LEARN MODERN JAVASCRIPT METHODS BY BUILDING FOOTBALL TEAM CARDS

Introduction:

One common aspect of building web applications: processing datasets, and then outputting information to the screen. In this sports team cards project, you'll learn how to work with DOM manipulation, object destructuring, event handling, and data filtering.

This project will cover concepts like, default parameters, `Object.freeze()`, and reinforce your knowledge of the `switch` statement and `map()` method.

Step 1:

In this project, you will build a set of football team cards and learn about nested objects, object destructuring, and default parameters. All of the HTML and CSS for this project has been provided for you.

Start by accessing the `id` called `"team"` from the HTML document and storing it in a `const` variable called `teamName`.

Remember, you can use the `getElementById` method for this.

NOTE: The numbers for the team are organized alphabetically by last name. This differs from conventional numbering where the numbers correspond with what is on the player's jerseys.

Step 2:

Next, access the `id` called `"sport"` from the HTML document and store it in a `const` variable called `typeOfSport`. Below that

variable, assign the `id` of `"year"` to a `const` variable called `worldCupYear`.

Step 3:

Next, access the `id` called `"head-coach"` from the HTML document and store it in a `const` variable called `headCoach`. Below that variable, assign the `id` of `"player-cards"` to a `const` variable called `playerCards`.

Step 4:

Create one more `const` variable called `playersDropdownList` and assign it the `id` of `"players"` using the `getElementById` method.

Step 5:

Now it is time to build out the data structure that will hold all of the information for your football team.

Below the variables you just created, create a new `const` variable called `myFavoriteFootballTeam` and assign it an empty object.

Step 6:

Inside the `myFavoriteFootballTeam` object, add a new property with a key named `team` and a string value of `"Argentina"`.

Step 7:

Below the `team` property, add four property key-value pairs.

1. A key named `sport` and a string value of `"Football"`.
2. A key named `year` and a number value of `1986`.
3. A key named `isWorldCupWinner` and a boolean value set to `true`.
4. A new key called `headCoach` with a value of an empty object.

Step 8:

Inside the headCoach object, add a property with a key of `coachName` and a string value of `"Carlos Bilardo"`. Below that property, add another key called `matches` with a number value of `7`.

Step 9:

Below the `headCoach` property, create a new property with a key named `players` with the value of an empty array.

Step 10:

Inside that `players` array, create a new object with the following properties:

Example Code:

```
name: "Sergio Almirón"

position: "forward"

number: 1

isCaptain: false

nickname: null
```

NOTE: The numbers for the team are organized alphabetically by last name. This differs from conventional numbering where the numbers correspond with what is on the player's jerseys.

Step 11:

Below that object, create a new object with the following properties:

Example Code:

```
name: "Sergio Batista"

position: "midfielder"

number: 2

isCaptain: false

nickname: null
```

Step 12:

The rest of the data for the `myFavoriteFootballTeam.players` array has been filled out for you.

The next step is to ensure that you can't modify this object by adding or removing any properties. We are going to use a method called `Object.freeze(obj)` which will freeze this object and prevent any changes being made to it.

Use the `Object.freeze()` method to freeze the `myFavoriteFootballTeam` object.

Step 13:

The next step is to access the key called `sport` from the `myFavoriteFootballTeam` object and assign it to a new `const` variable called `sport`.

Remember you can use dot notation for this.

Step 14:

Below the `sport` variable, access the key called `team` from the `myFavoriteFootballTeam` object and assign it to a new `const` variable called `team`.

Step 15:

In the last two steps, you have been accessing properties from the `myFavoriteFootballTeam` object using dot notation and assigning them to new `const` variables. But in JavaScript, there is an easier way to accomplish the same goal.

The object destructuring syntax allows you to unpack values from arrays and objects:

Example Code:

```
const developerObj = {
  name: "Jessica Wilkins",
  isDeveloper: true
};
// Object destructuring
const { name, isDeveloper } = developerObj;
```

Rewrite the two lines of code below using the new destructuring syntax. Your answer should be one line of code.

Step 16:

Next, add the `year` and `players` to your destructuring assignment.

Step 17:

Now you need to access the `coachName` value from the `myFavoriteFootballTeam.headCoach` object using the destructuring syntax.

Step 18:

Now you need to start displaying the team's information on the screen.

Below your destructuring assignments, assign the sport variable to typeOfSport.textContent.

Once you complete that task, you should see the result in the preview window.

Step 19:

Next, assign the `team` variable to `teamName.textContent`.

Step 20:

Assign the `year` variable to `worldCupYear.textContent`.

Below that, assign the `coachName` variable to `headCoach.textContent`.

You should now see all of that information displayed on the screen below `Team stats`.

Step 21:

Now you will start building out the function that will show player cards based on the selections made by the user in the `Filter Teammates` dropdown menu.

Start by creating an empty arrow function called `setPlayerCards`. You do not need to add a parameter because that will be taken care of in the next step.

