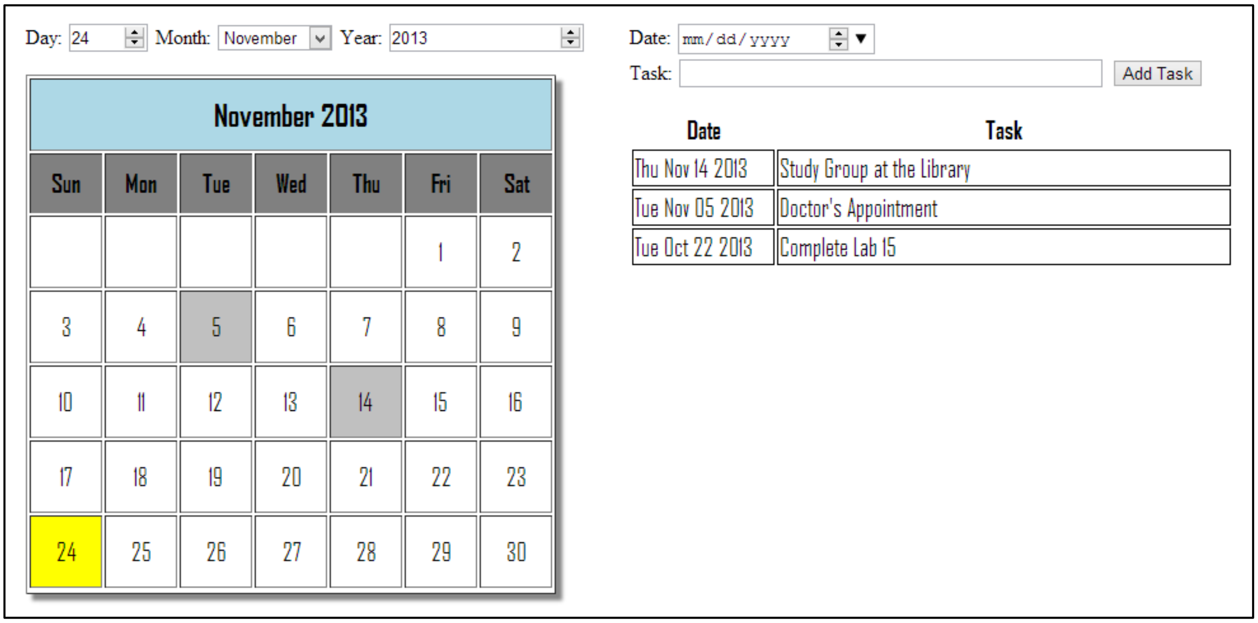
LAB 11
50 POINTS

OVERVIEW

In this Lab, you will create an interactive calendar and task list. The Lab combines the knowledge from several topics covered in class (HTML 5, CSS and JavaScript). Here is an example of the webpage.



Day: 24 Month: November Year: 2013

Date: mm/dd/yyyy ▼

Task: Add Task

November 2013

Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Date	Task
Thu Nov 14 2013	Study Group at the Library
Tue Nov 05 2013	Doctor's Appointment
Tue Oct 22 2013	Complete Lab 15

Once the page is loaded, the application will get the current date of the computer and will setup the form components with Day, Month and Year. Then, the information from the form components will be used to create a calendar for the corresponding date. The specified day will be highlighted in yellow.

The user can change the year or the month and the calendar is automatically recreated.

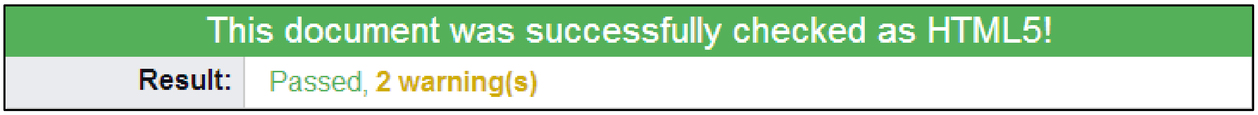
For the tasks, the user will select a specific date from the dropdown option, will type a description for the task, and will click on the button Add Task. The information from the form will be added to an array with the list of tasks. Then, the application will loop through the array and will print a table with the list of tasks.

