Sample CS 142 Midterm Examination

Winter Quarter 2019

You have 1.5 hours (90 minutes) for this examination; the number of points for each question indicates roughly how many minutes you should spend on that question. Make sure you print your name and sign the Honor Code below. During the examination you may consult two double-sided pages of notes; all other sources of information, including laptops, cell phones, etc. are prohibited.

I acknowledge and accept the Stanford University Honor Code. I have neither given nor received aid in answering the questions on this examination.

_____
(Signature)

____Solution_____
(Print your name, legibly!)

_____@stanford.edu
(Stanford email account for grading database key)

Problem	#1	#2	#3	#4	#5	#6	Total
Score							
Max	15	15	10	20	15	15	90

Problem #1 (15 points)

Consider the following HTML document:

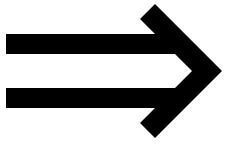
```
<html>
  <head>
    <style>
      html, body { height: 100% }
      div#coolDiv { height: 100% }
    </style>
  </head>
  <body>
    <div id="coolDiv"></div>
    <script type="text/javascript"src="events_and_dom.js"> </script>
  </body>
</html>
```

The provided HTML, creates an empty HTML document, save for the `body`, `html`, and "coolDiv" `div` all being 100% of the page. On the following page write the contents of `"events_and_dom.js"` that will allow a user to click and drag their mouse along the page. Upon releasing the mouse, place a purple rectangle `div` with one vertex beginning where the user clicked and the other, opposite corner where the user released the mouse (the rectangle needn't render until the user releases the mouse). The interior of the purple rectangle should display the area of the rectangle (i.e. height * width).

If you don't remember the exact names for DOM functionality you are permitted to guess and document your guess. For example, if you believe the DOM exports an event called mouseRotation that would be useful you could write:

> `mouseRotation` is an DOM event raised when the mouse is rotated and contains an event with the property `degrees` that is the number of degrees the mouse was rotated

Be sure your inventions actually match DOM functionality, `mouseRotation` doesn't. Please use the white space on the following page to write your answer.



Problem 1 continued:

Contents of "events_and_dom.js" :

```
var mouseDownCoords;
var coolDiv = document.getElementById('coolDiv');

coolDiv.addEventListener('mousedown', function(event) {
     var clientX = event.clientX;
     var clientY = event.clientY;
     mouseDownCoords = {"x": clientX, "y": clientY};
});

coolDiv.addEventListener('mouseup', function(event) {
     var clientX = event.clientX;
     var clientY = event.clientY;

     var newRect = document.createElement('div')

     // color
     newRect.style.backgroundColor = 'purple';

     // positioning
     newRect.style.position = 'fixed';
     newRect.style.left = Math.min(clientX, mouseDownCoords.x);
     newRect.style.top = Math.min(clientY, mouseDownCoords.y);

     // dimensions
     var width = Math.abs(clientX - mouseDownCoords.x);
     var height = Math.abs(clientY - mouseDownCoords.y);
     newRect.style.width = width;
     newRect.style.height = height;

     newRect.innerHTML = width * height;

     coolDiv.appendChild(newRect);
});
```

Problem #2 (15 points)

In old style JavaScript it was not uncommon to see functions starting with the line:

```
var self = this;
```

The effect of the statement is to create two different names for the same object within the function. From a programming language point of view, having aliases for an object is discouraged since it can cause confusion for the programmers and challenges for the compiler optimization pass.

(a) Describe why JavaScript programmers felt that this was a good programming pattern to adopt. Provide an example JavaScript code fragment that supports your answer.

Nested functions in JavaScript cause a confusing redefining of the `this` keyword. For example:

```
function a() {
    this.a = 1;
    function b() {
       this.a ...
    }
}
```

The two `this` above could refer to different objects particularly if function b is being used as a callback. By making the local variable named `self` refer to the definition of `this` when the function is called we can have consistent access to it both inside and outside of nested function definitions.

(b) Explain how the use of ECMAScript 6 extensions in ReactJS makes use of `self` variable pattern less frequent.

