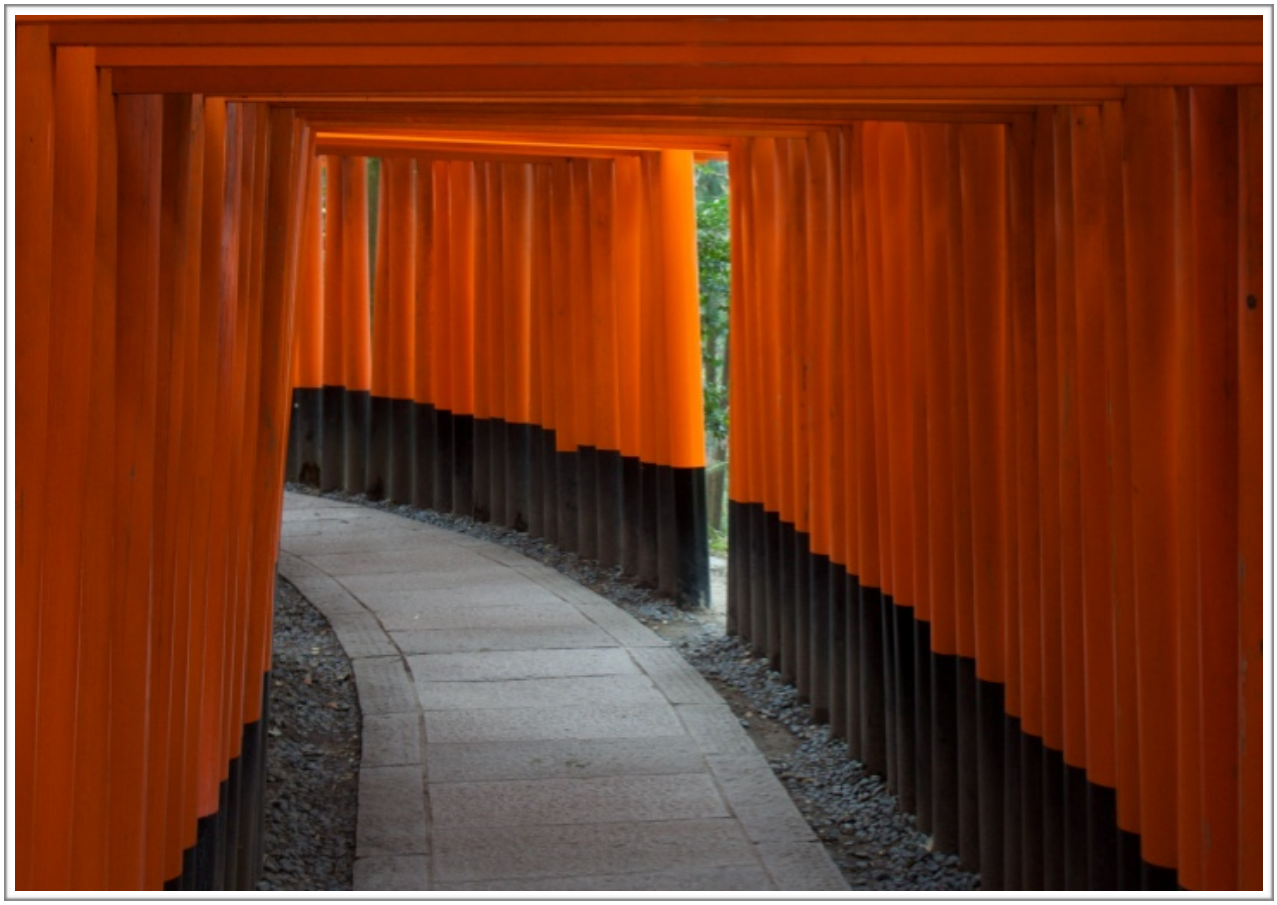
Net-centric introduction to computing

Daily Calendar



m Jenkin + h Roumani

1. Introduction

This lab deals with building a daily schedule in HTML, properly styling it in CSS, and adding sufficient JavaScript so that the table highlights the current day and time. Although the intent is to customize this application so that it runs within the Android environment — in particular within the EECS 1012 application provided — you will write the code so that it works in a generic browser on a generic computer platform.

As with all labs in EECS 1012, each lab is laid out as an ePub. It is expected that you will have read the lab electronic book prior to attending the lab. In order to encourage this, each lab has an associated set of 'pre lab' assignments that must be completed prior to attending the lab. The lab monitor will not allow you to participate in the lab if you cannot demonstrate that you have completed the pre-lab assignment prior to attending the lab itself. Nor will you receive grade credit for this lab if the pre lab exercises are not completed properly prior to your lab. Exercises in the lab are to be documented in your ePortfolio.