Finally, if the month being displayed has one or more tasks, the day with the tasks in the calendar will be highlighted.

CODE VALIDATION

You must validate your HTML 5 and CSS code.

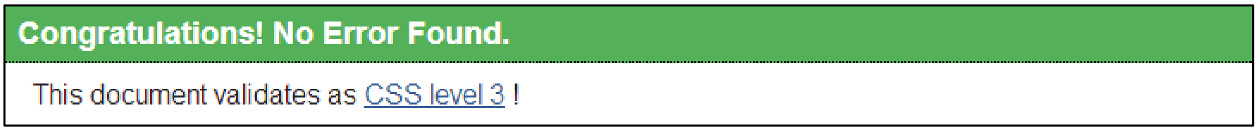
http://validator.w3.org/check



This document was successfully checked as HTML5!

Result: Passed, 2 warning(s)

http://jigsaw.w3.org/css-validator/validator



Congratulations! No Error Found.

This document validates as CSS level 3 !

FILES AND FOLDERS

Create a new folder with your name + the word Lab2 (ex: JohnDoe_Lab11).
The first thing you need to do is to create the folders that will hold the necessary files.

- Create a folder called **js** for the JavaScript files
- Create a file called **calendar.js** in the folder js.
- Create a file called **task.js** in the folder js.
- Create a folder called **css** for the style sheet file.
- Create a file called **calendar.css** in the folder css.
- Create a file called **Lab11.html** in the root folder of the Lab.

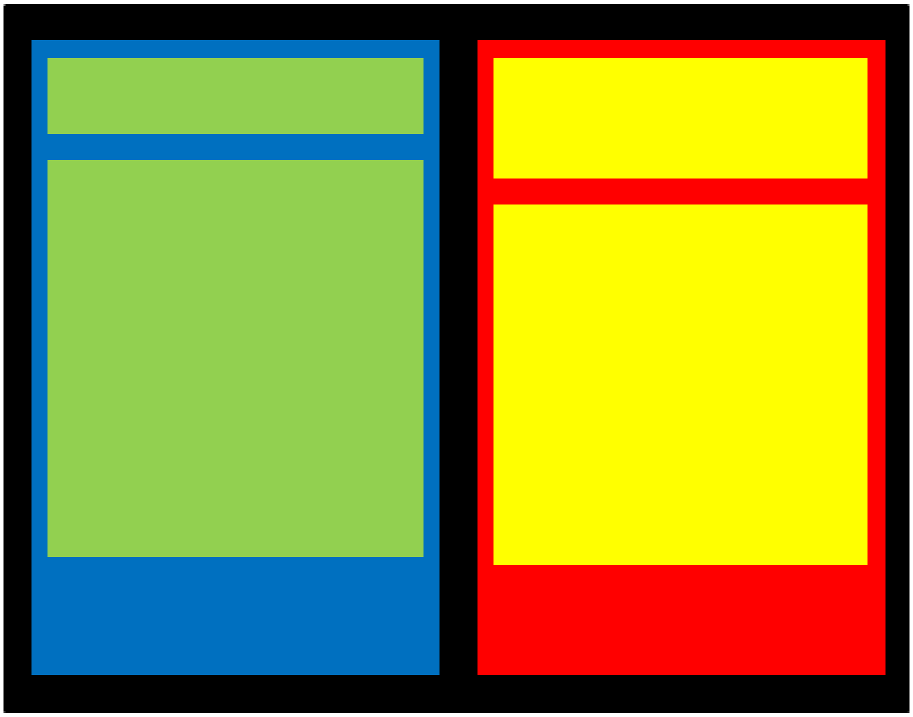
In the next sections, we will discuss the steps required to complete this web application. Sometimes you will see some source code, and sometimes you must find the appropriate code to complete the steps.

LAB 11

The file must have the basic content for HTML5. Remember that you must validate the file with HTML5 (experimental).

In the <head> area of the html file, create a link to each of the JavaScript files and to the CSS file. Notice that the files are inside a specific folder. So, you will need to inform the location of the files, including the folder name.

