Pulse Innovations A Sony Company

Pulseball Rankings Table

Introduction »

The purpose of this exercise is to evaluate your ability to develop in JavaScript and implement a specification. The primary goal is to produce a web application that implements a rankings predictor for the fictitious game of Pulseball. Secondary goals are to make something that looks nice or has additional features.

Thought should be given to all aspects of development in particular code quality and clarity. Consideration should also be given to UI/UX elements.

The exercise should be completed entirely in JavaScript and run without a back-end.

The Game of Pulseball »

In the international game of Pulseball, various series and tournaments contribute to the overall ranking of the teams. Some games are played at home, some games are played away and some games (especially those at the tournament level) are played on neutral ground.

To calculate the number of points earned or lost by each team after a game, we first subtract the second team's rating from the first's. To suppress the advantage playing at home might give for the home team, its rating is temporarily increased by 3 points when performing the calculations.

If the rating points difference is greater than 10, then it is capped at 10. If the rating points difference is lower than -10, then it is capped at -10.

1. If the first team wins, then it gains 1 - (rating difference / 10) points and the opposing team's rating loses the same amount

2. if the first team loses, then it loses 1 + (rating difference / 10) points and the opposing team's rating wins the same amount
3. In the case of a draw, each team loses or wins the points calculated by dividing the rating difference by 10.

Provided the current table is defined by the following JSON:

```
[
  { "team": { "name": "Australia", "id": 32 }, "pos": 1, "pts": 54.23
},
  { "team": { "name": "New Zealand", "id": 62 }, "pos": 2, "pts":
54.00 },
  { "team": { "name": "France", "id": 2 }, "pos": 3, "pts": 52.95 },
  { "team": { "name": "England", "id": 1 }, "pos": 4, "pts": 52.32 },
  { "team": { "name": "Romania", "id": 24 }, "pos": 5, "pts": 43.50 }
]
```

Where "pos" is the position of the team in the table and "pts" are the rating points for that team.

If France (the first team) is defeated by England (the second team) 23-19 in France, the match is represented as follows:

```
{
  "matchId": 2524,
  "description": "Match 2",
  "venue": {
    "id": 900,
    "name": "Stadium",
    "city": "Paris",
    "country": "France"
  },
  "teams": [
    {
      "id": 2,
      "name": "France",
      "abbreviation": "FRA"
    },
    {
      "id": 1,
      "name": "England",
      "abbreviation": "ENG"
    }
  ],
  "scores": [
    19,
    23
  ],
  "status": "C",
  "outcome": "B"
}
```

Where the outcome can be “A” for team 1 win, “B” for team 2 win, “D” for a draw, “N” for no result.

Status is an enum representing the current status of the match, ‘U’ for upcoming matches, ‘L’ for live matches and ‘C’ for completed matches.

Applying the match data to a global function with the signature `PULSEBALL.addMatch( match )` would create a new rankings table:

```
[
  { "team": { "name": "Australia", "id": 32 }, "pos": 1, "pts": 54.23
},
  { "team": { "name": "New Zealand", "id": 62 }, "pos": 2, "pts":
54.00 },
  { "team": { "name": "England", "id": 1 }, "pos": 4, "pts": 53.68 },
  { "team": { "name": "France", "id": 2 }, "pos": 3, "pts": 51.59 },
  { "team": { "name": "Romania", "id": 24 }, "pos": 5, "pts": 43.50 }
]
```

Where the new points values are rounded to two decimal places.

When the rankings are updated the new rankings table is used to calculate the ratings points difference. This way it is possible to predict the results of several matches and see how the rankings would be updated.

Task »

Build a rankings predictor web application in JavaScript that gets initialised through a function `PULSEBALL.init( rankingsJson )`, where `rankingsJson` is in the format of the rankings table JSON examples above and updates with `PULSEBALL.addMatch( match )`.

Optional extensions include:

- documentation
- automated build process
- testing (unit etc.)
- user interface
- version control