Compare two array or object with JavaScript

Posted on November 9, 2019 | by Prashant Yadav

Posted in Arrays, Interview, Javascript

Practically prepare for your JavaScript interview

- JavaScript Revision
- JavaScript-Concept Based Problems
- Data Structures
- Algorithms
- Machine Coding
- Web Fundamentals

Advertisements

Learn how to compare array or object with javascript.

There are no specific built-in methods available to us to compare two different arrays or objects in javascript. So we will see how we can create our own custom function which will compare two different objects.

Basic approach to compare array and object in javascript.

The most basic approach is to convert the whole array or the object to a string then compare if those strings are equal or not.

To convert an array or object we will be using JSON.stringify().

```
let a = [1, 2, 3, 4, 5];
let b = [1, 2, 3, 4, 5];

// "[1, 2, 3, 4, 5]"==="[1, 2, 3, 4, 5]"
console.log(JSON.stringify(a) === JSON.stringify(b));

//true
```

This may seem to be working fine, but this approach fails when we change the order of the elements.

Copy

```
let a = [1, 2, 3, 4, 5];
let b = [1, 2, 4, 5, 3];

// "[1, 2, 3, 4, 5]"==="[1, 2, 4, 5, 3]"
console.log(JSON.stringify(a) === JSON.stringify(b));

//false
```

Technically both the arrays are having the same elements and are equal in length but it fails in the comparison. The same happens with the objects.

Copy

```
let a = {a: 1, b: 2, c: 3};
let b = {b: 2, a: 1, c: 3};

//"{'a':1,'b':2,'c':3}" "{'b':2,'a':1,'c':3}"
console.log(JSON.stringify(a) === JSON.stringify(b));

//false
```

Comparing only arrays (Single and Multi dimensional).

We will create our own custom function which will compare only two different arrays.

In this, we will check

- If both the inputs are array or not.
- If the current element is array (muti-dimensional) then recursively check its elements with the corresponding element.
- Else compare both the elements and return the result.

Copy

```
let compare = (arr1,arr2) => {
   //If not array then return false
   if(!arr1 || !arr2) return false;

   if(arr1.length !== arr2.length) return false;

   let result;

    for(let i = 0; i < arr1.length; i++){
      if(Array.isArray(arr1[i]) && Array.isArray(arr2[i])){
          result = compare(arr1[i], arr2[i]);
        }else if(arr1[i] === arr2[i]){
          result = true;
        }else{
          result = false;
        }

      if(!result){
        break;
      }
    }

   return result
}

console.log(compare([1,2,3],[1,2,3]));
// true

console.log(compare([1,2],[1,2,3]));
//false

console.log(compare([[1, 2], [3, 4]],[[1, 2],[3, 4, 6]]));
//false

console.log(compare([[1, 2], [3, 4]],[[1, 2], [3, 4]]));
//true

console.log(compare([1, 2, [3, 4, 5]],[1, 2, [3, 4, 5]]));
//true

console.log(compare([], [1, 2, [3, 4, 5]]));
//false
```

This method works great if you only have arrays and nested arrays. If it will contain an object or function then it will fail.

Method 3:- Compare array or object with javascript.

This is a much more robust way to check if two different arrays or objects are equal or not.

With this method, we will be comparing more complex arrays. Something like this where an array can have any value possible.

Copy

```
let arr = [1, 'Prashant', 3, {
        a: 42,
        b: 'another thing',
        c: function () {
                console.log('running!');
        }
}, 5];
```

To perfectly compare an array or object we need to check multiple things.

- If they are same type (Object or Array).
- They have the same number of elements.
- Each element is equal to the other element which we are comparing with.
- They are of same type (array, object, string, number, function).
- They have same value.

If the element itself is an array or object then we need to compare all its items with the items of the corresponding array or object.

First, we will create a helper function that will be taking the two different inputs which need to be compared.

Copy

```
let compare = (current, other) => {
   //Comparison will be done here.
}
```

We will be performing different comparison tests in our function and if any of them fail then we will return `false`.

Some basic tests

First check if the inputs are either array or object.

We are not comparing anything other than an array or object. So test it to make sure that the inputs we get are either array or object.