The <body> will have several <div> tags to help to organize the content as shown in the figure below. IMPORTANT: The colors in the figure are used only to illustrate the divs. You don't actually need background colors for the divs.

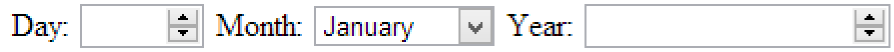


- Wrapper div (in black) has a size 950px and is centered on the screen

- Left div (in yellow) has a size of 50% and will float
- Right div (in red) has a size of 50% and will float
- TopLeft div (in green) to show the form components for the day, month and year
- BottomLeft div (in green) to show the calendar
- TopRight div (in yellow) to show the form to add a task
- BottomRight div (in yellow) to show the list of tasks

Now that you have the information about the structure for the file, you have to write the code to create the appropriate divs as well as the style sheet to produce the expected results. Keep in mind that the divs are nested.

Inside the TopLeft div, you will create the options for the user to input the Day, the Month and the Year as shown below. Notice that the day and year are using a specific type of input available for HTML 5 code. Each component must have and id. Please use the names day, month and year respectively for the ids.



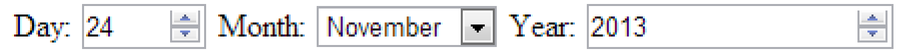
Day: Month: January Year:

The day has a minimum = 1 and a maximum = 31. This will help to control the range of numbers. Include an ID called **myyear**.
The month is created with a <select>. For the <option> tags you will need to inform the value, which is the number corresponding to a month and you need to define the name of the month that will be displayed to the user. For example, the option for January would be **<option value="0">January</option>**. It means the user will see the name of the month, but if selected, only the corresponding value is available. Include an ID called **mymonth**.
The year has a minimum = 100. Include an ID called **myday**.

Now that the components have been created, we need to create a JavaScript function that will get the current date of the computer and will use the information to initialize the day, month and year components. The example above shows that the computer date was November 24, 2013.

To do that, let's create a function called **setDate** inside the calendar.js file. The function will create a variable called **today** using the code new Date(). This is similar to the example in the slides of the file. With the variable today created, you can use the functions **getDate**() to get today's day, the function **getMonth**() to get today's month, and the function **getFullYear**() to get the four digits of the year. The result of these functions will be assigned to the components myyear, mymonth and myday using the code **document.getElementById("myday").value = xxxxx**. Replace xxxxx with the appropriate function from today's date. Similarly, create the code to get the element id "mymonth" and "myyear".

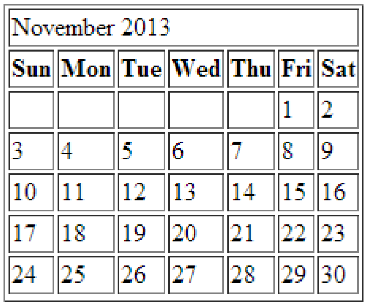
When the webpage is loaded, the system must call the function setDate() created above and the day, month and year components would be automatically extracted from today's date and used to populate the components. In the Lab3.html create the attribute **onload="setDate();"** in the <body> tag. When the page is loaded the components should have the information corresponding to the current date of the computer, as shown in the figure below. In the example, the date was November 24, 2013.



Day: 24 Month: November Year: 2013

At this point, you should check if today's date is added to the components on the webpage. Take the time also to validate your code using HTML 5. If you don't get the same result, STOP and fix your code before moving on to the next steps.

Next, we will dynamically create a calendar based on the information day, month and year selected above. Following is an example of the calendar without any css style. We will apply the css configuration later.



November 2013						
Sun	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Let's start by creating a function called **buildCalendar()** in the calendar.js file. Then, let's call the function when the page loads. We already have an attribute onload in the <body>. To add another function call, you can just add the name of the new function as shown below.
<body onload="setDate(); buildCalendar();">

Now, when the page loads, the system will first set the date, and then will build the calendar. Well, not yet. Our buildCalendar() function is empty.

Before we write any code, let's think about the details of the calendar.