2. Background

It is not intended that you read through all of the background material in your first review of this lab. Rather you should skim the material presented here and then go to the actual exercises associated with the lab. Some of the material in the Background is just that, background. That being said, you may find that parts of the lab are easier to complete, or make more sense, if you have at least a passing familiarity with the material presented here.

2.1 Daily schedules

Daily schedules are a fact of life. Given the complexity of remembering schedules and the potential consequences associated with making errors, daily schedule programs and devices are very common. The basic goal of such systems is to provide a display similar to that shown in Figure 1.

	Monday	Tuesday	Wednesday	Thursday	Friday
8:30-9:30					
9:30-10:30					
10:30-11:30	EECS 1012		EECS 1012		EECS 1012
11:30-12:30					
12:30-1:30					
1:30-2:30					
2:30-3:30					
3:30-4:30					
4:30-5:30					
5:30-6:30					
6:30-7:30					
7:30-8:30					
8:30-9:30					
9:30-10:30					

Figure 1. Basic daily schedule display

The idea is to represent an individual's person schedule and then to enable the software to remind the user about upcoming events and the current time. Google, Apple and others offer such software systems as key elements of their software infrastructure. For the simple daily schedule that you will develop in this lab, you should fill in the calendar with the specifics of **your** class schedule.

2.2 Tables in HTML

Tables are common structures found in most layout languages. They can be used not only to structure tabular data, but also to provide a useful structure for other things. For example, to lay out two images beside each other, one rather simple way of doing this — and ensuring that the images stay beside each other — is to put them in a table consisting of one row.

HTML Tables begin and end with the tags <table> …. </table>. Within the table environment data is defined in terms of table rows. Each table row starts with <tr> and ends with </tr>. Each table element or table data is defined with the <td> tag. It is also possible to identify table elements that are 'headers' using the <th> tag. So, a very simple table consisting of three rows and two columns with a header row and a header column is specified as shown in Figure 2. When writing HTML space between elements is ignored so the indentation and other spacing is just there to help humans read it. (The HTML shown in Figure 2 is actually a screen capture of the atom editor, so all of the indentation shown there is a product of the editor trying to make the programmer's life more simple.) The table text (the table data) could be anything. Here we have used the pair row, column for each element to highlight the layout process.

```
<table>
  <tr>
    <th> </th>
    <th> col one </th>
    <th> col two  </th>
    <th> col three </th>
  </th>
  <tr>
    <th> row one </th>
    <td> 1,1 </td>
    <td> 1,2 </td>
    <td> 1,3 </td>
  </tr>
  <tr>
    <th> row two </th>
    <td> 2,1 </td>
    <td> 2,2 </td>
    <td> 2,3 </td>
  </tr>
  <tr>
    <th> row three </th>
    <td> 3,1 </td>
    <td> 3,2 </td>
    <td> 3,3 </td>
  </tr>
</table>
```

Figure 2. A simple table in HTML

Figure 3 shows the resulting table. The table has been laid out as a sequence of rows, each of which consists of a sequence of columns. As shown here, the table is not particularly pretty to look at. As you can see, by default <th> and <td> elements are formatted differently, but other than that there is not much to say about the formatting of the table. Prior to the introduction of CSS, it was common to use attributes of the various HTML table elements to provide style information about the table to enhance the appearance of its layout. And it is still possible to do it that way, although resist the temptation. It is almost always better to use properly structured CSS to separate the structure of the document from the way in which it should be displayed. This is considered in the next section.

	col one	col two	col three
row one	1,1	1,2	1,3
row two	2,1	2,2	2,3
row three	3,1	3,2	3,3

Figure 3. The Table specified by the code given in Figure 2.

It is possible to do quite complex things with tables. It is possible, for example, to have <td> or <th> elements that span across columns, to make elements that are multiple rows high, and so on. But for this lab, being able to lay out elements in rows and columns is sufficient. Full details on the subtle complexities of table environment in HTML can be found on the web. See here, for example.

2.3 Formatting tables using CSS

As with other HTML structures, tables can be formatted using cascading style sheets. For example, if we wanted to have a box around all tables, we could link to a CSS that defines the table formatting element shown in Figure 4.

```
1  table {
2    border: 1px solid black;
3  }
4
```

Figure 4: CSS to put a box around all tables.

Which would cause the table to look as shown in Figure 5.

	col one	col two	col three
row one	1,1	1,2	1,3
row two	2,1	2,2	2,3
row three	3,1	3,2	3,3

Figure 4: Table with a box around it.

