VIRTUALMIND

Javascript

A. The following code suffers from a known condition called “Pyramid of Doom”: If we were to chain more server calls together, then the `PlayerDetailsController.showTeammatesClick` method would go too deep and become very unstable. This doesn’t allow for a good way to handle error, or application state, if we were to react to each call in particular.

Tip: Check what $.ajax returns and its supported methods/hooks

```
var PlayerService = {
	getPlayerTeamId: function(playerId, callback) {
		$.ajax({
			url: "/player/" + playerId + "/team",
			success: function(team) {
				callback(team.id)
			}
		});
	},
	getPlayers: function(teamId, callback) {
		$.ajax({
			url: "/team/" + teamId + "/player",
			success: callback
		});
	}
};

var PlayerDetailsController = {
	playerId: 8,
	showTeammatesClick: function() {
		PlayerService.getPlayerTeamId(this.playerId, function(teamId) {
			PlayerService.getPlayers(teamId, function(playerList) {
				// Render playerList
			});
		});
	}
};
```

Refactor the code to use promises. Some Acceptance Criteria on the new code:

- Keep the object definitions the same as in the example.
- Keep the function signatures and interfaces exactly as they are, except for `getPlayerTeamId` and `getPlayers`, which should not expect the `callback` parameter.
- Do not use callback functions in any way
- If `showTeammatesClick` is called, then the `playerList` must be rendered at some point, assuming that we have a stable communication with the server

A.2) Extra points for doing A) with async/await
(Please paste below links to your answers)

https://github.com/julianojcs/virtualmind

B. Collections Exercise

C. React Refactor Exercise

D. What kind of language is Javascript? *(remember, more than one (or none) options are possible)*

1. Strongly typed
2. Weakly typed <<<<
3. Dynamic <<<<
4. Prototype based <<<<
5. Functional <<<<
6. Static
7. Structured

E. Mark which of the following characteristics Javascript presents

1. Polymorphism <<<<
2. Inheritance <<<<
3. Encapsulation <<<<
4. Dynamic binding (The ability to switch an object's method at runtime) <<<<
5. Open recursion (Characteristic that implies that the "this" reference is dynamically bound) <<<<

F. Is Javascript Object Oriented?

1. Yes <<<<
2. No

Briefly describe why:

JavaScript can function as a procedural and an object oriented language. Javascript provides some features to implement object-oriented programs, such as polymorphism, encapsulation, inheritance (via prototyping), so it is a prototype-based language (not a class-based object-oriented).

G. What does a closure allow in Javascript?

1. Encapsulating code inside the scope of a function. <<<<
2. Allows declared variables to be accessible inside child scopes and inaccessible from parent scopes. <<<<
3. Allows declared variables to be accessible inside parent scopes and inaccessible from child scopes.
4. Currying <<<<
5. Event Bubbling <<<<

H. How would you deal with global scope in Javascript?:

1. Encapsulating components in functions <<<<
2. Using AMD or CommonJS Modules <<<<
3. Putting all the components under a same object
4. Using global variables

The answers for questions A, B and C are at the link below:

https://github.com/julianojcs/virtualmind