响应式网页设计流程及弹性图片研究

11302010003 宋波

摘要：响应式网页设计是当今移动互联网时代重要的网页技术，它能使用户在不同设备上（如智能手机，平板电脑，台式电脑，甚至未来的手表，眼镜等），浏览网页时获得最佳的用户体验。而在响应式网页设计的众多细节技术中，弹性图片是重要的一环。本文将首先介绍响应式网页设计的基本要素及流程，之后对弹性图片进行研究分析。

**简介**

随着科学技术的不断发展，人们浏览网页不仅仅局限在传统的台式电脑上了，用户可以在越来越多的设备上进行网页浏览，例如手机，平板电脑，超级本，台式电脑等。同种设备的分辨率也不尽相同，例如现在台式电脑的屏幕有小有大，分辨率从1024\*768至1920\*1080不等，五花八门的手机、平板电脑的分辨率也基本上不同，再加上新产生的视网膜屏幕越来越多地运用到手机，平板电脑，笔记本电脑上，越来越多的问题由此而生。传统的固定布局的网页设计的局限性越来越明显，因此，一种能够根据不同屏幕，不同分辨率，采取不同的布局，不同的字体，不同的图片等进行适应的响应式网页设计技术应运而生。在这之中，由于图片相对较高的大小，以及特殊的缩放特性，弹性图片成为了响应式网页设计的重点之一。本文将首先通过响应式网页设计的几个关键技术，如媒体查询、流体布局、列布局等来论述该设计的基本流程。之后，从不同角度如载入速度，视网膜显示等方面分析论述弹性图片技术。

媒体查询

响应式网页设计的基础技术便是媒体查询（Media Queries）。媒体查询的前身是CSS2中的媒体类型(Media Type)，可以通过它对不同的设备指定不同的样式。W3C总共列出了10种媒体类型，常见的有all(全部)，screen（屏幕），print（打印及打印预览）等。可以看出，这些媒体类型的种类较少，局限性非常大，无法适应种类繁多的移动设备。而媒体查询在CSS3中的出现有效地弥补了这一缺点。

在CSS3中，可以通过媒体查询设置不同类型的媒体条件，并根据对应的条件，给相应符合条件的媒体调用相对应的样式表。媒体查询可以通过HTML中的link标签引入，或是在CSS中直接引入，如下所示：

1．link方式引入

<link rel="stylesheet" media="screen and (max-width: 980px)" href="middle.css" />

2．@import方式引入

@media screen and (max-width: 980px) {

选择器 {

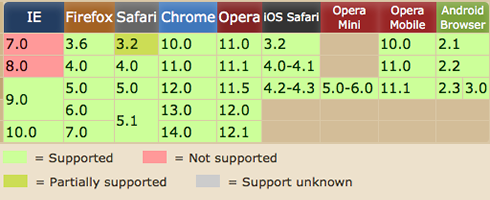
属性：属性值；

}

}

以上的媒体查询的内容由三部分组成：

1. 媒体类型：如screen，与CSS2中的相同。
2. 媒体特性：以(属性：值)的形式指定所选媒体的特性。如(max-width: 980px)指定了最大宽度不超过980px的媒体。常用的属性由图表1所示。
3. 关键字：用于组合不同的媒体特性，有and，not，only三种。and表示需满足两种组合的媒体特性，not表示排除某种制定的媒体类型，only指定某种特定的媒体类型，可以用来排除不支持媒体查询的浏览器。



图表 2 浏览器兼容性

图表 1 常见属性

此外，由于媒体查询是CSS3引入的特性，浏览器的兼容性存在一些问题，具体兼容性情况由图表2所示。

布局

通过媒体查询，我们可以根据不同的屏幕来指定不同的布局来达到优化用户体验的效果。在什么样的屏幕下用什么样的布局是一个很复杂的问题，现有的解决方案有非常多，在这里我们选取一种常见的流体布局加上列布局的方案对基本的响应式网页设计布局技术进行阐述。

当屏幕的分辨率宽度大于980px时，屏幕中间的主容器宽度为980px；当小于980px时，布局不再采用固定布局来而将采用流体宽度布局来替代；如果小于650px，边栏的宽度和主内容的宽度也变成全屏，采用列布局进行显示。典型样例如图表三所示。

