WA2056 Building HTML5 Based Mobile Web Sites

Student Labs

Web Age Solutions Inc.

Table of Contents

Lab 1 - Using Media Rules to Build Universal Web Site

A universal web site uses a single web application to serve content for both mobile and desktop browsers. One of doing this is to return the same HTML markup but use a different stylesheet to render the content differently in different devices. CSS media rule is a simple way of achieving this. In this lab, we will develop two different style sheets for mobile and desktop screens.

Part 1 - Basic Approach

We will have two CSS files:

1. Mobile CSS file for screen width 480px or less.
2. Desktop CSS file for screen width larger than 480px.

Note: CSS specification also allows you to use the media type handheld for mobile device and screen for desktop. But, media type is very poorly supported. Screen width or height based rules are much safer.

Part 2 - Design the HTML Page

__1. In the web server document root (C:\Program Files\Apache Software Foundation\Apache2.2\htdocs) create a new file called **universal.html**.

__2. Open it with Notepad and enter the following code.

```
<!DOCTYPE html>

<html>
<head>

</head>

<body>

<div id="header">
<p>This is the header area</p>
<p>Home | About Us | Contact Us | News</p>
<hr/>
</div>

<div id="content">
<p>This is the main body area</p>
<img src="flower.jpg" id="big_image"/>
</div>

<div id="footer">
<hr/>
```

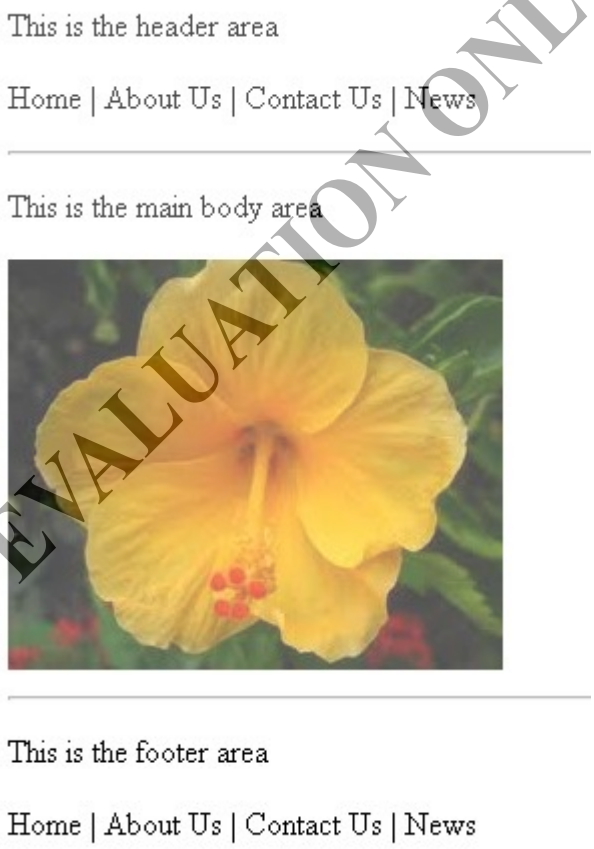
```
<p>This is the footer area</p>
<p>Home | About Us | Contact Us | News</p>
</div>

</body>

</html>
```

__3. Save changes.

__4. Copy **C:\LabFiles\flower.jpg** to the web server document root (C:\Program Files\Apache Software Foundation\Apache2.2\htdocs).

__5. In a browser (Chrome or Safari is encouraged) the URL:

```
http://localhost/universal.html
```



Note that the section links (Home, About Us, etc.) are repeated in the header and footer. Our goal is to hide the header entirely in a mobile device. That way, users can view most of the content right away without scrolling. For desktop, we will hide the footer.

__6. Close the browser.

Part 3 - Design the Stylesheet

__1. Within the <head> </head> tag, enter these styles:

```
<style  type="text/css">
@media only screen and (min-width: 481px) {
#footer {
      display: none;
}
#header {
      display: block;
}
}

@media only screen and (max-width: 480px) {
#footer {
      display: block;
}
#header {
      display: none;
}
#big_image {
      display:none;
}
}
</style>
```

__2. Save changes.

__3. For media rules to work properly in any Webkit based browser (as in Safari, iOS and Android), you will need to add this meta tag within <head> element.

```
<meta name="viewport" content="width=device-width" />
```

__4. Save changes and close the file.

Part 4 - Test Using Desktop Browser

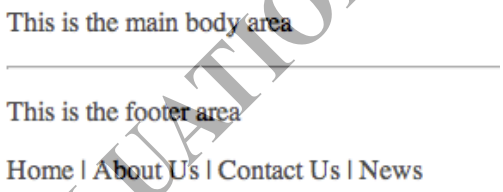
__1. Open a browser. Make it fairly wide to simulate a desktop environment.

__2. Enter the URL:

```
http://localhost/universal.html
```

__3. Make sure that header and image are displayed, but the footer is hidden.

__4. Try shrinking the width of the browser. At some point (i.e., below 480px width), the header and image will get hidden and the footer will get displayed.



