

# P1: students array demo using map, filter, find, reduce

The screenshot shows a VS Code editor on the left with a JavaScript file named `app.js`. The code defines a `students` array and performs several operations: filtering for high scores, finding a specific student, calculating the average score, and reducing the array to a survey. On the right, a browser window at `127.0.0.1:5500/index.html` displays the DevTools console. The console shows the results of these operations: `updateStudents` (an array of 5 objects), `highScores` (an array of 1 object), `specificId` (an object), `averageScore` (73), and `survey` (an object).

```
const highScores = students.filter((student) => {
  if(student.score > 85) {
    return student;
  }
  // if(student.score > 85) return student; (其他写法)
  // return student.score > 85;(其他写法)
});
console.log('highScores', highScores);

const specificId = students.find((student) => {
  return student.id === 1;
});
console.log('specificId', specificId);

const averageScore = students.reduce((scoreTotal, student) => {
  // console.log('student', student);
  // console.log('scoreTotal', scoreTotal);
  return scoreTotal + student.score;
}, 0) / students.length;
console.log('averageScore', averageScore);

const survey = students.reduce((survey, student) => {
  // console.log('favoriteScore', student.favoriteCourse);
  const favCourse = student.favoriteCourse;
  if(survey[favCourse]) {
    survey[favCourse] = survey[favCourse] + 1;
  } else {
    survey[favCourse] = 1;
  }
  return survey;
}, {});
console.log('survey', survey);
```

DevTools Console Output:

- `updateStudents`: `Array(5)`
  - `0`: `Array(5)`
    - `favoriteCourse`: "Psychology"
    - `id`: 1
    - `name`: "Cooper"
    - `score`: 50
  - `1`: `[[Prototype]]: Object`
  - `2`: `{id: 2, name: 'Vincent', score: 100, favoriteCourse: 'english'}`
  - `3`: `{id: 3, name: 'Rex', score: 75, favoriteCourse: 'PE'}`
  - `4`: `{id: 4, name: 'Rick', score: 60, favoriteCourse: 'yoga'}`
  - `5`: `{id: 5, name: 'Jason', score: 80, favoriteCourse: 'cooking'}`
- `highScores`: `Array(1)`
  - `0`: `{id: 2, name: 'Vincent', score: 100, favoriteCourse: 'english'}`
- `specificId`: `Object`
- `averageScore`: `73`
- `survey`: `Object`
  - `PE`: 1
  - `Psychology`: 1
  - `cooking`: 1
  - `english`: 1
  - `yoga`: 1

# P2: getRandomUser three times

The screenshot shows a VS Code editor on the left with a JavaScript file named `script.js`. The code defines a `main` function that interacts with a web page. It includes buttons for adding a user, doubling money, showing only millionaires, sorting by richest, and calculating entire wealth. On the right, a browser window at `https://randomuser.me/api` displays the DOM Array Methods. The console shows the results of these operations: `random user data` (an array of 1 object), `data` (an array of 1 object), `random user data` (an array of 1 object), `data` (an array of 1 object), `random user data` (an array of 1 object), and `data` (an array of 1 object).

```
const main = document.querySelector("#main");
const addUserBtn = document.querySelector("#add-user");
const doubleBtn = document.querySelector("#double");
const showMillionairesBtn = document.querySelector("#show-millionaires");
const sortBtn = document.querySelector("#sortBtn");
const calculateBtn = document.querySelector("#calculateBtn");

let data = [];

const addData = (obj) => {
  data.push(obj);
  console.log('data', data);
};

const getRandomUser = async () => {
  const res = await fetch('https://randomuser.me/api');
  const data = await res.json();
  console.log('random user data', data);
  const user = data.results[0];
  const newUser = {
    name: `${user.name.first} ${user.name.last}`,
    money: Math.floor(Math.random() * 10000000)
  };
  addData(newUser);
};

getRandomUser();
getRandomUser();
getRandomUser();

addUserBtn.addEventListener('click', getRandomUser);
```

DOM Array Methods:

- `random user data`: `{results: Array(1), info: {}}`
- `data`: `[[-]]`
- `random user data`: `{results: Array(1), info: {}}`
- `data`: `[[-]]`
- `random user data`: `{results: Array(1), info: {}}`
- `data`: `[[-]]`

Buttons:

- Add User
- Double Money
- Show Only Millionaires
- Sort by Richest
- Calculate entire Wealth

