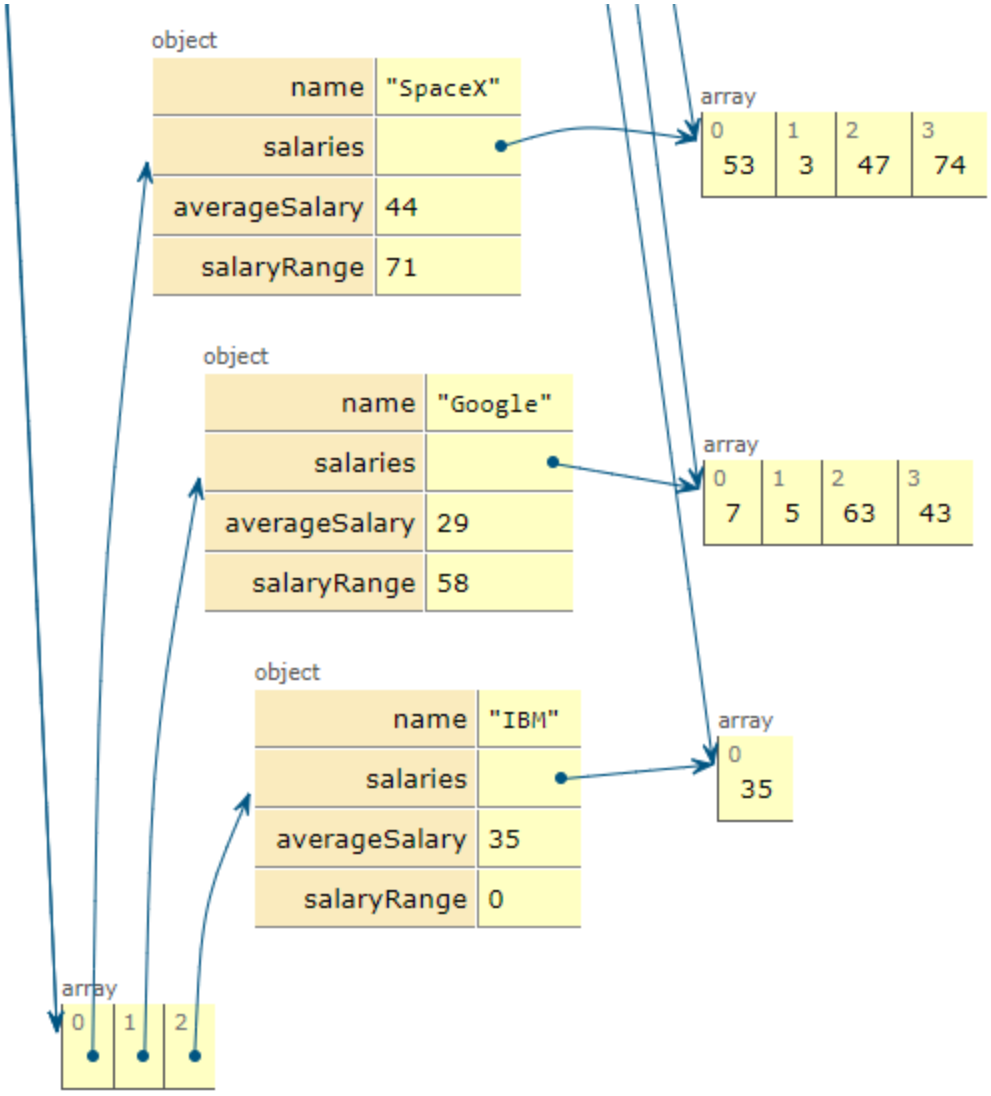
Aide au DS 2024

Méthodes

🥷Passez votre chemin si vous savez transformer la Base de Données en triant les compagnies par écart de salaires !