Part 5 - Testing Using Mobile Device

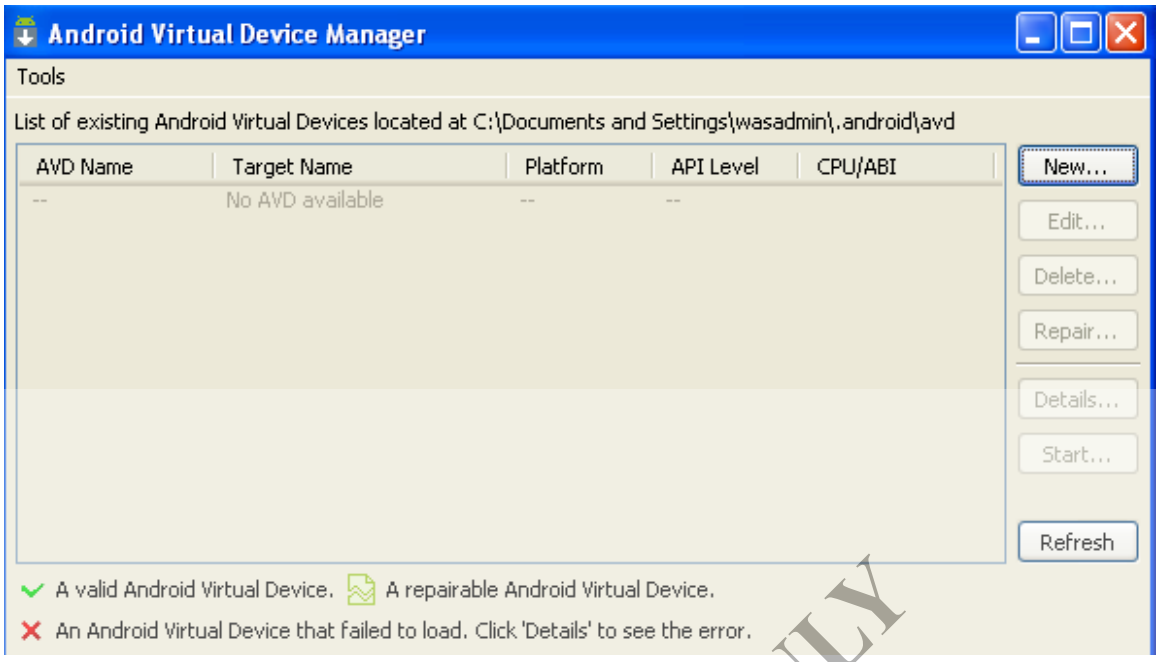
Next, we will use the Android device emulator to test the page on a mobile browser. First you will need to create an emulator, which is also known as an Android Virtual Device (AVD). An emulator is typically used for testing and debugging Android applications.

__1. Open a file browser and navigate to.

```
C:\Software\android-sdk-windows
```

__2. Open the AVD Manager by running this file:

```
AVD Manager.exe
```



__3. Click **New**.

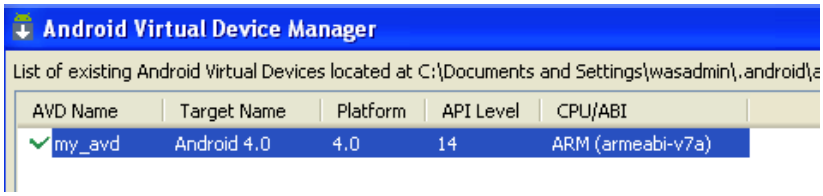
__4. Enter **my_avd** as the name.

__5. Select **Android 4.0 – API Level 14** as the target.

__6. Make sure that screen resolution is set to **Default (WVGA800)**. This resolution is a good approximation of most phones and fits nicely within most computer screens.

__7. Click **Create AVD**.

__8. Click **OK** inside the result confirmation dialog. The AVD should now appear in the list of AVDs, as shown below.



Note, the AVD files are created in **.android/avd** folder within your user's home directory.

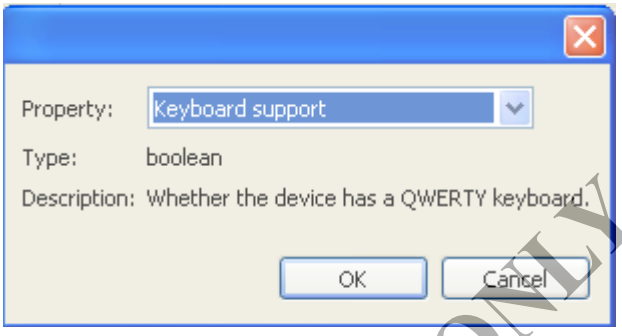
When a user taps inside an HTML form field in an Android phone's web browser, a soft (virtual) keyboard is normally displayed. In order to simulate this behavior, we need to modify the emulator settings.

__9. Select **my_avd**.

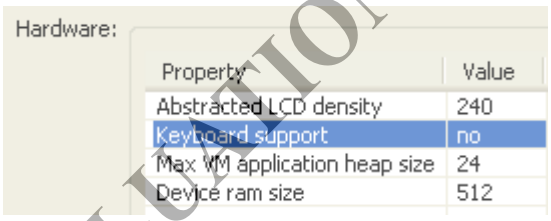
__10. Click the **Edit** button.