图表 3 常见响应式网页布局

为强调重点，简化冗余部分，我们尽将整个pagewarp（页面容器）分为header（包括导航栏，logo，描述等），content(主内容区)，sidebar（边栏）。其简化过的HTML代码如下所示：

<div id="pagewrap">

<header id="header">

<hgroup>

<h1 id="logo">Demo</h1>

<h2 id=" description">Site Description</h2>

</hgroup>

<nav>

<ul id="nav"> … </ul>

</nav>

</header>

<div id="content"> … </div>

<aside id="sidebar"> … </aside>

</div>

大于980px时的简化CSS代码如下所示，pagewrap宽度设为980px，header高度设为160px，content宽度设为600px并将其设置为左浮动，sidebar设为280px并将其设置为右浮动。

#pagewrap {

width: 980px;

margin: 0 auto;

}

#header {

height: 160px;

}

#content {

width: 600px;

float: left;

}

#sidebar {

width: 280px;

float: right;

}

}

当页面宽度小于980px时，显然不能再以px进行固定布局，所以将各宽度以相对百分比来替代，代码如下所示：

@media screen and (max-width: 980px) {

#pagewrap {

width: 95%;

}

#content {

width: 60%;

padding: 3% 4%;

}

#sidebar {

width: 30%;

padding: 8% 7%;

margin-bottom: 10px;

}

}

而当页面宽度小于650px时，需要将页面布局改为列布局以达到良好的显示效果，基本的方法为奖定位由float改为static，宽度根据情况改为100%或是auto，代码如下所示：

@media screen and (max-width: 650px) {

#header {

height: auto;

}

#nav {

position: static;

}

#logo {

margin: 15px 100px 5px 0;

position: static;

}

#description {

margin: 0 0 15px;

position: static;

}

#content {

width: auto;

float: none;

margin: 20px 0;

}

#sidebar {

width: 100%;

float: none;

margin: 0;

}

}

弹性图片

图片是网页中的重要组成部分，在之前的响应式网页设计流程中，只涉及到了图片及文字的部分。而对于图片，我们需要根据屏幕的情况，设置成相对应的大小来完整地展示出图片样貌，最基本的解决方式如下：

img {max-width:100%;}

该方法使图片在不同的情况下完整地显示出来，但是由于图片大小较大，缩放后显示效果差异较大等特性，该方法有诸多问题，如下所示：

1. 对于移动设备来说，所需的图片较小，如使用桌面电脑的图片，大小过大，所需流量过多，也减慢了反应时间，降低用户体验。
2. 如使用适应移动端的小图片，对于桌面电脑等较大屏幕设备来说，显示效果较差
3. 视网膜屏幕的引进，使得屏幕的大小与分辨率的大小不成正比，如将用于大屏幕的图片用在小屏幕等分辨率的设备，显示效果差

对于这些问题，有着多种的解决方案，下面将从客户端的CSS,HTML,JAVASCRIPT和服务器端等四个方面进行研究分析。

CSS的弹性图片

媒体查询方式

在CSS3中，实现弹性图片的方法之一是前文所述的媒体查询技术。使用前文所述的媒体特性如max-width等可以根据不同的屏幕尺寸选择不同的图片，但这不能解决视网膜屏幕的问题。

因此，在 W3C Media Queries Level 3中引入了媒体特性resolution，并派生出两个媒体特性min-resolution以及max-resolution。顾名思义，可以根据这两种媒体特性针对不同的分辨率结合屏幕大小类型选择合适的图片，代码如下所示：

@media only screen and (-o-min-device-pixel-ratio: 2/1),

only screen and (min--moz-device-pixel-ratio: 2),

only screen and (-webkit-min-device-pixel-ratio: 2),

only screen and (min-resolution: 240dpi),

only screen and (min-resolution: 2dppx) { … }

其中dppx为level 3中新引入的单位，与pixel-ratio一样，用于视网膜显示。

这种方法具有两种问题，其一是对于“非正方形像素”的设备，min-resolution查询中指定的值必须与最稀疏尺寸进行比较，max-resolution查询中必须与最密集尺寸进行比较，对于无前缀的resolution，从不查询非正方形像素。其二是对在不同的浏览器内核中，对于pixel-ratio的定义不一致。由此可见，使用媒体查询实现弹性图片较为麻烦，可维护性不高。

