What is considered bad about the getters and setters that you can make with JavaScript's get / set syntax?

Model Short Answer

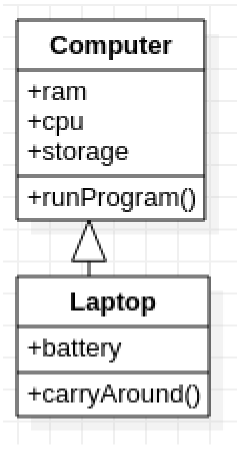
You can accidentally make extra properties on the object if you mis-spel the name of the setter. Using a real method you'd get a error if you mis-spel the name.

Question 2 2: Short Answer/Essay - 1.0 point Edit

Implement the following UML diagram in JavaScript using the new ES6 class syntax.

The runProgram() method should take a string that indicates which program, and then console.log "running: " + program

The carryAround() method takes no parameters, and console.log "carrying laptop: cpu" + cpu +" ram: " + ram + " storage: " + storage + " battery: " + battery



Model Short Answer

```
class Computer {
    constructor(cpu, ram, storage) {
        this.cpu = cpu;
        this.ram = ram;
        this.storage = storage;
    }
    runProgram(program) {
        console.log("running: " + program);
    }
}
```

```
class Laptop extends Computer() {
    constructor(cpu, ram, storage, battery) {
        super(cpu, ram, storage);
        this.battery = battery;
    }
    carryAround() {
        console.log("carrying laptop:  cpu" + this.cpu +" ram: "
        + this.ram + " storage: " + this.storage + " battery: " + this.battery);
    }
}
```

What, or more specifically _when_ does the following jQuery code do:

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

Model Short Answer
A function passed into jQuery is considered a window onload event handler.

Question 2 2: Short Answer/Essay - 1.0 point Edit

What is meant with 'a jQuery object'?

Model Short Answer
jQuery does not return actual elements, instead it returns them wrapped in 'a jQuery object'.

The objects that jQuery returns from its functions often represent a group of elements (zero or more elements) onto which further jQuery operations can be applied.

Question [3] 3: Short Answer / Essay - [1.0] point Edit

Write some jQuery code to:

- Create a input tag with type="text", value="hello world" name="hello"
- Add the tag you created to the bottom of the form that has id "helloForm"

Model Short Answer

```
$("#helloForm").append($("<input>", {
  type: "text",
  name: "hello",
  value: "hello world"
}));
```

Explain what is Event Bubbling is:

Model Short Answer
When an event happens (say a mouse click), then the event is first sent to the most specific element (right under the mouse pointer), after which the event is also given to the parent element where it started, and it's parent, and so on until even html body sees the event.

This process of going wider and wider (parent, parent, parent) is known as event bubbling

Question [2] 2: Short Answer/ Essay - [1.0] point Edit

Explain what the difference is between event.stopPropegation() and event.stopImmediatePropegation() for jQuery event handlers.

Model Short Answer
While both stopPropegation() and stopImmeditatePropegation() stop the event from bubbling (going to the parent), stopImmeditatePropegation() also stops any other event handlers that may be on the current element from executing

Question [3] 3: Short Answer/Essay - [1.0] point Edit

Explain what exactly happens when you return false from a jQuery event handler

Model Short Answer
Returning false is the same as:
event.preventDefault()
event.stopPropegation()

Question [4] 4: Short Answer/Essay - [1.0] point Edit

Explain how asynchronous callbacks work, even though JavaScript is single threaded (can only ever do one thing at a time)

Model Short Answer
JavaScript has a event loop, which has a queue of additional tasks waiting to be performed after the current task finishes.

Question [1] 1: Short Answer/Essay - [1.0] point Edit

Explain when a callback function given to process.nextTick() executes

Model Short Answer
functions in the process.nextTick() queue will be executed before Node moves on to the next phase in the event loop. They have the highest priority of any scheduled callback.

