World Wide Web Information Systems Development

Assignment 4

Overview

This assignment explores using JavaScript to manipulate HTML and CSS. This assignment can be done locally using your web browser or via a web server.)

Task

Located later on this page download the provided **a4.html** and **a4.css** files. Create an empty **a4.js** file in the same directory and load **a4.html** in to your web browser. You will see **three squares in the middle of the browser window**. There are three tasks in this assignment:

1. Use JavaScript to associate clicking the Update colour square so that it updates the square's colour (as described below).
2. Use JavaScript to associate clicking the Update counter square so that it updates the text in the square with the counter's value.
3. Use JavaScript to associate clicking the Update background square so that it toggles the background colour between black and white.

All items are described below and only involve edits to the a4.js file. Remember to frequently reload your web page after saving changes to check the results. Should you've issues, use Firefox' or Chrome's Web Developer debugger tool.

Associating Events With Specific Elements

The provided HTML has some elements that have id attributes set. In this assignment, you will only be editing a4.js. How does one associate events with HTML elements when one cannot edit the HTML? Since the **a4.js** file is already being loaded by the HTML file, one simply has to associate the **onLoad** event with a function in JavaScript. This is done by placing the following line at the top of **a4.js**:

```
1. window.addEventListener("load", init);
```

where **init** is the name of the function that you will write to associate each event and element.

Writing init()

To define a function in JavaScript prefix the name with function and after the name add parentheses and any needed arguments' names, e.g., you can define **init()** as follows:

```
1. function init()
2. {
3. }
```

Within **init()**, one needs to "find" the elements with a specific **id** attribute. This is done by calling **document.getElementById()**, e.g.,

```
1. var element_node = document.getElementById("some-id-value");
```

If the **id** value is found, then a non-**null** value is returned, otherwise **null** is returned. In this assignment will will assume the **id** values searched for are always found.

Once the element node has been found, there are a number of functions and data values that can be accessed. To associate an event with an element node call **addEventListener("eventName", function_to_call)**, e.g.,

```
1. element_node.addEventListener("click", myfunc);
```

which would associate (mouse) clicking (i.e., the "onClick" event) in that element node with calling the function myfunc.

In this assignment, you want to establish the following associations:

- the **onClick** event in the id="**colour**" element with calling the **update_colour** function,

- the **onClick** event in the id="**counter**" element with calling the **update_counter** function, and,
- the **onClick** event in the id="**background**" element with calling the **update_background** function.

Writing update_colour()

When a function is associated with an event, JavaScript will pass a (single) object to that function containing event information. Since the name "event" has special meaning in JavaScript, the name of this argument is usually abbreviated to evt, e.g.,

```
1. function update_colour(evt)
2. {
3. }
```

Before the update_colour function, however, write:

```
1. var last_colour = 0;
```

which declares a global variable in JavaScript. You will use this to toggle through eight colours 0 to 7 where the 3 bits indicate the absence (if 0) or presence (if 1) of the colour components red, green, and blue respectively.

- Like C, Java, etc. one can use bitwise-AND, i.e., "&", to test whether or not a bit is set, e.g., var red = (last_colour & 4) ? '0' : 'F';
- In JavaScript if a variable/value is a character or a string, one can concatenate such using the + (plus) operator, e.g., "#"+red+"FF" would produce the string with the following content: "#", followed by whatever was in red, followed by "FF".
- By starting last_colour at 0, one can ++last_colour to get the next colour. When last_colour == 8, set last_colour back to 0.

A problem with changing the colour is that it might become unreadable with respect to the background colour, thus, this function will set both as follows:

- Determine the red, green, and blue components from last_colour.
  - Hint: The red bitmask has a value of 4, green bitmask has a value of 2, and blue bitmask has a value of 1.
- Compute the foreground colour to be set by setting a variable with the CSS hexadecimal colour based on red, green, and blue.
  - Hint 1: Remember that CSS hexadecimal colours start with "#". Since this variable is a string, start by assigning "#" to it.
  - Hint 2: Use string concatenation (i.e., the + operator) to append additional hexadecimal digits. You only need one digit per colour which will either be "0" or "F" depending on whether or not it is set.
- Compute the background colour to be set by inverting the colour you computed in the previous bullet, i.e., where "F" appears set "0" and where "0" appears set "F".

