Solutions CS 142 Midterm Examination

Spring Quarter 2017

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)

_____

(Print your name, legibly!)

_____@stanford.edu

(Stanford email account for grading database key)

Problem	#1	#2	#3	#4	#5	#6	#7	#8	#9	Total
Score										
Max	10	10	10	10	12	10	12	8	8	90

Problem #1 (10 points)

(a) 5 points [~½ point each]

1. <h1>
2. <p>
3.
4.
5. <b>
6. <i>

(b) 3 points [2 for the correct tags, 1 for the explanation]

<b> and <i>, these are unnecessary since they are related to styling and can thus be replaced by CSS. It is good practice to divide structure into HTML and styling in CSS. It is semantically more correct to replace these with and . Bonus: <h1> can also be replaced by changing the font size in the CSS.

(c) 2 points [1 for the correct reason, 1 for the explanation]

The main disagreement is over how strict syntax errors should be between the two. XHTML is more strict over things such as unknown tags and elements overlapping. HTML is more lax and has quirkier syntax.

Problem #2 (10 points)

A. nothing (2 points)
B. throw
C. bar
D. wash2
E. throw2
F. bar2
G. false
H. false
I. true

Explanation

Apple.wash is a static function for the class. Static functions are not available on instances of a class. Therefore, there is no wash attribute on myApple, so undefined is returned. Another way to define wash in ECMAScript 2015 (6th edition) would be

```
class Apple {
  static wash() {...}
}
```

throw is a function on Apple's prototype. bar is a function on Object's prototype. From MDN:

> When it comes to inheritance, JavaScript only has one construct: objects. Each object has a private property (referred to as [[Prototype]]) which holds a link to another object called its **prototype**. That prototype object has a prototype of its own, and so on until an object is reached with `null` as its prototype. By definition, `null` has no prototype, and acts as the final link in this **prototype chain**.
>
> Nearly all objects in JavaScript are instances of `Object` which sits on the top of a prototype chain.

Therefore, myApple.throw refers to Apple.prototype.throw, myApple.bar refers to Object.prototype.bar.

Part 3:

the functions are being compared with an equality operator. Functions can be compared this way because they are objects. True or false will be returned depending on whether the objects are equal. Part G is false because yourApple.throw refers to the new class property only existing on that instance (yourApple). Part H is false because the yourApple.wash function !== undefined. Part I is true because Object's prototype.bar only exists once in memory and was changed.

Problem #3 (10 points)

URLs/Links

(A) You would like to load the page **`junipers.html`** from **`www.trees.org`**, using SSL encryption. Additionally, you want to pass the parameter **`species`** to the server with a value of **`californica`**. Please construct the URL you would use to access this page, and **label all components of the URL**.

`https://www.trees.org/junipers.html?species=californica`

scheme hostname hierarchical portion query parameters

(including port :80 was not necessary but also not penalized)

Grading:

4 pts (one for each component)
-0.5 pts for syntax errors (eg '/' instead of '?' in the query portion)
-1 pt for a completely incorrect part (eg interpreting the query as a fragment)

(B) Having navigated to the page above, you see it contains several links. For each of the links below, circle the correct option:

```
<a href="/junipers.html#~cultivation">Link 1</a>
```

The type of this link is:	**Absolute**	Relative	Other
This link contains invalid characters:	True	**False**	
This link causes a page reload:	**True**	False	

```
<a href="/spruces.html#classification&naming">Link 2</a>
```

The type of this link is:	**Absolute**	Relative	Other
This link contains invalid characters:	**True**	False	
This link causes a page reload:	**True**	False	

```
<a href="#juniper_berry">Link 3</a>
```

The type of this link is:	Absolute	Relative	**Other**
This link contains invalid characters:	True	**False**	
This link causes a page reload:	True	**False**	

Grading:

6 pts

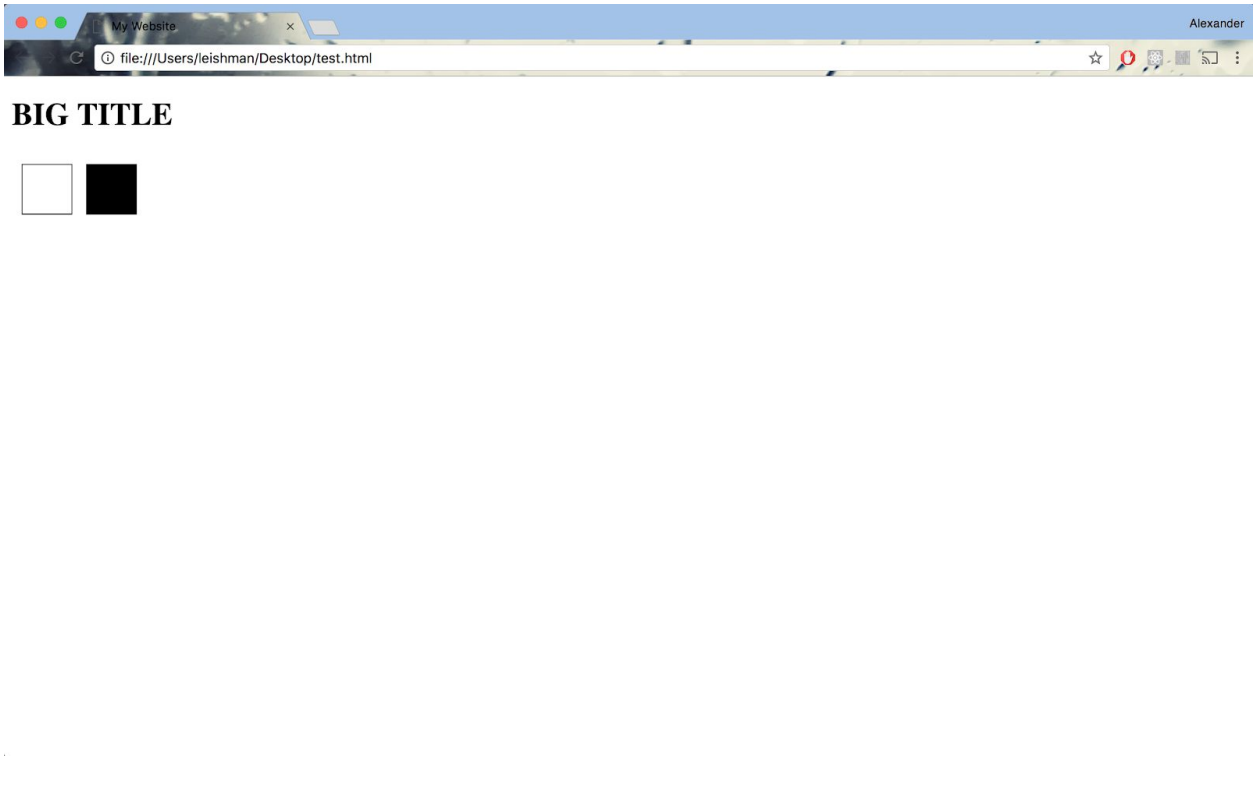
-1 pt for incorrect link type

-0.5 pts for each incorrect True/False question

Problem #4 (10 points)

Answer/Notes:

The below image is how this page renders in a browser. The .container div is invisible and the blocks are side-by-side because of the inline-block styling applied to them. Note the title is the browser tab, not rendered on the page.



Rubric:

- [- 2] not inline block
- [-2] missing title
- [-1] case incorrect
- [-2] getting a box styling incorrect
- [-2] everything inline
- [-2] outline box

Problem #5 (12 points)

Rubric:

-1.5 ea. for incorrect a/b

-3 ea. for incorrect c/d/e

Solutions:

a. The program is first started in the browser

1. 0
2. 0
3. 0

b. After step (a), button 1 is pushed 2 times

change in click-counts:

1. +2
2. +2
3. +2

(desired answer: 2/2/2, assuming previous parts correct)

c. After step (b), button 3 is pushed 3 times

change in click-counts:

1. +0
2. +0
3. +3

(desired answer: 2/2/5, assuming previous parts correct)

d. After step (c), button 1 is pushed 1 time

change in click-counts:

1. +1
2. +1
3. +0

(desired answer: 3/3/5, assuming previous parts correct)

e. After step (d), button 2 is pushed 2 times

change in click-counts:

1. +0

2. +2

3. +0

(desired answer: 3/5/5, assuming previous parts correct)

Notes:

For parts b-e, only the change in click-counts were considered for grading. A few students received deductions even though they wrote down 3/5/5 in part e, because their change in click-counts (based on their numbers from the previous step) were incorrect.

a left-to-right order (i.e., a->d->b->e->c) instead of the order specified by the instructions. These students were given a penalty of -3, and then graded according to that order (max deduction is still -12).

Common mistakes:

a. 0/0/0 -> 2/0/0 -> 2/0/3 -> 3/0/3 -> 3/2/3

Here, it was incorrectly assumed that each read only looked at the current scope.

b. 0/0/0 -> 2/2/2 -> 2/2/5 -> 3/3/6 -> 3/5/6

Here, it was incorrectly assumed that clicking button 1 in d also increments the count variable in button 3's scope.

c. 0/0/0 -> 2/0/0 -> 2/3/3 -> 3/3/3 -> 3/5/5

Here, there was the incorrect assumption that each read only looked at the current scope. There was also the incorrect assumption that ng-repeat created a new scope that is shared for all the items being iterated over (the hint on the exam stated that a new scope was created for every iteration).