Copy

```
let compare = (current, other) => {
  // Get the inputs  type
  let currentType = Object.prototype.toString.call(current);

  // Get the other type
  let otherType = Object.prototype.toString.call(other);

  // If items are not an object or array, return false
  if (['[object Array]', '[object Object]'].indexOf(currentType) < 0 || ['[object
Array]', '[object Object]'].indexOf(otherType) < 0) return false;

  // Other comparisons will happen here

  //If all tests are passed then
  return true
}
```

Check if both values are of same type.

If one value is of type array and another value is of type object then they are not equal so eliminate them.

Advertisements

Copy

```
let compare = (current, other) => {
  // Get the inputs  type
  let currentType = Object.prototype.toString.call(current);

  // Get the other type
  let otherType = Object.prototype.toString.call(other);

  // If items are not an object or array, return false
  if (['[object Array]', '[object Object]'].indexOf(currentType) < 0 || ['[object
Array]', '[object Object]'].indexOf(otherType) < 0) return false;

  // If the two inputs are not the same type, return false
  if (currentType !== otherType) return false;

  // Other comparisons will happen here

  //If all tests are passed then
  return true
}
```

Check if both values are of equal size.

Compare the length of the both the inputs.

Copy

```
let compare = (current, other) => {
  // Get the inputs  type
  let currentType = Object.prototype.toString.call(current);

  // Get the other type
  let otherType = Object.prototype.toString.call(other);

  // If items are not an object or array, return false
  if (['[object Array]', '[object Object]'].indexOf(currentType) < 0 || ['[object
Array]', '[object Object]'].indexOf(otherType) < 0) return false;

  // If the two inputs are not the same type, return false
  if (currentType !== otherType) return false;

  // Compare the length of the length of the two items
  let currentLen = currentType === '[object Array]' ? current.length :
Object.keys(current).length;
  let otherLen = otherType === '[object Array]' ? other.length :
Object.keys(other).length;
  if (currentLen !== otherLen) return false;

  // Other comparisons will happen here

  //If all tests are passed then
  return true
}
```

Check if elements of both input match.

Check if the elements inside both the elements are same or not.

Copy

```
let compare = (current, other) => {
  // Get the inputs  type
  let currentType = Object.prototype.toString.call(current);

  // Get the other type
  let otherType = Object.prototype.toString.call(other);

  // If items are not an object or array, return false
  if (['[object Array]', '[object Object]'].indexOf(currentType) < 0 || ['[object
Array]', '[object Object]'].indexOf(otherType) < 0) return false;

  // If the two inputs are not the same type, return false
  if (currentType !== otherType) return false;

  // Compare the length of the length of the two items
  let currentLen = currentType === '[object Array]' ? current.length :
Object.keys(current).length;
  let otherLen = otherType === '[object Array]' ? other.length :
Object.keys(other).length;
  if (currentLen !== otherLen) return false;

  // Compare properties
   if (currentType === '[object Array]') {
        for (var i = 0; i < currentLen; i++) {
            // Compare the item
        }
   } else {
        for (var key in current) {
            if (current.hasOwnProperty(key)) {
                // Compare the item
            }
        }
   }

  // Other comparisons will happen here

  //If all tests are passed then
  return true
}
```

For comparing values of both the inputs we will be creating a helper function. That can be used for both the comparison and will check other possible test cases.

Comparison function to check values of array or object

Again check the type of the element first.

Copy

```
let equal = (item1, item2) => {
  // Get the object type
  let itemType = Object.prototype.toString.call(item1);

  // If an object or array, compare recursively
  if (['[object Array]', '[object Object]'].indexOf(itemType) >= 0) {
      if (!compare(item1, item2)) return false;
  }
}
```

A simple comparison

Check if both inputs are of same type or not.

Copy

```
let equal = (item1, item2) => {
  // Get the object type
  let itemType = Object.prototype.toString.call(item1);

  // If an object or array, compare recursively
  if (['[object Array]', '[object Object]'].indexOf(itemType) >= 0) {
      if (!compare(item1, item2)) return false;
  }else{
     // If the two items are not the same type, return false
     if (itemType !== Object.prototype.toString.call(item2)) return false;
  }
}
```

Check if elements are equal.