- The full set of style parameters that can be applied to tables and their elements can be found in standard CCS references, including the textbook. CSS properties that are particularly useful for tables include:
- width, height - to define the size of an element.
- vertical-align - to define how material in adjacent elements should line up.
- white-space - to define how material in an element should be laid out internally.

3. Exercises

The goal of this lab is to construct an application that displays a customized weekly calendar within a web page, and to have the web page highlight the current time so as to remind you where you should be. As with other labs in this course, lab exercises are broken down into three sections, A, B and C. Exercises in section A are Pre-Lab exercises. All Pre-lab exercises **must** be completed prior to attending your lab. You will not be allowed to participate in a supervised lab if you have not completed these exercised prior to attending the lab. Furthermore, you will not receive course credit for the lab if you do not complete the pre-lab exercise. This lab is an unsuperised lab. You are expected to do this lab with your partner without the direct supervision of a TA.

3A. Pre-lab

1. Complete the Moodle quiz associated with this lab that can be found on the course Moodle site. You will not receive course credit for the lab without completing this quiz successfully. You are expected to complete this quiz on your own, but you are free to use other resources (e.g., the web, this ePub, and the provided source code to answer the questions).

B. In-lab

1. Obtain an experimental kit from the lab monitor. The kit should contain:
    - An Android device, power adapter, and micro USB cable.
    - Ensure that the Android device is properly charged (in any event, plug it in when you get it so that it is charging when not being used).
2. Obtain a laptop from the lab monitor, along with video camera and tripod. Alternatively use can use your own laptop.
3. Power up the laptop and start up the virtual machine. Once in the virtual machine locate the directory ‘server’. Within this directory there should be a directory ‘calendar’. If no such directory exists create it using the mkdir

command. All of the files that you will create in this lab will be created within the calendar directory within the server directory.

4. Using the atom application, create a file 'index.html' in the schedule directory. (Save this file to disk so that the live html preview works properly.) In this file create a html file (start with <html> ending with </html> with

   4.1. An empty <head> … </head> section.

   4.2. An empty <body> … </body> section.

5. Now within the <body> … </body> section first create a <h1> …. </h1> section with the tag id="info".

6. Following the <h1> section create a <table> … </table> structure that represents the university 'days and times of the week' as shown in Figure 1. (Note the formatting will be different than that shown in Figure 1, you will get to that shortly.) You need five days (Monday through Friday) and hourly time slots from 8:30 am until 9:30 pm. Use table headers <th> for the top row and the hour column.

7. Save the file. Make sure everything looks good in the preview within atom.

8. Launch the firefox browser within the virtual box and load the file index.html. Browse to this location using the 'Computer' entry in the 'Places' menu at the top of the screen.

9. Load the calendar into the application on the Android device. To do this, open the Html2Apk application and load the page http://ip:8000/zip/calendar where you replace ip with the ip address of the host machine. **Note:** if you are behind a router it is important to have the tablet and laptop connected to the same router and to use the address assigned to the laptop by the router. Rotate the screen, re-start the application, observe what files were downloaded to the application on the Android device.

10. Create a CSS file calendar.css in the calendar directory. Edit this file so that it has the same style formatting as shown below

```
table
{
  border: 1px solid black;
```

```
  table-layout: fixed;
  width: 100%;
}

th, td
{
  border: 1px solid black;
  overlfow: hidden;
  width: 16%;
}
```

Note what this style sheet does. It provides borders around the table, th and td entries. Furthermore, it sets the table to be 100% of the page wide, and each th or td box to be 16% of the page wide. This 16% was chosen so that the contents of the table (as laid out) fully fills the box horizontally.

11. Add a link to this stylesheet in the index.html file. You can do this by adding the line <link href="calendar.css" rel="stylesheet" type="text/css"/> to the <head> section of index.html. Your page should now look (more or less) like that shown in Figure 1. View the page in Firefox within the virtual box and install the calendar in the Android. Swap from portrait to landscape. Does the page re-format to the full width of the display?

12. Now we are going to start the process of having your web page update itself based on the time of day. Create a file calendar.js in the calendar directory, and add a link to calendar.js in the <head> of index.html using the line <script src="calendar.js" type="text/javascript"></script>.

