Back to Week 1

JS | Arrays & Objects

LESSON

Learning Goals

After this lesson you will be able to:

- Understand the term `data structure`
- Understand how / why data structures are often `nested`
- Access values from deeply nested structures

What is a `Data Structure` (Recap)

A data structure is *a particular way of organizing data.*

✔ *The better we can organize our data, the better we can represent people, places, objects, items; the world around us, in code.*

For instance, an array of strings would be a good structure to organize a list of students.

```
const students = [
  "Bob",
  "Susy",
  "Ted",
  "Sarah",
  "Bill"
];
```

Explain this code

The *data* is *structured* in the format of a list. To retrieve a particular item, we need to reference the index of the array.

```
console.log(students[0]);
// Bob
```

Explain this code

`Objects` are another way of structuring our data. They are good for *labeling* data and building more complex structures.

```
const bob   = { name: "Bob", age: 17 };
const susy  = { name: "Susy", age: 18 };
const ted   = { name: "Ted", age: 18 };
const sarah = { name: "Sarah", age: 20 };
const bill  = { name: "Bill", age: 19 };
```

Explain this code

We can access specific `values` by referencing `keys`.

```
console.log(bob.name); // <== Bob

console.log(susy.age); // <== 18
```

Explain this code

Occasionally, we're going to have very complex data that needs to be structured differently.

Nested Data Structures

Let's take the student example through a few iterations.

Objects in Arrays

In reality, a better solution for the list of students would be an **array of objects**. Each student is an object and a collection of them forms the array of students.

```
const students = [
  { name: "Bob", age: 17 },
  { name: "Susy", age: 18 },
  { name: "Ted", age: 18 },
  { name: "Sarah", age: 20 },
  { name: "Bill", age: 19 }
];
```

Explain this code

`students[<index>]` is going to be a student object! Let's grab Sarah's name.

```
console.log(students[3].name); // <== Sarah
```

Explain this code

Adding To Arrays

As previously discussed, we can use the `.push()` method to add things to arrays. This applies also to adding objects to arrays.

```
students.push({ name: "Steve", age: 25 });

const bob = { name: "Bob", age: 21 };
students.push(bob);
```

Explain this code

📝 Time to practice

In the *students* array above:

- Get the value of the first student's name
- Get the age of the student named Sarah

Arrays in Arrays

A Simple Example

Sometimes we need to have such a data structure to nest an array inside of an array. This is called a `two-dimensional array`.

Let's take a look at a simple example.

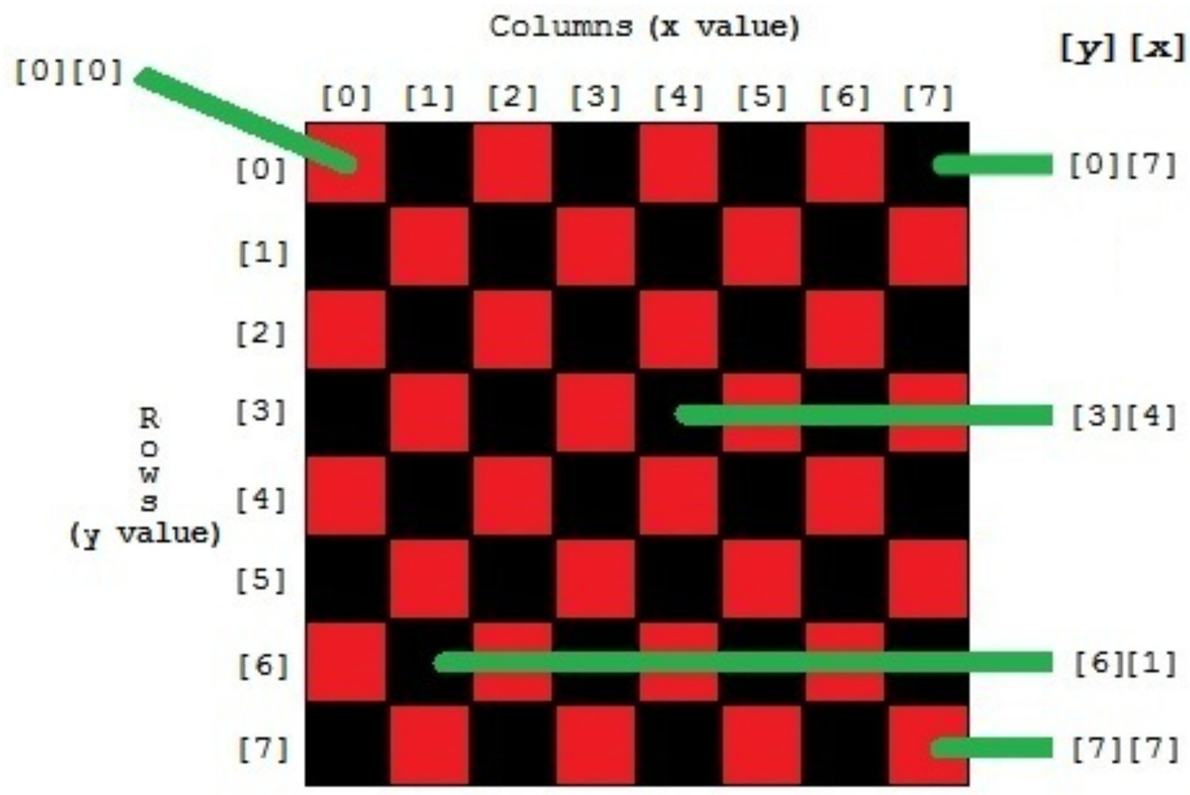
```
const twoD = [
  ["Bob", "Susy", "Ted"],
  ["Lilly", "Sarah", "Bill"],
  ["Thomas", "Barry", "Alex"]
]
```

