Please complete the following JavaScript questions using your normal development workflow. Document any research done to complete a question.

1. **Fix the below JavaScript code so that the correct index is printed to console.log on each iteration.**

```
(function() {

    var index,
        length = 10;

    for (index = 0; index < this.length; index++) {
        setTimeout(function() {
            console.log(index);
        }, 100);
    }

})();
```

2. **Modify the below JavaScript so that it is called just after the DOM has loaded. No legacy browser support required.**

```
(function() {
    document.getElementById("test").innerHTML = "Hello World";
})();
```

3. **Modify the below code so that it will only display a message if the user is using Internet Explorer 7**

```
(function() {
        alert("Hello World");
})();
```

4. **Modify the below JavaScript code so that it uses a closure to return the response.**

```
(function() {

        function hello(name, age) {
                return name + ", who is " + age + " years old, says hi!");
        }

        console.log(hello('John', 33));
}();
```

5. **Finish the below JavaScript by implementing a simple flow control function (flow) that can take the provided array of functions and process them asynchronously before making a final callback.**

```
(function() {

    var array = [
        function(callback) {
            console.log("first function called in " + (new Date().getTime() - timestamp) + "ms");
            callback();
        },

        function(callback) {
            console.log("second function called in " + (new Date().getTime() - timestamp) +
"ms");
            callback();
        },

        function(callback) {
            console.log("third function called in " + (new Date().getTime() - timestamp) + "ms");
            callback();
        }
    ],
    timestamp = new Date().getTime();

    function flow(array, callback) {
    }

    flow(array, function() {
        console.log("all functions finished in " + (new Date().getTime() - timestamp) + "ms");
    });

})();
```

6. **Modify the below code so that the return value can also be returned with a callback function (if a callback function has been specified).**

```
(function() {
    function isArray(array) {
        return typeof(array) === "object" && (array instanceof Array);
    }

    var result = isArray([
        "item1",
        "item2",
        "item3"
    ]);

    console.log("isArray: " + result);

})();
```

7. Optimize the below JavaScript to minimize the number of redraws and reflows required.

```
(function() {
    var element,
        index,
        length,
        content = document.getElementById("content"),
        data = [{
            id: 1,
            name: "John",
            color: "green"
        }, {
            id: 2,
            name: "Sally",
            color: "pink"
        }, {
            id: 3,
            name: "Andrew",
            color: "blue"
        }, {
            id: 4,
            name: "Katie",
            color: "purple"
        }],

    for (index = 0; index < data.length; index++) {
        element = document.createElement("li");
        content.appendChild(element);
        element.setAttribute("id", data[index].id);
        element.innerHTML = "<strong>" + data[index].name + "</strong>";
        element.style.color = data[index].color;
    }
})();
```

8. **Using the below JavaScript code as a starting point, implement a chain-able DOM Wrapper API that operates in a similar fashion to jQuery's API (No native prototype extensions).**

```
(function() {
NodeList.prototype.show = function() {
        var array = Array.prototype.slice.call(this, 0);

        array.forEach(function(node) {
                node.style.display = “block”;
        });
}

NodeList.prototype.hide = function() {
        var array = Array.prototype.slice.call(this, 0);

        array.forEach(function(node) {
                node.style.display = “none”;
        });
}

        document.querySelectorAll(“#test”).show();
        document.querySelectorAll(“#test”).hide();
})();
```

9. **The below JavaScript is used to handle mousemove events amongst 3 divs which are nested inside each other. Find and fix the problem which is causing too many events to get fired.**

```
(function() {
        var boxes = [
                        document.getElementById("box1"),
                        document.getElementById("box2"),
                        document.getElementById("box3")
           ];

        boxes[0].addEventListener("mousemove", function(event) {
                console.log("Box 1");
        });

        boxes[1].addEventListener("mousemove", function(event) {
                console.log("Box 2");
        });

        boxes[2].addEventListener("mousemove", function(event) {
                console.log("Box 3");
        });
})();
```