Spread Operators and Rest Parameters

Spread Operator ...

Old way:

```
var myArray = [5, 10, 50];
Math.max(myArray);    // Error: NaN
Math.max.apply(Math, myArray);    // 50
```

The Math.max() method doesn't support arrays; it accepts only numbers. When an array is passed to the Math.max() function, it throws an error. But when the apply() method is used, the array is sent as individual numbers, so the Math.max() method can handle it.

Spread Operator ...

Fortunately, with the introduction of the spread operator in ECMAScript 6, we no longer need to use the `apply()` method. With the spread operator, we can easily expand an expression into multiple arguments:

New way (ES6):

```
var myArray = [5, 10, 50];
Math.max(...myArray);    // 50
```

Here, the spread operator expands `myArray` to create individual values for the function.

Spread Operator ...

it can be used multiple times and can be mixed with other arguments in a function call:

```
function myFunction() {
  for(var i in arguments){
    console.log(arguments[i]);
  }
}
var params = [10, 15];
myFunction(5, ...params, 20, ...[25]);    // 5 10 15 20 25
```

Another advantage of the spread operator is that it can easily be used with constructors:

```
new Date(...[2016, 5, 6]);    // Mon Jun 06 2016 00:00:00 GMT-0700 (Pacific Daylight Time)
```

Rest Parameter

The rest parameter has the same syntax as the spread operator, but instead of expanding an array into parameters, it collects parameters and turns them into an array.

```
function myFunction(...options) {
    return options;
}
myFunction('a', 'b', 'c');      // ["a", "b", "c"]
```

Rest Parameter

A rest parameter is particularly useful when creating a variadic function (a function that accepts a variable number of arguments). Having the benefit of being arrays, rest parameters can readily replace the `arguments` object

Old way:

```
function checkSubstrings(string) {
  for (var i = 1; i < arguments.length; i++) {
    if (string.indexOf(arguments[i]) === -1) {
      return false;
    }
  }
  return true;
}
checkSubstrings('this is a string', 'is', 'this');   // true
```

Rest Parameter

That last function checks whether a string contains a number of substrings. The first problem with this function is that we have to look inside the `function`'s body to see that it takes multiple arguments. The second problem is that the iteration must start from `1` instead of `0`, because `arguments[0]` points to the first argument. If we later decide to add another parameter before or after the string, we might forget to update the loop. With the rest parameters, we easily avoid these problems:

New way (ES6):

```
function checkSubstrings(string, ...keys) {
  for (var key of keys) {
    if (string.indexOf(key) === -1) {
      return false;
    }
  }
  return true;
}
checkSubstrings('this is a string', 'is', 'this');   // true
```