13. Now edit calendar.js so that it contains the following text. function updatePage()
{

```
  var d = new Date();
  var e = document.getElementById("info");
  e.innerHTML = "Time is now " + d;
}

function startUpdate()
{
  updatePage();
  window.setInterval(updatePage, 10 * 1000);
}

window.onload=startUpdate;
```

This text does a number of things. First, it defines the window.onload event handler so that startUpdate() is invoked when the page is fully loaded. Second, the startUpdate() method first invokes updatePage(), but furthermore it registers updatePage as a callback that will be invoked every 10 * 1000 milliseconds (every 10 seconds). Finally, look at updatePage(). It gets the system date and replaces the innerHTML of the "info" element with the date. You may remember that the <h1> header at the top of the file was given this id, so the effect of this code is to put the time at the top of the page, updated every 10 seconds.

14. Browse to the index.html using firefox and download the calendar to the android device. Observe that the time is properly updated in both. Also observe that the Android device has downloaded the JavaScript, CSS and HTML files. (The application actually downloads everything it can find in the directory, including the contents of any sub-directories.)

15. We would now like to modify the application so that it highlights (in red) the current time. To do this we need to do a couple of things. The first, is we need to have a way to refer to individual day-time combinations on the schedule. There are a number of ways to do this, we will choose what is perhaps the easiest, but also an approach that requires editing the html. Edit the index.html file so that each of the day-time boxes have an id field. Specifically, we want each day-time box to have an id field of the format "timeD:HH" where D is 1 for Monday, 2 for Tuesday, and so on. And HH is 8 for the 8:30-9:30 slot, 9 for the 9:30-10:30 slot, ending with 21 for the 9:30 pm slot. (So the last slot on Friday evening has the id "time5:21".)

16. Now we need to adjust the JavaScript so that it takes the current time and works out from that time the corresponding day (in the range 1..5) and hour (in the range 8..21). Fortunately the Date() object has methods getDay(), getHours() and getMinutes() (see here) that gets this information in exactly the format we want. To see this, insert the following text to the JavaScript before updating the innerHTML, and set the innerHTML to be day + " " + hour + " " + minutes. Test the effects of this using the atom preview window.

```
var day = d.getDay();
var hour = d.getHours();
```

```
var minutes = d.getMinutes();
```

17. Now construct the name of the day-hour block in the html table that corresponds to the time. You might have though this could be done as "time" + day + ":" + hour. And this almost works, but not quite. The problem is that classes at York start on the half hour. So the time 9:00 corresponds to the 8:30 block, not the 9:30 block. We can 'fix' this by reducing the hour by 1 if the minutes are less than 30. Insert the following text into the JavaScript and update the innerHTML as shown below. The info tag at the top of the page should now be showing the appropriate block name for the current time.

```
if (minutes < 30)
    hour = hour - 1;

var block = "time" + day + ":" + hour;
e.innerHTML = "Time is now " + block;
```

18. Now lets work on highlighting the current time on the calendar. Now the entire week is not represented on the calendar nor or all the hours of the day. So not all 'block' correspond to real elements in the document. Fortunately, getElementById returns null if the element is not found. So we can avoid that particular problem easily. Insert the following into the JavaScript after the block value has been computed and run the code. Assuming you are doing this on a weekday during regular hours, one of the blocks will turn red. Almost there.

```
var c = document.getElementById(block);
if (c != null)
   c.style.background = 'red';
```

19. Unfortunately, the code as written above does have one bug. If you let the code run for a while you will see that eventually the entire day turns red. Although we turn 'on' the portion of the table that corresponds to now, we don't turn off that part that might have been turned on. Now turning the 'old time' off is a bit tricky. We could turn off all old times, but that might result in a flickering of the current time as we turn it off and on. A more elegant solution is to remember what time we turned on last, and turn it off if we need to. So, to do that we introduce a global variable lastTime and initialize it to null, indicating the we have not turned on any square yet. We do this in the global context (when assigning the window.onload event handler. So insert var lastTime=null; just

before where the window.onload handler was set to startUpdate. Finally, we need to maintain the value of lastTime and update the background colour of the lastTime location to be white if needed. To do that, replace the updating of the background colour to be red with the following code. Note that this does not change the background colour to white if lastTime is null or if lastTime and c are the same, and it updates lastTime to be c.

```
if ((lastTime != null) && (lastTime != c))
   lastTime.style.background = 'white';

if (c != null)
   c.style.background = 'red';

 lastTime = c;
```

20. Clean up the code, setting the <h1> field to something useful (like "My Schedule") or similar, and deploy the code to the Android device.
21. Create an ePortfolio event for this lab. Show the final application running in both the virtual box and on the android device. Can you configure the Android device so that the screen stays in landscape mode even when rotated? What problems did you encounter during the lab?

