In order to get into practice test1, you have to have the Proctorio Chrome plugin installed/enabled. Please do this and answer the only actual question in the practice quiz – the last question entitled JS Programming Question.

- **Overview** There will be a total of 100 points on the test.
- **Matching** There will be a matching question with 10-15 terms taken from these quizzes: Web Dev, (not Web Design), JS UI, JS Obj, AJAX.
- **Creating and Styling HTML Elements using JS** There will be a couple of questions with JS code that creates and styles some HTML elements. You will be asked to select which image best represent how the page would render. You need to understand nesting of HTML elements, fixed position, absolute position, and float.
- **Order of JS execution** To demonstrate that you understand the order of execution of JS code, you will be shown some JS code and asked what it will output to the console. Basically, the lines of code are just executed from top to bottom, skipping over function definitions -- functions are only run if they are invoked.
- **Associative Array** There will be a couple of questions that ask you about associative array notation, even if you do not know the names of the properties in the object.
- **Scratch Paper** Feel free to write your thoughts here (not graded). There will be 2-3 of these "fake essay questions" in the test.
- **JavaScript Objects** Code will be presented that calls a "make" function twice, returning two objects. Public methods will be invoked on these objects. You will be asked what will show on the page (or what displays in the console). You just need to keep track of the values of the properties of each object as the various public methods affect them.
- **Private/Public Data Members and Methods** Given a "make" function, you will be asked to identify private and public (non-function) data members of the object returned by the make function. You will also be asked to identify private and public methods (functions) of the object returned by the make function.
- **Iterating through DOM elements, adding events, custom properties** In Mod 2 Drop Down, there was sample code that accessed all elements (of a given HTML tag or of a given CSS class) that were in a specific div. In Mod 5 Click Sort, you added custom properties to DOM elements, so that these properties could be used later when/if that DOM element was involved in an event. You learned that the keyword "this" can mean the DOM element involved in an event. There will be a couple of questions where code adds custom properties to DOM elements so that later (e.g., when that element is clicked), these properties are available to help perform the click functionality.
**JS Call Back Functions**

You will be asked what you would see in the console after a page with some JS code (that uses call backs) were to be run.

A call back function is a function that is passed to provider code. The provider code will call back the "call back" function typically after something happens that the provider knows about and needs to communicate back to the calling consumer code.

A great example of why call backs are useful is our ajax function. When our ajax function knows that the data has arrived, it "calls back" the calling consumer code, passing to it the data extracted from the page.

But here is a different example of a call back function, not related to AJAX.

```javascript
function provider (callBackOne, callBackTwo) {
    function makeRandomNumber(low, high) { //
        var random = Math.random(); // returns some real number between 0 and 1
        var diff = high - low + 1; // determine the distance between the high and low.
        // Math.floor returns an integer truncating any real number after the decimal point.
        return Math.floor(random * diff + low);
    }
    var random = makeRandomNumber(1, 100);
    if (random <= 50) {
        callBackOne("CallBackOne says the magic number is " + random);
    } else {
        callBackTwo();
    }
}

function myFuncOne(inputParam) {
    console.log("myFuncOne called with input: " + inputParam);
}

function myFuncTwo() {
    console.log("myFuncTwo was called (no input supplied)"");
}
for (var i = 0; i < 6; i++) {
    provider(myFuncOne, myFuncTwo);
}
```

When the above code is run, the provider function is called 6 times. Each time the provider code is run, it will call back either myFuncOne or myFuncTwo, depending on the random number that's generated.

**AJAX order of execution** You will be asked to show what prints in the console given code that makes two AJAX calls. Remember that all the code that initiates both AJAX calls will run first, but then there is a very long time (maybe seconds in which millions of lines of JS code could be run) before either data set becomes available and only then are the call back functions executed.
**Modify function `MakeObj` below as follows:**

- "params" shall have an additional optional property named `altTitleSize`.
- When the user's mouse enters object `o` (onmouseenter), the title size shall be changed to the `altTitleSize` (if it was provided). When the user's mouse leaves object `o` (onmouseout), the title size shall revert back to its original size.
- Add a public method (to the object returned by `MakeObj`) that allows the HTML coder to change the `altTitleSize` property whenever they want to.

Use spaces to indent. Keep in mind that you cannot change any existing code, only add to it. Do not retyping all the existing code, just type in the new code you are adding. Your code will be inserted between the comments "YOUR ANSWER STARTS HERE" and "YOUR ANSWER ENDS HERE".

```javascript
"use strict";

function MakeObj(params)
{
    var o = document.createElement("div");
    var titleDOM = document.createElement("h2");
    var title = params.title || "Untitled";
    var titleSize = params.titleSize || "1.25rem";
    var text = params.text || "Text coming soon...";
    o.appendChild(titleDOM);
    var textDOM = document.createElement("p");
    o.appendChild(textDOM);
    // ***** YOUR ANSWER STARTS HERE *****
    // ***** YOUR ANSWER ENDS HERE *****

    function refresh()
    {
        titleDOM.innerHTML = title;
        titleDOM.style.fontSize = titleSize;
        textDOM.innerHTML = text;
    }
    o.changeTitle = function (newTitle) {
        title = newTitle;
        refresh();
        refresh();
        return o;
    }
    var one = MakeObj({
        title: "JavaScript Primer",
        titleSize: "2rem",
        text: "Lots of Stuff"
    });
    var two = MakeObj({});
    document.getElementById("content").appendChild(one);
    document.getElementById("content").appendChild(two);
    two.changeTitle("Web APIs Made Easy");
}:
```