/** Scripts for managing the array-example document. Note: this is pret
ty rudimentary; the current logic does NOT attempt to reset and restart the
game through any user interaction with a control ... instead, the user is asked to "reload" the page, which will clear and reset all of the
top-level variables and call the generateQuestion procedure. */

/* Variable declarations */

var colorNames = ["blue", "green", "red"]; // array of color names (strings).
var currentIndex = 0; // keep track of currently selected color name.
var correctResponses = 0; // for keeping score (maybe later?)

//_______end variable declarations.

//_______methods (functions & procedures) definitions.

/** preconditions: assumes that the array and currentIndex are set
* by the caller BEFORE calling this procedure!
* postconditions: generates a question by choosing the current color
* name (by currentIndex) from the colorNames array.
* Note: called by both onload event and by the evalResponse() procedure.
*/

function generateQuestion() {
  updateScore();
  uncheckRadioButtons(); // clears any errant settings between question s
  /*
  * Because this could be called from evalResponse while at the
  * last element in the array (looking at the last color), we need
  * to check here.
  */
  if( currentIndex >= colorNames.length ) { // note the negation of "<"
    alert("Out of colors. Reload this page to play again, or close th e tab.");
    return;
  }
  document.getElementById("prompt").innerHTML="What is the complement for the color " + colorNames[ currentIndex ] + "? ";
}

/** preconditions: none.
* postconditions: unchecks (clears) all radio buttons named "complement ".
*/
function uncheckRadioButtons() {
    var radioButtons = document.getElementsByName("complement");
    /* an example of a "bounded iteration," because we
    * know the size (length) of the radioButtons array
    * at the time that the code below is executed.
    */
    for( var index=0; index < radioButtons.length; index++ ) {
        radioButtons[ index ].checked=false;
    }
}

function updateScore() {
    document.getElementById("score").innerHTML="Current Score: " + correctResponses + " /" + colorNames.length;
}

function evalResponse() {
    if( currentIndex >= colorNames.length ) { // note the negation of "<"
        alert( "Out of colors. Reload this page to play again, or close the tab." );
        return;
    }
    /* retrieves the next color name from the
    * array, using the currentIndex. AFTER doing this, it
    * updates the currentIndex by 1.
    * This code can be written on two lines for clarity:
    * forColor = colorNames[ currentIndex ];
    * currentIndex = currentIndex + 1;
    */
    var forColor = colorNames[ currentIndex++ ];
    /* This is the "pattern" for using radio buttons. You need to concern
* yourself with the "details" at this time, unless you wish to
* use radio buttons in your own documents.
* Radio Buttons, by the way, are more commonly used in "forms process
* ing,"* which does not concern us because we are writing "client side" scr
ipts.
* /
* if(( forColor == "red" && document.getElementById("cyan").checked ===
   true ) ||
  ( forColor == "green" && document.getElementById("magenta").checke
d === true ) ||
  ( forColor == "blue" && document.getElementById("yellow").checked
== true ) )
{
  correctResponses++;
  /* we can either update the scores within this procedure,
   * or we can depend upon the common procedure "generateQuestion()"
   * to update the score before it generates the "next" question.
   * I leave this logic here for you to uncomment just to satisfy
   * your curiosity.
   */
  //updateScore();
} else {
  correctResponses--;
  /* see the comment above. */
  //updateScore();
}
/*
 * Ask: why is it "safe" to call this here?
 * Hint: look at the first line of code in this procedure..
 */
generateQuestion();
} //_______end methods definitions.