

COMP 322 Internet Systems Fall 2006 Assignment 5

Due in the digital drop box Wednesday, Oct. 25 by 11:00 PM.

1. The following code contains a JavaScript function (with a fragment removed), `moveToEnd()`, with three formal parameters: `ar` (an array), `start` (an index in `ar`), and `len` (an integer). It returns an array that is like `ar` except the `len` elements beginning at index `start` have been removed from their positions in the array and added to the end. It does not change the array `ar`. For example, if `testAr` is `[1, 3, 5, 7, 9]`, then

```
moveToEnd(testAr, 1, 2)
```

returns `[1, 7, 9, 3, 5]`. If there are not `len` elements from index `start` to the end of the array, then this function should simply return a copy of `ar`.

Supply the missing code. Note that test code is already included. This file (with the fragments missing) is available on the Web page where you got this assignment.

Hint: Use the `splice()` and `concat()` methods. Note that `ar.splice(start, len)` does not change `ar`, and returns `[]` (the empty array), if there are not `len` elements from index `start` to the end of array `ar`.

```
<html>
<head>
<title>Assignment 5, Problem 1</title>
<script type = "text/javascript">
  function moveToEnd( ar, start, len )
  {
    var ar1 = ar.concat(); // Make a copy of ar

    // Fill in the missing code

  }

  var testAr = [ 1, 3, 5, 7, 9 ];
  document.write( "<p>testAr: [" + testAr + "]<br>"
    + "moveToEnd(testAr, 1, 2): ["
    + moveToEnd(testAr, 1, 2) + "]</p>" );
</script>
</head>
<body>
</body>
</html>
```

2. The following code (with fragments missing) prompts the user for his/her name and age. It then attempts to match the string input from the prompt against a regular expression that defines the legal input. If this match succeeds, it echoes back the name and age entered. Otherwise, it flags the input as illegal. The regular expression does not distinguish between upper- and lowercase letters. It defines the following pattern:

- the beginning-of-string anchor (so that the user can't enter superfluous characters at the beginning),
- the sequence of letters (upper- or lowercase) 'n', 'a', 'm', 'e',
- a colon (':'),
- zero or more spaces or a tab (`\t`),
- one or more word characters (remembered for backreference),
- a semicolon (';'),
- zero or more spaces or a tab (`\t`),
- the sequence of letters (upper- or lowercase) 'a', 'g', 'e',
- a colon (':'),
- zero or more spaces or a tab (`\t`),
- a sequence of one to three digits (remembered for backreference), then
- the end-of-string anchor (so that the user can't enter superfluous characters at the end),.

The following are examples of valid input:

```
name: Bob; age: 21
```

```
name:Fred; age: 33
```

Since the match is case insensitive, the following are also valid input:

```
Name: Bob; Age: 21
```

```
NAME: bOB; aGe: 21
```

When you echo back the name and age, you'll use two of the implicit variables `RegExp.$1` to `RegExp.$9`. Recall that these have the values that were matched to the parts of the regular expression that were enclosed in parentheses. The thing to watch out for here is that you will have to use parentheses also to group alternatives (the operands of `|`). These parenthesized expressions must be counted when you determine which of these implicit variables to use.

Each missing fragment is labeled with a Greek letter. Each Greek letter is repeated after the code with a description of the missing code at the indicated position. This file (with the fragments missing) is available on the Web page where you got this assignment.

```
<html>
<head>
<title>Assignment 5, Problem 2</title>
<script type = "text/javascript">
  var str,
      reg = α_____ ;

  str = prompt( "Enter your first name and age in the format\n"
              + "Name: FirstName; Age: 99",
              "Name: FirstName; Age: 99" );
  if ( str.match( reg ) )
    document.write( "<p>Your name is " + β_____
                  + " and your age is " + γ_____ + "</p>" );
  else
    document.write( "<p>Illegal input</p>" );
</script>
</head>
<body>
</body>
</html>
```

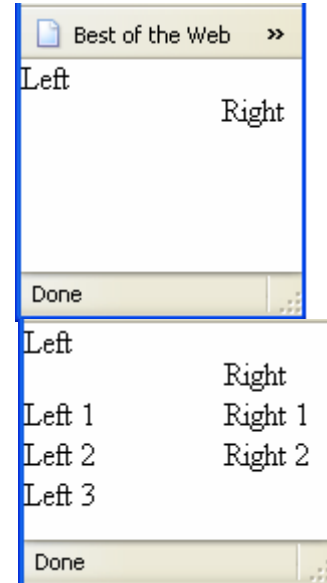
α : The required regular expression

β : The implicit variable (one of `RegExp.$1` to `RegExp.$9`) that matches the user's name.

