UNIVERSITY OF TORONTO
Faculty of Arts and Science
Midterm Exam – Fall 2011
CSC 309
Instructor – Eyal de Lara
Duration – 1 hours
Examination Aids: One single-sided hand-written 8.5"x11" page containing notes

NAME _____

STUDENT NUMBER _____

PART I (Short Answers): _____ /25
PART II (Programming): _____/75
Total: _____/100

Part I of this examination is a series of short-answer questions.

Part II of this examination is a programming exercise.

Answer all questions in the spaces provided on this examination paper. There is no need to use more space than is provided. If you must, then use the back of the examination paper and so indicate in your answer.

General Advice:

- Skim through the entire exam before beginning your detailed work, to get a sense of where best to spend your time; if you get stuck on one question, go on to another and return to the difficult question later.
- Show your work, not just the final answer. Partial credit will be granted in cases that demonstrate correct reasoning, even if an error leads to an incorrect final result.
- For the programming exercise, if unsure of API details or language syntax details, try your best and include a comment indicating what you are attempting to achieve.

Good luck!

Part I – Short Answer Questions

1. [8 points] Briefly describe two shortcomings of the original HTML implementation that were addressed by the introduction of CSS.

   The mix of content description and style into a single file lead to the following problems (among others):

   - Low ability to customize to different platforms

   - Low programmer productivity

   - Low system performance due to bloated documents

2. - HTTP

   a) [3 points] What is the main difference between HTTP1.0 and HTTP1.1?

   HTTP 1.1 uses persistent connections and can handle multiple requests over a single connection.

   b) [5 points] Despite its differences from HTTP1.0, HTTP1.1 remains a stateless protocol. Why is this the case?

   HTTP 1.1 does not keep track of session state, only TCP state.

3.- [9 points] Give 3 examples of limitations that browsers typically place over both JavaScript and Java Applets for security reasons.

No file/directory access defined in the language
No raw network access.
Same origin policy: can only read props of documents and windows from the same place (host, port, protocol)
Cannot read history
Cannot hide/show menubar, status line, scrollbars cannot close a window not opened by itself

Part II - Programming Question

4- [25 points] Using JavaScript and XHTML write a simple application that shows the current date and a running clock. Format the date and time as shown in Figure 1.

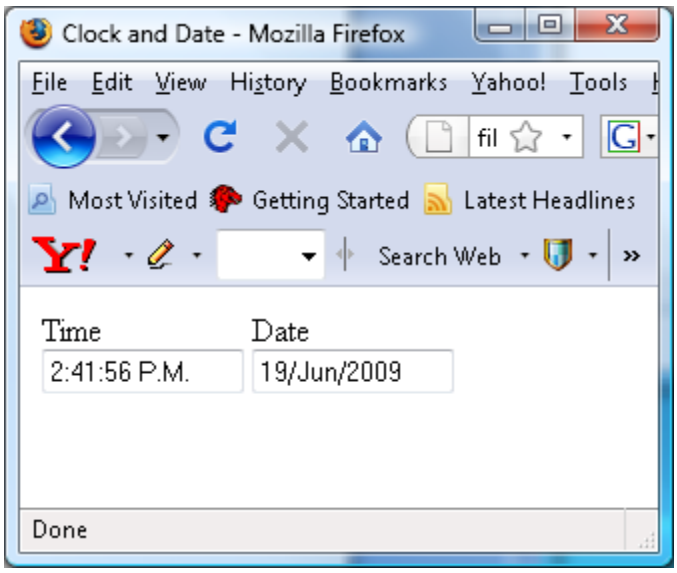

Figure 1

Write your application using the following methods of the JavaScript built-in Date class:

getDate
Returns the day of the month (1-31) for the specified date according to local time.

getFullYear
Returns the year (4 digits for 4-digit years) of the specified date according to local time.

getHours
Returns the hour (0-23) in the specified date according to local time.

getMinutes
Returns the minutes (0-59) in the specified date according to local time.

getMonth
Returns the month (0-11) in the specified date according to local time.

getSeconds
Returns the seconds (0-59) in the specified date according to local time.

Additional space for question 4

```
<html>
<head>
    <title>Clock and Date</title>

      <script Language="JavaScript">

      var month =
['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec
'];

      function gettheDate() {
            Today = new Date();
            TheDate =  Today.getDate() + "/" + month[Today.getMonth()]
+ "/" + Today.getFullYear();
            document.getElementById('thedate').value = TheDate;
      }

      function startclock () {
            gettheDate()
            showtime();
      }

      function showtime () {
            var now = new Date();
            var hours = now.getHours();
            var minutes = now.getMinutes();
            var seconds = now.getSeconds()
            var timeValue = "" + ((hours >12) ? hours -12 :hours)
            timeValue += ((minutes < 10) ? ":0" : ":") + minutes
            timeValue += ((seconds < 10) ? ":0" : ":") + seconds
            timeValue += (hours >= 12) ? " P.M." : " A.M."
            document.getElementById('thetime').value = timeValue;
            setTimeout("showtime()",1000);
      }

      </script>

</head>
<body onload="startclock()">
      <table>
            <tr>
                  <td>Time<br /><input type="text" id="thetime" size=12
value=""></td>
                  <td>Date<br /><input type="text" id="thedate" size=12
value=""></td>
            </tr>
      </table>
</body>
</html>
```

5.- [25 points] Using JavaScript and HTML implement a simple application that gives the illusion that an arrow extends and shrinks when a button is clicked. For example, Figures 2 shows the arrow as it is being extended after the user clicks the extend button.

