Proceedings,
The 2013 ASEE North Midwest Section Conference,
October 17-18, 2013, Fargo, North Dakota.

ASEE-NMWSC2013-0057

Adaptive Mobile Web Pages Using HTML5 and CSS for Engineering Faculty

Wen-Chen Hu
Department of Computer Science
University of North Dakota
Grand Forks, ND 58202-9015
wenchen@cs.und.edu

Naima Kaabouch
Department of Electrical Engineering
University of North Dakota
Grand Forks, ND 58202-7165
naima.kaabouch@engr.und.edu

Hung-Jen Yang
Department of Industrial Technology Education
National Kaohsiung Normal University
Kaohsiung City, Taiwan 80201
hjyang@nknucc.nknu.edu.tw

Hongyu Guo
Department of Computer Science
University of Houston – Victoria
Victoria, TX 77901
guoh@uhv.edu

Abstract

Many engineering faculty members have their class materials like lecture slides and assignments posted on the Internet, so students can easily access them anytime. Nowadays students also like to access the online class materials anywhere via their mobile devices such as smartphones or tablet computers. However, traditional Web pages are distorted or become awkward to use when they are displayed on devices. Additionally, mobile features of handheld devices such as geolocation information, which could be useful for engineering education, are usually ignored. The introduction of HTML5 and the newest CSS have greatly solved this problem. This paper introduces the mobile features of HTML5 and CSS to engineering faculty for building adaptive mobile class Web pages.

Introduction

Using Web to deliver engineering education has been a trend. Many engineering students are used to do various things on the Web, e.g., printing class slides and assignments, checking the class announcements, posting questions, interactively discussing issues with fellow students, etc. On the other hand, IDC, a market research firm, predicted that more users would access the Internet wirelessly via a mobile device compared to the users using wired connection by 2015[4]. Among the wireless devices, many of them are small-screen handheld devices like smartphones. Nowadays students also like to access the online class materials anywhere via their mobile handheld devices such as smartphones. However, traditional Web pages are distorted or become awkward to use when they are displayed on devices. In order to solve this problem, engineering

faculty had to develop two versions of a Web page in the past, one for desktops or laptops and another one for devices. It is time and effort consuming and the consistency is a major issue.

Advanced mobile browsers such as Opera Mini[8,11] are able to adapt themselves to various Web pages as shown in Figure 1.a. However, not all mobile browsers have this kind of features; e.g., the Android Browser[2] shown in Figure 1.b does not present the page as good as the Opera Mini does. The problem is greatly relieved since the introduction of HTML5 (HyperText Markup Language) and the newest CSS (Cascading Style Sheets). This article tries to help engineering educators to develop their class Web pages automatically adaptive for both desktop or laptop computers and mobile handheld devices by using HTML5 and CSS. Additionally, mobile features of HTML5 and CSS may be useful for engineering education and some of them will be discussed.

(a) (b)

Figure 1: A page displayed on two mobile browsers: (a) Opera Mini-Simulator and (b) Android Emulator

Organization of the rest of this paper is as follows. The background information, mobile browsers, is introduced in Section 2. Sections 3 and 4 show the mobile features of HTML5 and CSS, respectively. The HTML5 features include (i) multiple style sheets, (ii) viewport, (iii) geolocation, and (iv) other mobile features, and the CSS features include (i) relative sizes vs. absolute sizes and (ii) CSS float. A summary of this paper is given in the final section.

Mobile Browsers

Mobile browsers are miniaturized versions of desktop browsers such as Mozilla Firefox and Microsoft Internet Explorer. They provide the graphical user interfaces that enable mobile users to interact with mobile applications.

Client Computers or Devices

Desktop and laptop computers are on the client-side of electronic commerce systems, whereas mobile handheld devices are for mobile commerce systems. An Internet-enabled mobile handheld device is a small general-purpose, programmable, battery-powered computer that is

capable of handling the front end of mobile commerce applications and can be operated comfortably while being held in one hand. It is the device via which mobile users interact directly with mobile commerce applications. The differences between these two client machines are given in Table 1[4]. There are other kinds of computers such as tablet computers, which are a special kind of PCs.