Now that the background and foreground colours have been computed, you want to set them. Within a function triggered by an event, the **this** keyword will refer to the element. To set an element's CSS attribute access its **.style** member followed by the CSS property and assign that property to the string you want to set it to. For example, suppose you want to set the CSS **color** attribute to the string "**#FFF**", then you would write:

```
1. this.color = "#FFF";
```

For CSS attributes that have dashes, "-", in their names, **delete the dash and capitalize the next letter**, e.g.,

```
1. this.backgroundColor = "#000";
```

ASIDE: A dash is the subtraction operator in JavaScript. JavaScript will properly map the name to the proper CSS attribute name.

Finish this function by assigning the foreground and background colours as described above.

Before continuing further test your code by repeatedly clicking in the update colour square after saving **a4.js** and reloading **a4.html**. If everything works, the square will cycle through 8 colours with the text always remaining visible.

Writing update_counter()

Declare a global variable called **counter** and set it to **0**. After such define an **update_counter** function whose body does the following:

- Assigns **this.innerHTML** to the string **" Count = " + counter + ". "**.
  - Notice that the string is HTML text.
  - The innerHTML property will destroy any HTML inside the element that this refers to.
- Increments counter (e.g., use ++).

Save and reload, then test by repeatedly clicking in the update counter square to see if the text is updated.

Writing update_background()

Declare a global variable called **bcolor** and set it to **0**. After such define an **update_background** function whose body does the following:

- if **bcolor** is **0**, then set the page background to "#000", i.e., black, and set bcolor to 1.
- If **bcolor** is **1**, then set the page background to "#FFF", i.e., white, and set bcolor to 0.

To set the entire page's CSS attributes, you need to use:

```
1. document.documentElement.style
```

followed by ".", then the property needed, e.g., ".backgroundColor", and then assign it to the desired colour.

Save and reload, then test by repeatedly clicking in the update background square to see if the page's background colour changes between white and black.

Submission

Since a4.html and a4.css were provided, **only submit your a4.js file.**

a4.css	10 February 2021, 4:55 PM
a4.html	10 February 2021, 4:55 PM

Submission status

Attempt number	This is attempt 1.
Submission status	No attempt
Grading status	Not marked
Due date	Wednesday, 24 February 2021, 11:59 PM
Time remaining	6 days 8 hours

Grading criteria

Update colour	There are many errors, warnings, issues, and/or code does not work as required. ***0 points***	There are some errors, warnings, or code execution issues. ***1 points***	There is an error, a warning, or a coding issue that is simple to correct and with such, everything works as required. ***2 points***	Update colour works as required without any errors or warnings (including in the JavaScript console). ***3 points***
Update counter	There are many errors, warnings, issues, and/or code does not work as required. ***0 points***	There is an error, a warning, or a coding issue that is simple to correct and with such, everything works as required. ***1 points***	Update counter works as required without any errors or warnings (including in the JavaScript console). ***2 points***	
Update page background	There are many errors, warnings, issues, and/or code does not work as required. ***0 points***	There is an error, a warning, or a coding issue that is simple to correct and with such, everything works as required. ***1 points***	Update page background works as required without any errors or warnings (including in the JavaScript console). ***2 points***	
Assignment Requirements Conformance	Not all instructions were followed. ***0 points***	Some instructions were followed. ***1 points***	Nearly all instructions were followed. ***2 points***	All instructions were followed. ***3 points***
Late Penalty (Deduction)	Late. All marks deducted. ***-10 points***	Late but submitted within 24 hours of deadline or late without penalty deadline. ***-2 points***	Not late or late without penalty (as decided by or with permission of instructor). ***0 points***	

Last modified -

Submission comments ▸ Comments (0)

Add submission

You have not made a submission yet.

◄ Week 4 Lecture Files

Jump to...

Week 5 Lecture Files ►