Problem #6 (10 points)

Part (a)
undefined
red
Blue

Part (b)
undefined
red
Green

Explanation for parts (a) and (b):
`console.log(this && this.color)` prints `'this'` if it can be converted to false; otherwise, prints `'this.color'`. `'this'` within the context of the the anonymous function being returned refers to the Global object (in the browser, this is `window`, in node.js, it's just the global object). That's because the value of `'this'` was changed when the code entered the anonymous function and `'this'` no longer refers to the `CrayonStore` object. Therefore, `'this'` (the global object) can't be converted to false, so the value of `'this.color'` which is `undefined` was printed.

The var `'color'` is within a closure and not a property of `CrayonStore`. `crayonStore.color = 'black';` doesn't change what 'color' is.

'`gcolor`' is a global variable that was first set to blue and changed to green.

Part (c)
`undefined`

Part (d)
`undefined`

Explanation for parts (c) and (d):
Indexing variables within loops (i.e. `'idx'` in this problem) persist after a loop is done so when you try to call `getFunc()`, `'idx'` has a value of 2. `retObj[colors[2]]` will return `'undefined'` because `colors` only has two items and the code is trying to get the third item.

Rubric (10 points total)

Parts (a) (3 points) and (b) (3 points)
+1 point for each correct output
'undefined && SOMECOLOR' was not accepted as an answer to the output

Parts (c) (3 points) and (d) (3 points)
+2 points for each correct output

Problem #7 (12 points)

a) 4 points
Correct answers:

- Changing the structure / layout of the DOM may invalidate the code (could access incorrect element)
    - -1 for not mentioning something like this
- Unclear which node is being used, affected by whitespace
- Long / cumbersome way to write
- Need to know the DOM structure a priori to write code like this, not always the case

b) 4 points
1 points for valid code fragment, something like this:

```
<body>
      <div id=”a”>
            <p>Diff.</p>      # parentNode is <div id=”a”>, offsetParent is <body>
      </div>

      <div id=”b” style="position: relative;">
            <p>Same.</p>      # parentNode is <div id=”b”>, offsetParent is <div
      id=”b”>
      </div>
</body>
```

3 points for description / explanation of the difference:
The `offsetParent` property refers to the nearest *positioned* (non-static) ancestor element. If there is no such element, `offsetParent` will refer to the `body` element. The `parentNode` property refers to the parent of the given node (i.e. the nearest ancestor element, regardless of positioning).

c) 4 points
1 point for pointing out the difference:
Bubble phase starts at the innermost DOM element with a registered event listener and propagates outwards. Capture phase starts at the outermost DOM element with a registered event listener and propagates inwards.

3 points for explanation:
Bubble phase is more commonly used because it lets the most specific element react first, which is often the desired effect from a UX perspective (e.g. in capture phase, it would be impossible to click a button inside a clickable div without first registering the div click listener).

Problem #8 (8 points)

(a) Any of them except `px`. You want the unit to be device independent and px depends on the pixel density of your device.

Full Points(4): Must include discussion on device independent units and exclude px.
Partial Points(0-3): Depends on the validity of justification. For example, stating that you would use em because relative distances are good for responsive design could earn 3 points. Points taken off for incorrect statements and including px.

(b) The dimensions could be picked based on screen sizes of popular devices that the app will be used on. For example, one could go with approximate screen sizes for phones, tablets, desktops, etc

```
@media only screen and (min-width: 768px) {
        /* tablets and desktop layout */ }
 @media only screen and (max-width: 767px) {
        /* phones */ }
@media only screen and (max-width: 767px) and (orientation: portrait) {
        /* portrait phones */ }
```

Full Points(4): Must include some discussion device screen sizes.
Partial Points(0-3): Points taken off if device screen sizes not mentioned.

Problem #9 (8 points)

(a) The browser's navigation bar (e.g. forward/back buttons, refresh button, bookmarks) can be a challenge for JavaScript frameworks like Angular.js. Explain the basic problem with it and describe the solution used to make it work with JavaScript frameworks.

3 points

Navigating away from the web application or pushing the refresh button will cause the JavaScript execution to terminate, losing any web app state stored in JavaScript (e.g. where in the application the user is, what the user is doing, etc.) One solution for this is to store enough the web application state in the URL so that lost state can be restored. Deep linking is a name that is giving for this.

-2 for not mentioned termination but talking about URLs/single page applications

(b) For each of the following items from web application development, which (if any) of the web view components (i.e. Model/View/Controller) would you expect to take the brunt of the challenges to implement the item. Justify your answer.

(i) Internationalization

2 points

Internationalize means that things like text content need to be selectable based on the user's preferences or locale. Text content that previously could be hard-code in the HTML view component will need to be fetched based on the user. The model component of a view will need to be expanded to include this text content.

(ii) Accessibility

2 points

In class we presented ARIA which extended the HTML (view component) to include text description for images/icons/etc. so that screen readers and present them to a user that can't see them.

(iii) End-to-End testing

1 point

End-to-end testing involves running the actually web app which unless something is done wrong shouldn't impact components.