## P3: add 6 users to the DOM

script.js

index.html

script.js > updateDOM > tempData > providedData.map() callback

const showMillionairesBtn = document.querySelector('#show-millionaires')

const sortBtn = document.querySelector('#sortBtn')

const calculateBtn = document.querySelector('#calculateBtn')

let data = [];

const updateDOM = (providedData = data) => {

let tempData = providedData.map((item) => {

return `<div class="person"><strong>\${item.name}</strong> \${formatMoney

});

tempData = tempData.join('');

console.log('tempData', tempData);

main.innerHTML = `<h2><strong>Person</strong> Wealth</h2>\${tempData}`

}

const addData = (obj) => {

data.push(obj);

console.log('data', data);

updateDOM();

}

// Format number as money - https://stackoverflow.com/questions/149055/how-to-

function formatMoney(number) {

return '\$' + number.toFixed(2).replace(/\d(?=(\d{3})+\.)/g, '\$&','');

}

const getRandomUser = async () => {

const res = await fetch('https://randomuser.me/api');

const data = await res.json();

console.log('random user data', data)

const user = data.results[0];

const newUser = {

name: `\${user.name.first} \${user.name.last}`,

money: Math.floor(Math.random()\*10000000)

}

const updateDOM = (providedData = data) => {

let tempData = providedData.map((item) => {

return `<div class="person"><strong>\${item.name}</strong> \${formatMoney

});

tempData = tempData.join('');

console.log('tempData', tempData);

main.innerHTML = `<h2><strong>Person</strong> Wealth</h2>\${tempData}`

}

const addData = (obj) => {

data.push(obj);

console.log('data', data);

updateDOM();

}

// Format number as money - https://stackoverflow.com/questions/149055/how-to-

function formatMoney(number) {

return '\$' + number.toFixed(2).replace(/\d(?=(\d{3})+\.)/g, '\$&','');

}

const getRandomUser = async () => {

const res = await fetch('https://randomuser.me/api');

const data = await res.json();

console.log('random user data', data)

const user = data.results[0];

const newUser = {

name: `\${user.name.first} \${user.name.last}`,

money: Math.floor(Math.random()\*10000000)

}

https://randomuser.me/api

DOM Array Methods

127.0.0.1:5500/index.html

Person

Add User

Double Money

Show Only Millionaires

Sort by Richest

Calculate entire Wealth

Alfred Romero \$9,545,506.00

Greg Grant \$1,396,870.00

Sebastian Rasmussen \$8,847,540.00

Aksel Lindquist \$5,335,007.00

Nika Van der Kooi \$1,254,037.00

Claudine Lopes \$1,569,684.00

Wendy Miller \$1,255,330.00

Felix Wong \$9,969,581.00

Pablo Bernard \$6,421,975.00

Camila Terry \$1,067,943.00

## P4: add 6 users first, then filter condition set to > 30000000

script.js

index.html

script.js > updateDOM > tempData > providedData.map() callback

ratio: 2;

showCondition: 30000000;

const updateDOM = (providedData = data) => {

let tempData = providedData.map((item) => {

return `<div class="person"><strong>\${item.name}</strong> \${

formatMoney(

item.money

)}</div>`;

});

tempData = tempData.join('');

console.log('tempData', tempData);

main.innerHTML = `<h2><strong>Person</strong> Wealth</h2>\${

tempData

}`;

const addData = (obj) => {

data.push(obj);

console.log('data', data);

updateDOM();

}

// Format number as money - https://stackoverflow.com/questions/149055/how-to-format-numbers-as-currency-string

function formatMoney(number) {

return '\$' + number.toFixed(2).replace(/\d(?=(\d{3})+\.)/g, '\$&','');

}

const getRandomUser = async () => {

const res = await fetch('https://randomuser.me/api');

const data = await res.json();

