Notes :

**Modulus :**

5 % 2 = 1 because

Math.floor(5 / 2) = 2 (Quotient)

2 \* 2 = 4

5 - 4 = 1 (Remainder)

11 % 3 = 2 because

Math.floor(11 / 3 ) = 3 (Quotient)

3 \* 3 = 9

11 – 9 = 2 (Remainder)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

const firstName = "Ada";

const lastLetter = firstName[0];

example use *bracket notation*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

multi-dimensional array

const teams = [["Bulls", 23], ["White Sox", 45]];

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Array**

Equality Operator ( == )

Strict Equality Operator ( === )

Inequality Operator ( != )

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Object**

Ada dua acara untuk mengakses *value* dari sebuah *property :*

1 – Dot Notation (.) -> myObject.name

2- Bracket Notation ([]) -> myObject[“name”]

Penggunaan lain dari *Bracket Notation* pada objek adalah untuk mengakses *properti* yang disimpan sebagai nilai variable :

const dogs = {

Fido: "Mutt",

Hunter: "Doberman",

Snoopie: "Beagle"

};

const myDog = "Hunter";

const myBreed = dogs[myDog];

console.log(myBreed);

String “Doberman ” akan ditampilkan pada *console[[1]](#footnote-1)*

Meghapus property object -> delete object.name

Mengecek *property* didalam suatu *object* :

* object.hasOwnProperty("name");
* object.hasOwnProperty(name);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Use Multiple Conditional (Ternary) Operators**

*Ternary operator* bisa dibuat lebih dari satu cabang

function findGreaterOrEqual(a, b) {

if (a === b) {

return "a and b are equal";

}

else if (a > b) {

return "a is greater";

}

else {

return "b is greater";

}

}

Diubah menjadi :

function findGreaterOrEqual(a, b) {

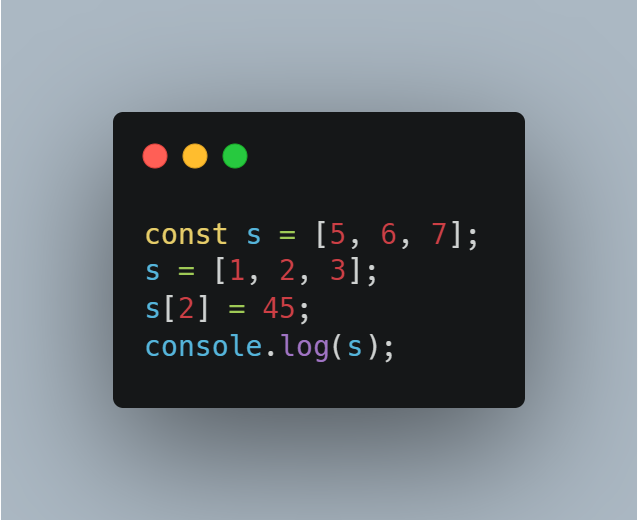
return (a === b) ? "a and b are equal" : (a > b) ? "a is greater": "b is greater";

}

**Javascript Lanjutan / ES6**

**Object freeze**

Pada umumnya deklarasi menggunakan variable *const* untuk mencegah perubahan pada nilai, kecuali jika variable tersebut ingin dirubah nilainnya maka *let* sebagai gantinya.

Meskipun demikian *const* dapat diubah nilainya pada *Array*  dan *Object* :

Maka “S” menghasilkan nilai [ 5 , 6 , 45 ] bukan [1, 2, 3] karena nilai *const* tidak bisa diubah akan tetapi dengan mendetailkan salah satu nilai s (sperti : s[2] = 45) maka nilai tersebut masih bisa dirubah.

Maka dari itu *method* “Object.freeze” dapat mencegah itu semua.