	Desktop and Laptop Computers	**Mobile Handheld Devices**
Browser	Desktop browsers	Mobile browsers
Functions	Full	Limited
Major Input Methods	Keyboards and mice	Stylus and soft keyboards
Major Output Methods	Screens and printers	Screens
Mobility	Low	High
Networking	Wired	Wireless and mobile
Transmission Bandwidth	High	Low
Power Supply	Electrical outlets	Batteries
Screen	Normal	Small
Size	Desktop	Handheld
Weight	Normal	Light

Table 1: Differences between desktop & laptop computers and handheld devices

Major Mobile Browsers

A number of mobile browsers are currently available. Five popular ones are (i) Apple's Safari, (ii) Android Browser, (iii) Opera Mini, (iv) Google's Chrome, and (v) Microsoft's Internet Explorer[9]. Table 2 compares these five mobile browsers. Some companies also provide mobile browser emulators/simulators such as Opera Mini Simulator that enable developers to test their products on desktop computers because small devices are not convenient for mobile application development.

	Safari	**Android Browser**	**Opera Mini**	**Chrome**	**Internet Explorer**
Developer	Apple	Google	Opera	Google	Microsoft
Mobile Traffic as of April 3, 2013	1st (61.79%)	2nd (21.86%)	3rd (8.4%)	4th (2.43%)	5th (1.99%)
Rendering Engine	WebKit	WebKit	Presto	WebKit	Trident
Host Operating Systems	OS X and iOS	Android	Android, Windows Phone, iOS, etc.	Android and iOS	Windows Phone

Latest Release	6.0	4.0	7	28.0	10.0
License	Freeware	Freeware	Proprietary	Freeware	Proprietary

Table 2: A comparison of the five leading mobile browsers

Mobile Browser Features

Due to the limited resources of handheld devices, mobile browsers differ from traditional desktop browsers in the following ways: (i) smaller windows, (ii) smaller footprints, and (iii) fewer functions and multimedia features. However, there are some features that are unique for mobile browsers. For example, ACCESS' NetFront Browser[1] includes Smart-Fit Rendering technology, which intelligently adapts standard Web pages to fit the screen width of any mobile device enabling an intuitive and rapid vertical scrolling process, without degrading the quality or usability of the pages being browsed. Concretely, the following process is performed as follows: (i) images larger than the screen width are scaled down to fit the screen width and (ii) tables larger than the screen width are split and laid out vertically as shown in Figure 2.

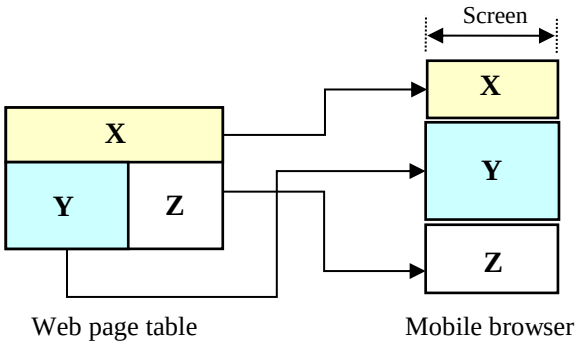


Figure 2: A Web page table split by ACCESS' NetFront Browser

HTML5 for Mobile Web Pages

There are a couple of versions of HTML (HyperText Markup Language) such as (i) HTML, each version of HTML adding more tags, (ii) HTML 4.0, trying to clean up the standard by marking some of the tags as deprecated, and (iii) XHTML (eXtensible HTML), a reformulation of HTML 4.0 in XML. Though HTML was well established and defined for most of the requirements of static Web pages, there were several areas that were missing from the previous HTML specifications. Without those missing specifications, each browser or software had its unique way of specifying certain objects. HTML5[15] is therefore proposed to improve and standardize the HTML specifications. Key mobile features of HTML5 including (i) media types, (ii) viewport, (iii) geolocation APIs, and (v) other mobile features are discussed next.

