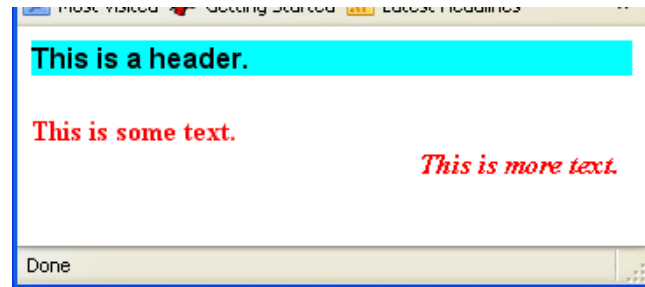


COMP 322 Internet Systems Fall 2008 Exam 1—Solutions

50 points total

1 (13 pts.). The following HTML document (listed with gaps) is rendered as shown at right. The topmost sentence is an h4 header; it has a cyan background and is in Helvetica font. The next sentence down is the first part of a div element. The last sentence is an em element in the same paragraph. The HTML document references an external style sheet named `elprob1.css`. This style sheet defines



- a rule for h4 elements, specifying that they are in Helvetica font with a cyan background,
- a rule for em elements, specifying that they are shifted down ½ centimeter and right 100 pixels relative to their normal positions, and
- a class, `emph`, specifying red color and bold.

In the following listing, missing code is identified by Greek letters. The letters are repeated after the listing along with descriptions of the missing code; you there supply the missing code. After the descriptions of the missing code, you are asked to supply the entire style sheet (the file `elprob1.css`).

```
<html>
<head>
  <title>Problem 2</title>
  _____
  _____
</head>
<body>
  <h4>This is a header.</h4>
  <div _____>This is some text.
    <em>This is more text.</em>
  </div>
</body>
</html>
```

α (2 lines, 3 pts.): Specify a link to the file `elprob1.css`, used as a style sheet.

Answer

```
<link rel = "stylesheet" type = "text/css"
      href = "elprob1.css">
```

β (2 pts.): Apply the class `emph` to this div element.

Answer

```
class = "emph"
```

Write the entire external style sheet, the file `elprob1.css`. (8 pts) (This is done in 3 lines.)

Answer

```
h4    { background-color: cyan; font-family: helvetica }
em    { position: relative; left: 100px; top: 0.5cm }
.emph { color: red; font-weight: bold }
```

2 (16 pts.). The HTML document on the following page consists of JavaScript code executed when the document is loaded. It prompts the user for his/her name then outputs in an alert box either

Welcome back, _____

or

You are not registered, _____

depending on whether the last name is recognized as a registered user's name. (In the code, only three user's names are checked; a realistic program would check many more names.) What occupies the missing text above is the first name the user enters. The user might enter just one name. Or, besides entering his/her last name, a user might enter his/her first name and even middle name, but only the last name is checked. So, when the full name is converted to an array of strings, the important elements are the one at index 0 and the last element. Function `start`, invoked when the document is loaded, does most of the work. The other function, `user`, returns `true` if the name passed to it is recognized as a user's name.

Note that, in JavaScript, strings are compared by value not by reference (as in C or C++). Thus, if `str1` and `str2` are strings, then

```
str1 == str2
```

compares them for equality while

```
str1 != str2
```

compares them for inequality.

In the following listing, missing code is identified by Greek letters. These letters are repeated on the next page; you there supply the missing code.

```

<html>
<head>
<title>Exam 1</title>
<script>
  var users = ["Smith", "Jones", "Johnson"];

  function start()
  {
    var name, nameArray,
        greeting;

    name = α _____,
          _____ );
    nameArray = β _____;

    if ( γ _____ )
      greeting = "Welcome back, " + nameArray[0];
    else
      greeting = "You are not registered, " + nameArray[0];

    window.alert( greeting );
  }

  function user( name )
  {
    var i = 0;

    while ( δ _____ )
      i ++;

    if ( i == users.length )
      return false;
    else
      return true;
  }
</script>
</head>

<body onload = "start()">
</body>
</html>

```

α (3 pts.): Open a prompt dialog box that has the prompt "Enter your name" and default "John Doe".

Answer

```

window.prompt( "Enter your name",
              "John Doe" )

```

β (3.5 pts.): Convert the string assigned to name to an array of its substrings delimited by spaces.

Answer

```

name.split( " " )

```

γ (4.5 pts.): Call function user(), passing it the last element in array nameArray.

Answer

```

user( nameArray[nameArray.length - 1] )

```

δ (5 pts.): The condition that index i hasn't gone beyond the end of the array users and name is different from users[i].