We can simply perform the strict equality check `===` to check if both the elements are equal or not. However if the element is a function then we will need to convert it to a string and then compare it.

We will use `toString()` method to convert the function to a string.

Copy

```
let equal = (item1, item2) => {
  // Get the object type
  let itemType = Object.prototype.toString.call(item1);

  // If an object or array, compare recursively
  if (['[object Array]', '[object Object]'].indexOf(itemType) >= 0) {
      if (!compare(item1, item2)) return false;
  }else{
     // If the two items are not the same type, return false
     if (itemType !== Object.prototype.toString.call(item2)) return false;

     // If it's a function, convert to a string and compare
     // Otherwise, just compare
     if (itemType === '[object Function]') {
        if (item1.toString() !== item2.toString()) return false;
     } else {
        if (item1 !== item2) return false;
     }
  }
};
```

We can use this equality checker function in our compare function and get the result.

Complete function to compare two arrays or objects.

Copy

```
let compare = (current, other) => {
  // Get the inputs  type
  let currentType = Object.prototype.toString.call(current);

  // Get the other type
  let otherType = Object.prototype.toString.call(other);

  // If items are not an object or array, return false
  if (['[object Array]', '[object Object]'].indexOf(currentType) < 0 || ['[object
Array]', '[object Object]'].indexOf(otherType) < 0) return false;

  // If the two inputs are not the same type, return false
  if (currentType !== otherType) return false;

  // Compare the length of the length of the two items
  let currentLen = currentType === '[object Array]' ? current.length :
Object.keys(current).length;
  let otherLen = otherType === '[object Array]' ? other.length :
Object.keys(other).length;
  if (currentLen !== otherLen) return false;

  //Helper function to check the equality
  let equal = (item1, item2) => {
      // Get the object type
      let itemType = Object.prototype.toString.call(item1);

      // If an object or array, compare recursively
      if (['[object Array]', '[object Object]'].indexOf(itemType) >= 0) {
          if (!compare(item1, item2)) return false;
      }else{
          // If the two items are not the same type, return false
          if (itemType !== Object.prototype.toString.call(item2)) return false;

          // If it's a function, convert to a string and compare
          // Otherwise, just compare
          if (itemType === '[object Function]') {
                if (item1.toString() !== item2.toString()) return false;
          } else {
                if (item1 !== item2) return false;
          }
      }
   };

  // Compare properties
   if (currentType === '[object Array]') {
        for (var i = 0; i < currentLen; i++) {
            // Compare the item
            if (equal(current[i], other[i]) === false) return false;
        }
   } else {
        for (var key in current) {
            if (current.hasOwnProperty(key)) {
                // Compare the item
                if (equal(current[key], other[key]) === false) return false;
            }
        }
   }

  //If all tests are passed then
  return true
}
```

Copy

```
let arr1 = [1, 2, 3, 4, 5];
let arr2 = [1, 3, 2, 4, 5];
console.log(compare(arr1, arr2));
// returns false

let arrObj1 = [1, 2, {
        a: 1,
        b: 2,
        c: 3,
  d: function(){
    console.log("abcd");
  }
}, 4, 5];
let arrObj2 = [1, 2, {
        c: 3,
        b: 2,
        a: 1,
  d: function(){
    console.log("abcd");
  }
}, 4, 5];
console.log(compare(arrObj1, arrObj2));
// returns true

let arr4 = [[1, 2], [3, 4, 5]];
let arr3 = [[1, 2], [3, 4, 5]];
console.log(compare(arr4, arr3));
// returns true
```

If you want you can use other libraries as well like _.isEqual(obj1, obj2) of lodash.

Recommended Posts:

- useOnClickOutside() hook in React
- Fetch request and response interceptor
- Currying – part 4
- Highlight the words in the string
- Find numbers that appear twice in an array.
- Create a toggle function in JavaScript
- Create a basic implementation of a streams API
- Convert string to array in javascript
- 3 different ways to hide DOM element using Javascript
- Implement browser history in JavaScript

Prev Next