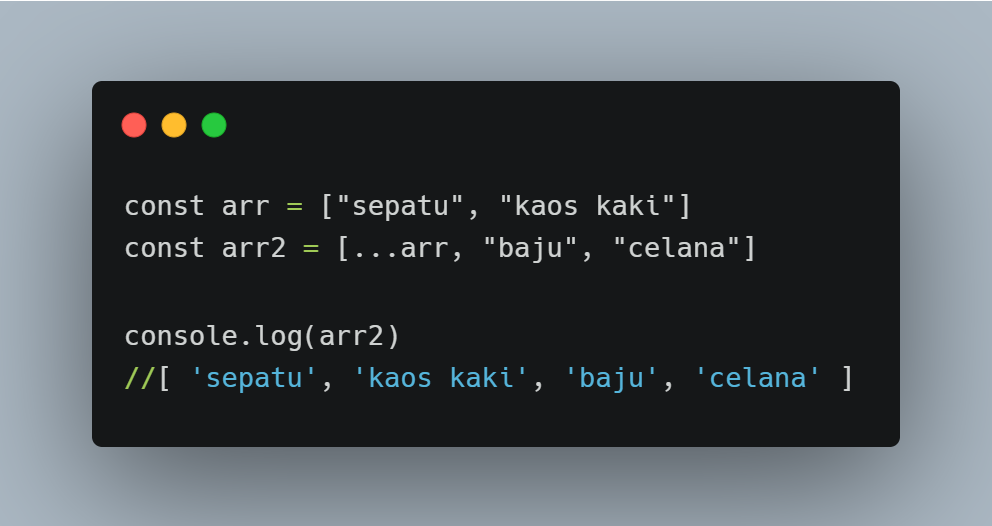
Contoh :

Console akan menampilkanan obj { name : “FreeCodeCamp”, review : “Awesome” } karena perubahan tersebut dicegah dengan Object.freeze.

**Rest Parameters**

****

**Spread Operator**

****

**Destructuring Assignment to Extract Values from Objects**

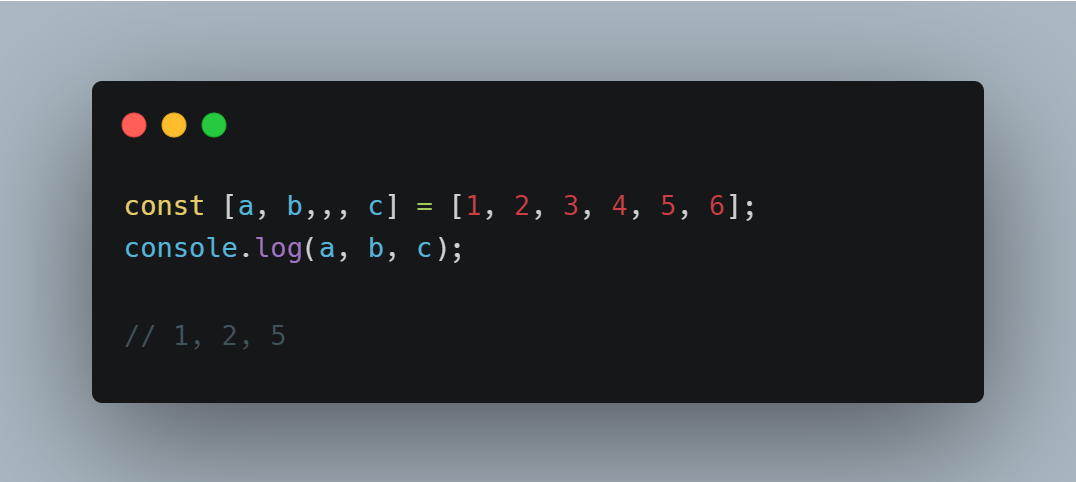
****

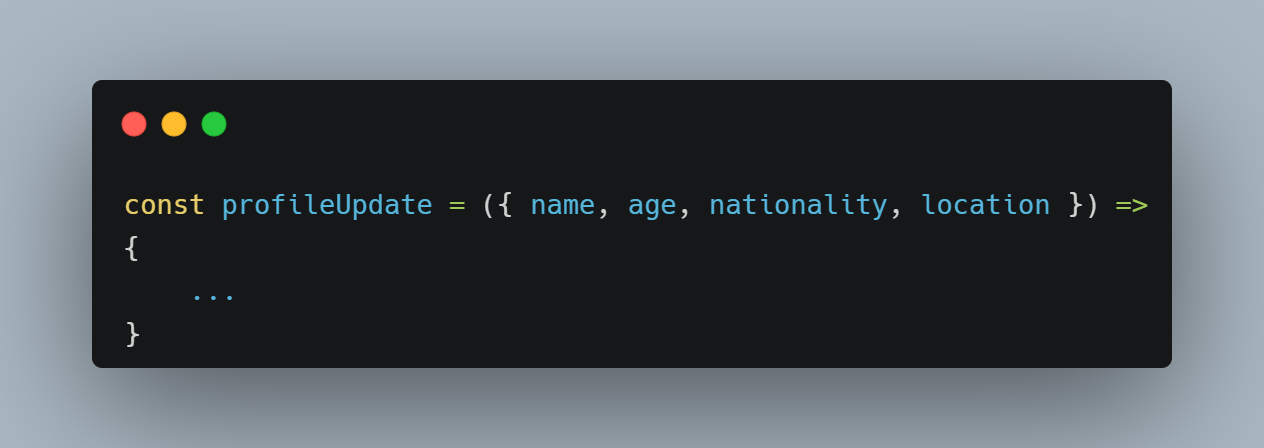
**Destructuring Assignment to Assign Variables from Objects**

****

Variable “username” akan menyimpan nilai dari variable “user.name” yaitu “John Doe” dan seterusnya.

