

Both problems are equally weighted.

1. The following HTML document executes a function `go` when it is loaded. This function prompts for a product descriptor and checks whether the string the user enters is a valid product identifier. A product descriptor consists of the following in the order listed (and nothing more—**hint**: use some anchors):

- An upper case letter (the series)
- Two or three digits (the number)
- One of the uppercase letters X, Y, or Z (the version)

The series, number, and version are all remembered. If the string supplied by the user is a valid product descriptor, then the function raises an alert box whose text is a string of the form

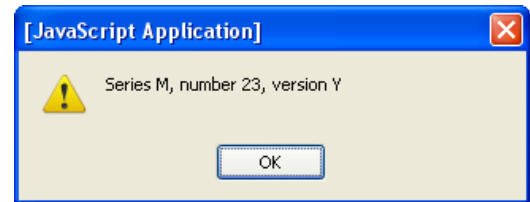
Series `<series>`, number `<number>`, version `<version>`

where `<series>`, `<number>`, and `<version>` are the substrings remembered from the match. For example, if in the prompt the user enters

M23Y

then the alert shown at right is raised.

The following is a listing of the document with gaps identified by Greek letters. After the listing, the Greek letters are repeated with descriptions of the corresponding code. You then supply the missing code.



```
<html>
<head>
  <title>Quiz 3, Problem 1</title>
  <script type="text/javascript">
    function go()
    {
      var regexp = α,
          str = window.prompt( "A product descriptor" );

      if ( str.match(regexp) ) {
        var str1 =
          β
          _____;

        window.alert( str1 );
      }
    }
  </script>
</head>
<body onload="go()">
</body>
</html>
```

α : the regular expression for matching valid product descriptor (as defined above)

Answer: `/^([A-Z])(\d{2,3})([XYZ])$/`

β : the string, as described above, that's the content of the alert box if there's a match. (This uses the three strings remembered from the match.)

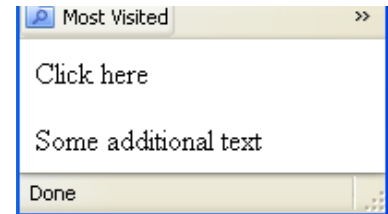
Answer: `"Series " + RegExp.$1 + ", number " +`
`RegExp.$2 + ", version " + RegExp.$3;`

2. The rendering of the document shown below with gaps initially is simply a paragraph whose content is **Click here**. The paragraph has an `id` attribute with value `"pclick"`. The body of the document also contains a `div` element whose content is initially empty and that has `"show"` as the value of its `id` attribute. When the user clicks the paragraph, the string

Some additional text

is made the content of the `div` element, and the rendering becomes as shown at right.

When the document is loaded, function `go()` is executed. This function adds the function `addText()` as a listener for click events in the paragraph. Function `addText()` simply makes the string **"Some additional text"** the content of the `div` element. The bodies of the two functions are missing in the listing. After the listing, you supply these bodies.



```
<html>
<head>
  <title>Quiz 3, Problem 2</title>
  <script type="text/javascript">
    function start()
    {
      <You supply the body>
    }
    function addText()
    {
      <You supply the body>
    }
  </script>
</head>
<body onload="start()">
  <p id="pclick">Click here</p>
  <div id="show"></div>
</body>
</html>
```

The body of `start()`:

Answer

```
var para = document.getElementById("pclick" );
para.addEventListener("click", addText, false);
```

The body of `addText()`:

Answer

```
var sh = document.getElementById("show" );
sh.innerHTML = "Some additional text";
```