<https://medium.com/@abustamam/for-loops-vs-foreach-in-javascript-7a977278a39e>

<http://qnimate.com/foreach-vs-for-of-vs-for-in-in-javascript/>

The for loop consistently came out over 40 times faster than the foreach.

***For loop***

 if you need to stop or break out of the loop before every element is visited, forEach() is not the right function. Here use for loop

***Array.forEach method***

foreach is an method that is available only in Array objects. It allows you to iterate through elements of an array. When invoked it takes a callback function and invokes the callback once for every array element. The callback can access both index and value of the array elements. foreach is available only for looping arrays.

**Improved Readability with forEach()**

students.forEach(function(student,index) {  
 console.log(student);  
});

*“But what if I want the iteration number too?”*

The callback function actually accepts three arguments:

* element value - student
* element index - index
* array being traversed (I have never used this)

you’re developing something that is very data intensive, and the extra performance of the for loop will make a difference, go with the for loop.

Otherwise, if you find forEach() to be more readable and aren’t iterating over huge data sets, use forEach()…

**var** foo **=** {

foo: 1,

bar: 2,

baz: 3

};

Procedural style:

**var** bar **=** {};

Object.keys(foo).forEach(**function** (prop) {

bar[prop] **=** **null**;

})

***For …in***

for in is used to loop through properties of an object. It can be any object. for in allows you to access the keys of the object but doesn’t provide reference to the values. In JavaScript object properties themselves have internal properties. One of the internal properties is [[Enumerable]]. for in will only walkthrough a property if it has [[Enumerbale]] set to true. It not used to iterate elements of an collection rather used to iterate properties of objects.

for...in*should not be used to iterate over an*[Array](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array)*where the index order is important.*

var arr = [  
 'Mike',   
 'Steven'  
];  
for (var i in arr) {  
 console.log(i);  
}

**wrong**

for...in actually iterates through the *enumerable properties* of an object/array. That means the *properties* that exist in the array, so the X that goes in object[X] or array[X]. For arrays, that’s the index. For objects, they’re the keys (*not* the values).

for loop that also works with objects. However, the problem with using it to work with objects is that if the object inherits anything from another object, or has methods on its prototypes, those will be included as well. Many times, you don’t want those to be included, so you’d have to do hasOwnProperty(X).

If you want to iterate through objects, I’d suggest using Object.keys().forEach(function(key) { ... });

***For ….of***

for...of iterates on the *other* side of iterables: so instead of iterating through the *keys* it iterates through the *values*.

var arr = [  
 'Mike',   
 'Steven'  
];  
for (var person of arr) {  
 console.log(person);  
}

Is there a problem with this? If you only care about the value, then for...ofwill work perfectly for you. However, if you also care about the *key* or *index*, then for...of will not give that to you. You *could* use for...in and simply get the value by using a “getter” (that is, arr[i]), but then you’re back in square one with for loops.

So, to summarize:

* Arrays: use for...of if you don’t care about index value. I like to use forEach for all my iterations though — no refactoring needed if I ever decide I do need the index value.
* Objects: use for...of if you don’t care about key value; Object.keys() if you do (and use obj[key] to get the value).
* Everything: use lodash if it is a possibility. There’s a reason it’s the most downloaded and depended-on library on NPM.

***Lodash***

1. Use [lodash](http://lodash.com/" \t "_blank). I use lodash in almost all my projects. In case you don’t know what lodash is, it’s basically a utility library in JavaScript that lets you do a lot of common things. lodash has a lot of helpful iteration methods, such as forEach and map. These also work on objects, which is really neat. Usage might look like this:

\_.forEach(arr, function(element, i) {  
 console.log(element);  
});