- We need to know what week day is the first day of the month for the date selected by the user. It will help us to define which column to use to print the first day of the month. For the the example above, the first day is Friday. All the table cells before that are empty.
- We also need to know the last day of the month (28, 29, 30 or 31). To print the days, we will use a loop. The information about the last day will help us to control when to stop the loop.
- We can only print in one of the 7 columns. So, we need a way to control when to stop printing in the same row and when to move down to the next row.
- To print the table, we need 3 parts. First is the header of the table. This part will not repeat and must be created before the loop. Second is the list of days for the month. Each day will be printed within the loop. Third is the end of the table. We close the table tag after we complete the loop.

- We will create the code to build the calendar in the function buildCalendar(). Then we will send the code into the specific div area identified by the id=”bottomleft”. That will force the calendar to appear in the correct position.

In the buildCalendar() function, create a variable called firstDate which is a new Date(). However, at this time, we can pass some parameters to the function. Inside the parenthesis, we will inform 3 parameters that must be separated by comma. These parameters correspond to the month and year of the current date selected, and we use value 1 to represent the first date. The parameters informed must be in the following sequence year followed by month followed by the number 1 (first day of the month). The year and the month are coming from the form components.

```
var firstDate = new Date(document.getElementById("myyear").value,
                         document.getElementById("mymonth").value,
                         1);
```

We need now to get the specific information (weekday, day, month and year) from the variable firstDate. For that, create a new variavle for each of the four items listed above and use the date functions to get the expected information. For example, to get the weekday we use the code **var weekDay = firstDate.getDay();** which will store the number of the weekday for the firstDate. Remember that 0=Sunday, 1=Monday and so on. Complete that for all four information listed above.

Next, we need to identify the last day of the month. There are different ways to accomplish this. We will use a simple trick. First, we will create a date with the first day of the next month. Then, we will create another date for the day prior from the step above. If the first step was to find the first day of the next month, the second step will find the last day of the month that we need. Finally, we use a date function to get only the number for the last day of the month. See the code below.

```
var newDay = new Date(year, month + 1, 1);
var lastDate = new Date(newDay - 1);
var lastDay = lastDate.getDate();
```

Now, we are ready to start printing the table.

Let’s create a variable called code that will contain the code to create the header of the table. This is similar to how we created a table for the temperature convert from Fahrenheit to Celsius. That includes the open table tag, the tr and th for the header. We will want to use CSS to add styles to the table. So, create a table with a class (not an id) called caltable. Here is the start of the code and you have to finish it. Use <th> for the weekday name. There will be several lines of code. Here is the first.

```
var code = "<table class='caltable' border='1'>"+
```

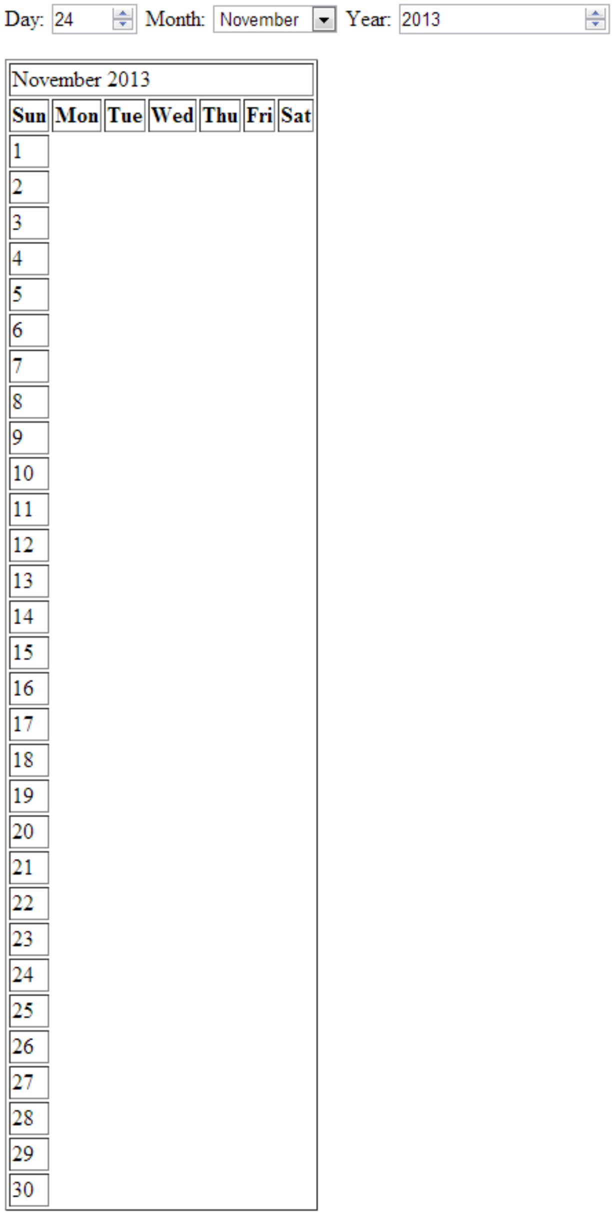
To print the name of the month, create a new function called **getMonthName()**. This function will check the month selected from the dropdown list and will return the name of the month. You need to getElementById and then access the text of the option selected (selectedIndex).