__11. Click the **New** button to the right of **Hardware**.

__12. In the pop-up dialog, select **Keyboard support** from the **Property** drop-down and click **OK**.



__13. Back in the *Edit Android Virtual Device (AVD)* dialog, change the value of **Keyboard support** from **yes** to **no**.



__14. Click the **Edit AVD** button to save the changes.

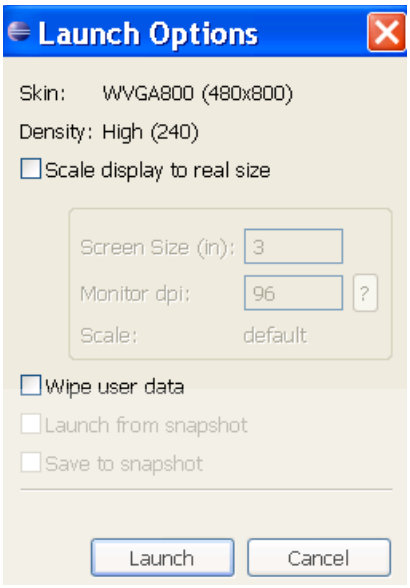
__15. Click **OK** inside the result confirmation dialog.

Now that we're finished creating our AVD, we'll start it up.

__16. Select **my_avd**.

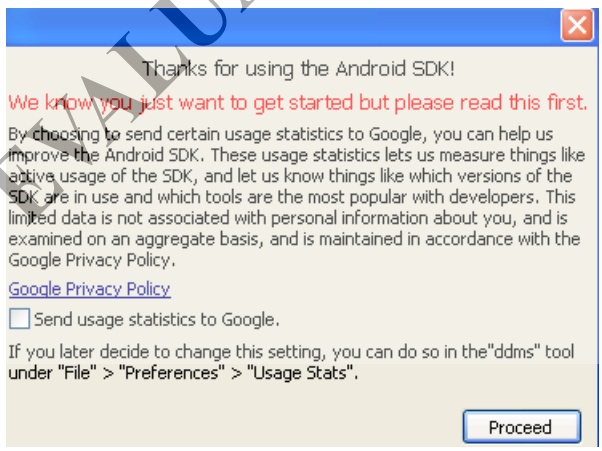
__17. Click **Start**.

__18. Click **Launch**.



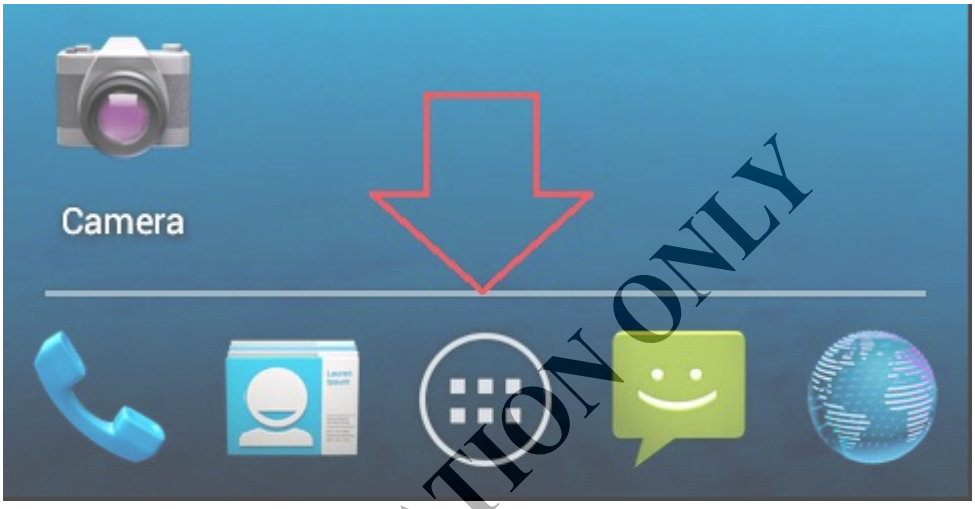
The AVD will be launched. It takes a while for the machine to boot. In the meanwhile, you will see an Android message.