Explain this code

In this structure, to reference an element, is to reference *an entire array*.

```
console.log(twoD[1]); // <== [ 'Lilly', 'Sarah', 'Bill' ]

console.log(twoD[1][0]); // <== 'Lilly'

console.log(twoD[0][0]); // <== 'Bob'

console.log(twoD[0][3]); // <== undefined

console.log(twoD[3][0]); // TypeError: Cannot read property '0' of undefined
```

Explain this code



A More Complex Example

Let's expand the student example to represent more than one classroom. This would be a list of lists containing students, or an **`array of arrays containing objects`**.

```js
// Names generated by faker: https://github.com/marak/Faker.js/

const classes = [
  [
    { firstName: 'Tomas', lastName: 'Bechtelar', age: 22 },
    { firstName: 'Nico', lastName: 'Schamberger', age: 26 },
    { firstName: 'Ashleigh', lastName: 'Kutch', age: 29 },
    { firstName: 'Lulu', lastName: 'Considine', age: 20 },
    { firstName: 'Garland', lastName: 'Waelchi', age: 21 }
  ],
  [
    { firstName: 'Charlie', lastName: 'Rolfson', age: 23 },
    { firstName: 'Austin', lastName: 'Schowalter', age: 26 },
    { firstName: 'Emie', lastName: 'Franecki', age: 29 },
    { firstName: 'Okey', lastName: 'Runte', age: 18 },
    { firstName: 'Jameson', lastName: 'Jakubowski', age: 22 }
  ],
  [
    { firstName: 'Antwan', lastName: 'Marquardt', age: 22 },
    { firstName: 'Eugenia', lastName: 'Nienow', age: 23 },
    { firstName: 'Keely', lastName: 'Hagenes', age: 29 },
    { firstName: 'Jazmin', lastName: 'Aufderhar', age: 29 },
    { firstName: 'Stanley', lastName: 'Hand', age: 22 }
  ],
  [
    { firstName: 'Vincent', lastName: 'Langworth', age: 20 },
    { firstName: 'Mervin', lastName: 'Blick', age: 28 },
    { firstName: 'Damien', lastName: 'Rohan', age: 28 },
    { firstName: 'Fabian', lastName: 'Kautzer', age: 22 },
    { firstName: 'Lilliana', lastName: 'Lesch', age: 26 }
  ],
  [
    { firstName: 'Antonette', lastName: 'Stokes', age: 25 },
    { firstName: 'Alexandrine', lastName: 'DuBuque', age: 22 },
    { firstName: 'Braeden', lastName: 'Walker', age: 26 },
    { firstName: 'Derick', lastName: 'Weber', age: 22 },
    { firstName: 'Robert', lastName: 'Beatty', age: 30 }
  ]
];
```

Explain this code

Examples

```js
console.log(classes[0]);
// [ { firstName: 'Tomas', lastName: 'Bechtelar', age: 22 },
//   { firstName: 'Nico', lastName: 'Schamberger', age: 26 },
//   { firstName: 'Ashleigh', lastName: 'Kutch', age: 29 },
//   { firstName: 'Lulu', lastName: 'Considine', age: 20 },
//   { firstName: 'Garland', lastName: 'Waelchi', age: 21 }
// ]

console.log(classes[0][2]);
// { firstName: 'Ashleigh', lastName: 'Kutch', age: 29 }

console.log(classes[0][2].firstName);
// 'Ashleigh'
```

Explain this code

📝 Time to practice

From the array of `classes`:

- Retrieve the second “classroom” of students
- Retrieve the first name “Antonette”
- Retrieve the age 18
- Retrieve the last name “Beatty”

