Javascript Problem Solving Questions

DOM | String | Arrays | Callbacks | Events | Recent features

Question1

You are building a webpage where you need to manipulate the DOM dynamically based on user data. A section of your page contains a list of users, and the goal is to allow the user to add a new user to the list.

1. Create a function addUser(name, email) that adds a new user object to an existing array of user objects. Then, create a new HTML element dynamically using JavaScript that displays the user's name and email on the page.
2. Update the page dynamically using DOM manipulation without reloading it.

Sample Output:

```
<ul id="user-list">
    <li>Name: John Doe, Email: john@example.com</li>
</ul>
```

Question2

You are tasked with validating user input and ensuring that it meets the required conditions. Write a function that takes an array of strings representing usernames and filters out those that don't meet the following criteria:

1. Length must be between 4 and 12 characters.
2. Must not contain special characters or spaces.

Sample Output:

```
let usernames = ["user_01", "john123", "invalid user", "xyz@!"];

let validateUsernames = (data) => {

        //todo

}

console.log(validateUsernames(usernames));
```

Question3

Use following share market sample data to answer the questions

```
{
  "market": {
    "name": "Global Stock Exchange",
    "location": "New York",
    "date": "2024-11-14",
    "indices": [
      {
        "name": "S&P 500",
        "symbol": "SPX",
        "companies": [
          {
            "name": "Apple Inc.",
```

```
        "symbol": "AAPL",
        "sector": "Technology",
        "currentPrice": 175.65,
        "priceChange": -2.35,
        "marketCap": "2.78T",
        "dividends": {
          "annualYield": 0.58,
          "payoutRatio": 24.5
        },
        "historicalPrices": [
          { "date": "2024-11-13", "close": 178.00 },
          { "date": "2024-11-12", "close": 176.50 },
          { "date": "2024-11-11", "close": 179.30 }
        ]
      },
      {
        "name": "Tesla Inc.",
        "symbol": "TSLA",
        "sector": "Automotive",
        "currentPrice": 250.75,
        "priceChange": +5.50,
        "marketCap": "880B",
        "dividends": {
          "annualYield": 0.00,
          "payoutRatio": 0.0
        },
        "historicalPrices": [
          { "date": "2024-11-13", "close": 245.25 },
          { "date": "2024-11-12", "close": 247.00 },
          { "date": "2024-11-11", "close": 244.00 }
        ]
      }
    ]
  },
  {
    "name": "NASDAQ",
    "symbol": "IXIC",
    "companies": [
      {
        "name": "Google LLC",
        "symbol": "GOOG",
        "sector": "Technology",
        "currentPrice": 2850.90,
        "priceChange": +18.75,
        "marketCap": "1.9T",
        "dividends": {
          "annualYield": 0.00,
          "payoutRatio": 0.0
        },
        "historicalPrices": [
          { "date": "2024-11-13", "close": 2832.15 },
          { "date": "2024-11-12", "close": 2835.00 },
          { "date": "2024-11-11", "close": 2825.20 }
        ]
      },
      {
        "name": "Microsoft Corp.",
        "symbol": "MSFT",
        "sector": "Technology",
        "currentPrice": 345.10,
        "priceChange": +4.20,
```

```
            "marketCap": "2.60T",
            "dividends": {
              "annualYield": 0.90,
              "payoutRatio": 32.5
            },
            "historicalPrices": [
              { "date": "2024-11-13", "close": 340.90 },
              { "date": "2024-11-12", "close": 341.00 },
              { "date": "2024-11-11", "close": 338.75 }
            ]
          }
        ]
      }
    ]
  }
}
```


Q1) Write a function filterPositivePriceChange(marketData) that filters and returns all companies that have a priceChange greater than 0.

Sample Output:

[{ "name": "Tesla Inc.", "symbol": "TSLA", "currentPrice": 250.75, "priceChange": 5.50 },

{ "name": "Google LLC", "symbol": "GOOG", "currentPrice": 2850.90, "priceChange": 18.75 },

{ "name": "Microsoft Corp.", "symbol": "MSFT", "currentPrice": 345.10, "priceChange": 4.20 }

]

Q2) Write a function findCompanyBySymbol(marketData, symbol) that returns the company details based on the provided symbol.

Sample Output:

{ "name": "Google LLC",

"symbol": "GOOG",

"sector": "Technology",

"currentPrice": 2850.90,

"priceChange": 18.75,

"marketCap": "1.9T",

"dividends": { "annualYield": 0.00, "payoutRatio": 0.0 },

"historicalPrices": [

{ "date": "2024-11-13", "close": 2832.15 },

{ "date": "2024-11-12", "close": 2835.00 },

{ "date": "2024-11-11", "close": 2825.20 }

] }

Q3) Write a function sortCompaniesByMarketCap(marketData) that returns the companies sorted by their marketCap in descending order.

