

ICT-4570 Homework 5

Purpose

In this assignment you will have the opportunity to explore the use of the HTML5 canvas and event handling in JavaScript to provide an interactive experience to users

What to Hand In

Canvas submission instructions:

Please combine multiple files into a single "zip" archive, and save it in a location that you will remember. When you are ready to submit the assignment solution, open the assignment in Canvas, and click on the Submit for Evaluation button at the top, attach the file, and click Submit for Evaluation at the bottom.

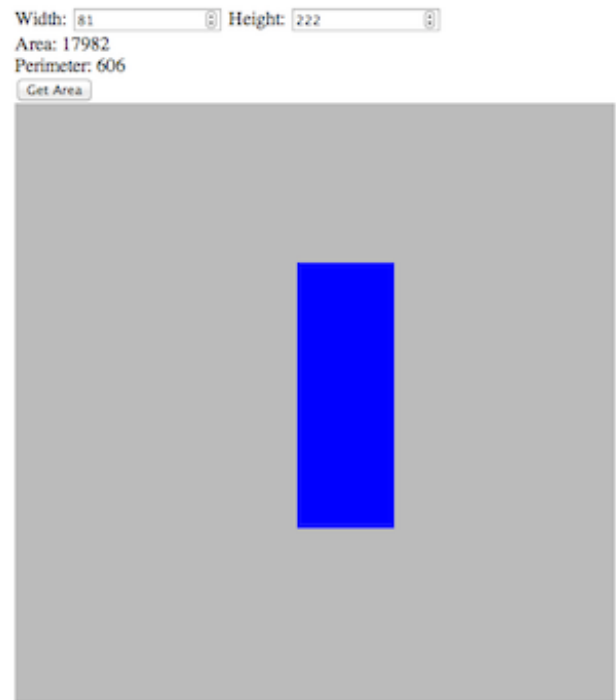
Problems

- I. Consider assignment 2, where you provide a form to collect a rectangle's width and height, and provide its perimeter and area. This assignment has similar requirements, except that the user will use the mouse to draw a rectangle on a canvas placed on the page, and the width, height, area, and perimeter will be extracted from the location of the user's action.
 1. A left mouse click will begin drawing the rectangle, the user will drag out the rectangle, and releasing the mouse will establish the final size of the rectangle.
 2. The current width and height of the rectangle should be fed back to the user while the mouse button is depressed. There are several ways to do this, including filling in the form values.
 3. If you provide feedback to the user through a popup or other mechanism, you may defer filling in the form until the user has released the mouse.
 4. Conversely, if the user fills out the form and presses submit, draw a rectangle on the canvas from 0,0 to the extent provided by the user.
 5. Canvas coordinates supported should be at least 500 by 500.
 6. You should ensure your form input does not accept sizes bigger than your canvas, or less than or equal to zero.
 7. You may choose the line and fill of the rectangle to your own liking, so long as they are visible.
- II. Provide a short paragraph describing your experience with this exercise.

What, if anything, made it difficult?

What, if anything, could have made this easier to accomplish?

The paragraph may be submitted as a separate text document, or may be included in your code comments.



Sample form with canvas

Notes

- The `canvas` is one of the key new components in HTML5. It is expected that JavaScript is

used to interact with it.

- The canvas should be styled with a border, or a background that makes the component visible to you (e.g., gray, or #bbb),
- Events you will care about include `mouseup` and `mousedown`
- Remember the width and height will be the *difference* between the start and current points
- Here is one approach to the problem:
 - Create `var` variables to store the starting X and Y coordinates of drawing the rectangle. These variables will be set when the `mousedown` event occurs. The event received by the event handler as `x` and `y` properties
 - Create a `mousedown` event handler function. It should set the starting `x` and `y` properties above.
 - You will also find it convenient to set another var, responsible for remembering that you are drawing the rectangle, to true. That way, when the mouse remains down, you'll know that you should draw the rectangle and update the form.
 - Create a method called `updateForm` which takes a width and height parameter. The body of the function can set the `width` and `height` fields on the input form, and call a method to draw that rectangle on the canvas.
 - Create a `mouseup` event handler function. It should update the form values with the latest information. It should also reset the var that remembers you are drawing a rectangle to false. Notice that you will not need to redraw the rectangle in this case, since the `mousemove` took care of this.
 - Create a `mousemove` event handler. This method should do *nothing* unless the variable remembering the mouse is down is set to true.

If it is set to true, you should clear the canvas, set the fillstyle to the color you want to draw the rectangle in, and then draw the rectangle (using `fillRect`).

You should also update the form values with the current width and height. Here, you have to remember that the width is the difference between the starting `x` and the current `x`. Similarly, the height is the difference between the starting `y` and the current `y`.
 - Create a method called `init` which
 - Attaches the `mouseup`, `mousedown`, and `mousemove` events to the canvas
 - Initializes the vars you need
 - You will need to attach `init` to the `onload` method for body
 - You can modify your `getarea` method from assignment 2 to add a call to draw a rectangle on the canvas from 0, 0 to the current values of the width and height from the form.
 - You will still need to attach the button to the slightly modified `getarea`
- One other note for the particular: You should also add an event listener for `mousedown` that invokes the same handler as `mouseup`. Do you see why?
- For IOS devices (and I presume other tablets, pads, and phones), the events handlers should be `touchstart`, `touchmove`, and `touchend`. Note that if you want to handle this event, it is a bit different than a mouse event, containing a `touchlist`, which might have multiple points. A `Touch` looks a lot like a mouse event.
- Notice that using this technique, when you press the "Get Area" button on the form, the rectangle is redrawn with the upper left corner at (0,0). You could use the fill color to differentiate these cases.
- When you draw your rectangle, you may find the width and height are negative! Switch them around before placing on the form.
- This problem can also be addressed by appending Scalable Vector Graphics (SVG) components to the DOM. In that case, the element and attributes to add are:
`<rect>` element with attributes `width` and `height`.
Styling is accomplished with the attributes `fill`, which takes a color specification, `stroke-width`, and `stroke`, which is the stroke color.

Simply replacing the contents of the container with the new element will update the result.

- Here is a sample drawarea function for the form, which assumes the canvas element is drawing:

```
function drawarea(wid, hgt) {  
    "use strict";  
    var canvas = window.document.getElementById('drawing');  
    var context = canvas.getContext('2d');  
    context.clearRect(0, 0, canvas.width, canvas.height);  
    context.fillStyle = '#FF0000';  
    context.fillRect(0, 0, wid, hgt);  
}
```

Evaluation

Criteria	Weight
HTML page, including a form and a canvas	15%
Form validation for width and height	15%
Form drawing of rectangle upon submit	15%
Canvas drawing of updating rectangle	15%
Canvas updating of width and height	15%
Updating of form after mouse release	15%
A cogent and concise paragraph summarizing your experience with this exercise	10%