Objects inside of Objects

Objects inside of objects can be tricky to deal with. Let’s create a `classRoom` object, which will have a `teacher` in it.

```
const classRoom = {
  teacher:  { firstName: 'Greg', lastName: 'Dach', age: 38 }
};
```

Explain this code

Remember, when we’re accessing a value inside of the `teacher` object, we have to go through the `classRoom` object first.

```
console.log(classRoom.teacher.firstName); // <== 'Greg'

console.log(classRoom.teacher.age); // <== 38
```

Explain this code

We can go as many levels deep as we want:

```
const classRoom = {
  teacher:  {
    firstName: 'Greg',
    lastName: 'Dach',
    age: 38,
    address: {
      street: "3085 Kelton Knolls",
      city: "Aldaside",
      state: "Maryland"
    }
  }
};
```

Explain this code

To get the *city* from the `address` object which is nested in the `teacher` object which is nested in the `classRoom` object:

```
console.log(classroom.teacher.address.city); // <== "Aldaside"
```

Explain this code

📝 Time to practice

Get back the teacher's age from the `classRoom` object.

Beyond

Let's represent the entire school system with `nested objects`. Let's start at the very bottom, and work our way up.

Classroom

A classroom has a teacher and a few students.

```
const classRoom = {
  teacher: { firstName: 'Marcelino', lastName: 'Padberg', age: 25 },
  students: [
    { firstName: 'Aliyah', lastName: 'Schulist', age: 18 },
    { firstName: 'Cleveland', lastName: 'Towne', age: 28 },
    { firstName: 'Jan', lastName: 'Quitzon', age: 18 },
    { firstName: 'Alaina', lastName: 'Runolfsdottir', age: 18 },
    { firstName: 'Gerhard', lastName: 'Bergstrom', age: 23 }
  ]
};

console.log(classRoom.students[2].firstName); // <== 'Jan'

console.log(classRoom.teacher.age); // <== 25
```

Explain this code

School

A school has a name, and many classrooms:

```
const school = {
  name: "Fake School 1",
  classRooms: [
    {
      teacher: { firstName: 'Marcelino', lastName: 'Padberg', age: 25 },
      students: [
        { firstName: 'Aliyah', lastName: 'Schulist', age: 18 },
        { firstName: 'Cleveland', lastName: 'Towne', age: 28 },
        { firstName: 'Jan', lastName: 'Quitzon', age: 18 },
        { firstName: 'Alaina', lastName: 'Runolfsdottir', age: 18 },
        { firstName: 'Gerhard', lastName: 'Bergstrom', age: 23 }
      ]
      },
    {
      teacher: { firstName: 'Edwardo', lastName: 'Schowalter', age: 28 },
      students: [
        { firstName: 'Manley', lastName: 'Doyle', age: 18 },
        { firstName: 'Maximilian', lastName: 'Gleichner', age: 19 },
        { firstName: 'Sid', lastName: 'Rohan', age: 30 },
        { firstName: 'Catalina', lastName: 'Hilpert', age: 27 },
        { firstName: 'Gerald', lastName: 'O\'Keefe', age: 26 }
      ]
    }
  ]
}

console.log(school.name); // <== "Fake School 1"

console.log(school.classRooms[1].students[4].firstName); // <== Gerald
```

