



Responsive Media



Responsive Media

- When it comes to rich experiences online, we have a love/hate relationship.
- Love
 - Beautiful images and interesting videos help to provide a deeper, more pleasant experience.
- Hate
 - Including many images and videos on a page results in a slow loading time, which can be very frustrating.



Responsive Media

- It takes careful consideration and planning to give users the best of both worlds:
 - a beautiful experience that loads as quickly as possible.
- We have already seen that the following help create Responsive Designs
 - Fluid Layouts
 - Viewports
 - Media Queries
 - Grid Based Layouts
- But there is still plenty of room to tidy things up



Responsive Media

- With this topic, we will discuss:
 - Why performance matters
 - How to conditionally load images
 - What responsive image solutions are available, and their limitations
 - How to swap out background images without downloading multiple images
 - How to conditionally load web fonts
 - What is ahead for responsive images
 - How to make embedded video scale while maintaining its aspect ratio
 - What to do with responsive advertising



What the problem?

- Not all Media can scale to the degree we want
 - If I try to scale too large the image start to pixelate
 - If I try to scale too small the focus of the image gets lost

Image Size Comparison

Normal Size

Enlarged



Image Size Comparison

Normal Size

Reduced





What the problem?

- However, image quality is only one part of the problem. Another problem is the image weight and the demand that places on performance
- With what we have been doing so far, the same image is being loaded regardless of the device in use.
- That means, an 624px lead-in image is being downloaded even on small screens where a 350px image is all that is needed.
- The page performance is visibly suffering.



Performance

- Poor performing has an impact on site impact
- Case Study
 - Shopzilla improved its page load time from 4 to 6 seconds to 1.5 seconds. The results were stunning. The site's conversion rate increased by 7 to 12 percent and page views jumped a whopping 25 percent.
 - Personal experience supports this
- The situation is much worse for mobile phones. Networks are slower, hardware is less capable, and you have to deal with the messy world of data limitations and transcoding methods.

Three common Performance Issues with Mobile Sites

- Download and Hide
- Download and Shrink
- Excess DOM

Download and Hide

- Two Approaches to hiding an item
 - `visibility:hidden`
 - `display:none`
- See them in use [here](#)

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      img.hidden {display:none;}
    </style>
  </head>

  <body>
    <h1>The next image will not display</h1>
    

</html>
```



Download and Hide

- The problem with this solution is it does not stop the download. It just prevent displaying the image after download
- So we have to think Mobile-Up and decide what images we need, and not Desktop-Down and decide which images to hide

Download and Shrink

- Conceptually similar to Download and Hide
- Here we download an image intended for a large screen device and shrink the image down to fit into a smaller display area
- The result is that we degrade performance by downloading too much image data
- We should download the right sized image.
 - Image resizers
 - Digital Asset Management (DAM) systems

Selectively Serving Images to Mobile

- So where do we start?
- Refer Back to Future Friendly Manifesto (Slide deck #11 - slides 22-27)

Selectively Serving Images to Mobile

- Step One
 - Remove all the graphics
- This gives us a bare bones solution and can represent a solution approach for our lower end platforms

HTML's data-* Attributes

- The data-* attributes is used to store custom data private to the page or application.
- The data-* attributes gives us the ability to embed custom data attributes on all HTML elements.
- The data-* attributes are supported in all major browsers.
- The attribute value can be any string

HTML's data-* Attributes

- The stored (custom) data can then be used in the page's JavaScript to create a more engaging user experience (without any Ajax calls or server-side database queries).
- The data-* attributes consist of two parts:
 - The attribute name should not contain any uppercase letters, and must be at least one character long after the prefix "data-"
 - The attribute value can be any string

HTML's data-* Attributes

- Example

```
<ul>
  <li data-animal-type="bird">
    Owl
  </li>
  <li data-animal-type="fish">
    Salmon
  </li>
  <li data-animal-type="spider">
    Tarantula
  </li>
</ul>
```

HTML's data-* Attributes

- **Important:**
 - Custom attributes prefixed with “data-” will be completely ignored by the user agent e.g., Browser.