__19. If you see a *Thanks for using the Android SDK* window, uncheck the box and click Proceed.

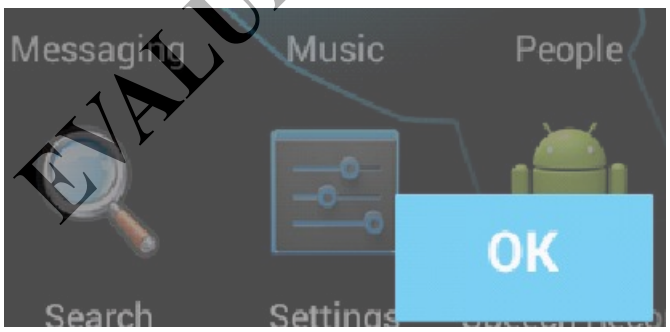


__20. Click **OK** when you see the image below.

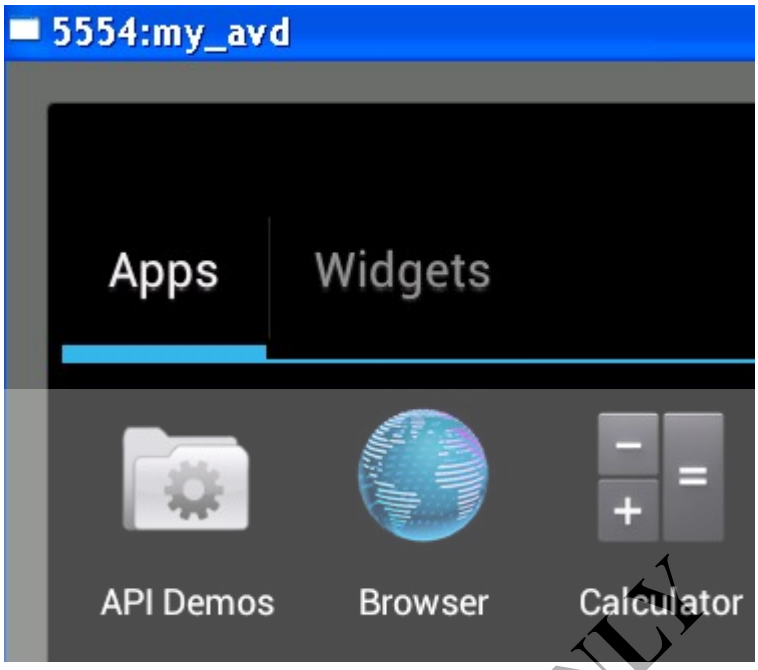
__21. Now click on the Application icon 



__22. Click **OK** again.



__23. The **Apps** page will be shown.



__24. Click the **Browser** app to open the internal web browser.

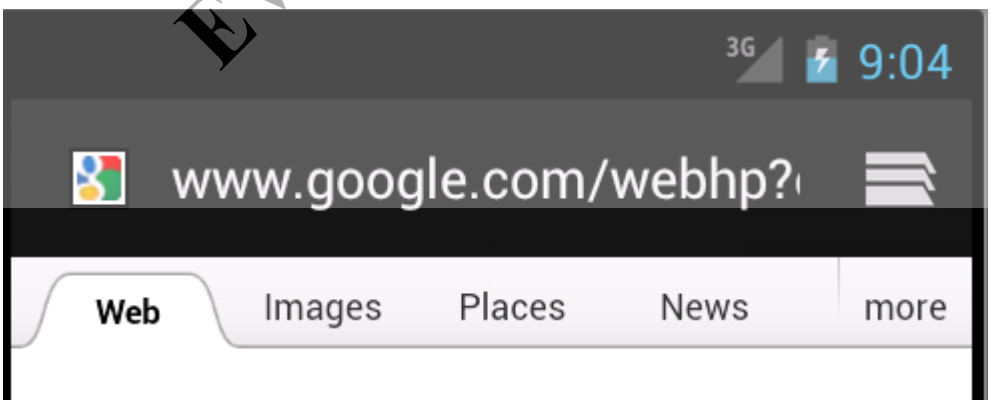
Next, we will navigate to the web page inside the mobile browser that you previously tested using a desktop browser. To access the web server from Android, we either have to use the IP address or edit the HOSTS file. We will take the former approach.

__25. Find out the IP address of your machine (by running the **ipconfig** command from a a new command prompt).

__26. Back in the Android emulator, enter the following URL in the location bar of the mobile browser:

http://<YOUR_IP_ADDRESS>/universal.html

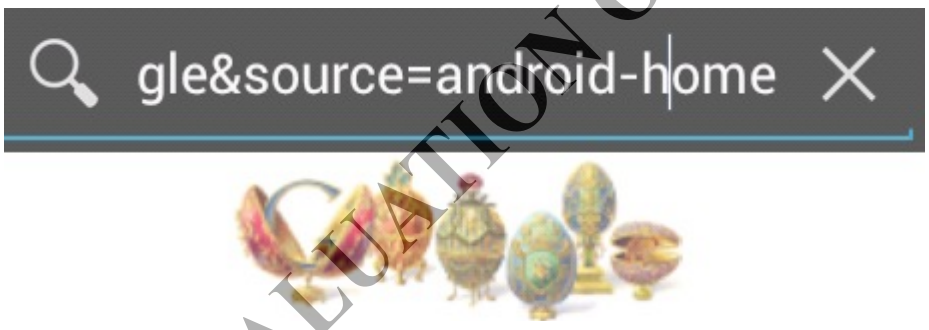
- If the location bar is not visible, then drag down (i.e., scroll down) to view it.



- When you click inside the location bar, a soft (virtual) keyboard will be displayed at the bottom of the browser, which you'll need to use to input the URL.

Note: If you try to type the URL using your computer keyboard or the emulator keyboard (displayed on the right side of the emulator window), you won't be able to. This is because we disabled keyboard support when creating the emulator.