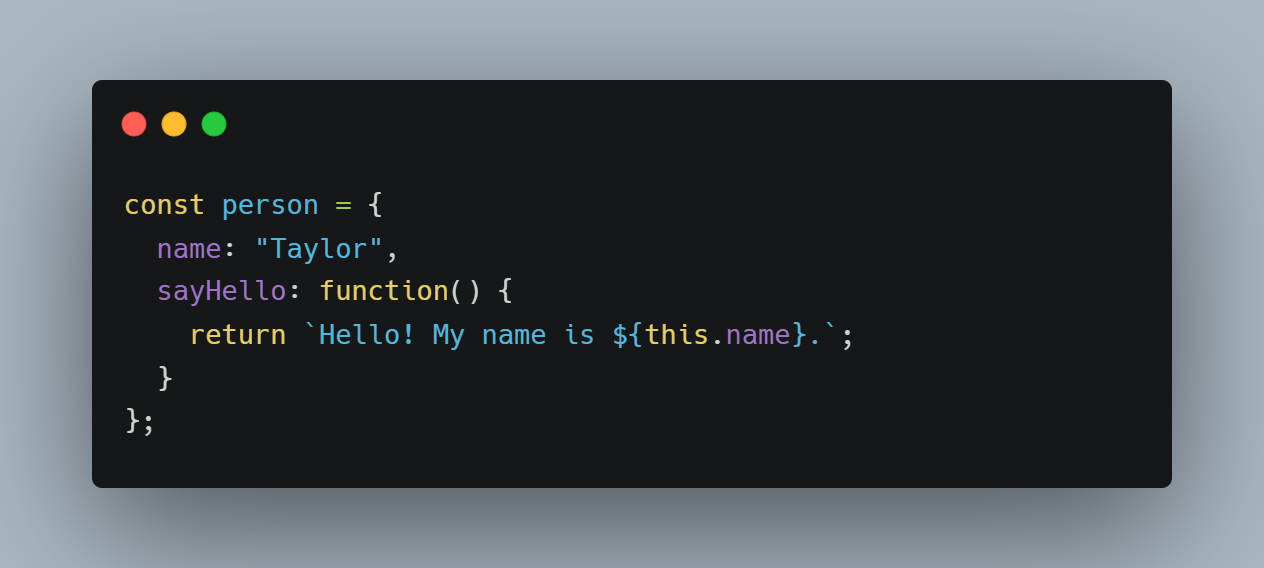
**Destructuring Assignment to Assign Variables from Arrays**

****

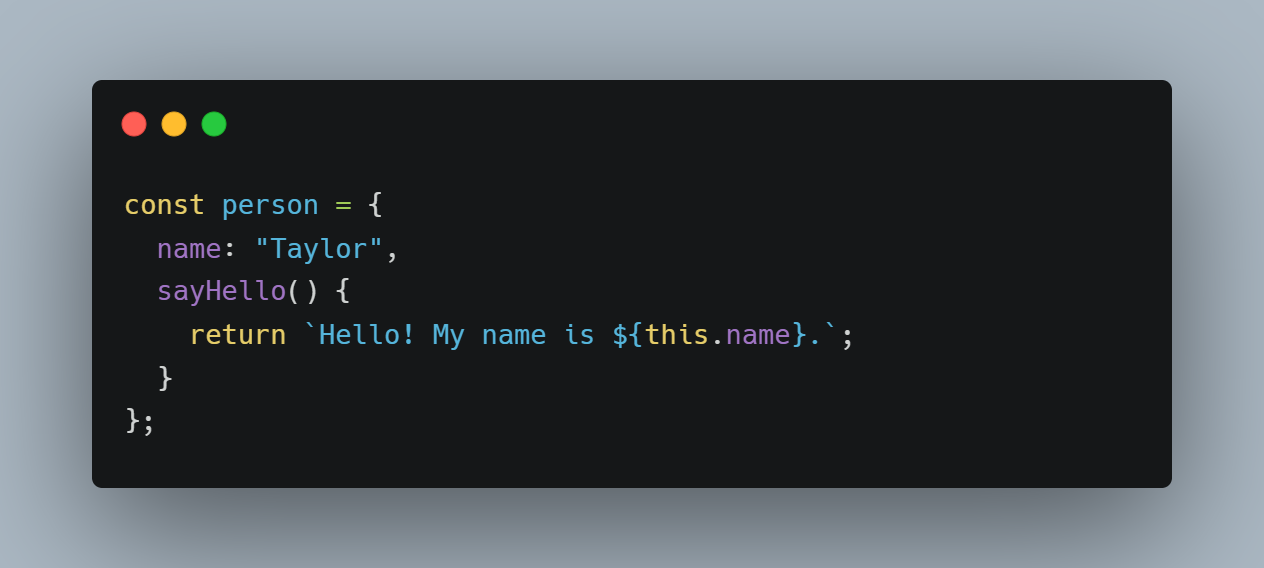
**Destructuring Assignment to Pass an Object as a Function's Parameters**