```
<ul class = "slats">
  <li data-src = "images/ball.jpg" class = "group">
    <a href = "#" >
      <h3> Kicker connects on record 13 field goals </h3>
    </a>
  </li>
  <li data-src = "images/goal_post.jpg" class = "group">
    <a href = "#">
      <h3> Your favorite team loses to that team no one
likes </h3>
    </a>
  </li>
  <li data-src = "images/ball_field.jpg" class = "group">
    <a href = "#">
      <h3> The Scarecrows Win 42-0 </h3>
    </a>
  </li>
</ul>
```

```
q : function( query) {
  if (document.querySelectorAll) {
    var res = document.querySelectorAll(query);
  }else {
    var d = document,
        a = d.styleSheets[0] || d.createStyleSheet();
    a.addRule( query, 'f:b');
    for(var l = d.all, b = 0, c =[], f = l.length;
        b < f; b + +) {
      l[ b].currentStyle.f && c.push(l[ b]);
      a.removeRule( 0);
      var res = c;
    }
    return res;
  }
}
```

- 
- The JavaScript takes a Selector and returns the elements that match it

```
//load in the images
var lazy = Utils.q('[data-src]');
for (var I = 0 ; I < lazy.length; i++){
    var source = lazy[i].getAttribute('data-src');
    //create the image
    var img = new Image();
    img.src = source;
    //Insert image inside of the link
    lazy[i].insertBefore(img, lazy.firstChild);
}
```

Line 2 grabs any elements with a data-src attribute applied.

Then, in line 3 the script loops through those elements.

In lines 4– 7, the script creates a new image for each element using the value of the data-src attribute.

The script then inserts the new image (line 9) as the first element within the link.

But this still load all images



The `matchMedia()` Method

- `matchMedia()` method is a native JavaScript that lets you pass in a CSS media query and receive information on whether or not the media query is a match
- `matchMedia()` return a `MediaQueryList` that has the following properties
 - `matches` – a true/false value showing if the media query matches and
 - `media` – the media query passed in the `matchMedia()`

Selectively Serving Images to Mobile