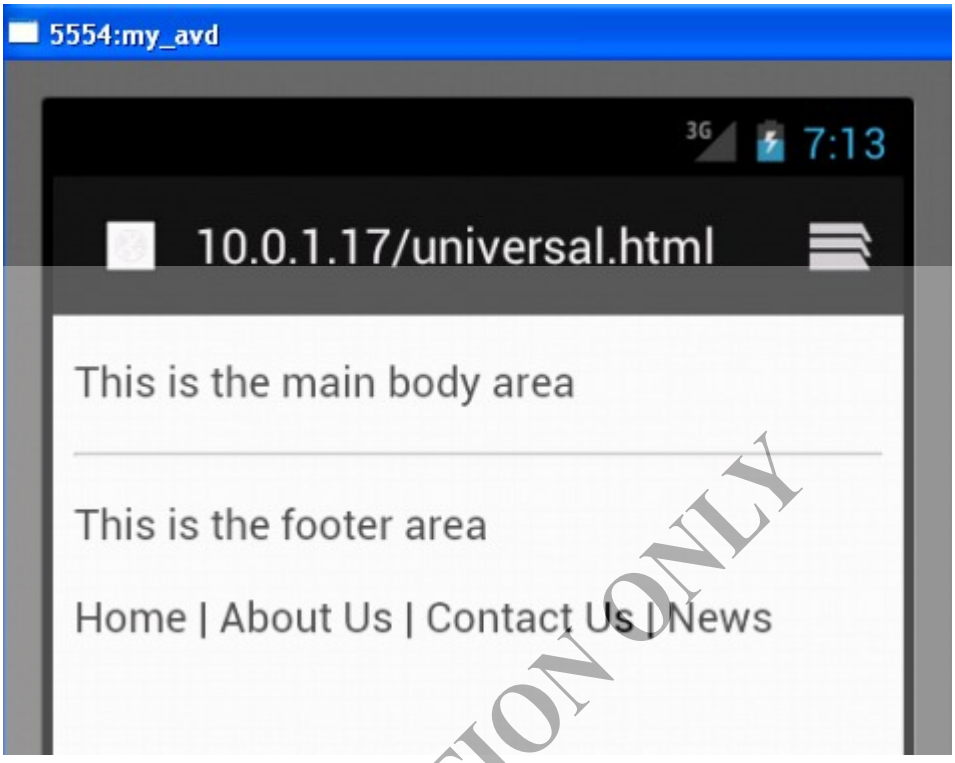
- To delete the current URL displayed in the location bar, click inside the location bar again. An **X** should appear to the right of the URL. Click the **X**.



- Enter the URL above. To type the colon (:) and forward slashes (/), click the **?123** key on the keyboard. To type the remaining letters, click the **ABC** key.
- Click the **Go** key to navigate to the URL.

__27. Verify that the header and image are hidden, but the footer is displayed.

Note: You may need to double click inside the browser to zoom in.



__28. Leave the emulator running, since we will be using it in upcoming labs too.

Part 6 - Review

In this lab, we learned how to create a universal web site using the CSS media rule. We created two sets of styles, one for mobile devices and another one for the desktop.

Lab 2 - Device Specific Rendering from JavaScript

In a previous lab, you used CSS media rules to do alternate rendering of the same HTML content. In this lab, you will achieve that goal using JavaScript. Using JavaScript, we will detect if the browser is running in a mobile device and then change the layout. This approach comes in handy in Web 2.0 style web sites where data is fetched using Ajax and then rendered using the JavaScript DOM API.

We will continue to work with universal.html. The logic will be the same. If running on a mobile device, we will hide the header and the image. Otherwise, we will hide the footer.

Part 1 - Write Device Detection Function

__1. Open **universal.html** in an editor.

__2. Comment out the entire **<style> </style>** tag so it has no effect.

__3. Save changes.

__4. Refresh the page in a browser and make sure that both the header, footer, and image are shown.

__5. Within the <head> </head> tag, add this code.

```
<script>
function isMobile() {
        var agent = navigator.userAgent.toLowerCase();
        var tokens = ["android", "iphone", "ipad", "ipod"];

        for (var i = 0; i < tokens.length; ++i) {
                if (agent.indexOf(tokens[i], 0) >= 0) {
                        return true;
                }
        }
        return false;
}
</script>
```

__6. Save changes.

Part 2 - Hide Elements on Load

Now, we will write JavaScript code that will hide the appropriate section of the page.

__1. Within the <script> </script> tag, add this code.

```
window.onload = function() {
        var hideIds = new Array();

        if (isMobile()) {
                hideIds.push("header");
                hideIds.push("big_image");
        } else {
                hideIds.push("footer");
        }

        for (var i = 0; i < hideIds.length; i++) {
           document.getElementById(hideIds[i]).style.display = "none";
        }
}
```

__2. Save changes and close the file.

Part 3 - Test

Note: This lab is best tested using Safari; Safari should be installed as part of the setup for this course. It has a menu item – **Develop > User Agent** which allows you to impersonate a desktop browser and iPhone or iPad browser. If you cannot use Safari then go to step 5 and do the Emulator test.

