

CS 142 Midterm Examination

Spring Quarter 2023

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)

Only the front side of the exam pages will be scanned. Do not write answers on the back of the pages.

Problem #1 (12 points)

React is frequently used to build **single-page applications** that support **deep linking**. Answer the following questions, including explanations demonstrating your understanding of single-page applications, deep linking, and React.

A) Is it possible to build a single-page application without deep linking?

Yes, it's possible to build a SPA without deep linking. A SPA just means that content on the page is loaded with a singular HTML file (i.e. browser page), and any new views are created by updating the page. The SPA web app would be served as a single browser page, and there wouldn't be any URLs that point to the views inside the app (i.e. deep linking)

B) Would it be possible to support deep linking in an application that wasn't a single-page application?

Yes. The original web applications were constructed of multiple HTML documents, each served at a different URL. Every view effectively could be accessed using the link to that URL. The users navigate to different views using hyperlinks, much like deep-linking brings to SPAs.

... Problem #1 continued on the next page ...

... Problem #1 continued ...

C) Would it be possible to construct a React application that wasn't a single-page application?

Yes. It's simple to create a multi page application in React just by having multiple HTML files that service different React components upon request.

Problem #2 (6 points)

Both React and the JavaScript web application framework that preceded it (AngularJS) allowed the programmers to describe views using HTML-like code with templates containing JavaScript expressions embedded in it. Both React and AngularJS needed to detect if any of the template expressions changed to determine if it could display the view without having to run the code to re-render it. AngularJS did this by checking **all** the template expressions in a view to see if anything changed. This approach resulted in sluggish performance for views with many template expressions, such as a view containing a dense table of values.

Describe how React got around checking every JSX template expression when displaying these complex views that AngularJS had problems with.

React only re-renders a component when either the props or state change. This allows React to avoid checking the template expressions in a component. This approach allows React to omit many of the template expression checks that AngularJS needs to perform on complex views.

Problem #3 (8 points)

Most web application frameworks, including the React framework we used in this class, follow the **model–view–controller** (MVC) software design pattern. Since React runs in the browser's environment that supports **HTML**, **CSS**, and **Javascript**, each of the MVC parts needs to be implemented using one of these three. For example, React controllers can use the JavaScript language, and its model data can be held in JavaScript data structures. Answer the following questions, including an explanation.

A) Can a React view also be written in JavaScript?

React views are generated by functions that return a JavaScript tree created by calls to `React.createElement`. They are written in JavaScript. Even if the view is coded in JSX, the Babel transpiler will convert the JSX to JavaScript calling `React.createElement`.

B) Can a React controller be written in HTML?

No. HTML is a declarative language for describing document layout. As such, it lacks loops, conditional statements, subroutines calls, etc., that are needed to perform the controller functionally, like model fetching and user event handling.

Problem #4 (12 points)

Assume you have the following HTML document:

```
<html id="H">
  <body id="B">
    <div id="D1">
      Div1
      <div id="D2">
        Div2
        <div id="D3">Div3</div>
      </div>
      <div id="D4">Div4</div>
    </div>
  </body>
</html>
```

If you looked up one of the `div` DOM nodes by ID (i.e., `getElementById` of D1-D4) you would be at the start of two singly-linked lists formed by the properties `offsetParent` and `parentNode`. The linked list formed by these properties always goes up the tree. We define the length of these lists as the number of DOM nodes on it (e.g., `e.offsetParent.offsetParent`) up to and including the body node. Note the `offsetParent` of the body node is null while its `parentNode` is the `html` tag. We don't include the `html` node in the length of either list.

- A. Write a CSS style sheet for the HTML document that makes the `offsetParent` and `nodeParent` lists have the same length for every `div` node. If it is not possible, explain why not.

```
div {
  position: relative;
}
```

Making every `div` a start of a new positioning context means that the `offsetParent` of every `div` node children will be the `div` node. The `offsetParent` and `nodeParent` pointers will be the same.

... Problem #4 continued on the next page ...

... Problem #4 continued ...

- B. Write a CSS style sheet that makes the `offsetParent` list longer than `parentNode` list. If it is not possible, explain why not.

This is not possible -- we can only make `offsetParent` list equal to (like we did above) or shorter (e.g., if we assign `position` to `fixed`, so `parentOffset` list of each div will be of length 1).

- C. Is it possible for children of a node (e.g. D2 and D4 of D1) to have different length lists? Answer the question with an explanation for both the `offsetParent` and `nodeParent` lists.

`offsetParent` lists of siblings can be different: if one sibling's position is fixed and another one's is relative. `parentNode` lists cannot be different for siblings because it just contains nodes corresponding to tags in the html

Problem #5 (8 points)

Assume you have the following HTML document (same as Problem #4):

```
<html id="H">
  <body id="B">
    <div id="D1">
      Div1
      <div id="D2">
        Div2
        <div id="D3">Div3</div>
      </div>
      <div id="D4">Div4</div>
    </div>
  </body>
</html>
```

Assume you have some JavaScript that adds both a bubble and a capture event 'click' handlers on every node that has an id property. The handler does a `console.log` of the event's phase ('bubble' or 'capture'), `target.id`, and `currentTarget.id`.