Avant	Après
[{ name: "Google", salaries: [7, 5, 63, 43] }, { name: "IBM", salaries: [35] }, { name: "SpaceX", salaries: [53, 3, 47, 74] }];	[{name: 'SpaceX', salaries: [53, 3, 47, 74], averageSalary: 44, salaryRange: **71** }, {name: 'Google', salaries: [7, 5, 63, 43], averageSalary: 29, salaryRange: **58** }, { name: 'IBM', salaries: [35], averageSalary: 35, salaryRange: **0** }]

Aide au DS 2024



[lien]

🚀Trouvez les compagnies avec au moins un salaire plus grand que 60k

1. const companiesWithHighSalaries = companies.filter(company =>
2. company.salaries.some(salary => salary > 60)
3.);
4.

5. console.log('Companies with at least one salary higher than 60k:', companiesWithHighSalaries);

🚀Quelle est la compagnie avec le plus(haut) salaire ?

Ne pas utiliser la version avec forEach suivante :

```
1. const findCompanyWithLowestSalary = (companies) => {
2.   let lowestSalary = Infinity;
3.   let companyWithLowestSalary = null;
4.
5.   companies.forEach(company => {
6.     const minSalary = Math.min(...company.salaries);
7.     if (minSalary < lowestSalary) {
8.       lowestSalary = minSalary;
9.       companyWithLowestSalary = company.name;
10.    }
11. });
12.
13. return companyWithLowestSalary;
14.};
15.
16.const companyWithLowestSalary =
    findCompanyWithLowestSalary(companies);
17.console.log(`The company with the lowest salary is:
    ${companyWithLowestSalary}`);
```

Utilisez la version avec reduce

Aide au DS 2024

```
const findCompanyWithLowestSalary = (companies) => {
  return companies.reduce((acc, company) => {
    const minSalary = Math.min(...company.salaries);
    if (minSalary < acc.lowestSalary) {
      return { lowestSalary: minSalary, companyName: company.name };
    }
    return acc;
  }, { lowestSalary: Infinity, companyName: null }).companyName;
};
```

🚀 Remplacer Math.max par un reduce

```
const findCompanyWithHighestSalary = (companies) => {
  return companies.reduce((acc, company) => {
    const maxSalary = Math.max(...company.salaries);
    if (maxSalary > acc.highestSalary) {
      return { highestSalary: maxSalary, companyName: company.name };
    }
    return acc;
  }, { highestSalary: -Infinity, companyName: null }).companyName;
};

```

```
const max = (a, b) => (a > b ? a : b);

const findCompanyWithHighestSalary = (companies) => {
  return companies.reduce((acc, company) => {
```

```
    const maxSalary = company.salaries.reduce(max, -Infinity);
    if (maxSalary > acc.highestSalary) {
      return { highestSalary: maxSalary, companyName: company.name };
    }
    return acc;
  }, { highestSalary: -Infinity, companyName: null });
};

const { companyName: companyWithHighestSalary } =
  findCompanyWithHighestSalary(companies);
console.log(`The company with the highest salary is:
  ${companyWithHighestSalary}`);
```

Lig; 13 : destructuring for the fun.

🚀 Ajoutez l'attribut **averageSalary** à chaque compagnie en utilisant l'opérateur de décomposition (...)

```
const calculateAverage = (salaries) => {
  const total = salaries.reduce((sum, salary) => sum + salary, 0);
  return Math.floor(total / salaries.length);
};

const companiesWithAverageSalaries = companies.map(company => ({
  ...company,
  averageSalary: calculateAverage(company.salaries)
}));
```

```
console.log('Companies with their average salaries:',
    companiesWithAverageSalaries);
```