Multiple Style Sheets

One simple way to solve the above problem of ill presentations of mobile Web pages is to provide each page with two style sheets, one for desktops or laptops and another one for devices. There are server- and client- side approaches to find the type of client browsers. For server-side approach[10], the environment variable, `HTTP_USER_AGENT`, contains the information of client browsers such as

Mozilla/5.0 (Windows NT 6.1; WOW64; rv:22.0) Gecko/20100101 Firefox/22.0

Opera/9.80 (J2ME/MIDP; Opera Mini/7.1.32422/30.3558; U; en) Presto/2.8.119 Version/11.10 or

Mozilla/5.0 (linux; U; Android 2.3.4; en-us; google_sdk Build/GINGERBREAD) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1

Based on the information, the server then provides an appropriate style sheet for the browser. However, this approach has the disadvantage of slowness because it requires the responses from servers. On the other hand, the client-side approach is faster compared to the server-side one. Without needing server-side scripts, the `media` attribute of the tag `link` specifies how a document is to be presented on different media: computer screen, paper, mobile browser, etc[14]. Two of the most common media types are

- *Screen*, which is intended primarily for color computer screens, and
- *Handheld*, which is intended for handheld devices (typically small screen, limited bandwidth).

For example, the following two lines can be added to a Web page:

```
<link rel="stylesheet" href="site.css"   media="screen"   />
<link rel="stylesheet" href="mobile.css" media="handheld" />
```

The browsers will select the corresponding style sheet based on their types. For example, if the browser is hosted by a computer, the style sheet `site.css` will be used. On the other hand, if the browser is hosted by a smartphone, the style sheet `mobile.css` will be used. Figure 3 shows the examples with or without using the multiple style sheets.

(a) (b)

Figure 3: A page (a) without media types specified and (b) added a media type: handheld

Viewport

A Web page is rendered in a virtual "window" (the viewport), where users can pan and zoom to see different areas of the page. The viewport size and the initial zoom factor are introduced in the HTML meta tag[7,16]. The partial syntax is given as follows:

```
<meta name="viewport" content="width=device-width, initial-
scale=1, maximum-scale=1">
```

where

- The `width` property specifies the size of the viewport.
- The `initial-scale` property controls the zoom level when the page is first loaded.
- The `maximum-scale`, `minimum-scale`, and `user-scalable` properties control how users are allowed to zoom the page in or out.

For example,

```
<meta name="viewport" content="width=400, initial-scale=1">
```

yields a width of 400px and zoom level is 1 when the page is first loaded. Figure 4 shows two screen shots with and without using the following viewport command:

```
<meta name="viewport" content="width=320">
```

(a) (b)

Figure 4: (a) A page without viewport specified and (b) the page added a viewport and width

Geolocation APIs

HTML5 Geolocation[12] is dense and powerful. It provides several functions of managing the geographical location of a person or device. Table 3 shows the five interfaces provided by the HTML5 Geolocation.

Interfaces of HTML5 Geolocation	
Interface	**Description**
`Geolocation`	The `Geolocation` is the main object in the API and is used to obtain the location information of devices and users.
`PositionOptions`	The `getCurrentPosition` and `watchPosition` methods accept `PositionOptions` objects as their third argument.
`Position`	Same as `Coordinates`
`Coordinates`	The coordinates of a geographical location
`PositionError`	The error information if the geographical information cannot be obtained

Table 3: HTML5 Geolocation interfaces

`Geolocation` is the most important interface of the HTML5 Geolocation. Table 4 shows the three methods, `clearWatch`, `getCurrentPosition`, and `watchPosition`, provided by the `Geolocation` interface.

Methods of `Geolocation` Interface	
Method	**Description**
`clearWatch`	Stop the watch process identified by the `watchID` argument.
`getCurrentPosition`	Get the current location in latitude and longitude coordinates.
`watchPosition`	Continue to monitor the position and invoke callback function when position changes.

Table 4: Methods of HTML5 `Geolocation` interface

The `getCurrentPosition` method returns the current position of a device. The `watchPosition` method monitors the device's position. When the position changes, the callback function must be invoked with a new `Position` object, reflecting the current location of the device. Table 5 gives the attributes of the `Postion` interface.