Image-set方式

在最新的W3C CSS Image Values and Replaced Content Module Level 4中引入了原生语法image-set，根据不同的分辨率模式选择图片，语法如下所示：

<image-set> = image-set( [ <image-set-decl>, ]\* [ <image-set-decl> | <color>] )

<image-set-decl> = [ <image> | <string> ] <resolution>

具体结合实例为：

background-image: -webkit-image-set(

url(…/normal.png) 1x,url(…/retina.png) 2x);/\* Retina \*/

image-set方法相对媒体查询方法来说有两种优点：

1. 当使用媒体查询时，各种不同分辨率下的图片很可能散落在几十行代码里，而image-set将其聚集在了一起，提高了代码的可维护性。
2. 媒体查询告诉浏览器在不同的场景下使用哪种图片，而image-set方法本质上是给浏览器提供了选择。虽然在当下这两种方式的实际效果是一样的，但在理想的情况下会有巨大的差别。在高分辨率的视网膜屏幕下，媒体查询不论在任何情况下都会匹配较大的图片，因为它的限定条件已经限制住了。而image-set是给浏览器提供了选择，浏览器可以在一些情况下，例如在低网速的情况下选择1x的普通图片，这也是弹性图片的最终目标之一。

当然，image-set方式在当今也有着极大的局限性：

1. 仅支持backgroud-image属性。
2. 目前仅是webkit浏览器的私有属性，不过由于大多数具有视网膜显示的设备都使用的是webkit内核的，所以影响范围不大。
3. 仅能用url()方式取值，不支持color，gradient()等方法。

总的来说，在当今CSS3的时代，实现弹性图片的实际方法是使用image-set和媒体查询方法并用。但在CSS4或更远的未来，image-set方式也许会成为主流，配合上更先进的浏览器，使弹性图片变得更加智能化。

HTML5的弹性图片

CSS有媒体查询方式和image-set方式实现背景图片等弹性图片的问题，但html中的<img>标签的弹性图片问题无法由此解决。在HTML5中引入了<picture>标签来解决这一问题，这一标签仍是一个草案，其规范不断在变动中，所以选取2013年4月最新的规范示例进行阐述：

<picture width="500" height="500">

<source media="(min-width: 45em)" srcset="large-1.jpg 1x, large-2.jpg 2x">

<source media="(min-width: 18em)" srcset="med-1.jpg 1x, med-2.jpg 2x">

<source srcset="small-1.jpg 1x, small-2.jpg 2x">

<img src="small-1.jpg" alt="">

<p>Accessible text</p>

</picture>

<picture>标签中可以添加<source>标签来提供不同的选择。其中，media属性前文所述的<link>标签中的media属性一样，提供了媒体查询的功能选择屏幕的大小，而srcset属性则类似前文所述的image-set为视网膜显示的设备提供更好的图片，并且srcset具有fallback特性，如果一个视网膜显示的设备请求一张较大的2x图片出现问题比如资源无法加载等，srcset会自动跳过该选择，使用下一张备选图片。

由于该标签仍在草案中，所以浏览器的支持是非常差的，就目前来说不具备应用性。但与image-set一样，该标签为未来根据用户请求图片的网速来选择相应的图片等更加智能的技术提供了基础。

JAVASCRIPT的弹性图片

运用客户端脚本实现弹性图片最基础的原生方法就是检测屏幕或可见区域的大小（如window.screen.height，window.screen.width检测屏幕分辨率的高和宽）来动态地改变<img>标签中的src属性来加载适应屏幕的图片。

这种方式显然有许多问题：

1. 每适用一个<img>标签就需要打量的条件语句来进行判断，会产生非常多的冗余代码，可维护性较差。
2. 如果用户禁用客户端脚本将无法实现弹性图片。

由于第一点问题的存在，派生出了许多第三方的JAVASCRIPT脚本或是JQUERY的插件，在实际的应用中也基本上是运用这些脚本，在这里选取比较著名的有JQUERY PICTURE插件进行阐述。