🚀🚀🚀 Triez les compagnies par le plus gros écart entre salaires.

```
const companies = [
  { name: "Google", salaries: [7, 5, 63, 43] },
  { name: "IBM", salaries: [35] },
  { name: "SpaceX", salaries: [53, 3, 47, 74] },
];

const calculateStats = (salaries) => {
  const { total, maxSalary, minSalary } = salaries.reduce(
    (acc, salary) => {
      acc.total += salary;
      acc.maxSalary = Math.max(acc.maxSalary, salary);
      acc.minSalary = Math.min(acc.minSalary, salary);
      return acc;
    },
    { total: 0, maxSalary: -Infinity, minSalary: Infinity }
  );

  const averageSalary = Math.floor(total / salaries.length);
  const salaryRange = maxSalary - minSalary;

  return { averageSalary, salaryRange };
};

```

```
console.log("Companies before transformation:", companies);

// Transform companies to include averageSalary and salaryRange
const companiesWithStats = companies.map((company) => {
  const { averageSalary, salaryRange } = calculateStats(company.salaries);
  return {
    ...company,
    averageSalary,
    salaryRange,
  };
});

// Sort companies by salary range in descending order
const sortedCompanies = companiesWithStats.sort(
  (a, b) => b.salaryRange - a.salaryRange
);

// Log the sorted companies
console.log(
  "Companies after transformation and sorting by salary range:",
  sortedCompanies
);
```

Aide au DS 2024

Arbre

Voici un équivalent de getElementByTagName

```
1. /**
2.  * Traverse the DOM tree and collect elements by tag name.
3.  * @param {string} tagName - The tag name to match elements.
4.  * @param {Element} element - The root element to start the traversal
    (default is document.body).
5.  * @param {Element[]} result - The array to store matching elements
    (default is an empty array).
6.  * @returns {Element[]} - The array of matching elements.
7.  */
8. const getElementsByTagName = function (tagName, element =
    document.body, result = []) {
9.   if (element.nodeName.toLowerCase() === tagName.toLowerCase()) {
10.    result.push(element);
11.  }
12.
13.  for (let c of element.children) {
14.    getElementsByTagName(tagName, c, result);
15.  }
16.
17.  return result;
18.};
19.
```

```
20. const matchingElements = getElementsByTagName('p');
```

Voici un équivalent de querySelectorAll. La magie du code vient de l'utilisation de la méthode **matches.**

```
1.  const getElementsBySelector = function (selector, element =
    document.body, result = []) {
2.    if (element.matches(selector)) {
3.      result.push(element);
4.    }
5.
6.    for (let c of element.children) {
7.      getElementsBySelector(selector, c, result);
8.    }
9.
10.   return result;
11.};
```

Aide au DS 2024

Pizzas

🍕 PizzaJS

Aide au DS 2024

🥷Glossaire

Trouvez le code pour créer un glossaire

```
[
  "Pepperoni",
  "Mushrooms",
  "Onions",
  "Sausage",
  "Bacon",
  "Extra cheese",
  "Black olives",
...
  "Capers",
  "Eggplant",
  "Feta cheese",
  "Goat cheese",
  "Gorgonzola",
  "Jalapenos",
  "Kalamata olives",
  "Parmesan cheese",
  "Provolone cheese",
..
  "Pesto",
  "Prosciutto",
  "Roasted red peppers",
  "Shrimp",
  "Smoked salmon",
  "Truffle oil",
  "Wild mushrooms"
];
```

```
{
  "a": [
    "Anchovies",
    "Artichokes",
    "Arugula"
  ],
  "b": [
    "Bacon",
    "Black olives",
    "Basil",
    "Beef",
    "Broccoli"
  ],
  "c": [
    "Chicken",
    "Capers",
    "Caramelized onions",
    "Chorizo",
    "Clams",
    "Corn",
    "Crushed red pepper"
  ],
..
  "w": [
    "Wild mushrooms"
  ],
  "z": [
    "Zucchini"
  ]
}
```

🆘Help :

Aide au DS 2024

Le code suivant doit être exécuté dans visualStudio avec nodejs avec la commande > **node glossaire.js**

https://gist.github.com/dupontdenis/57117cc6b23146b73c8ea16c7448a05b