Javascript shorthand

JavaScript shorthand không chỉ tăng tốc quá trình compile mà còn làm cho các câu lệnh ngắn hơn, do đó dẫn đến việc load trang nhanh hơn. Shorthand về bản chất thì nó vẫn giống cách viết longhand, chỉ là chúng là viết tắt của cùng một thứ trong một định dạng nhỏ gọn hơn. Chúng là một trong những kỹ thuật tối ưu hóa code đơn giản nhất.

Có nhiều cách viết tắt JavaScript, tuy nhiên lại không có bất kỳ 1 hướng dẫn chính thức nào, nó hoàn toàn được định nghĩa bởi các lập trình viên có kinh nghiệm và chia sẻ cho cộng đồng. Dưới đây là một số cú pháp shorthand mà mình thường dùng cũng như tham khảo ở các blog công nghệ nổi tiếng.

1. Decimal numbers

Nếu bạn thường xuyên làm việc với số thập phân lớn thì cú pháp dưới đây rất tiện dụng, vì bạn không cần phải gõ tất cả các số 0, chỉ cần thay thế chúng bằng ký hiệu `e`. Chẳng hạn, `1e8` có nghĩa là thêm tám số không sau số 1, nó bằng 100.000.000.

Số sau chữ cái `e` cho biết số 0 xuất hiện sau chữ số trước `e`. Ví dụ, `16e4` là 160000.

```
/* Shorthand */
var myVar = 1e8;

/* Longhand */
var myVar = 100000000;
```

2. Tăng và giảm

Cú pháp tăng dần được tạo thành từ hai dấu `+`, điều đó có nghĩa là giá trị của một biến sẽ được tăng thêm một. Tương tự, cú pháp giảm dần bao gồm hai dấu `-`, và nó có nghĩa là biến sẽ được giảm đi một.

Hai cú pháp này chỉ có thể được sử dụng với kiểu dữ liệu là `number` trong Javascript. Chúng có vai trò không thể thiếu trong các vòng lặp, trường hợp sử dụng thường xuyên nhất của chúng là vòng lặp `for`.

```
/* Shorthand */
i++;
j--;

/* Longhand */
i=i+1;
j=j-1;
```

3. Cộng, trừ, nhân, chia

Có một cú pháp cho một trong bốn phép toán cơ bản: cộng, trừ, nhân và chia. Chúng hoạt động tương tự như các toán tử tăng và giảm, chỉ khác một chút là bạn có thể thay đổi giá trị của một biến theo bất kỳ số nào (không chỉ bằng 1).

Trong ví dụ dưới đây, biến `i` được tăng 5, `j` giảm đi 3, `k` được nhân với 10 và `l` chia cho 2.

```
/* Shorthand */
i+=5;
j-=3;
k*=10;
l/=2;

/* Longhand */
i=i+5;
j=j-3;
k=k*10;
l=l/2;
```

4. Xác định vị trí của ký tự trong string

`charAt()` là một trong những function thường xuyên sử dụng nhất, nó sẽ trả về chuỗi tại một vị trí xác định (ví dụ, vị trí thứ 5 của một chuỗi). Nhưng thay vào đó, có một cú pháp shorthand đơn giản hơn mà bạn có thể sử dụng, chỉ cần thêm vị trí ký tự được đặt trong dấu ngoặc vuông ngay phía sau chuỗi đó.