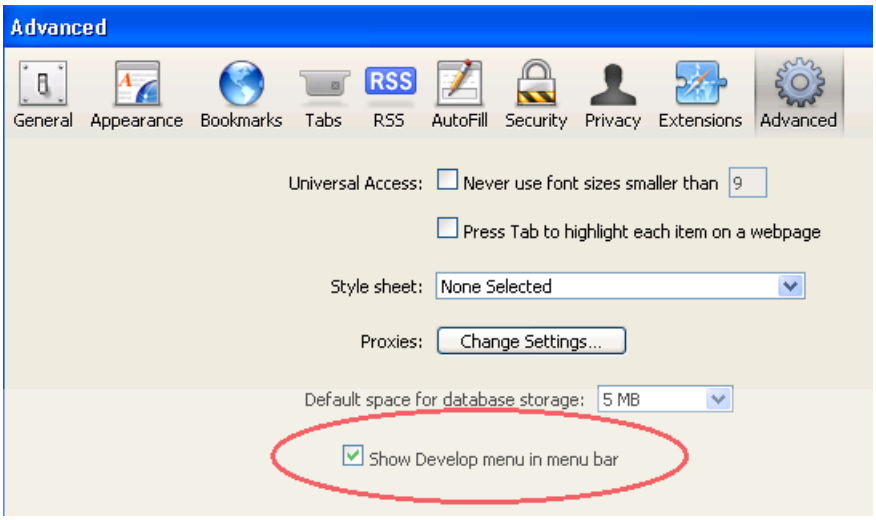
__1. Open a browser and enter the URL:

```
http://localhost/universal.html
```

__2. By default, you will see the desktop version. That is, the header and image will be shown and the footer will be hidden.

Note: You may need to double click inside the browser to zoom in.

__3. If you are using Safari, you need to enable the Developer menu. Select **Edit->Preferences** from the menu bar. In the *Advanced* dialog, click the **Advanced** button in the toolbar and check the **Show Develop menu in menu bar** checkbox. Close the dialog afterwards.



__4. Now, from the menu bar, select **Develop > User Agent > Safari iOS 4.3.3 – iPhone**. The page should refresh automatically and you should see the mobile version. That is, the header and image will be hidden and the footer will be shown.

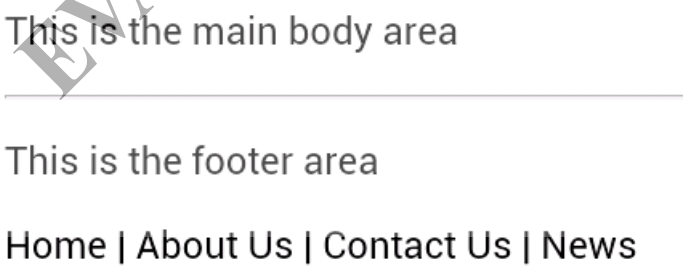
Next, we will test the web page in the Android emulator. If you closed the Android emulator, refer to Lab 1 for instructions on launching it and opening the internal web browser.

__5. In the Android emulator's web browser, enter the following URL or refresh the page:

```
http://<YOUR_IP_ADDRESS>/universal.html
```

Note: Remember that you'll need to use the soft keyboard to input the URL. You may also need to drag down (i.e., scroll down) to see the location bar.

__6. Verify that the header and image are hidden and the footer is shown.



__7. Close all browsers.

__8. Close my_avd emulator window and close the ADV Manager window.

Part 4 - Review

Many Web 2.0 style applications use the JavaScript DOM API to manipulate content shown in a page. In a universal web site, you may need to do device specific rendering. In this lab, we learned how to do that.

Lab 3 - Create a Basic HTML5 Layout

In this lab, you will create an HTML5 template which will serve as the starting point for creating new HTML files in upcoming labs.

This lab will help you understand what a very simple HTML5 document looks like. Specifically, you will learn to declare the DOCTYPE and character encoding.

When creating HTML5 files, you will use Notepad as a text editor. Of course, you're free to use another text editor if you'd prefer.

Also, we'll be using Google Chrome as a web browser, since it provides the best support for HTML5 in a Windows environment.

Note: On the Mac, Safari provides the best support for HTML5. Feel free to experiment with other web browsers, such as Firefox 4 and Internet Explorer 9. However, we can't guarantee that all the HTML5 pages you create will render properly in those other browsers. If you want to see how well your browser supports HTML5, navigate to **http://www.html5test.com** from within the associated browser.

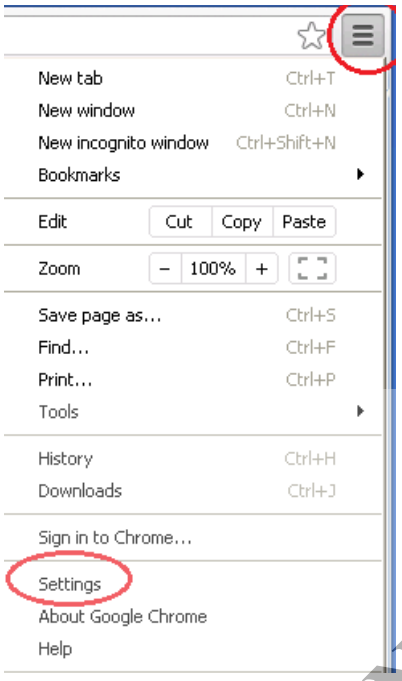
Note: These labs were written using Chrome 23.0.x. If you're using a other version of Chrome, you may see slight differences in the GUI.