Question [2] 2: Short Answer/Essay - [1.0] point Edit

Write a Node.js module called duck.js

Your module should export a Duck class which has the following properties:
- flying: boolean (starts false)
- quaking: boolean (starts false)
- xPos (starts at 0)
- yPos (starts at 0)

And the following methods

- takeOff() // sets flying to true
- land() // sets flying to false
- startQuacking() // sets quacking to true
- stopQuacking() // sets quacking to false
- moveTo(x, y) // changes the x and y to the given x, y and consle.logs the current status

Example status messages:
Duck is swimming to ${x},${y}
Duck is swimming to ${x},${y} while quacking
Duck is flying to ${x}, ${y}
Duck is flying to ${x}, ${y} while quacking

Model Short Answer

```
class Duck {
    constructor() {
        this.quaking = false;
        this.flying = false;
        this.xPos = 0;
        this.yPos = 0;
    }

    takeOff(){
        this.flying = true;
    }

    land() {
        this.flying = false;
    }

    startQuacking() {
        this.quacking = true;
    }

    stopQuacking() {
        this.quacking = false;
    }

    moveTo(x, y) {
        this.xPos = x;
        this.yPos = y;
        let msg = "Duck is "
        if (this.flying) {
            msg += `flying to ${x},${y} `;
        } else {
            msg += `swimming to ${x},${y} `;
        }
        if (this.quacking) {
            msg += "while quacking";
        }
        console.log(msg);
    }
}

module.exports = Duck;
```

Question [3] 3: Short Answer/Essay - [1.0] point Edit

Write a testDuck.js file to test the different functions exposed by the duck.js module (just run the methods, no need for Mocha / Chai)

Model Short Answer

```
const Duck = require("./duck");
var duck = new Duck();

duck.moveTo(1, 1);
duck.takeOff();
duck.moveTo(1, 2);
duck.startQuacking();
duck.moveTo(1, 3);
duck.land();
duck.moveTo(2, 3);
duck.stopQuacking();
duck.moveTo(3,3);
```

Question 1 1: Short Answer/Essay - 1.0 point Edit

Explain what the Post/Redirect/Get pattern is

Model Short Answer
After a POST request the server doesn't return HTML, instead it returns a redirect (303) with a URL which the browser GETs to show the user the result

Question 2 2: Short Answer/Essay - 1.0 point Edit

Write a small list application in Express.

GET to / should show the list (an unordered list based on an array variable that is local to the module), and also have a link on the page to /add
GET to /add should show a form with a single input that allows the user to enter text and a submit button
POST to /add should receive what was entered, add it to the array and then redirect to the list

Important: do not try to make valid html for any of these pages -- just use the bare minimum to create the requested functionality!

Model Short Answer

```
const express = require('express');
const app = express();

let list = [];

app.use(express.urlencoded({ extended: false }));

app.get('/', (req, res) => {
  let output = "<ul>";
  for (i of list) {
      output += `<li>${i}</li>`;
  }
  output += "</ul><a href='/add'>add</href>";
  res.send(output);
});
```

```
app.get('/add', (req, res) => {
    res.send(`<form method="post"><input name="item" /><input type="submit" /></form>
`);
});

app.post('/add', (req, res) => {
    list.push(req.body.item);
    res.redirect(303, "/");
});

app.listen(3000);
```

Question 1 1: Short Answer/Essay - 1.0 point Edit

Explain the MVC pattern

Model Short Answer
Model, View, Control these are the 3 main concerns of almost any piece of software that renders a graphical user interface (GUI).

Model is the code that relates to the data(base). View is the code the relates to creating the GUI, and Control is the code that ties them together.

Question [2] 2: Short Answer/Essay - [1.0] point Edit

Create the same list application as we made for the previous quiz, but this time use templates to output proper (completely valid) HTML.

Your application should respond to the following requests:

GET to / should show the list (an unordered list based on an array variable that is local to the module), and also have a link on the page to /add
GET to /add should show a form with a single input that allows the user to enter text and a submit button
POST to /add should receive what was entered, add it to the array and then redirect to the list

This time you should use templates and output 100% valid HTML5. You should submit:

- Your Express application
- Your views (templates)

Model Short Answer

```
const path = require('path');

const express = require('express');

const app = express();


let list = [];

app.set('view engine', 'ejs');

app.set('views', path.join(__dirname, 'view'));


app.use(express.urlencoded({ extended: false }));


app.get('/', (req, res) => {

  res.locals.list = list;

  res.render("list");

});


app.get('/add', (req, res) => {

    res.sendFile(path.join(__dirname, 'view', 'form.html'));
});

app.post('/add', (req, res) => {
    list.push(req.body.item);
    res.redirect(302, "/");
});

app.listen(3000);
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>List</title>
</head>

<body>
    <ul>
        <% for(let i of list) { %>
            <li><%= i %></li>
        <% } // close for loop %>
    </ul>
    <a href="/add">add</a>
</body>

</html>
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Form</title>
</head>
```

```
<body>
    <form method="post">
        <input name="item" />
        <input type="submit" />
    </form>
</body>

</html>
```

Question 1 1: Short Answer/Essay - 1.0 point

Edit

Give one way in which sessions and cookies are similar, then give one way in which sessions and cookies are different.