DOM Array Methods

127.0.0.1:5500/index.html

Person Wealth

Add User

Double Money

Show Only Millionaires

Sort by Richest

Calculate entire Wealth

Vilma Hiltunen \$4,621,975.00

Tara Nelson \$7,653,706.00

Galileu Almeida \$3,408,206.00

Kent Lawson \$9,969,581.00

Anna Cox \$9,322,892.00

Ella Wong \$6,730,365.00

# P5: use config for three buttons

File Edit Selection View Go Run Terminal Help

scriptjs - theme - Visual Studio Code

scriptjs x index.html

scriptjs > showMillionaires > filteredData > data.filter() callback

```
45   name: `${user.name.first} ${user.name.last}`,
46   money: Math.floor(Math.random() * 1000000),
47 };
48 addData(newUser);
49 };
50
51 getRandomUser();
52 getRandomUser();
53 getRandomUser();
54
55 const getRandomFiveUser = async () => {
56   for (let i = 0; i < config.numRandom; i++) {
57     const res = await fetch('https://randomuser.me/api');
58     const data = await res.json();
59     console.log('random user data', data);
60     const user = data.results[0];
61     const newUser = {
62       name: `${user.name.first} ${user.name.last}`,
63       money: Math.floor(Math.random() * config.showCondition),
64     };
65     addData(newUser);
66   }
67 };
68
69 const doubleMoney = () => {
70   data = data.map((user) => {
71     return {
72       name: user.name,
73       money: user.money * 1.5,
74     };
75   });
76   updateDOM();
77 };
78
```

Ln 81, Col 32 Spaces: 4 UTF-8 CRLF JavaScript Port: 5500 Prettier

DOM Array Methods

w06\_xc.pdf

127.0.0.1:5500/index.html

Microsoft Teams TKU 淡江大學首頁 Google Drive OneDrive Java IDE 下載 - 安... Class Beautiful Free Imag...

## DOM Array Methods

Add User (5)

Raise 1.5 Money

Show > 2000000

Sort by Richest

Calculate entire Wealth

Person	Wealth
Soraya Jean Webb	\$16,148,909.00
Lena Webb	\$10,213,765.00
Elias Poulsen	\$27,366,675.00
Jonas Alves	\$9,941,153.00
Kuzey Çetin	\$28,169,647.00

Identify your project's root folder to open source files in Visual Studio Code and sync changes. [Learn more](#)

Set root folder Don't show again

Console

Default levels

```
$16,148,909.00</div><div class="person"><strong>Lena Webb</strong> $10,213,765.00</div><div class="person"><strong>Elias Poulsen</strong> $27,366,675.00</div><div class="person"><strong>Jonas Alves</strong> $9,941,153.00</div>
random user data
> (results: Array(1), info: {...})
data
▼ (8) [(-), (-), (-), (-), (-), (-), (-), (-)]
  ▶ 0: {name: 'Lincoln Robinson', money: 1873878}
  ▶ 1: {name: 'Uma Østbye', money: 34322}
  ▶ 2: {name: 'Sander Christensen', money: 429114}
  ▶ 3: {name: 'Soraya Jean', money: 16148909}
  ▶ 4: {name: 'Lena Webb', money: 10213765}
  ▶ 5: {name: 'Elias Poulsen', money: 27366675}
  ▶ 6: {name: 'Jonas Alves', money: 9941153}
  ▶ 7: {name: 'Kuzey Çetin', money: 28169647}
  length: 8
  ▶ [[Prototype]]: Array(8)
tempData <div class="person">
<strong>Lincoln Robinson</strong> $1,873,878.00</div>
<div class="person"><strong>Uma Østbye</strong> $34,322.00</div><div class="person"><strong>Sander Christensen</strong> $429,114.00</div><div class="person"><strong>Soraya Jean</strong> $16,148,909.00</div><div class="person"><strong>Lena Webb</strong> $10,213,765.00</div><div class="person"><strong>Elias Poulsen</strong> $27,366,675.00</div>
<div class="person"><strong>Jonas Alves</strong> $9,941,153.00</div><div class="person"><strong>Kuzey Çetin</strong> $28,169,647.00</div>
filteredData
▼ (5) [(-), (-), (-), (-), (-)]
  ▶ 0: {name: 'Soraya Jean', money: 16148909}
  ▶ 1: {name: 'Lena Webb', money: 10213765}
  ▶ 2: {name: 'Elias Poulsen', money: 27366675}
  ▶ 3: {name: 'Jonas Alves', money: 9941153}
  ▶ 4: {name: 'Kuzey Çetin', money: 28169647}
  length: 5
  ▶ [[Prototype]]: Array(0)
tempData <div class="person">
<strong>Soraya Jean</strong> $16,148,909.00</div><div class="person"><strong>Lena Webb</strong> $10,213,765.00</div><div class="person"><strong>Elias Poulsen</strong> $27,366,675.00</div><div class="person"><strong>Jonas Alves</strong> $9,941,153.00</div><div class="person"><strong>Kuzey Çetin</strong> $28,169,647.00</div>
```

上午 10:35 2022/3/31