```
if (window.matchMedia("(min-width: 37.5em)").matches) {  
    //load in the images  
    var lazy = Utils.q('[data-src]');  
    for (var i = 0 ; i < lazy.length; i++){  
        var source = lazy[i].getAttribute('data-src');  
        //create the image  
        var img = new Image();  
        img.src = source;  
        //Insert image inside of the link  
        lazy[i].insertBefore(img, lazy.firstChild);  
    }  
}
```

Selectively Serving Images to Mobile

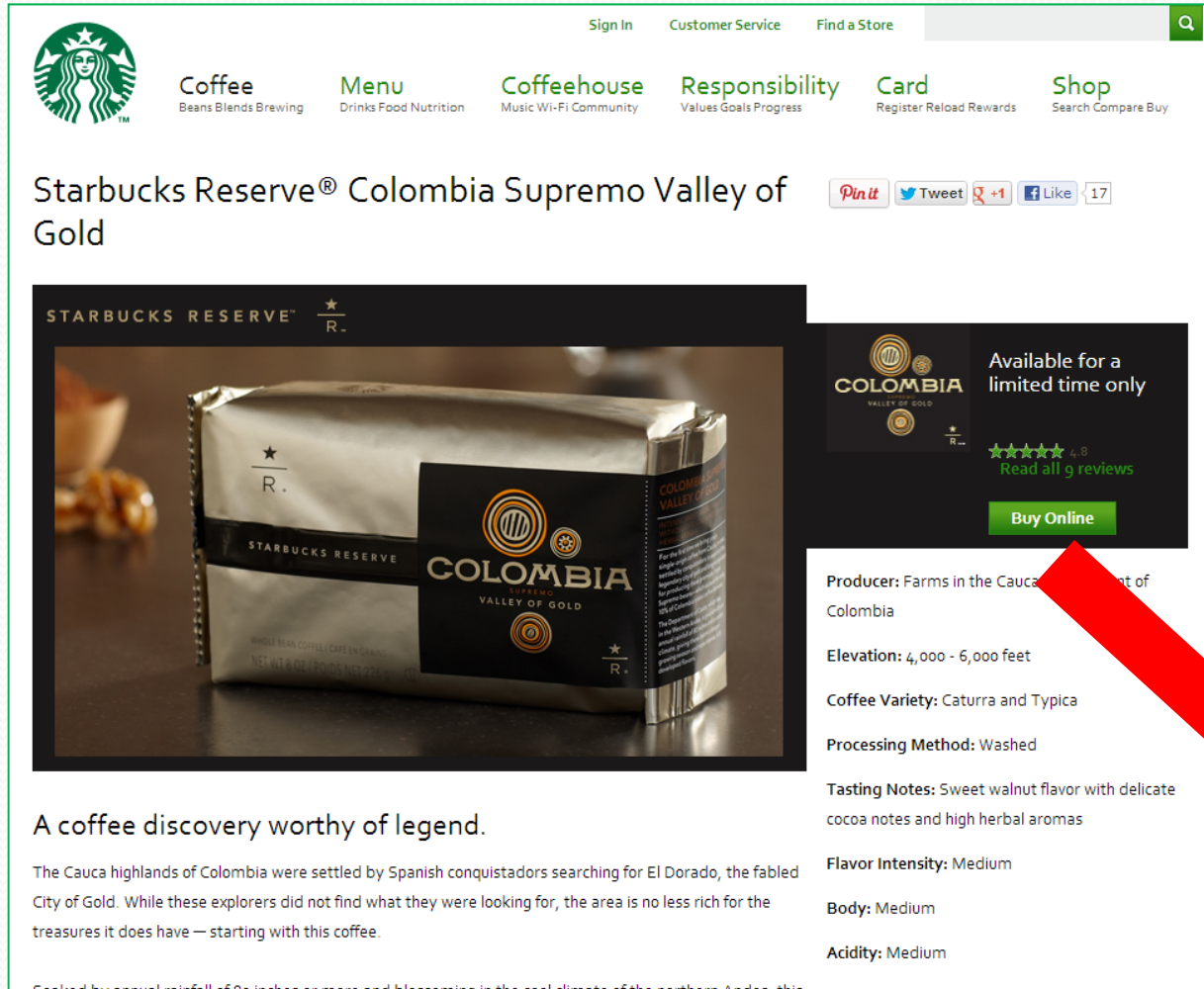
- Restricts loading images to displays greater than 37.5em wide
- For small screen devices
 - Reduced the number of HTTP call by 3
 - Reduced the size of the size of the page (3 images less to download)
- This is a Performance gain!!!!



Responsive Design

- Now we start to see what we have to do to create responsive design.
 - We have to write JavaScript to allow us to intelligently place content in our Fluid layouts

Consider Again



The screenshot shows the Starbucks website interface for the Starbucks Reserve Colombia Supremo Valley of Gold coffee. At the top, there is a navigation bar with links for Sign In, Customer Service, Find a Store, and a search icon. Below this are menu categories: Coffee (Beans Blends Brewing), Menu (Drinks Food Nutrition), Coffeehouse (Music Wi-Fi Community), Responsibility (Values Goals Progress), Card (Register Reload Rewards), and Shop (Search Compare Buy). The main heading is "Starbucks Reserve® Colombia Supremo Valley of Gold" with social media sharing options for Pin it, Tweet, +1, Like, and 17. A large image shows a bag of the coffee. To the right of the image is a dark overlay with the text "Available for a limited time only", a 4.8 star rating, and a "Buy Online" button. Below the image and overlay is a detailed description of the coffee, including its producer, elevation, variety, processing method, and tasting notes.

STARBUCKS RESERVE®
R.

STARBUCKS RESERVE
COLOMBIA
SUPREMO
VALLEY OF GOLD

Available for a limited time only

★★★★★ 4.8
Read all 9 reviews

Buy Online

Producer: Farms in the Cauca Department of Colombia

Elevation: 4,000 - 6,000 feet

Coffee Variety: Caturra and Typica

Processing Method: Washed

Tasting Notes: Sweet walnut flavor with delicate cocoa notes and high herbal aromas

Flavor Intensity: Medium

Body: Medium

Acidity: Medium

A coffee discovery worthy of legend.

The Cauca highlands of Colombia were settled by Spanish conquistadors searching for El Dorado, the fabled City of Gold. While these explorers did not find what they were looking for, the area is no less rich for the treasures it does have — starting with this coffee.

Soaked by annual rainfall of 80 inches or more and blossoming in the cool climate of the northern Andes, this



Performance Issues with Mobile Sites

- Download and Hide
- Download and Shrink
- Excess DOM



Excess DOM

- If the same HTML is returned to all browsers, then mobile sites will get HTML that is far more complicated than necessary
- More complicated DOM lead to higher memory consumption and a slower site



Responsive Image Strategies

- We do not always get the luxury of selecting whether or not to load an image. Sometimes we have to load an image
- For Example:
 - Logo images



Responsive Image Strategies

- There currently three strategies for handling images:
 - Fighting the Browser
 - Resignation
 - Going to the server

Responsive Image Strategies –

Fighting the Browser

- Most front-end solutions attempt to fight the browser. They try their best to switch which image is loaded before the browser can download the wrong one.
- This is an increasingly difficult task. Browsers want pages to load quickly, so they go to extreme lengths to download images as quickly as possible.
- Of course, this is a good thing— you want your site to load as quickly as possible. It is really only annoying when you want to beat them to it.

Responsive Image Strategies -

Resignation

- This approach admits defeat to the browser. We cannot adequately control which image gets loaded by the Browser.
- So lets think Mobile-Up
- Load the small images first and for bigger screen devices load larger images later.
- Not ideal as this results in the larger screen devices make two or more HTTP requests per image

Responsive Image Strategies –

Going to the Server

- Finally, a few methods use the server and some form of detection to determine which image to load.
- This method does not have to race the Browser, because all the logic is executed before the browser ever sees the HTML.
- This is not a Future Friendly solution, as we are having to maintain information about every single device that might access the site.
 - This is very difficult given the proliferation of devices

Image Scaling

- As an example we will use Sencha.io Src created by James Pearce
- To use it, pick a URL for an image and prefix it with
 - <http://src.sencha.io/> and
 - Some scaling value.

Sencha.io Src

- Consider the following image for the TV show Fringe
 - <http://www.funwallz.com/image/tv-fringe-anna-torv-olivia-dunham-fresh-new-hd-wallpaper-28634.jpg>
- Image at 100%
 - <http://src.sencha.io/http://www.funwallz.com/image/tv-fringe-anna-torv-olivia-dunham-fresh-new-hd-wallpaper-28634.jpg>
- Image scaled to 300px
 - `http://src.sencha.io/300/http://www.funwallz.com/image/tv-fringe-anna-torv-olivia-dunham-fresh-new-hd-wallpaper-28634.jpg`



Sencha.io Src

- Sencha.io SRC also caches the request so that the image is not regenerated every time.

Sencha.io Src – Some Challenges

- Sencha.io Src sizes based on 100% of the screen width not the size of a container in the fluid layout.
- You would have to handle this with some JavaScript – mode coding
- Scaling is based on the entire image not just a portion of the image. When scaling down an image you can lose perspective. You may want to crop the image

Sencha.io Src – Some Challenges

- This is a third party service
 - What if the service provider goes out of business?
 - What if the service URL changes?
 - What if the Terms of Condition Change?
 - What if the image is Classified or considered business critical?



Adaptive Images

- Adaptive Images detects a site's visitor's screen size and automatically creates, caches, and delivers device appropriate re-scaled versions of your web page's embedded HTML images.
- No mark-up changes needed. It is intended for use with Responsive Designs and to be combined with Fluid Image techniques.



Adaptive Images

- Adaptive images are an excellent solution for an existing site where you may not have time to restructure your markup or code.
- Generated images are also cached to help eliminate repeated image regeneration

Adaptive Images

- Getting it up and running is a simple three-step process:
 1. Place the .htaccess and adaptive-images.php files that are included in the download into your root folder.
 2. Create an ai-cache folder and grant it write permissions.
 3. Add the following line of JavaScript to the head of your document:

```
<script>
    document.cookie = `resolution = ` +
    Math.max(screen.width, screen.height) + `;
    path = / `;
</script >
```

Adaptive Images

- Some challenges with Adaptive Images include
 - A single URL for an image actually refers to multiple images. The web server dynamically selects, or generates, the appropriate size image. This will cause issues with CDNs as they use the URL for cache keys.
 - Now we no longer have a 1:1 relationship between a URL and the image



Chicken and Egg Dilemma

- To improve performance, browsers want to be able to be able to download images as soon as possible, before the page layout is known
- Developers, on the other hand, rely on knowledge on the page layout to be able to determine which image to load

Responsive Images – Best

Approach

- Ultimately there is no definitive solution currently exist for responsive images
 - Image Resizing allows you to create appropriate sized images for your layout, but you will be dependent on some external service
 - Adaptive Images allow you to create, on the server, images sized appropriate for the device so the HTML is simpler



Web Fonts

- Webfonts are a font format with a specific license that permits web designers and developers to use real typography online without losing the advantages of live text — dynamic, searchable, accessible content
- Until recently, typography on the Web was very limited font-wise. Most websites could only display text using the small selection of common system fonts already installed on the user's computer.



Web Fonts

- Designers who wanted to introduce some flair, character or individuality to their work would instead need to create raster images — with very limited options for size and placement and serious implications for those all-important Google search results — or use substitution software like Flash or Javascript to render their choice of font, even though that software itself was not universally available or taken up by endusers.
- Neither solution was a credible option for extended body text either, with the substitution techniques in particular requiring a great deal of processor power to render large amounts of text.



Web Fonts

- `@font-face`. The CSS declaration that web designers and programmers use to specify a font. That font file will either be saved in a directory on your host server much like the images are, or it could be hosted by a specialist third party service provider. It's been around since the late '90s, and in a way was the first, if not the simplest step on the road to webfonts.

@font-face file types

- We need the following file types in order to have the full 92+% support.
 - Internet Explorer (all versions): **EOT**
 - Safari (3.2+): **TTF / OTF**
 - iPhone (3.1) **SVG**
 - Chrome (all versions): **SVG** (TTF/OTF added 25th Jan 2010)
 - Firefox (3.5+): **TTF/OTF** (.WOFF added 3.6)
 - Opera (10+) **TTF/OTF**
- **So, .eot + .ttf / .otf + svg + woff = best support possible.**

@font-face Example