Explain this code

School System

A school system has many schools in it and is the final result.

```
const schoolSystem = {
  schools: [
    {
      name: "Fake School 1",
      classRooms: [
        {
          teacher: { firstName: 'Marcelino', lastName: 'Padberg', age: 25 },
          students: [
              { firstName: 'Aliyah', lastName: 'Schulist', age: 18 },
              { firstName: 'Cleveland', lastName: 'Towne', age: 28 },
              { firstName: 'Jan', lastName: 'Quitzon', age: 18 },
              { firstName: 'Alaina', lastName: 'Runolfsdottir', age: 18 },
              { firstName: 'Gerhard', lastName: 'Bergstrom', age: 23 }
          ]
      },
      {
        teacher: { firstName: 'Edwardo', lastName: 'Schowalter', age: 28 },
        students: [
          { firstName: 'Manley', lastName: 'Doyle', age: 18 },
          { firstName: 'Maximilian', lastName: 'Gleichner', age: 19 },
          { firstName: 'Sid', lastName: 'Rohan', age: 30 },
          { firstName: 'Catalina', lastName: 'Hilpert', age: 27 },
          { firstName: 'Gerald', lastName: 'O\'Keefe', age: 26 }
        ]
      }
     ]
    },
    {
     name: "Fake School 2",
     classRooms: [
       {
         teacher: { firstName: 'Lucas', lastName: 'Schroeder', age: 29 },
         students: [
           { firstName: 'Giuseppe', lastName: 'Hegmann', age: 24 },
           { firstName: 'Jennyfer', lastName: 'Hane', age: 19 },
           { firstName: 'Mikayla', lastName: 'Braun', age: 23 },
           { firstName: 'Rickie', lastName: 'White', age: 22 },
           { firstName: 'Rose', lastName: 'Collins', age: 30 }
         ]
       },
       {
         teacher: { firstName: 'Green', lastName: 'Sauer', age: 25 },
         students: [
           { firstName: 'Melany', lastName: 'Welch', age: 25 },
           { firstName: 'Paxton', lastName: 'Corkery', age: 22 },
           { firstName: 'Nellie', lastName: 'Hauck', age: 26 },
           { firstName: 'Eunice', lastName: 'Hirthe', age: 26 },
           { firstName: 'Aylin', lastName: 'Barrows', age: 26 }
          ]
       }
     ]
       },
       {
    name: "Fake School 3",
    classRooms: [
      {
```

```
            teacher: { firstName: 'Nikko', lastName: 'Crist', age: 42 },
            students: [
              { firstName: 'Christop', lastName: 'Hahn', age: 26 },
              { firstName: 'Newell', lastName: 'Kemmer', age: 27 },
              { firstName: 'Katheryn', lastName: 'Heller', age: 26 },
              { firstName: 'Saul', lastName: 'Heathcote', age: 20 },
              { firstName: 'Maudie', lastName: 'Haley', age: 30 }
            ]
          },
          {
            teacher: { firstName: 'Nathanael', lastName: 'Hansson', age: 50 },
            students: [
              { firstName: 'Jensen', lastName: 'Reichel', age: 21 },
              { firstName: 'Lois', lastName: 'Kulas', age: 28 },
              { firstName: 'Caterina', lastName: 'Wolff', age: 28 },
              { firstName: 'Dahlia', lastName: 'Collier', age: 24 },
              { firstName: 'Linwood', lastName: 'Langosh', age: 26 }
            ]
          }
        ]
      }
    ]
};

console.log(schoolSystem.schools[1].name); // <== Fake School 2

console.log(schoolSystem.schools[1]);
//{ name: 'Fake School 2',
//  classRooms:
//    [
//        { teacher: [Object], students: [Array] },
//        { teacher: [Object], students: [Array] }
//    ]
// }

console.log(schoolSystem.schools[1].classRooms[0]);

// {
//      teacher: { firstName: 'Lucas', lastName: 'Schroeder', age: 29 },
//      students: [
//              { firstName: 'Giuseppe', lastName: 'Hegmann', age: 24 },
//              { firstName: 'Jennyfer', lastName: 'Hane', age: 19 },
//              { firstName: 'Mikayla', lastName: 'Braun', age: 23 },
//              { firstName: 'Rickie', lastName: 'White', age: 22 },
//              { firstName: 'Rose', lastName: 'Collins', age: 30 }
//      ]
// }


console.log(schoolSystem.schools[1].classRooms[0].students[1]);

// <== { firstName: 'Jennyfer', lastName: 'Hane', age: 19 }


console.log(schoolSystem.schools[1].classRooms[0].students[1].firstName); // <== Jennyfer
```

Explain this code

📝 **Time to practice**

- Add a student with the name of Lucille D. Lozano to Fake School 2, in the first classroom.
- Retrieve the “Fake School 3” object
- Retrieve the teacher with the first name of “Nathanael”
- Retrieve the student with the first name of “Saul”

Real World Applications

Databases

Data Structures, specifically arrays and hashes in JavaScript, are incredibly important. Later on when we get to databases, most of our database objects are going to be in the form of nested objects and arrays.

Web APIs

When we’re trying to get information in the future from APIs (web services that give us back information), it’s going to be formatted much like our JavaScript objects.

For instance, let’s take a look at the **PunkAPI**, that brings us a random beer, with its ingredients and other information.

In the link above, PunkAPI gives us a random beer from its database. You can notice that the element that the API send us is a nested object with a lot of info.

Summary

In this lesson, we learned about all different varieties of JavaScript structures and how to access the data inside of them. We covered arrays in arrays, objects in arrays, and many different combinations of the two.

We also took a look at how all that might look in the real world applications.

All in all, data structures in programming are **SUPER** important. There are dozens of other structures in different languages, but in JavaScript the array and object are core.

Data Structures let us represent the world around us more effectively, and you will be a more efficient programmer if you develop the skill to structure well information needed to develop your apps successfully.

Extra Resources

- Back to Basics: JavaScript Objects
- MDN Array
- MDN Object

Mark as completed

PREVIOUS

JS | Data Types in JavaScript - Objects

NEXT

LAB | JavaScript Clue