The next step is to create a loop to print all the days between the first and the last day of the month. Each day will be printed inside a <td> </td>. You will also need a <tr> </tr> for each row. For example:

code = code + "<tr><td>" + i + "</td></tr>";

Write a for loop or while loop to print the days. After the loop is complete, close the table tag.

The result should be similar to the picture below. Test it!

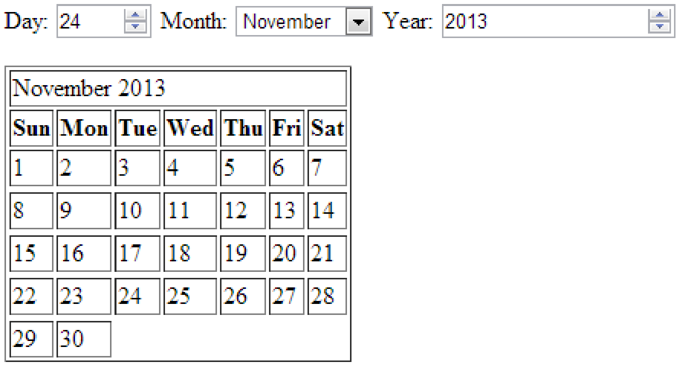


Day: 24 Month: November Year: 2013

November 2013						
Sun	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						
29						
30						

The problem is that we need to complete a row with the weekdays before moving down to the next row. To fix that, we will use a variable to count the number of columns printed. So, before the loop we need to create a variable col = 0. Col 0 is Sunday and col 6 is Saturday. Every time we print a column within the loop we need to increment the value of col. When the count of col is greater than 6, we reset the value to zero. This approach will also need a change on how we print the rows and columns. We

want to start the <tr> tag when count is equal to zero. And, we want to close the <tr> tag when col is greater than 6, which the same if statement used to reset the count of col.
The result should be similar to the picture below. Test it!



Day: 24 Month: November Year: 2013

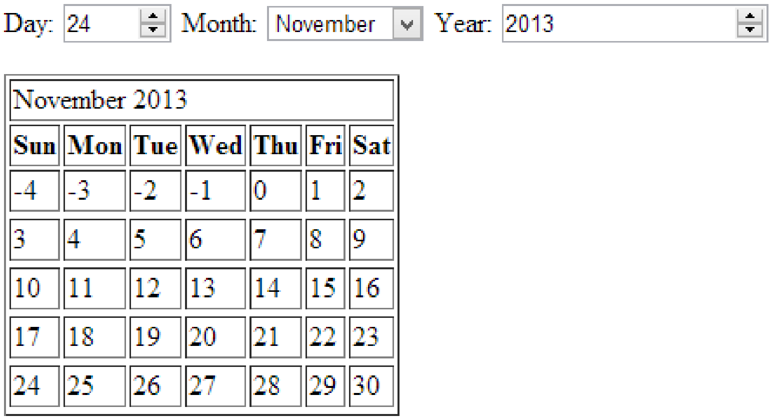
November 2013

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

Our next problem is to start the first day of the month in the correct column. For our example, the first day of November is Friday. That means that columns for Sun, Mon, Tue, Wed and Thu prior to Fri should be empty.