**Declarative Functions with ES6**

Penulisan function dalam object bisa ditulis seperti ini :

****

Dalam ES6, kamu bisa meringkas dengan menghilangkan tulisan “*function”* dan “:” (titik dua) secara bersamaan.



**Class Syntax to Define a Constructor Function**

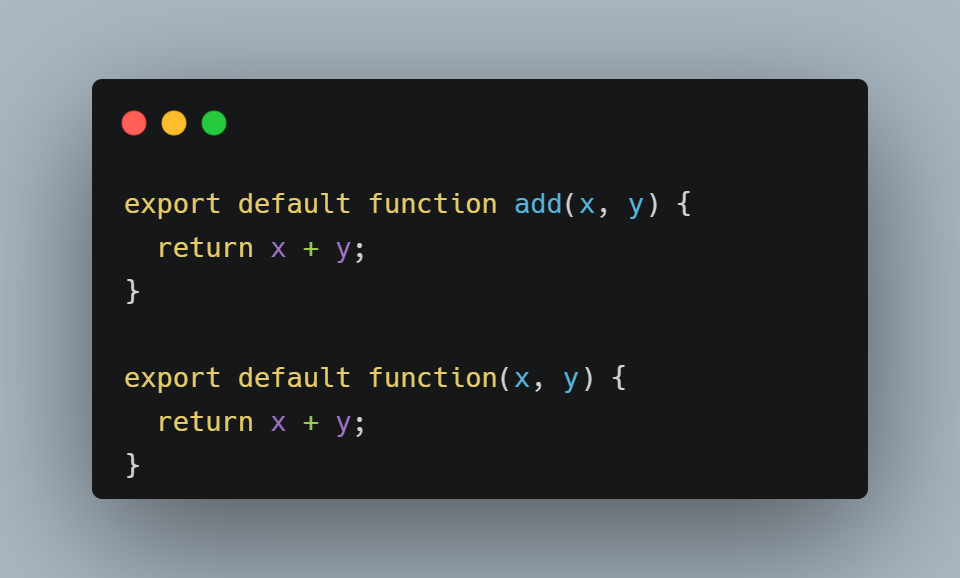
****

**Getter and Setter**

****

For more read about it let’s check out :

* [MDN Mozilla](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/get?retiredLocale=id)
* [W3School](https://www.w3schools.com/js/js_object_accessors.asp)

**Export Fallback with export default**

**Promise**

****Promise adalah Sebuah *constructor function,* untuk membuatnya dibutuhkan sebuah keyword *new* dan dua parameter  *resolve* dan *reject.*



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Regex**

Singkatan dari *regular expression* adalah sebuah pola untuk mencocokan kumpulan karakter yang terdapat pada *string*.

**.test()**

Untuk mencari kata “the” dalam sebuah *string* “” bisa menggunakan dengan menulis regex : /the/.