EMCAScript 6 has arrow functions and they do not refine `this` and they can be used to implement nested functions and use the this keyword consistently both in the method and inside of nested arrow functions.

Problem 2 continued….

(c) The JavaScript language supports *prototype-based inheritance* and *closures*. Under prototype-based inheritance, an object inherits properties from another object called its prototype object (e.g. `obj.__proto__` returns the prototype object of object `obj`.) An object is considered in a closure if the object can be referenced through the closure and won't be collected by the garbage collector. Demonstrate your knowledge of closures by answering the following questions:

(1) Is it possible for the prototype object of an object (`obj.__proto__`) to be in a closure but `obj` itself would not be in the closure? If it is not possible, explain why. If it is possible, show some code that creates the situation of an object with its prototype in a closure but the object itself is not.

Yes, JavaScript creates closures around function definitions. So all it takes for an object to be in a closure is to have it referenced by a function definition. For example:

```
var obj = {a:1};
function f() {
   return obj;
}
```

object refers to obj
obj is in the closure of f()

`obj` is now part of the closure associated with function f. If we just use obj as the prototype object:

```
Function Con() { }
Con.prototype = obj;
```

object refers to Con instance
Con's prototype is obj,
which is in the closure of f()

Then `new Con()` would return an object whose prototype is in a closure but the object itself is not.

(2) Is it possible for an object to be in a closure yet its prototype object not be in the closure? If it is not possible, explain why. If it is possible, show some code that creates the situation of an object with its prototype in a closure but the object itself is not.

No, Unlike the case above, it is always possible to go from an object to its prototype object. (e.g. Lookups that miss in the object look in `object.__proto__`) this means that if an object is in a closure its prototype must be in the closure as well.

Problem # 3 (10 points)

A web application using Ruby on Rails, an early server-side web application framework, displays a view of a particular product with the URL:

```
http://www.example.com/product.rb?ID=11526&IT=5f7d3d
```

When the web application was ported to use a new framework like AngularJS or ReactJS the same product view had a simple URL of:

```
http://www.example.com
```

(a) Describe how these modern frameworks can show this particular product view without anything but the website name in the URL.

The SPA can make API calls to fetch data as requested without navigating to a new page.

single page application

(b) After a few years with the simple URL, the web application developers deployed a new version that when showing the product the URL:

```
http://www.example.com/#ID=11526&IT=5f7d3d
```

Explain the advantages the web application developers hoped to achieve by adding the characters back to the URL.

One of the main features is deep linking, which maintains some session state in URL so that you can refresh, bookmark, share, etc.

Problem #4 (20 points)

For the following HTML document:

```
<html>
  <head>
    <style>
      .blue {  color: blue; }
      #red { color: red; }
      #red2 { color: red; }
      div {  color: yellow; }
    </style>
  </head>
  <body>

      <div>
          <div class='blue'> sho </div>


          <div class='blue' id='red'> neel </div>


          <div> mendel </div>


          <div id='red2'> kesler </div>


          <div class='blue' style='{color: green}'> sam </div>


      </div>
 </body>
</html>
```

a) Below the five `div` regions in bold above, write the color of the text will be set to by the CSS rules. Use the space after the text to write your answer.

Blue
Red
Yellow
Red
Green

Problem 4 continued….

CSS:

```
myWideDiv {
    top: 0px;
    width: 142px;
    height: 32px;
    padding-top: 4px;
    padding-bottom: 4px;
    padding-left: 4px;
    padding-right: 4px;
    margin-left: 4px;
    margin-right: 9px;
    margin-top: 10px;
    margin-bottom: 6px;
    background-color: blue;
}
#two {
  background-color: red;
}
```

HTML:

```
<div class="myWideDiv">X</div>
<div id="two">Y</div>
```

The above HTML and CSS draws two boxes in the window: one with a blue background and the letter "X" followed by a box with a red background and the letter "Y" in it.

(b) How tall is the blue box (i.e. div with id="myWideDiv") in pixels on the screen? Explain your answer.

We know that padding will also appear as the blue color so we take padding-bottom + padding-top + height and get 40px.