Part 1 - Clear the Browser Cache and Location Settings

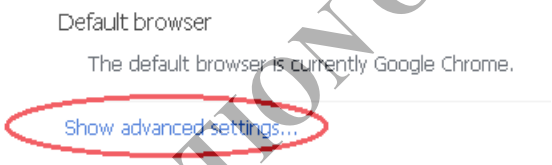
In this part, you will clear Chrome's cache and location settings. This will help ensure that any cached files and stored location settings will not interfere with any of the labs.

__1. Open Chrome by selecting **Start->All Programs->Google Chrome->Google Chrome**.

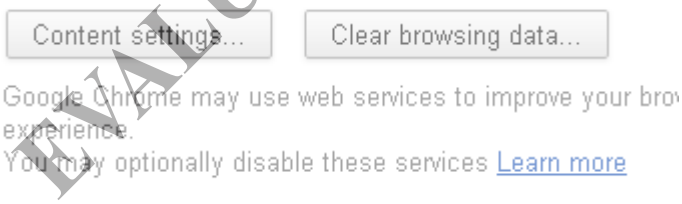
__2. Click the icon to the right of the location bar and select **Settings** from the drop-down menu. Older Chrome versions will show Options instead Settings.



__3. Under Settings expand Show advanced settings...



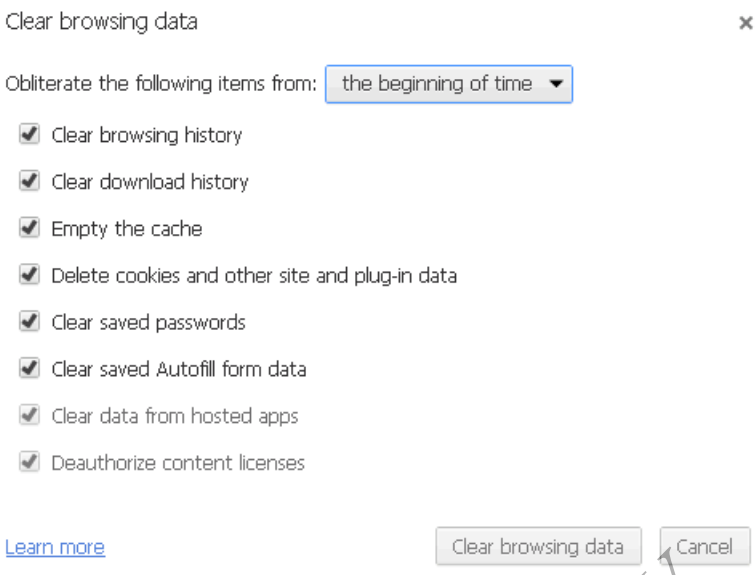
__4. Under Privacy click the **Clear browsing data**.



__5. Check all the checkboxes.

__6. Select **the beginning of time** option from the **Obliterate the following items from** drop-down box.

__7. Click the **Clear browsing data** button.

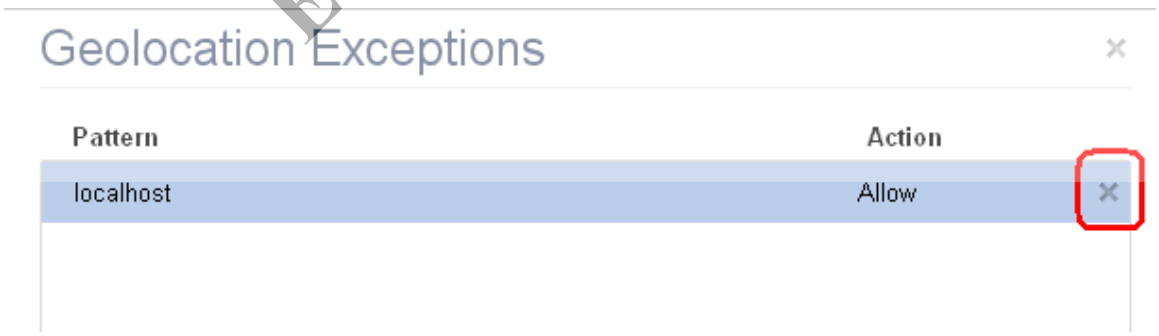


Wait for the clearing process to complete.

__8. Click the **Content settings** button, which is located to the left of the **Clear browsing data** button.

__9. Scroll down in the browser and click the **Manage exceptions** button under the **Location** section.

__10. If any exceptions appear in the panel, select the exception and click the **X** to the right of it to delete the exception.