To draw the arrow use three small images empty.gif, end.gif and mid.gif, which draw an empty box, an arrow head and a straight line, respectively. Assume all images are available on the current directory. The collapsed version of the arrow should show only the arrow heard, while the extended arrow should show 7 straight lines topped with an arrow head. The total time it takes to extend or collapse the arrow show be around 350 milliseconds.

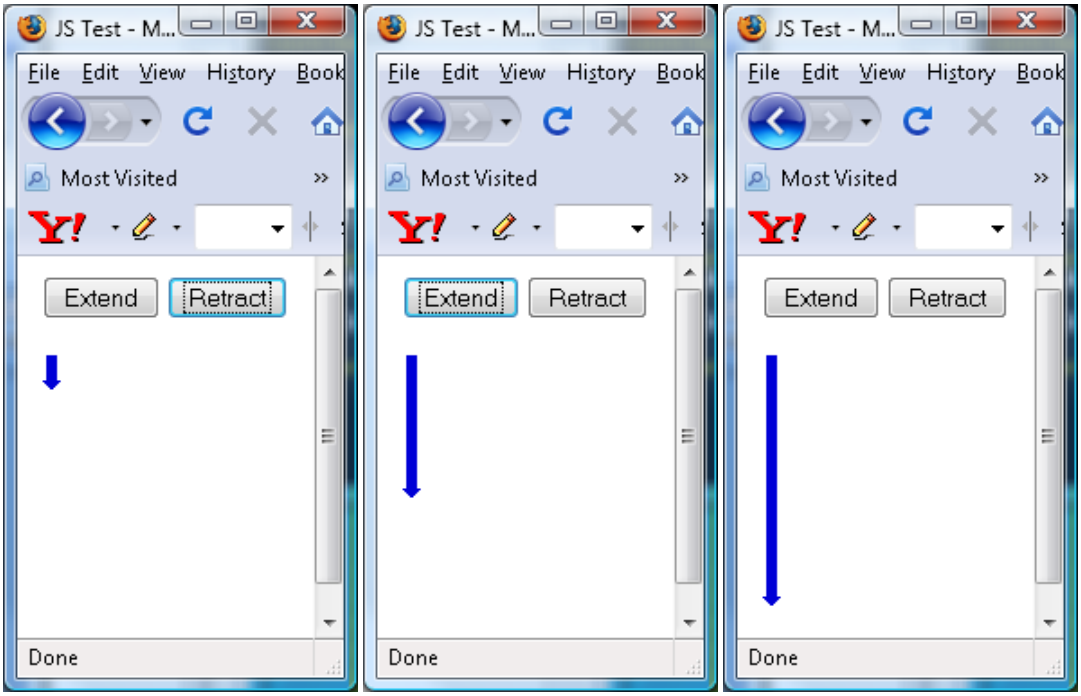

Figure 2

Additional space for question 5

```
<html>
<head>
<title>Arrow Animation</title>
<script language="JavaScript">

var first=0  // The index of the first image in the arrow
var i=first-1
function drawLine() {
  if (++i < (first+6)) {
     document.images[i].src="mid.gif";
     document.images[i+1].src="end.gif";
     setTimeout("drawLine()",50);
     }
  }

function clearLine() {
  if (--i >= first) {
     document.images[i+1].src="empty.gif";
     document.images[i].src="end.gif";
     setTimeout("clearLine()",50);
     }
  }

//  -->
</script>
</head>
<body>
      <input type="button" value="Extend" onclick="drawLine()">
      <input type="button" value="Retract" onclick="clearLine()"><br>
      <br>
      <img src="end.gif"   height=20 width=10><br>
      <img src="empty.gif" height=20 width=10><br>
      <img src="empty.gif" height=20 width=10><br>
      <img src="empty.gif" height=20 width=10><br>
      <img src="empty.gif" height=20 width=10><br>
      <img src="empty.gif" height=20 width=10><br>
      <img src="empty.gif" height=20 width=10><br>
      <img src="empty.gif" height=20 width=10><br>
</body>
</html>
```

6.- [25 points] Using exclusively CSS and the HTML tags <html>,<head>,<link>,<body>, and design a page that mimics the layout shown in Figure 3.