Tuy nhiên hãy lưu ý rằng cú pháp `charAt()` bắt đầu tính từ vị trí số 0. Do đó, ở ví dụ dưới đây thì `myString[4]` sẽ trả về ký tự thứ 5 trong chuỗi.

```
var myString = "Happy birthday";

/* Shorthand */
myString[4];
// y

/* Longhand */
myString.charAt(4);
// y
```

5. Sử dụng toán tử điều kiện

Các toán tử có điều kiện (ternary) thường được sử dụng trong câu lệnh `if-else`. Nó thường bao gồm ba phần như sau:

1. Các điều kiện
2. Điều gì xảy ra nếu điều kiện là đúng (if)
3. Điều gì xảy ra nếu điều kiện là sai (else)

Trong ví dụ dưới đây, chúng ta gửi một tin nhắn đơn giản (bên trong biến `message`). Tuy nhiên, nếu sử dụng shorthand thì chỉ là một dòng code duy nhất thay vì 3 dòng.

```
var age = 17;

/* Shorthand */
var message = age >= 18 ? "Allowed" : "Denied";

/* Longhand */
if( age >= 18) {
  var message = "Allowed";
} else {
  var message = "Denied";
}
```

6. Kiểm tra sự tồn tại của 1 biến

Thường xảy ra khi bạn cần kiểm tra xem một biến có tồn tại hay không. Hãy xem sự khác biệt rõ ràng khi sử dụng shorthand và cú pháp thông thường như ví dụ dưới đây:

Cụ thể như câu lệnh `if(myVar)` không chỉ đơn giản kiểm tra xem biến đó không phải đúng hay sai mà còn làm rất nhiều việc khác nữa. Cụ thể, nó còn kiểm tra xem biến đó có phải là `undefined`, `empty`, `null`, và `false`.

```
var myVar = 99;

/* Shorthand */
if( myVar ) {
  console.log("The myVar variable is defined AND it's not empty
  AND not null AND not false.");
}

/* Longhand */
if( typeof myVar !== "undefined" && myVar !==  "" && myVar !== null
&& myVar !== 0 && myVar !== false  ) {
  console.log("The myVar variable is defined AND it's not empty
  AND not null AND not false.");
}
```

7. Điều kiện IF

Trong 1 câu lệnh `if` thường chúng ta có nhiều điều kiện để return về 1 giá trị nào đó, nhưng thay vì viết 1 cú pháp dài dòng chúng ta có thể sử dụng 1 cú pháp ngắn gọn hơn như ví dụ dưới đây.

```
/* Shorthand */
([1,5,7].indexOf(x) !=- 1) && alert('X has some value!');

/* Longhand */
if(x == 1 || x == 5 || x == 7)  {
  console.log('X has some value');
}
```

8. Switch case

Chúng ta có thể tái cấu trúc lại câu lệnh `switch case` thành một `object`, làm cho cú pháp trở nên ngắn gọn hơn nhiều.

```
/* Shorthand */
var cases = {
  1: doSomething,
  2: doSomethingElse,
  3: doSomethingElseAndOver
};

/* Longhand */
switch (something) {
  case 1:
    doSomething();
  break;
case 2:
    doSomethingElse();
  break;
case 3:
    doSomethingElseAndOver();
  break;
  // And so on...
}
```

9. Khai báo biến

Nếu bạn khai báo nhiều biến với cùng 1 từ khóa (`var, let, const`) thì bạn nên dùng cú pháp dưới đây để rút gọn việc khai báo quá nhiều biến cùng loại trên nhiều dòng.

```
/* Shorthand */
let x, y, z = 3;

/* Longhand */
let x;
let y;
let z = 3;
```

10. Arrow functions

Các function cổ điển có vẻ rất dễ đọc và viết ở dạng đơn giản, nhưng chúng có xu hướng trở nên hơi dài dòng và khó hiểu khi bạn bắt đầu lồng chúng vào các lệnh gọi hàm khác. Thay vì đó bạn nên sử dụng arrow function nếu muốn sau này kiểm soát code của mình dễ dàng hơn.

```
/* Shorthand */
sayHello = name => console.log('Hello', name);

setTimeout(() => console.log('Loaded'), 2000);

list.forEach(item => console.log(item));

/* Longhand */
function sayHello(name) {
  console.log('Hello', name);
}

setTimeout(function() {
  console.log('Loaded')
}, 2000);

list.forEach(function(item) {
  console.log(item);
});
```