Answer

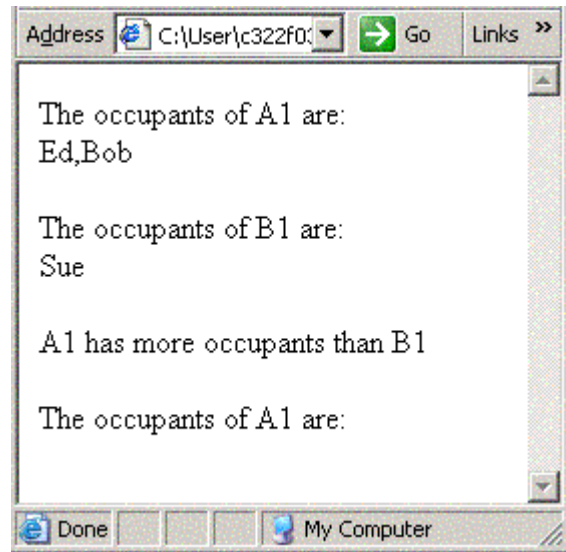
```

i < users.length && name != users[i]

```

3 (21 pts.). The following HTML document links to two JavaScript files, one of which (`Apartment.js`) defines the object prototype `Apartment`, which has two instance variables, `number` (a string representing the apartment number, e.g., “A2”), and `occupants` (an array of strings that are the names of the occupants). There are the usual `get` instance methods for `number` and `occupants`, and there is the usual `set` instance method for `number`. There are two instance methods to modify `occupants`. Method `addOccupant()` takes a string (a name) and adds it to the `occupants` instance array, and `expelAll()` sets the `occupants` instance array of the current instance to the empty array. The constructor function, `Apartment()`, has one formal parameter, for the number. The default for instance variable `number` is “A0”; instance variable `occupants` is always initialized to the empty array. Finally, there is one class method, `moreOccupants()`, which takes two `Apartment` instances, `apt1` and `apt2`, and returns `true` if `apt1` has more occupants than `apt2` (else returns `false`).

The second JavaScript file (`testApartment.js`), the test file for this prototype, first constructs two instances of `Apartment`. One, with number “A1”, is assigned to variable `apt1`, and the other, with number “B1”, is assigned to `apt2`. It then adds occupants “Ed” and “Bob” to `apt1` and “Sue” to `apt2`. Next, it sends marked-up text to the browser that reports the occupants of each apartment. Then it sends text to the browser reporting which apartment has more occupants. Finally, the test code expels all occupants of `apt1` and again sends text to the browser reporting all occupants of `apt1` (of which there are now none). The rendering of the HTML document is shown at right.



```
<html>
<head>
<title>Problem 2</title>
<script type = "text/javascript"
      src = "Apartment.js">
</script>
<script type = "text/javascript"
      src = "testApartment.js">
</script>
</head>
</html>
```

The two JavaScript files are listed below with gaps labeled with Greek letters. After each file, the labels are repeated with descriptions of the code that should fill the corresponding gaps. You there supply the code.

File Apartment.js:

```
function Apartment(number)
{
  this.number = α_____i;
  this.occupants = β____i;
  this.getnumber = γ_____i;
  this.setnumber = δ_____i;
  this.getoccupants = ε_____i }i;

  this.toString = this.getnumber;
}

function Apartment_addOccupant( occupant )
{
  ζ_____i;
}

function Apartment_expelAll()
{
  η_____i;
}

Apartment.prototype.addOccupant = Apartment_addOccupant;
Apartment.prototype.expelAll = Apartment_expelAll;

function Apartment_moreOccupants( apt1, apt2 )
{
  return θ_____i;
}

Apartment.moreOccupants = Apartment_moreOccupants;
```

α (1.5 pts.): the formal parameter number or (if this isn't supplied) the default "A0"

Answer

number || "A0"

β (1 pt.): the empty array

Answer

[]

γ (1.5 pts.): the usual get method for number

Answer

function() { return this.number; }

δ (1.5 pts.): the usual set method for number

Answer

function(number) { this.number = number; }

ε (1.5 pts.): the usual `get` method for `occupants`

Answer

```
function() { return this.occupants; }
```

ζ (2 pts.): Add formal parameter `occupant` to the end of the `occupants` array of the current instance.

Answer

```
this.occupants[ this.occupants.length ] = occupant
```

η (2 pts.): Set the `occupants` array of the current instance to the empty array.

Answer

```
this.occupants = []
```

θ (2.5 pts.): an expression that is true if apartment `apt1` has more occupants than apartment `apt2`

Answer

```
apt1.occupants.length > apt2.occupants.length
```

File testApartment.js:

```

var apt1 = ι_____,
    apt2 = κ_____;

λ_____;
_____;
_____;

document.write( "<p>The occupants of " + apt1 + " are:<br>"
                + apt1.getOccupants() + "</p>" );
document.write( "<p>The occupants of " + apt2 + " are:<br>"
                + apt2.getOccupants() + "</p>" );

if ( Apartment.moreOccupants( apt1, apt2 ) )
    document.write( "<p>" + apt1 + " has more occupants than "
                   + apt2 + "</p>" );
else
    document.write( "<p>" + apt1 + " has more occupants than "
                   + apt2 + "</p>" );

μ_____;

document.write( "<p>The occupants of " + apt1 + "
are:<br>"
                + apt1.getOccupants() + "</p>" );

```

ι (1.5 pts.): a new apartment with name "A1"

Answer

```
new Apartment( "A1" )
```

κ (1.5 pts.): new apartment with name "B1"

Answer

```
new Apartment( "B1" )
```

λ (3 pts.) (3 lines): Add occupants Ed and Bob to apt1 and Sue to apt2.

Answer

```
apt1.addOccupant( "Ed" );
apt1.addOccupant( "Bob" );
apt2.addOccupant( "Sue" );
```

μ (1.5 pts.): Expel all the occupants from apartment apt1.

Answer

```
apt1.expelAll()
```