__11. Click the **X** in the upper right corner of the **Geolocation Exceptions** panel to close the panel.

__12. Click the **X** in the upper right corner of the **Content Settings** panel to close the panel.

__13. Close the browser.

Part 2 - Create the Template

In this part, you will create the HTML5 template. The template will consist of the following elements:

- DOCTYPE
- html
- head
- meta (used to specify character encoding)
- title
- body

__1. Open Notepad by selecting **Start->All Programs->Accessories->Notepad**.

__2. Enter the **DOCTYPE** declaration, followed by the **html** element:

```
<!DOCTYPE html>
<html>
```

The DOCTYPE declaration must be the very first element in your HTML5 document. It even must appear before the html element.

Note: For those of you who are familiar with XML and XHTML (an XML compliant version of HTML), the DOCTYPE element is used to indicate the document type definition (DTD) that an XML document adheres to. A DTD is used to specify the grammar of an XML document.

However, an HTML5 document is not XML, so the DOCTYPE element is not required for this purpose. Instead, it's used to trigger "no-quirks mode" (a.k.a. "standards mode"). "No-quirks mode" implies the browser renders the page according to the HTML5 and CSS standards. "Quirks mode" implies it doesn't and instead tries to maintain backward compatibility with legacy pages created for older browser versions.

Also, notice that in HTML5 the DOCTYPE element is kept very minimal and simply includes the root element name. Normally, when entering a DOCTYPE element in an XHTML/XML document, you would need to specify several other pieces of information, including a URL pointing to the DTD (see example below). However, since the DOCTYPE element is not being used to reference a DTD, we can omit this

additional information.

e.g.,

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Note: The DOCTYPE element is case insensitive. You could have written the following instead:

```
<!DOCTYPE HTML>
<!doctype html>
```

__3. Define the document's character encoding inside the head element as shown below.

```
<head>
   <meta charset="utf-8"/>
   <title>HTML 5 Template</title>
</head>
```

Note: The character encoding declaration is shorter than the one you're accustomed to writing in HTML4:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

Note: HTML5 is not XML. Hence, element names are not case sensitive, elements don't require both a start and end tag, and attribute names don't need to be enclosed in quotes. Hence, you could have expressed the meta element in any of the ways listed below:

```
<META CHARSET="utf-8"/>
<META CHARSET=utf-8/>
<meta charset="utf-8">
<meta charset=utf-8/>
<Meta CharSet=Utf-8>
```

Having said that, we will use XML syntax going forward.

__4. Finally, add the **body** element and the **html** close tag to the document:

```
<body>
   Contents goes here...
</body>

</html>
```

Next, you will save the HTML file to the Apache Web Server document root.

__5. Open Windows Explorer.

__6. Create a folder called **layout** under the following directory (the Apache folder may be installed in other location, check with your instructor):

```
C:\Program Files\Apache Software Foundation\Apache2.2\htdocs
```

__7. Save in this folder the HTML file as **html5Template.html** within the layout folder.

Part 3 - Test

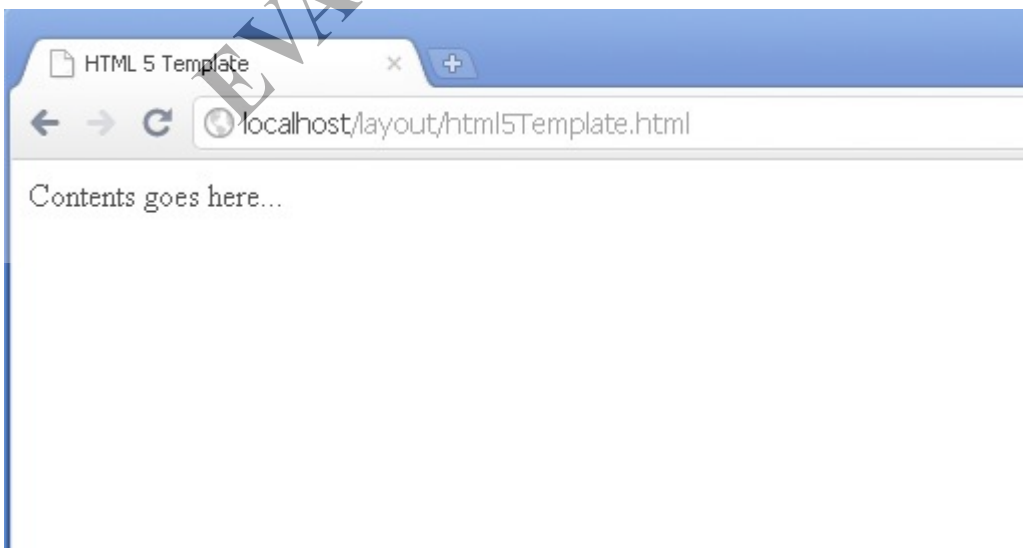
In order to test the HTML5 document, we'll use Google Chrome, since it provides the best support for HTML5.

__1. Open the **Chrome** browser.

__2. Enter the following URL into the address bar:

```
http://localhost/layout/html5Template.html
```

__3. You should see the following screen:



__4. Close the html file.

Part 4 - Review

In this lab, you created a template for your HTML5 files. In the process, you learned how to declare DOCTYPE and meta elements**.**