γ : The implicit variable that matches the user's age.

3. This problem requires you to use the Node instance method `cloneNode(deep)`, which returns a duplicate of the node that invokes it. The duplicate node has no parent. Cloning an `Element` copies all attributes and their values. Argument *deep* is Boolean. If it is `true` and the node is an `Element`, then a *deep copy* is made: the copy includes all the content of the original. Otherwise, the copy is made of only the original node (without any content even if the original is an `Element`).

The following `.html` file (with gaps) specifies a document whose body has two `p` elements, one with content `Left`, the other with content `Right`. The first has `id` “`p1`”, the second `id` “`p2`”. Both have the CSS properties `position` with value “`absolute`” and `top` with value “`0`”. The first has “`0`” for the value of its `left` property; the second has “`100`” for the value of this property. The initial rendering is shown at right. Note that Mozilla offsets the second paragraph a bit from the top. When `Left` is clicked, a new paragraph with content `Left 1` appears below `Left`. If `Left` is clicked again, `Left 2` appears below `Left 1`. Each time `Left` is clicked, a new paragraph `Left n` (`n` being the number of times `Left` has been clicked) appears below the last paragraph. Clicking `Right` has a similar effect except that the new content is `Right n` and appears below the `Right` paragraph. Thus, for example, after `Left` has been clicked three times and `Right` twice (in any order), the rendering is as shown at right.



This behavior is achieved with a function `doit`, which is assigned as the `click` event listener to both paragraphs. Five global variables are maintained. Variables `leftCounter` and `rightCounter` keep track of how often the left and right paragraphs, respectively, have been clicked. Variables `leftTop` and `rightTop` are incremented by the value of variable `topIncr` each time the left and right paragraphs, respectively, are clicked; they give the offsets of the two paragraphs from the top of the browser window.

When a paragraph is clicked, `doit` first makes a deep copy of the node corresponding to the clicked paragraph. (The `Event` object has a reference to this object.) If the clicked paragraph was the left one (determined again with information from the `Event` object), then the value of `leftCounter` is appended to the paragraph’s content, the value of its `top` property is set to `leftTop`, and `leftTop` is incremented by `topIncr`. If the clicked paragraph is the right paragraph, analogous steps are performed. In any case, the clone is added to the children of the body.

The following listing has gaps labeled with Greek letters. These letters are repeated after the listing with a description of what should fill the corresponding gap. The file (with gaps) can be downloaded from where you downloaded this assignment.

```
<html>
<head>
<title>Assignment 5, Problem 3</title>
<script type="text/javascript">
  var leftCounter = 0,
      rightCounter = 0,
      topIncr = 20,
      leftTop = 20,
      rightTop = 20;

  function start()
  {
    α _____;
    β _____;
  }

  function doit( event )
  {
    var newP = γ _____;

    if ( δ _____ ) {
      newP.innerHTML += " " + ++leftCounter;
      ε _____;
      leftTop += topIncr;
    }
    else {
      newP.innerHTML += " " + ++rightCounter;
      ζ _____;
      rightTop += topIncr;
    }

    η _____;
  }
</script>
</head>
<body onload="start()">
  <p id="p1" style="position: absolute; left:0; top: 0">
    Left
  </p>
  <p id="p2" style="position: absolute; left:100; top: 0">
    Right
  </p>
</body>
</html>
```

- α : Add `doit` as the `click` event listener for the first paragraph. Do this with one statement. Do not use a variable for the reference to the node corresponding to the first paragraph.
- β : Add `doit` as the `click` event listener for the second paragraph. Again do this with one statement.
- γ : A clone of the clicked paragraph. (Use the information in `event`.)
- δ : An expression that is true if the first paragraph was clicked. (Use the information in `event`.)
- ϵ : Set the value of the `top` property of this paragraph to `leftTop`. Use `setProperty`.
- ζ : Set the value of the `top` property of this paragraph to `rightTop`. Use `setProperty`.
- η : Add the clone to children of the body of the document.