`Position` Interface			
Attribute	**Value**	**Unit**	**Description**
`coords.latitude`	double	degrees	The latitude of postion
`coords.longitude`	double	degrees	The longitude of position
`coords.accuracy`	double or null	meters	The accuracy of position
`coords.altitude`	double or null	meters	The altitude above the mean sea level
`coords.altitudeAccuracy`	double or null	meters	The altitude accuracy of position
`coords.heading`	double or null	degrees clockwise	The heading from the true north

`coords.speed`	double or null	meters/second	The speed of the device
`Timestamp`	DOMTimeStamp	like the Date object	The date/time of the response

Table 5: Attributes of the `Position` interface

The following example briefly explains how to implement an LBS (location-based service) application by using HTML5 Geolocation and Google Maps APIs[3], so the proposed LBS application is made simple on purpose. It is to find classmates, who are constantly moving. For example, an assignment is due tomorrow and a student is urgent to find a classmate to help solving his/her assignment problems. The proposed LBS application keeps track of the whereabouts of the classmates, who have signed up the application, by using the methods `watchPosition` and `clearWatch` of HTML5 Geolocation. Once the classmates' locations are located, the direction between the student and the selected classmate is then given by using Google Maps APIs. Program 1[5] is used by the server to send a map with a direction to the client by using the PHP command `echo`. It uses a couple of Google Maps APIs to draw a direction on a map:

- The class `google.maps.DirectionsService` computes directions between two or more places.
- The class `google.maps.DirectionsRenderer` renders directions retrieved in the form of a `DirectionsResult` object retrieved from the `DirectionsService`.
- The method `google.maps.DirectionsService.route` initiates a direction request to the `DirectionsService`. It requires passing a callback function, which executes upon completion of the service request.

```
<?php
echo  '<!DOCTYPE HTML>
<html>
 <body onLoad="initialize( )">
  <div id="map canvas" style="width:96%;height:400px"></div>

  <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false">
  </script>
  <script>
   var  directionsDisplay;
   var  directionsService = new google.maps.DirectionsService( );
   var  map;

   function  initialize( ) {
    directionsDisplay = new google.maps.DirectionsRenderer( );
    var  GF = new google.maps.LatLng( 47.9252569, -97.032855 );   // Changeable
    var  mapOptions = {
     zoom: 12,
     mapTypeId: google.maps.MapTypeId.ROADMAP,
     center: GF
    }
    map = new google.maps.Map(
     document.getElementById('map_canvas'), mapOptions );
    directionsDisplay.setMap( map );
    calcRoute( );
   }

   function calcRoute( ) {
```

```
    var  request = {
     origin: "the ralph, grand forks, nd",                        // Changeable
     destination: "streibel hall, grand forks, nd",               // Changeable
     travelMode: google.maps.DirectionsTravelMode.WALKING
    };
    directionsService.route( request, function( response, status ) {
     if ( status == google.maps.DirectionsStatus.OK )
      directionsDisplay.setDirections( response );
    } );
   }
  </script>
 </body>
</html>';
?>
```

Program 1: Drawing a direction on a map

Figure 5 shows an example of the Program 1 execution results. The origin and destination are "the ralph, grand forks, nd" and "streibel hall, grand forks, nd," respectively in the Program 1. Instead of text locations, the locations could use latitude and longitude, too. Another map can be easily generated by replacing the origin and destination.

Figure 5: An example of Program 1 execution results

Other HTML5 Mobile Features

Other HTML5 mobile features including (i) offline support, (ii) canvas drawing, (iii) advanced forms, and (iv) video and audio streaming support are introduced next.