A) If the user clicks on the string "Div2", show the `console.log` lines that will be written.

capture, D2, H

capture, D2, B

capture, D2, D1

capture, D2, D2

bubble, D2, D2

bubble, D2, D1

bubble, D2, B

bubble, D2, H

... Problem #5 continued on the next page ...

... Problem #5 continued...

- B) If the user then runs the following JavaScript code and then clicks on the string "Div4", show the `console.log` lines that will be written.

```
function ev(event) {  
    event.stopPropagation();  
}  
document.getElementById('D4').addEventListener('click', ev, false);  
document.getElementById('D1').addEventListener('click', ev, true);
```

capture, D4, H
capture, D4, B
capture, D4, D1

This question runs into something about event handling that wasn't presented in CS142. The issue is what is `event.target` when the user clicks on a Text node (e.g., string "Div 4"). In the past different browsers handled this differently but the current DOM standard will have the target set to the parent element. If the click is on "Div 4", the `event.target` will point to the `<div id="D4">` element. Most but not all events result in the target being set on the enclosing element.

Problem #6 (6 points)

- A) Explain why a new feature added to the JavaScript language standard can achieve wide usage much faster than additions to the language standards of compiled languages like Java or C++.

JavaScript in the web environment is often used with a transpiling step (e.g Babel) that will translate new JavaScript features into code that run in older JavaScript run times found in some browsers on the Internet. Because of this programmers can adopt new features as soon as the transpilers support them. Compiled languages need to wait until widespread adoption of the new compiler before it makes sense to develop software using the new language features.

- B) Consider the following code:

```
function MyObj() {  
  }  
MyObj.prototype = {  
  A: function A() { console.log("AP"); },  
  B: function B() { console.log("BP"); },  
  O: {C: function C() { console.log("CP"); }}  
};  
let o1 = new MyObj();  
let o2 = new MyObj();  
  
o1.A = function A() { console.log("A0"); };  
o1.O.C = function C() { console.log("CO"); };
```

For each expression, state what the console statement would print or if an exception error would occur:

o1.A(); **AO**

o1.B(); **BP**

o1.O.C(); **CO**

o2.A(); **AP**

o2.B(); **BP**

o2.O.C(); **CO**

Problem #7 (8 points)

The introduction of CSS to HTML discourages the use of some of the original HTML tags as it was believed that CSS addressed the tag's functionality better. For each of the following HTML tags, briefly describe if they would be better done with CSS or their use in HTML is still encouraged.

- A) P - HTML, since it describes part of a document (i.e. a paragraph).

- B) H1 - HTML, since it describes part of a document (i.e. a first-level header).

- C) PRE - CSS, this a styling directive that forces a particular set of text styles (fixed-width font, keep whitespace, etc.)

- D) IMG - HTML, as this tag is meant to hold image content and has semantic meaning.

- E) B - CSS, this a styling directive that forces a particular text style (font-weight).

- F) DIV - HTML. A div is just a generic container that wraps some content in the HTML document.

- G) TITLE - HTML, since it defines part of the document (the page title displayed in the browser tab).

Problem #8 (8 points)

An HTML document is served to a browser via the URL `http://localhost/a/b.html`.

What would be the URL sent to the web server if you clicked on the following hyperlinks in the HTML document?

A) `C` `/a/C.html` will be sent to `http://localhost`

B) `/c` `/c.html` will be sent to `http://localhost`

C) `/c#c` `/c.html` will be sent to `http://localhost`
(fragment isn't set)

D) `:80#c` `/a/:80` will be sent to `http://localhost`

Problem #9 (10 points)

When the following JavaScript code is executed, five log messages will be written to the console. Each log message will be two words.

```
try { console.log("one", varA); } catch(err) { console.log("one error"); }
function A() {
  try { console.log("two", varA); } catch(err) { console.log("two error"); }
  for (let i = 0; i < 1; i++) {
    var varA = 10;
    try { console.log("three", varA); } catch(err) { console.log("three error"); }
  }
  try { console.log("four", varA); } catch(err) { console.log("four error"); }
}
A();
try { console.log("five", varA); } catch(err) { console.log("five error"); }
```

A) Fill in the blanks to complete each log statement below. The first words have been given to you:

one error

two undefined

three 10

four 10

five error

B) If we change the line “var varA = 10;” to be “let varA = 10;”, what would the output be then?

one error

two error

three 10

four error

five error

Problem #10 (6 points)

Consider the following JavaScript function:

```
function A(arg) {  
  if (arg)  
    console.log("true", arg === true);  
  else  
    console.log("false", arg === false);  
}
```

For each of the following invocations of the function A, list what the output will be:

A) A(true); **true true**

B) A("false"); **true false**

C) A(0); **false false**

D) A(); **false false**

E) A([]); **true false**

F) A([0]); **true false**

Problem #11 (6 points)

Closures in JavaScript can sometimes cause problems. One problem is large data structures can sometimes be captured in closures and cause the program to have unused memory that can not be collected by garbage collection. Write a function that captures something in a closure and returns something to the caller so that the caller holds the return value.

You can use a function call to `gigabyte()` to allocate and return a gigabyte-sized data structure.

```
function capture() {  
  
    let variableForClosure = gigabyte();  
  
    function functionToCreateClosure() { // Creates closure  
        return variableForClosure; // Reference adds to closure  
    }  
  
    return functionToCreateClosure;  
  
}
```