We can use the weekday to solve this problem. The value of weekday is 0 for Sun, 1 for M, 2 for Tue, 3 for Wed, 4 for Thu, 5 for Fri and 6 for Sat. The first is Friday which is number 5. Now count how many columns should be empty. Yes, 5 again. So, we can use the weekday to add the empty spaces.

One simple solution is to change the value of the start of your for loop from 1 to 1-weekday. That would print the following result. Test it!

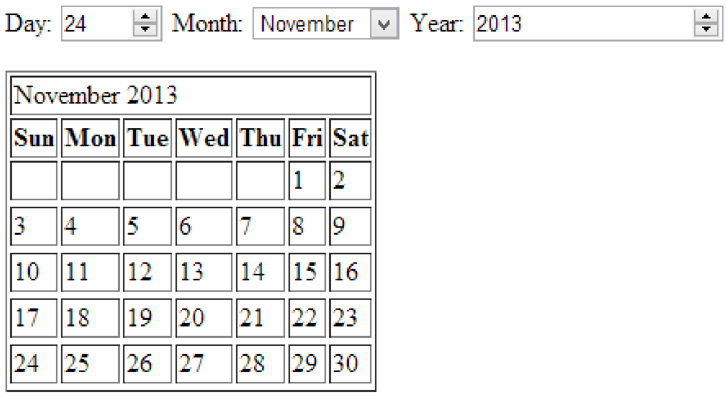


Day: 24 Month: November Year: 2013

November 2013

Sun	Mon	Tue	Wed	Thu	Fri	Sat
-4	-3	-2	-1	0	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Now, the only thing we need to do is to create an if-statement in the loop to print the day only when the variable controlling the loop (ex: i) is greater than 0. When the value is negative, we print an empty td (ex: **<td></td>**). Problem solved.
The result should be similar to the figure below. Test it!



Day: 24 Month: November Year: 2013

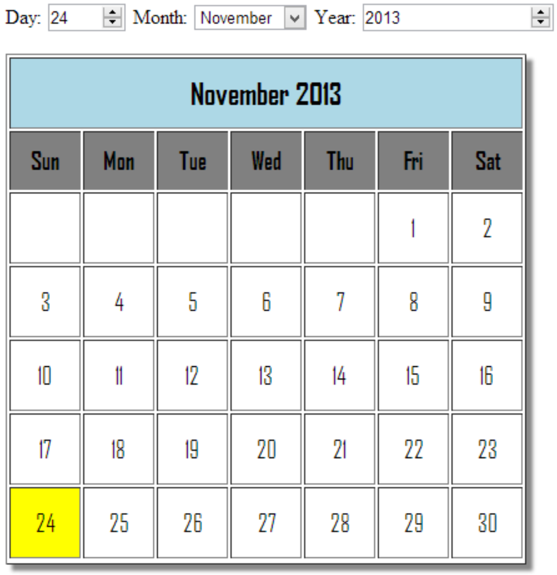
November 2013						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

We also want to highlight the target day selected by the user. For the example above, the target date is November 24, 2013. We want to make the target day in bold.

First, let's create a new variable (before entering the loop) called **targetDay**. The content from the box Day will be assigned to the variable targetDay. So, for the example above, the targetDay = 24.
Second, before we print the <td> with the day within the loop, we need to check if the day to be printed (ex: i) is equal to the targetDay. If yes, then we add an **id="dday"** to the <td> so that we can use CSS to add style to the targetDay. If not, we just print the day without id.

We also want to recreate the calendar everytime the user changes the day the month or the year using the form components. So, we need to add an **onchange="buildCalendar()"** to the <input> tag for the day, the <input> tag for the year, and the <select> for the month. This way, if any of them changes the calender is recreated.

Finally, you need to create the appropriate CSS code to add style to the calendar as show below. Notice that all the days have the same size <td>.



Day: 24 Month: November Year: 2013

November 2013						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30