Offline support: Internet connection may not be always available, especially when users are on the move. HTML5 supports offline accesses by providing application cache, which stores a Web application for offline browsing. In order to use application cache, an HTML manifest file has to be declared. For example, the manifest file `demo.appcache` is used for the following example:

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
 <body> ... </body>
</html>
```

The manifest file specifies how the file is cached by the browsers. After the browser visits the page first time and caches it, the browser will check the application cache for this page at the subsequent visits.

Canvas drawing: Graphics drawing could be useful for mobile Web applications and be interesting for mobile users. In the past, drawing graphics on the Web required tremendous works and might need the support of plug-ins. The introduction of the HTML5 `<canvas>` element greatly mitigates the problem. The `<canvas>` element is a container for graphics, where the developers can use JavaScript scripts on the fly to draw graphics, such as circles, boxes, and curves, in the container.

Advanced forms: Data entry is always troublesome for handheld devices because of their small sizes. HTML5 provides a couple of features to make form filling much easier, e.g., field validation, auto completion, and data encryption. Additionally, advanced forms can also save time because many of the new features can be handled by client browsers instead of servers.

Video and audio streaming support: In order to play video or audio clips on the Web, different browsers may use different video or audio formats and require the assistance of correct plug-ins. The inconsistency and extra plug-ins make the developers' jobs much difficult. HTML5 tries to solve these problems, so no plug-ins are needed to play video or audio clips if standard formats are applied.

CSS for Mobile Web Pages

Web developers used to put formatting tags such as `<font>` and `<p>` in the HTML scripts. When Web pages became larger and more complicated, the formatting tags made the HTML scripts difficult to read and maintain. CSS (Cascading Style Sheets)[13] was created to relieve the problem by removing the formatting tags from the HTML scripts and storing them in separate CSS files. Like most Web technologies and standards, CSS is updated constantly and the newest CSS is added with the features of facilitating mobile Web page development. This section discusses few CSS features that help creating adaptive Web pages for both desktop or laptop browsers and mobile browsers.

Relative Sizes vs. Absolute Sizes

Moving the Web contents left or right horizontally is usually not a pleasant experience for mobile users. There are two ways, absolute and relative sizes, to specify the size in CSS. In order to display the desired sizes in different browsers, use relative sizes instead of absolute sizes. For the example of Figure 6, instead of using an absolute size as shown in Figure 6.a:

```
table.w1 {width:640px;}
```

use the following relative size to better display the page as shown in Figure 6.b:

```
table.w1 {width:96%;}
```
or
```
table.w1 {width:auto;}
```

(a) (b)

Figure 6: A page using (a) an absolute size and (b) a relative size

Same thing can be applied to images. For the example of Figure 7, instead of using an absolute size as shown in Figure 7.a, use a relative size as below to get a better result in Figure 7.b:

```
img {max-width:96%;}
```

(a) (b)

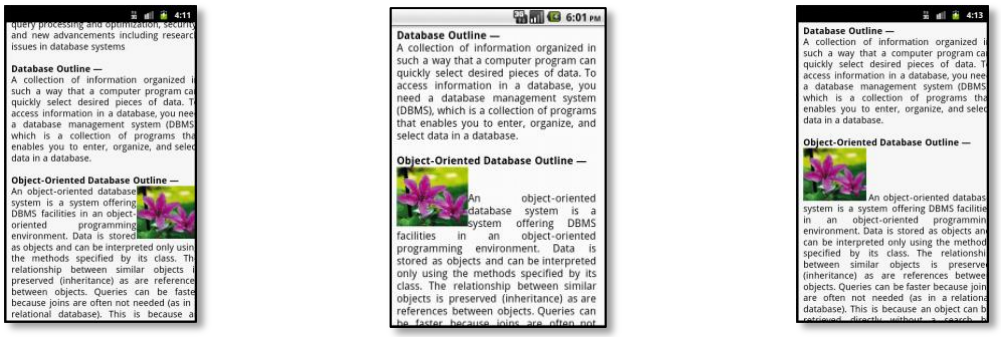
Figure 7: A page using (a) an absolute size and (b) a relative size

Disabling the Float Property

The CSS `float` property is used to push an element to the left or right and allows other elements to wrap around it. An element floated horizontally means it can only be floated left or right, not up or down. Figure 8 shows three screenshots of a page with an image floated to the right, left, or none. The following CSS rules apply the `float` property to all images:

```
img {float:right;},   img {float:left;},   or   img {float:none;}
```

Developers can pick any value as they desire.



(a) (b) (c)

Figure 8: A page with image float being (a) right, (b) left, and (c) none

Summary

The number of mobile Web users will surpass the number of traditional Web users in a short time. In order to keep their teaching methods up-to-date, engineering faculty must adapt themselves for the mobile ages. Instead of creating two versions of their class pages, one for the desktop or laptop computers and another one for mobile handheld devices, engineering faculty can create adaptive class Web pages by using HTML5 and the newest CSS. In addition, some mobile features of HTML5 may be useful for engineering education. For example, geolocation APIs can be used to find users' locations and the offline support allows users to check the cached pages without Internet connection. This paper introduces some of the mobile features of HTML5 and the newest CSS to engineering faculty, so they can apply the features to the construction of their adaptive class pages for both of the traditional and mobile Web users. Since HTML5 and CSS are constantly revised, it is expected more mobile features will be added to them. Interested readers may check the newest HTML5 and CSS specifications at the W3C Web site (http://www.w3.org/) from time to time.

References

1. ACCESS CO., LTD. (n.d.). *NetFront Browser*. Retrieved May 25, 2013, from http://gl.access-company.com/products/browser/browser/
2. Android. (n.d.). *Android Emulator*. Retrieved May 23, 2013, from http://developer.android.com/tools/help/emulator.html
3. Google. (n.d.). *Google Maps API*. Retrieved February 21, 2013, from https://developers.google.com/maps/
4. Hachman, M. (2011, September 12). *IDC: Mobile Internet Use to Pass PCs by 2015*. Retrieved April 6, 2013, from http://www.pcmag.com/article2/0,2817,2392796,00.asp
5. Hu, W.-C., Kaabouch, N., Yang, H.-J., and Wang, W. (2013, April 19-20). Location-based services using HTML5 Geolocation and Google Maps APIs. In *Proceedings of the Midwest Instruction and Computing Symposium (MICS 2013)*, La Crosse, Wisconsin, USA.
6. Hu, W.-C., Yang, H.-J., Chen, L., and Deshmukh, R. (2009, April 17-18). Web content adaptation for Internet-enabled mobile handheld devices. In *Proceedings of the Midwest Instruction and Computing Symposium (MICS 2009)*, Rapid City, South Dakota.
7. Mozilla Developer Network (MDN). (n.d.). *Using the Viewport Meta Tag to Control Layout on Mobile Browser*s. Retrieved July 3, 2013, from https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag

8. Opera Software. (2013). *Opera Mini-Simulator*. Retrieved June 10, 2013, from http://www.opera.com/developer/opera-mini-simulator
9. Paczkowski, J. (2013, April 3). *Safari Still Winning the Mobile Browser War*. Retrieved June 5, 2013, from http://allthingsd.com/20130403/safari-still-winning-the-mobile-browser-war/
10. Wikipedia. (n.d.). *Common Gateway Interface*. Retrieved July 12, 2013, from http://en.wikipedia.org/wiki/Common_Gateway_Interface
11. Wikipedia. (n.d.). *Opera Mini*. Retrieved May 25, 2013, from http://en.wikipedia.org/wiki/Opera_mini
12. World Wide Web Consortium (W3C). (2012, May 10). *Geolocation API Specification*. Retrieved February 6, 2013, from http://www.w3.org/TR/geolocation-API/
13. World Wide Web Consortium (W3C). (2011, June 07). *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. Retrieved June 16, 2013, from http://www.w3.org/TR/CSS2/
14. World Wide Web Consortium (W3C). (2011, June 07). *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification: Media Types*. Retrieved June 16, 2013, from http://www.w3.org/TR/CSS2/media.html
15. World Wide Web Consortium (W3C). (2011, May 25). *HTML5*. Retrieved March 13, 2013, from http://www.w3.org/TR/2011/WD-html5-20110525/
16. World Wide Web Consortium (W3C). (2013, May 20). *CSS Device Adaptation*. Retrieved June 18, 2013, from http://dev.w3.org/csswg/css-device-adapt/#the-atviewport-rule