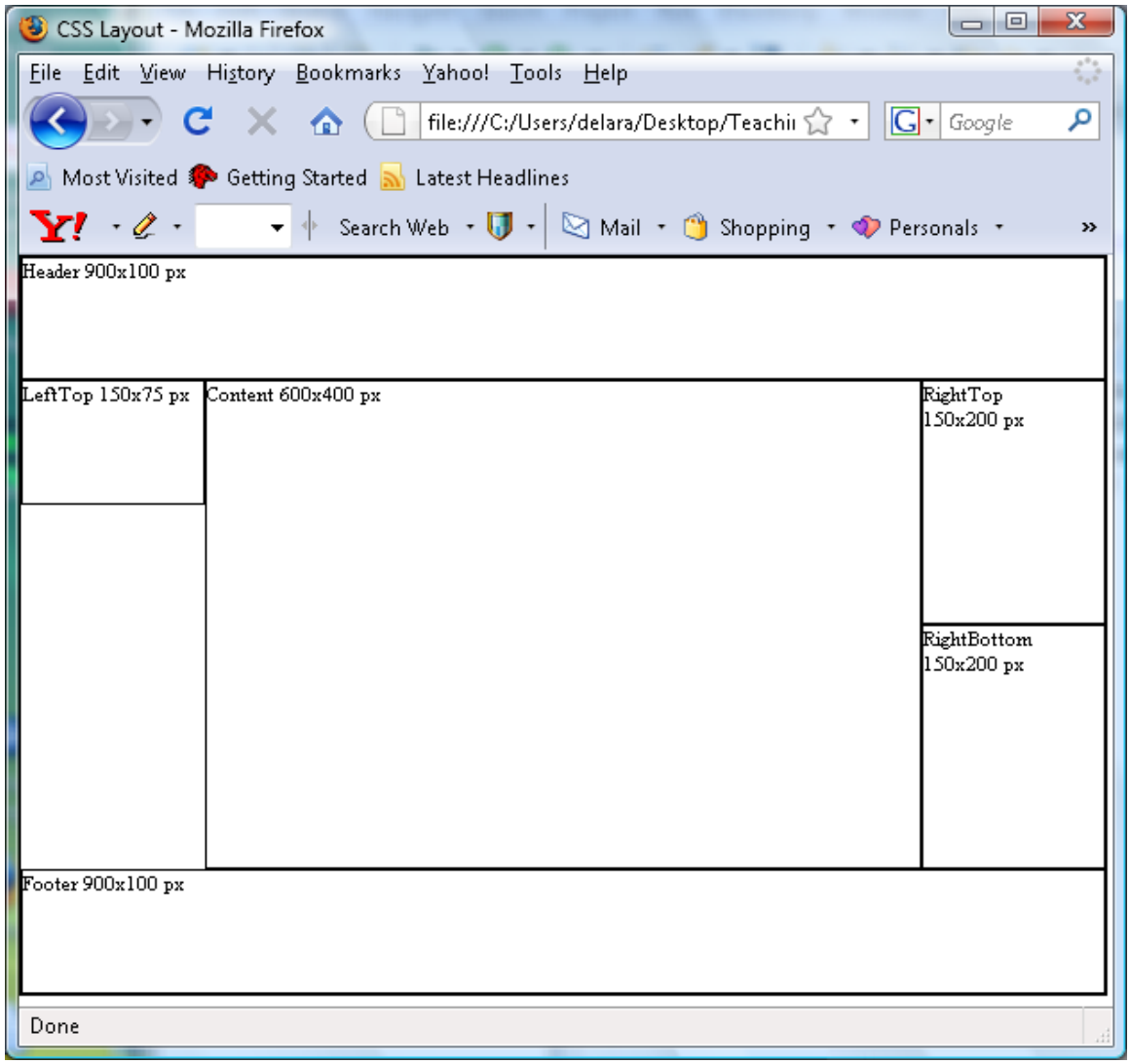

Figure 3

Additional space for question 6

layout.html

```
<html>
<head>
	<title>CSS Layout</title>
	<link href="layout.css" rel="stylesheet" type="text/css" />
</head>
<body>
	<div id="header">Header 900x100 px</div>
	<div id="righttop">RightTop <br />150x200 px</div>
	<div id="lefttop">LeftTop 150x75 px</div>
	<div id="content">Content 600x400 px</div>
	<div id="rightbottom">RightBottom 150x200 px</div>
	<div id="footer">Footer 900x100 px</div>

</body>
</html>
```

layout.css

```
body, div {
	padding:0;
	margin:0;
	border:solid 1px;
	width: 900px;
}
#header, #footer {
	width:898px;
	clear: both;
	height: 100px;
}
#righttop, #rightbottom{
	float:right;
	width:150px;
	height: 200px;
}
#rightbottom {
	clear: right;
}
#content {
	float:left;
	width:591px;
	height: 403px;
}
#lefttop {
	width:150px;
	height: 100px;
	float: left;
}
```