Sample Output:

```
[ { "name": "Microsoft Corp.", "symbol": "MSFT", "marketCap": "2.60T" },
{ "name": "Apple Inc.", "symbol": "AAPL", "marketCap": "2.78T" },
{ "name": "Google LLC", "symbol": "GOOG", "marketCap": "1.9T" },
{ "name": "Tesla Inc.", "symbol": "TSLA", "marketCap": "880B" } ]
```

Q4) Write a function findAndSortHighCapCompanies(marketData) that filters out companies with a market cap above 1 trillion and sorts them by priceChange in descending order.

Sample Output:

```
[
 { "name": "Microsoft Corp.", "symbol": "MSFT", "marketCap": "2.60T", "priceChange": 4.20 },
 { "name": "Apple Inc.", "symbol": "AAPL", "marketCap": "2.78T", "priceChange": -2.35 },
 { "name": "Google LLC", "symbol": "GOOG", "marketCap": "1.9T", "priceChange": 18.75 }
]
```

Question4

Use following share market sample data to answer the questions

```
const busRoutes = [
 // Route 1: From Location A to Location D
 [
  "Route 1: A → D",           // Route name
  ["A", "B", "C", "D"],       // Stops
  [120, 80, 150, 90],         // Passengers at each stop
  [5, 8, 6, 7]              // Time intervals between stops (in minutes)
 ],

 // Route 2: From Location B to Location E
 [
  "Route 2: B → E",           // Route name
  ["B", "C", "D", "E"],       // Stops
  [100, 140, 110, 60],        // Passengers at each stop
  [10, 7, 5, 10]            // Time intervals between stops (in minutes)
 ],

 // Route 3: From Location C to Location F
 [
  "Route 3: C → F",           // Route name
  ["C", "D", "E", "F"],       // Stops
  [80, 50, 120, 130],         // Passengers at each stop
  [15, 12, 10, 6]           // Time intervals between stops (in minutes)
 ],

 // Route 4: From Location A to Location E
 [
```

```
  "Route 4: A → E",           // Route name
  ["A", "B", "C", "D", "E"],   // Stops
  [130, 90, 80, 150, 70],      // Passengers at each stop
  [6, 8, 5, 9, 7]             // Time intervals between stops (in minutes)
 ]
];
```

Q1) Write a function filterRoutesByHighDemand(busRoutes) that filters and returns all routes where at least one stop has passenger demand greater than 100.

Sample Output:

```
[
 ["Route 1: A → D", ["A", "B", "C", "D"], [120, 80, 150, 90], [5, 8, 6, 7]],
 ["Route 2: B → E", ["B", "C", "D", "E"], [100, 140, 110, 60], [10, 7, 5, 10]],
 ["Route 3: C → F", ["C", "D", "E", "F"], [80, 50, 120, 130], [15, 12, 10, 6]],
 ["Route 4: A → E", ["A", "B", "C", "D", "E"], [130, 90, 80, 150, 70], [6, 8, 5, 9, 7]]
]
```

Q2) Write a function findRouteWithMaxDemand(busRoutes) that finds the route with the highest passenger demand at any stop and returns the route's name.

Sample Output:

```
"Route 1: A → D"  // This route has the highest passenger demand at stop 3 (150 passengers)
```

Q3) Write a function sortRoutesByAverageDemand(busRoutes) that sorts the routes by their average passenger demand in ascending order.

Sample Output:

```
[
 ["Route 3: C → F", ["C", "D", "E", "F"], [80, 50, 120, 130], [15, 12, 10, 6]],
 ["Route 2: B → E", ["B", "C", "D", "E"], [100, 140, 110, 60], [10, 7, 5, 10]],
 ["Route 4: A → E", ["A", "B", "C", "D", "E"], [130, 90, 80, 150, 70], [6, 8, 5, 9, 7]],
 ["Route 1: A → D", ["A", "B", "C", "D"], [120, 80, 150, 90], [5, 8, 6, 7]]
]
```

Q4) Write a function calculateTotalDemand(busRoutes) that calculates and returns an array where each element is an array with the route name and its total passenger demand.

Sample Output:

```
[
 ["Route 1: A → D", 440],
 ["Route 2: B → E", 410],
 ["Route 3: C → F", 380],
 ["Route 4: A → E", 520]
]
```

Q5) Write a function calculateTotalTravelTime(busRoutes) that returns an array where each element contains the route name and its total travel time.

Sample Output:

```
[
 ["Route 1: A → D", 26],
 ["Route 2: B → E", 32],
 ["Route 3: C → F", 43],
 ["Route 4: A → E", 35]
]
```