```
@font-face {  
    font-family: "Blooming Grove";  
    src: url('/fonts/examples/bgrove.ttf')  
        format('truetype'),  
        url('/fonts/examples/  
bgrove.otf')  
        format('opentype');  
}  
  
.example {  
    font-family: "Blooming Grove";  
    font-size: 1.5em;  
}
```



Media Queries

- You can include the `@font-face` in the Media Query rules so you can select appropriate font families for specific devices.



High-Resolutions Displays

- Creating images for high-resolution displays means creating larger images, which in turn means larger file size
- You do not want to pass these larger images to screens that do not need them.
- So how can we control which images get loaded on a device
 - Media Queries

High-Resolutions Displays

- We use Media Queries and specifically the resolution feature
 - `min-resolution` and
 - `-webkit-min-device-pixel-ratio` for WebKit based browsers

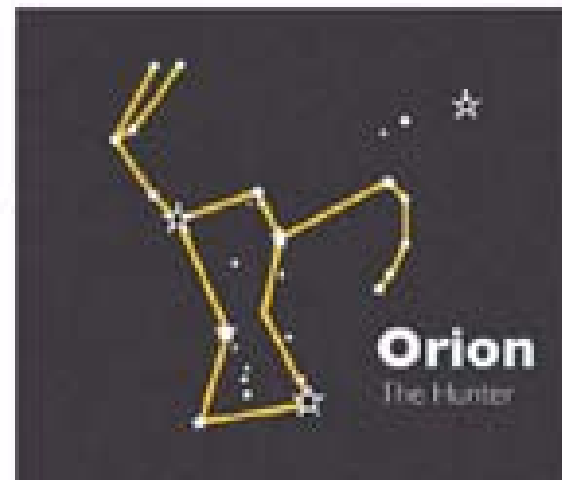
```
@media only screen and
      (-webkit-min-device=pixel-ration: 2),
      only screen and (min-resolution: 2dppx){
...
}
```



Scalable Vector Graphics (SVG)

- Another solution to scaling images is to choose the right technology
- Scalable Vector Graphic images are vector images that can scale without
 - Increasing the file size
 - Losing detail

Example with SVG





Scalable Vector Graphics (SVG)

- There are two main issues standing in the way of SVG
 - Lack of Browser support and
 - Lack of tool support

Scalable Vector Graphics (SVG)

– Browser Support

- Internet Explorer 8 and lower do not support SVG
- Default Browser on Android does not support SVG
- Other Browsers vary in the level of support

Scalable Vector Graphics (SVG)

– Tool Support

- The most popular tool used for creative design is Photoshop.
 - Photoshop is not designed to support SVG
- Adobe Illustrator does support SVG but that is a new (and costly license) and retraining on a new tool
 - Consider what it would be like if I made the Java people write in C# or
 - The C# people write in Java
 - That is the degree of challenge

Other Fixed-Width Assets - Video

- In HTML5 I can leverage some of the new features

```
video{  
    max-width: 100%;  
    height:    auto;  
}
```

Other Fixed-Width Assets -

Video

- Many sites pull their videos from a third party
 - YouTube
 - Vimeo

Using an iFrame

- In this case setting `max-width` to `100%` and `height` to `auto` means that `height` retains its original value but `width` scales.
 - This breaks the aspect ratio