这一插件基本支持所有主流浏览器，插件大小较小，使用也较为方便，其脚本代码仅需一行，如下所示：

$('figure, picture').picture();

对应的html代码为：

<figure data-media="picture1.png" data-media440="picture2.png" data-media600="picture3.png" title="A Half Brained Idea">

<a href="">

<noscript>

<img src="picture3.png" alt="demo">

</noscript>

</a>

</figure>

在<figure>标签中使用data属性设定不同宽度的图片。可以看出，插件的使用使得用户的脚本变得简洁，而HTML中的部分更趋近于前文所述的HTML中弹性图片的方法。该方法的优点是具体实现由插件完成，所以浏览器兼容性上更为出色，但由于网上插件众多，其可靠性需要程序员谨慎甄别。

服务器端的弹性图片

在服务器端实现弹性图片的基本思路为检测发送请求的浏览器信息，输出不同的img地址，通常需要与JAVASCRIPT配合使用。

在这里选取最常见的Adaptive Picture技术进行阐述，该技术需要将将配置文件和PHP脚本放入Apache服务器里网站的根目录中，还需在客户端的JAVASCRIPT里加入包含屏幕分辨率的cookie信息，如下所示：

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

当HTML中遇到<img>标签并向服务器请求图片时，cookie信息也会一并发送至服务器，Apache服务器收到后会查看配置文件并发现对图片的请求，服务器再转送至Adaptive Picture的PHP脚本文件，该脚本文件会查看所请求图片的信息并与cookie中的屏幕信息进行比较，如果图片过大会进行处理后再发送回客户端。

这种方式的优缺点也显而易见，前端的HTML,CSS,JS部分几乎没有额外的工作量，但与此同时，服务器的压力增大许多，对于一个图片较多，访问量较大的网站来说，这部分的比例是非常大的。

分析与总结

由于响应式网页设计出现时间不长，对于响应式图片技术的研究更可以说是刚开始起步，所以现有的解决方案五花八门，极不统一，从前端HTML到后端的服务器，似乎都可以解决这一问题，又似乎都有明显的缺陷。那么，什么才是正确的道路？

首先来看服务器端和客户端脚本的解决方案。两个非常严重的问题使服务器端的解决方案的局限性很大，无法成为标准，其一是前文所述的服务器压力增大的问题，另一个则是与响应式网页设计将前端与后端分离的原始初衷相违背。而客户端脚本的解决方案由于其可被用户禁用的特性再加上它的复杂性就无法成为标准。

而现实情况则是这两种解决方式都非常有市场，造成这一现象的重要原因之一就是程序员可以在设计网页初期配置好之后可以方便地使用而不用每次都要解决浏览器兼容性等方面的问题，这也是HTML5的picture标签以及image-set的现有问题。

我认为，未来的响应式网页设计的趋势是提供选择而不是固定在某些情况下的简单判断，让浏览器结合网速，缓存等环境信息请求到它认为在当前情况下的最好选择。而picture标签和image-set是实现这种目标的基础。我们可以期待，在HTML5的正式版和CSS4或者更远的未来，被更多浏览器所支持，功能健全，无需配置的标准picture标签和image-set将会成为实现弹性图片的主流。

参考文献

[1] Media Queries Level 3 Editor's Draft 13 April 2012 <http://w3c-test.org/csswg/mediaqueries3/#resolution>

[2] CSS Image Values and Replaced Content Module Level 4 Editor's Draft 11 April 2013 <http://dev.w3.org/csswg/css-images/>

[3] The picture element An HTML extension for adaptive images W3C Working Draft 26 February 2013 <http://www.w3.org/TR/html-picture-element/>

[4] Safari 6 and Chrome 21 add image-set to support retina images <http://blog.cloudfour.com/safari-6-and-chrome-21-add-image-set-to-support-retina-images/>

[5] Adaptive Images for Responsive Designs <http://24ways.org/2011/adaptive-images-for-responsive-designs/>

[6] The state of responsive images <http://www.netmagazine.com/features/state-responsive-images>

[7] Responsive设计和CSS3 Media Queries的结合 <http://www.w3cplus.com/css3/responsive-design-with-css3-media-queries>