://rafael.adm.br/css_browser_selector/

[8] Ateş, F. et al. Modernizr. Retrieved February 13, 2013, from: http://modernizr.com/

[9] Masinter, L. The "data" URL scheme. Retrieved February 13, 2013, from IETF: http://tools.ietf.org/html/rfc2397

[10] Can I Use... Data URIs. Retrieved February 13, 2013, from Can I Use…: http://caniuse.com/#search=uri

[11] Retina Display. Retrieved February 14, 2013, from Wikipedia: http://en.wikipedia.org/wiki/Retina_Display

[12] Edwards, M. Device pixel density tests. Retrieved February 14, 2013, from: http://bjango.com/articles/min-device-pixel-ratio/

```
/* SETING A COMMON IMAGE FOR DEVICES THAT
 * DO NOT SUPPORT HIDPI IMAGES WITH
 * CSS BROWSER SELECTOR +
 */
.no-hidpi .example_bg {
    background-image: url('bg_default.png');
}
.hidpi .img_default, .no-hidpi .img_hidpi {
    display: none !important;
}
/* SETING HIDPI IMAGE FOR
 * RETINA DISPLAYS WITH
 * CSS BROWSER SELECTOR +
 */
.retina_1x .example_bg {
    background-image: url('bg_hidpi_1x.png');
}
.retina_2x .example_bg {
    background-image: url('bg_hidpi_2x.png');
}
.retina_2x .img_default, .retina_2x .x1 {
    display: none !important;
}
.retina_1x .img_default, .retina_1x .x2 {
    display: none !important;
}
</style>

<div class="example_bg">
    <img class="img_hidpi x2" src="img_hidpi_2x.png"
alt="with pixel-ratio >= 2" />
    <img class="img_hidpi x1" src="img_hidpi_1x.png"
alt="with pixel-ratio > 1 and < 2" />
    <img class="img_default" src="img_default.png"
alt="with pixel-ratio 1 or default" />
</div>
```

Figure 4. HiDPI detection using CSS Browser Selector Plus