(c) How many pixels of space (if any) will there be between the blue and red boxes? Explain your answer.
The red div doesn't have margins so we only need to consider the blue div. We see that it has 6px bottom margin so we know that the red div will have 6px between its top and the blue div's bottom.

Problem 4 continued…

```
<html>
<body id="body">
  <div id="A">
    <div id="B">
      <div id="C">
        Text
      </div>
    </div>
  </div>
  <div id="D">
    <div id="E" style="position: relative">
      <div id="F">
        Text2
      </div>
    </div>
  </div>
</body>
</html>
```

(d) What is the depth of the DOM tree that would be generated from the above HTML. You can assume the DOM tree starts at `document.body` being at depth 1. Explain your answer.

Body->divA->divB->divC->TextNode(Text)
Body->divD->divE->divF->TextNode(Text2)
We see that both of these go down 5 levels, so we have our depth as 5.

(e) Describe the DOM node or nodes that are at the maximum depth.

There are two nodes at the maximum depth of 5. Both of these nodes are text nodes. One textNode has C as a parent and the other has F has a parent. One has the text set to Text and the other has it set to Text2.

(f) Write what each line would print out. Use the space below the console.log statement for your answer. Hint: The id property of a DOM node is the id of the element or null if that element doesn't have an id.

```
var div = document.getElementById("C");
console.log(div.id); C

console.log(div.offsetParent.id); body

console.log(div.parentNode.id); B


div = document.getElementById("F");
console.log(div.id); F

console.log(div.offsetParent.id); E

console.log(div.parentNode.id); E
```

Problem #5 (15 points)

(a) One of the differences between ReactJS and AngularJS is AngularJS is said to support two-way binding whereas ReactJS only supports one-way binding. Which directional binding is missing in React that AngularJS supports?

Angular supports view-to-model binding and model-to-view binding. React is missing the view-to-model binding.

(b) Describe a way you could use ReactJS to achieve AngularJS-style two-way binding in a ReactJS web application.

You can add 'onChange' event handlers to elements that then call setState and make the appropriate updates.

Problem 5 continued...

(c) AngularJS was known for having performance problems handling views with a large amount of model data such as a dense table of information. Describe the mechanism used by AngularJS that made these views a problem.

Angular automatically adds a watch to every variable in template expressions, and during a digest cycle each variable being watched is checked -- as such, having to run hundreds or thousands of these checks in a dense table can slow down performance.

(d) The use of the Virtual DOM abstraction in ReactJS enables the view rendering of a web application to happen either in the browser or ahead-of-time in the server. Describe the advantages of having a web application do rendering on the server.

- **Quicker initial page load**
- **Better search engine optimization**

(e) Describe the disadvantages of having all rendering done on the server.

- **Every subsequent page load will take longer than client-side rendering since we need to make a server request each time**

Problem #6 (15 points)

(a) If you take a good web application running in a window on a high resolution desktop machine and gradually reduce the size of the window you might notice that the web application shrinks everything in the view to fit into the smaller window but after some amount of shrinking web application radically alternatives its layout to work in the smaller window. Describe the mechanism used in web application that would explain this behavior.

This is the concept of responsive web design and the web app is likely using CSS breakpoints specified by @media queries to define layout switches for certain screen sizes.

(b) When testing web applications we typically have a test framework that helps the test writer develop and execute tests. Describe what would be important features of a test framework for:

(i) Unit testing

Components should be isolated and tested as individual modules. Interactions with other components should be replaced with mock implementations.

(ii) End-to-End testing

Should test the integration of components. Ideally these tests should run across the entire application testing various workflows. These types of tests require a scripting interface into the browser which can simulate user actions on the application.

(c) The ngAria modules of AngularJS provides a warning message if it sees an HTML img tag that does not have an attribute named `aria-label`. The `aria-label` attribute has no effect on the img tag and the view renders and the web app functions identically with and without it. Describe the purpose of requiring an attribute that has no effect on the web application.

The ngAria module is used to encourage good accessibility practices for websites. For example, the aria-label allows the user to add additional text descriptions to an html element (such as an image tag) that will not be rendered visually on the screen, but can provide additional details to screen readers to caption the visual elements of web page. This is extremely important for users with visual impairments.