Step 22:

Function parameters can be initialized with default values. If a function is called without an argument, then the default value will be used:

Example Code:

```
const greeting = (name = "Anonymous") => {
  return "Hello " + name;
}

console.log(greeting("John")); // Hello John

console.log(greeting()); // Hello Anonymous
```

Add a new parameter to your `setPlayerCards` function called `arr` and assign it a default value of `players`. Remember that you

destructured the `players` variable from the `myFavoriteFootballTeam` object on line `175`.

Step 23:

The next step is to create a new array that will be responsible for adding the player card information to the page.

Inside the `setPlayerCards` function, start by adding the `map` method to `arr` that will take in an empty callback function. Then, use the addition assignment `+=` operator to assign the new array to `playerCards.innerHTML`.

Remember that the `innerHTML` property gets, or in this case, sets the HTML markup for the `playerCards` element.

Step 24:

`arr` contains a series of objects that each contains a `name`, `position`, `number`, `isCaptain` and `nickname` property. In order to access each of those properties inside the callback function, you will need to use object destructuring to unpack them into variables.

Here is an example:

Example Code:

```
function myExampleFunction({ name, age, job, city }) {

}
```

Inside the parameter list in the callback function for the `map` method, unpack all 5 object properties from objects in `arr` using object destructuring.

Step 25:

Inside the body of the callback function, you will need to return a template literal `` which will contain the HTML content for the player cards.

Inside the template literals, add an empty `div` with a class of `"player-card"`.

Step 26:

Inside the `div`, add an `h2` element which contains the `name` parameter. Since you are working with template literals, you will need to use an embedded expression for the `name` parameter:

Example Code:

```
${expression goes here}
```

Step 27:

The next step would be to display the word `(Captain)` next to the player if they are listed as a captain for the team.

Before the `${name}` expression, add a new embedded expression. Inside that expression, use a ternary operator to check if `isCaptain` is true. If so, return `"(Captain)"` otherwise return an empty string.

Step 28:

Below the `h2` element, add a paragraph element with the text `Position:` and an embedded expression that contains the `position` parameter.

Step 29:

Below the paragraph element, add another paragraph element with the text `Number:` and an embedded expression that contains the `number` parameter.

Step 30:

Below your existing paragraph elements, add another paragraph element with the text `Nickname:` .

Step 31:

Next to the `Nickname:` text, add an embedded expression that will show the player's nickname if they have one.

Use a ternary operator to check if `nickname` is not `null`. If the player has a nickname, display `nickname` otherwise display `"N/A"`.

Step 32:

The `.map()` method will return a new array of `player-card` items separated by commas.

To remove the commas between each `player-card` so it does not show up on screen, chain the `.join()` method to the `.map()` method. Pass an empty string as the argument for the `.join()` method.

Step 33:

The next step is to create a function that will detect when a user makes a selection from the `playersDropdownList`.

Use the `.addEventListener()` method on `playersDropdownList`. Inside the event listener, pass in a `"change"` event type and an empty callback function.

Step 34:

For the callback function, pass in `e` as a parameter.

`e` represents an object which contains the information for that event.

Step 35:

Inside the callback function, add a `console.log` with the value of `e.target.value`.

Open the console, and make a selection from the teammates dropdown menu. You should see the value of that selection in the console.

`e.target.value` represents the `value` property from the `playersDropdownList` element. In future steps, you will use this value to show player cards based on the position they play.

Step 36:

Remove the `console.log` statement you created in the previous step.

The next step would be to reset the content for the `playerCards` element.

Inside the callback function, access the `innerHTML` property of the `playerCards` element and assign it a value of an empty string.

Step 37:

The next step would be to add a `switch` statement which will check for the user's selection from the player dropdown menu and filter out cards based on the player's positions.

Add a `switch` statement and use `e.target.value` for the expression.

Step 38:

If the user selects `Nicknames` from the dropdown menu you will want to filter out player cards that have a nickname.

Start by adding a `case` clause for `"nickname"` inside your `switch` statement.

Step 39:

Call the `setPlayerCards` function with an argument of `players.filter()`.

Inside the `filter` method, add a callback function with a parameter called `player` and implicitly return `player.nickname` is not `null`.

Step 40:

Before you can move onto the next `case`, you will need to add a `break` statement.

Below your `setPlayerCards` call, add a `break` statement.

Step 41:

Next, add a `case` clause for `"forward"`.

Inside that `case`, call the `setPlayerCards` function with an argument of `players.filter()`.

Inside the `filter()` method, add a callback function with a parameter of `player` that will check if `player.position` equals `"forward"`.

Lastly, add a `break` statement below the `setPlayerCards` function call.

Step 42:

Add a new `case` for `"midfielder"` that checks if `player.position` equals `"midfielder"` following the same pattern from the previous step.

Step 43:

Add a new `case` for `"defender"` that checks if `player.position` equals `"defender"` following the same pattern as the previous step.

Step 44:

Add a new `case` for `"goalkeeper"` that checks if `player.position` equals `"goalkeeper"` following the same pattern as the previous step.

Step 45:

The final step is to add a `default` clause if none of the other `case` clauses match the user selection.

For the `default` clause, call the `setPlayerCards` function without any arguments passed in.

Test out your dropdown menu, and you should see the player cards be filtered out by position or nickname.

Congratulations on completing the football team cards project.