Using an array through a switch() statement in Javascript

Asked 6 years, 2 months ago Active 1 year, 1 month ago Viewed 26k times

12

3

I'm trying to develop a simplified poker game through Javascript. I've listed all possible card combinations a given player might have in its hand ordered by its value, like this:

```
switch(sortedHand)
{
 //Pair
     case [1,1,4,3,2]: sortedHand.push(1,"Pair"); break;
     case [1,1,5,3,2]: sortedHand.push(2,"Pair"); break;
     case [1,1,5,4,2]: sortedHand.push(3,"Pair"); break;
     case [1,1,5,4,3]: sortedHand.push(4,"Pair"); break;
     case [1,1,6,3,2]: sortedHand.push(5,"Pair"); break;
     case [1,1,6,4,2]: sortedHand.push(6,"Pair"); break;
     case [1,1,6,4,3]: sortedHand.push(7,"Pair"); break;
     case [1,1,6,5,2]: sortedHand.push(8,"Pair"); break;
     case [1,1,6,5,3]: sortedHand.push(9,"Pair"); break;
     case [1,1,6,5,4]: sortedHand.push(10,"Pair"); break;
```

Even though the "sortedHand" array stores values succesfully (as I've seen through console.log), the switch() statement always returns the default case, and everyone gets an straight flush. I fear this is a matter of the literal approach I've used to declare possible array values to be compared with the whole of "sortedHand", but I don't know any better. Is it even possible to use switch() in such a manner?

javascript arrays switch-statement

asked Jul 23 '13 at 18:26

Guilherme de Abreu

69 1 1 5

case (Array) will always be false... – dandavis Jul 23 '13 at 18:31

6 Answers

22

You can try `switch` ing on a textual representation of the array.

```
switch(sortedHand.join(' '))
{
    //Pair
    case '1 1 4 3 2': sortedHand.push(1,"Pair"); break;
    case '1 1 5 3 2': sortedHand.push(2,"Pair"); break;
    case '1 1 5 4 2': sortedHand.push(3,"Pair"); break;
    case '1 1 5 4 3': sortedHand.push(4,"Pair"); break;
    // etc.
}
```

As an alternative to specifying every case directly, perhaps build a function dispatch table using an object and get rid of the switch entirely.

```
var dispatch = {};

// Build the table however you'd like, for your application
for (var i = 0; i < 10; i++) {
    (function(i) {
        var hand = ...; // Add your hand logic here
        dispatch[hand] = function() { sortedHand.push(i, "Pair"); };
    })(i);
}

// Execute your routine
dispatch[sortedHand.join(' ')]();
```

edited Jul 23 '13 at 18:39

answered Jul 23 '13 at 18:31

voithos

48.1k 9 74 104

Brilliant. It worked perfectly. – Guilherme de Abreu Jul 23 '13 at 19:03

> the switch() statement always returns the default case

6

That's because the comparison doesn't check the array contents, but the array object itself. Objects are considered equal by their identity, so nothing will be equal to an object instantiated by a literal.

> Is it even possible to use switch() in such a manner?

Yes, one can use objects in `switch` statements, but you would have to use references in the cases. Not applicable to your problem.

In your case, I'd suggest a stringification:

```
switch(sortedHand.join())
{
 //Pair
     case "1,1,4,3,2": sortedHand.push(1,"Pair"); break;
     case "1,1,5,3,2": sortedHand.push(2,"Pair"); break;
     case "1,1,5,4,2": sortedHand.push(3,"Pair"); break;
     case "1,1,5,4,3": sortedHand.push(4,"Pair"); break;
     case "1,1,6,3,2": sortedHand.push(5,"Pair"); break;
     case "1,1,6,4,2": sortedHand.push(6,"Pair"); break;
     case "1,1,6,4,3": sortedHand.push(7,"Pair"); break;
     case "1,1,6,5,2": sortedHand.push(8,"Pair"); break;
     case "1,1,6,5,3": sortedHand.push(9,"Pair"); break;
     case "1,1,6,5,4": sortedHand.push(10,"Pair"); break;
```

but I guess there's an even better, arithmetic solution to detect the patterns you're after. That would be shorter and faster, but I'm not sure what exactly this snippet is supposed to do.

answered Jul 23 '13 at 18:32

Bergi
409k 69 647 975

1

That will not quite work as you have it, but you can use `sortedHand.join(',')` and compare it with `[1,1,1,2,5].join(',')` which will compare the two arrays and should be true if their contents were the exact same (Be careful with `number` s typed as `strings` !) To be fair, though, I can't imagine why you would design your logic like that. Even a simple card game has hundreds of thousands of possible hands. You might do better using `underscore.js` 's collection managing functions as it will be simpler, and just a better practice.


13.3k 16 48 69

1

a faster, potentially reusable, and more flexible way of doing it is to use an object instead of case:

```
var ok= {
    '1 1 4 3 2':1,
    '1 1 5 3 2':2,
    '1 1 5 4 2':3,
    '1 1 5 4 3':4
}[ sortedHand.join(' ') ] ;
if(ok){ sortedHand.push( ok ,"Pair"); }
```

objects work great when one output is hinged on one input. if you need to do five things in each case, then you have to use case, but if you just need X to turn into Y, (a 1:1), Look Up Tables in the shape of Objects are ideal.

i imagine a RegExp can work here, i used them on a connect4 game to identify 4 in a row, but the above logic table should work as well or better than what you describe.

answered Jul 23 '13 at 18:34


dandavis
12.8k 3 28 32

1

There are 1274 possible combinations of 5 cards in a regular deck. Listing them all out in a switch statement is completely ridiculous. Why not just have a function count any duplicates to check for 2,3,4-of-a-kinds and then check for straights? (Your array doesn't show suit so I'm assuming you are leaving it out).

But if you really want to do it that way, you could use a string. Strings work with switches, and you can even use them like arrays. e.g. "123"[0] == '1'. You can change them back and forth user functions like parseInt.

answered Jul 23 '13 at 18:46


milestyle
866 7 14

You are right. I left them out. The deck is composed by six ranks only, there are 250 posibilities in total. I resorted for the switch method because I

0

Since no one suggested this, use a for loop and count the number of cards with exactly the given value. Having such a function you can call 'cardCount = count(sortedHand, cardNumber)'. And of cause looping through all possible card-numbers will give you the hands.

Since a given player can only have 1x2, 2x2, 1x3, 1x3+1x2, 1x4 or straights/streets, you can return an array of all hits being arrays/objects stating the count and the cardNumber involved. So [{2, 5}, {3, 6}] for a full house.

answered Aug 11 '18 at 21:49

Martin Kersten
3,098 3 28 51