Model Short Answer

Cookies and sessions are the same in that they both allow you to store a key/value pair.

Cookies and sessions are different in that cookie data is stored on the client (browser) machine, while session data is stored on the server.

Question 2 2: Short Answer/Essay - 1.0 point Edit

The code we wrote in the quiz over the last 2 days had a single list object for anyone that came to the website.

Update this code to give each visitor their own list, in other words, put the list into the session.

You can test to see if each visitor has their own list by connecting to your server with two different browsers (chrome and firefox, or edge, or opera, or ...).

You don't need to include your templates, just submit your updated app.js

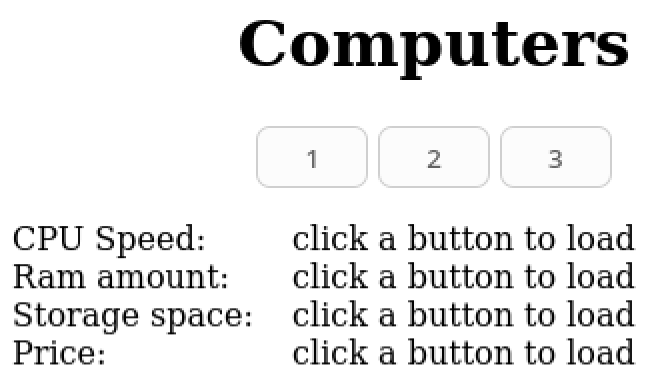
Model Short Answer

```
const path = require('path');

const express = require('express');

const app = express();

const session = require('express-session');


app.set('view engine', 'ejs');

app.set('views', path.join(__dirname, 'view'));


app.use(express.urlencoded({extended: false}));

app.use(session({

    secret: 'my super secrit secret',

    resave: false,

    saveUninitialized: true

}));

app.use((req, res, next) => {

    if (!req.session.list) {

        req.session.list = [];

    }

    next();

});
```

```
app.get('/', (req, res) => {
  res.locals.list = req.session.list;
  res.render('list');
});

app.get('/add', (req, res) => {
  res.sendFile(path.join(__dirname, 'view', 'form.html'));
});

app.post('/add', (req, res) => {
  req.session.list.push(req.body.item);
  res.redirect(302, '/');
});

app.listen(3000);
```

Make a page that looks like the screenshot (correct HTML5 and CSS).

Next question youll write AJAX code to retrieve JSON when a button is clicked and show the specifications for the computers related to the button



CPU Speed:	click a button to load
Ram amount:	click a button to load
Storage space:	click a button to load
Price:	click a button to load

Model Short Answer

```
<!DOCTYPE html>
<html>

<head>
    <title>Quiz</title>
    <meta charset="utf-8">
    <style>
        body {
            max-width: 450px;
            margin-left: auto;
            margin-right: auto;
        }
        h1, #btns { text-align: center; }
        span.label {
            display: inline-block;
            width: 135px;
        }
    </style>
</head>
```

```
<body>
    <h1>Computers</h1>
    <p id="btns">
        <button id="btn_1">1</button>
        <button id="btn_2">2</button>
        <button id="btn_3">3</button>
    </p>
    <div>
        <span class="label">CPU Speed:</span>
        <span id="cpuSpeed">click a button to load</span>
    </div>
    <div>
        <span class="label">Ram amount:</span>
        <span id="ramAmount">click a button to load</span>
    </div>
    <div>
        <span class="label">Storage space:</span>
        <span id="storage">click a button to load</span>
    </div>
    <div>
        <span class="label">Price:</span>
        <span id="price">click a button to load</span>
    </div>
</body>

</html>
```

Question [2] 2: Short Answer/Essay - [1.0] point Edit

Write AJAX code to GET JSON from "/computers" passing it a id parameter to indicate which button was pressed.

For example "/computers?id=1" should return:

```
{
    "cpu": "8 core 4Ghz",
    "ram": "16GB",
    "storage": "4TB NVME",
    "price": "$1500"
}
```

Your code should update the web page with this data.

Model Short Answer

```
"use strict";
$(() => {
    $("button").click(function () {
        $.get("computers.json", { "id": $(this).text() })
            .done(function (data) {
                $("#cpuSpeed").text(data.cpu);
                $("#ramAmount").text(data.ram);
                $("#storage").text(data.storage);
                $("#price").text(data.price);
            });
    });
});
```