C. Advanced

1. Replace the text for classes and other events in the table with hyperlinks to Moodle or web pages associated with them. You can now 'click' on events in your calendar.
2. In the CSS file, set the height of the table to be 90% of the page, and set the height of each td and tr element to be around 6%. Does the table "more or less" fill the screen?
3. In the CSS file, set the text-align style for h1 to be center, so that the title is properly centred.

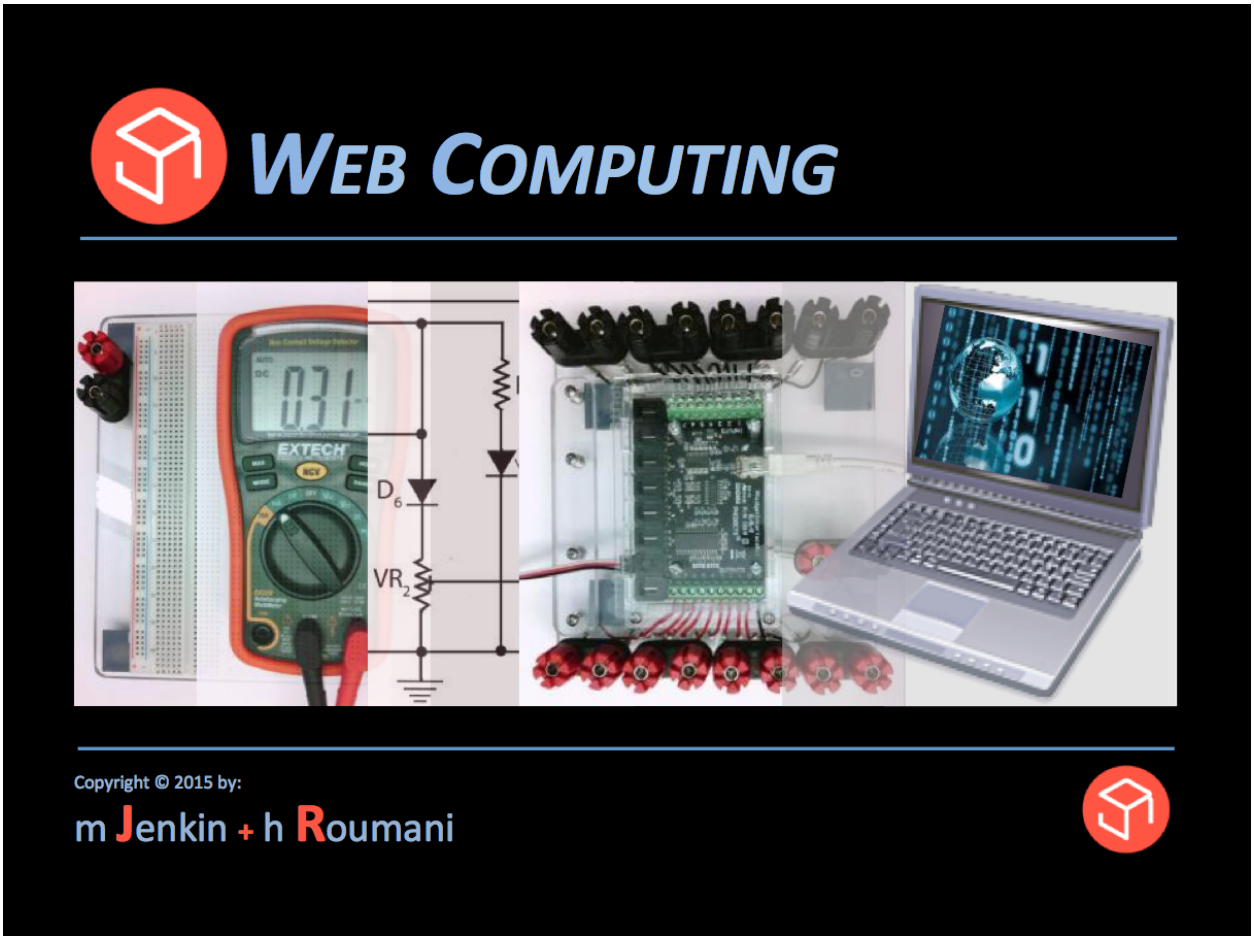
4. Further Reading

CSS Cheat Sheet - http://www.lesliefranke.com/files/reference/csscheatsheet.html

HTML Cheat Sheet - http://www.webmonkey.com/2010/02/html_cheatsheet/

JavaScript Cheat Sheet - http://www.cheat-sheets.org/saved-copy/jsquick.pdf

5. Credits


WEB COMPUTING
EXTECH
D6
VR2
Copyright © 2015 by:
m Jenkin + h Roumani