String Match and Search Methods

Strings offer `match` and `search` methods which can be used with regular expressions. `match` method will tell you whether the regular expression matches the string and also return an array containing the first match that occurs:

```
var source = "The kittens have mittens.";
var vowelMatches = /[aeiou]/;
console.log(source.match(vowelMatches)); //["e"]
```

The `search` method will tell you if a regular expression matches, and return the index on the string where the first match occurs:

```
console.log(source.search(vowelMatches)); //2
```

Both of these methods are very convenient and knowing what each one returns and how you can use it, can come in handy depending on what you're trying to accomplish with your regular expressions.

Regular Expression Exec Method

In addition to using regular expressions with the string methods, you can also use regular expression's own `exec` method. It returns an array, and that array contains the next index to match for each time the exec call is issued against that string. The regular expression also maintains a last index property containing the index of that last match.

```
var input = "The kittens have mittens";
var vowelMatch = /[aeiou]/g;

var match = vowelMatch.exec(input);
console.log(match); //["e"]
```

"e" is the first occurrence of this pattern inside of our source string, right there at the third position:

```
console.log(vowelMatch.lastIndex); //3
```

Let's copy and paste these two lines, and run them again:

```
var match = vowelMatch.exec(input);
console.log(match); //["i"]
console.log(vowelMatch.lastIndex); //6
```

You see that the second time the exec method run against the same strings, it keeps track of where it last found a match, and finds the next match. In this case, the next match is the vowel "i", which is at the 6th position.

So since the exec method keeps track of its position, and keeps on updating on a regular basis, it's a perfect tool for using within a loop.

Finding All Occurrences

So, using the exec method within a loop, lets you go through every occurrence of the pattern match inside of the string to get that information.

```
var input = "The kittens have mittens";
var vowelMatch = /[aeiou]/g;
var match;
while (match = vowelMatch.exec(input)) {
  console.log([match[0], "was found at", match.index].join(" "));
}
//"e was found at 2"
//"i was found at 5"
//"e was found at 8"
//"a was found at 13"
//"e was found at 15"
//"i was found at 18"
//"e was found at 21"
```

Source Property

Another property that the regular expression maintains is the `source` property, and that keeps track of the string that represents the regular expression that's actually being executed, the pattern that's being matched.

```
var vowelMatch = /[aeiou]/;
console.log(vowelMatch.source); //"[aeiou]"

var globalVowelMatch = /[aeiou]/g;
console.log(globalVowelMatch.source); //"[aeiou]"
```

One thing that's interesting to note, is that the global flag, the modifier outside of the slashes, doesn't get captured as part of that source property. In fact, JavaScript right now doesn't have a standard property that does capture that flag, but there is a way that you can figure it out.It's not exactly the most reliable way, but it's good to know about in a pinch.

The ‘flags’ Property

The `flags` property for regular expressions isn’t universally supported yet, but still let’s check it out:

```
console.log(globalVowelMatch.flags); //undefined or "g" (browser dependent)
console.log(globalVowelMatch.toString()); //"/[aeiou]/g"
console.log(globalVowelMatch.toString().match(/[gimuy]*$/)[0]); //"g"
```