Swipe →

VERSION COMPARISON IN JAVASCRIPT

IDEAS2IT

Why is version number important?

Version numbering is especially important in corporate settings, where products and services may rely upon features specific to a certain version of the software.

Syntax of version number

Given a version number **MAJOR.MINOR.PATCH**, and it indicates:

- **MAJOR** version is when you make incompatible API changes.
- **MINOR** version is when you add functionality in a backwards compatible manner.
- **PATCH** version is when you make backwards compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to this format.

Can we compare versions with String comparison?

NO!! Because string comparison is based on the Abstract Relational Comparison Algorithm in ES5.

```
“6.9” < “6.12”  // false
```

It keeps moving on to the next characters of both strings sequentially till the end of string, comparing the unicode value of each characters.

It then exits after comparing different unicode values.

```
'9' -> 57
'1' -> 49
```

In our case, the unicode value of the third character from first string (9) is greater. Hence we consider first string (6.9) to be greater.

Hence the output is **FALSE** which in our case will be **WRONG!!**

Version Comparison in Javascript

```
versionCompare = function (left, right) {
 if (typeof left != 'string' && typeof right !=
'string') {
   return false;
 }
 var a = left.split('.'),
   b = right.split('.'),
   len = Math.max(a.length, b.length);

 for (i = 0; i < len; i++) {
   if (
     (a[i] && !b[i] && parseInt(a[i]) > 0) ||
     parseInt(a[i]) > parseInt(b[i])
   ) {
     return 1;
   } else if (
     (b[i] && !a[i] && parseInt(b[i]) > 0) ||
     parseInt(a[i]) < parseInt(b[i])
   ) {
     return -1;
   }
 }
 return 0;
};
```

Algorithm

- First, Check whether both values passed to the function is string or not.
- Second, Split both the input strings by a ‘DOT’ to get multiple arrays.
- Third, Compare the value from first index of first array with value from first index of second array and so on till end of the largest array.
- As a result, Function will return one of the following response

```
-1      =  left is OLDER
 0      =  they are equal
 1      =  left is NEWER = right is OLDER
false   =  if one of input versions is invalid
```

Example

```
versionCompare('1.1', '1.2') => -1
versionCompare('1.1', '1.1') =>  0
versionCompare('1.2', '1.1') =>  1
versionCompare('2.23.3', '2.22.3') =>  1
```

IDEAS2IT