Salah satu cara untuk mencarinya dengan menggunakan *method* “.test()” dengan cara ini regex akan mengembalikan nilai true atau false ,**ingat huruf besar dan huruf kecil itu tidak sama didalam javascript!**

regex juga bisa memilki banyak pola,contoh :

regex juga bisa mencocokan sebuah string yang sama tanpa memperdulikan huruf besar dan kecil, dengan manmbahkan “i” dibelakangnya ,seperti “/freecodecamp/i”

apapun nilainya seperti “freeCodeCamp” ,” FreeCodeCamp”, “FreeCodecamp” atau “FrEeCodECamp” selama nilainya sama akan mengembalikan *true .*

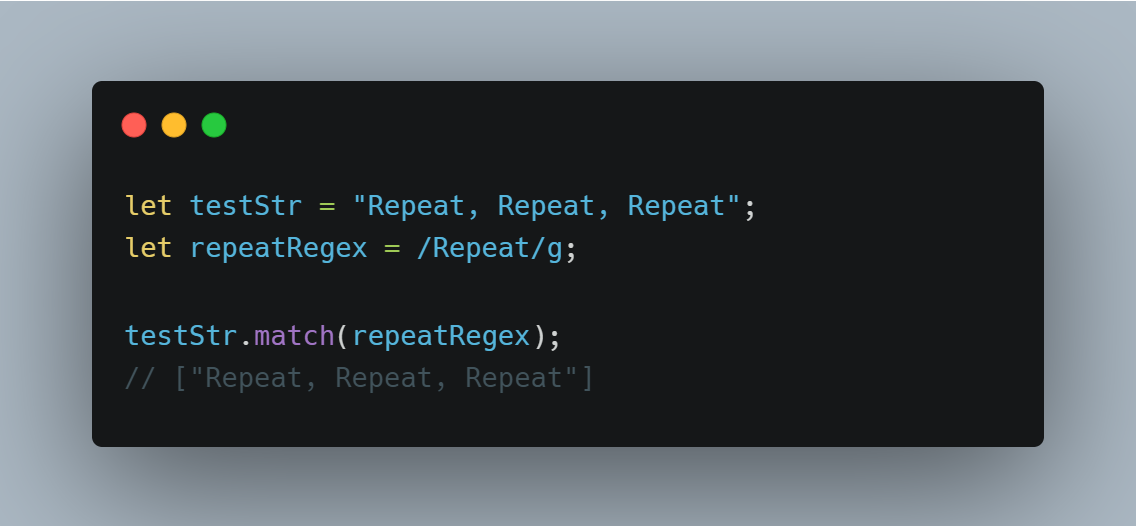
**.match()**

Kamu juga bisa mengestrak nilai yang sama dengan method ‘*.match()’.*



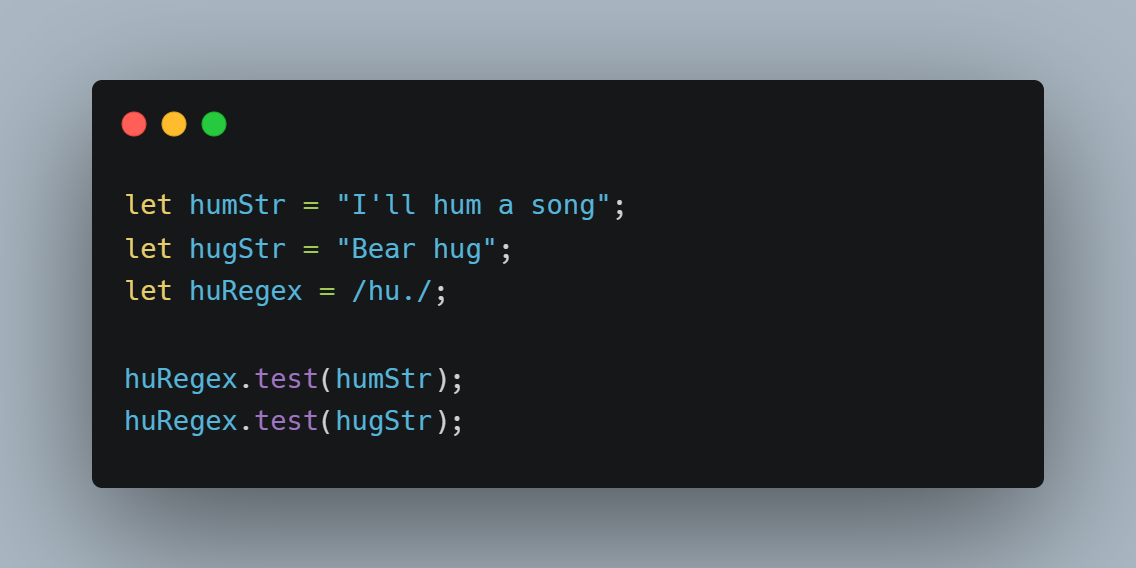
Contoh pertama akan menghasilkan [“Hello”] dan contoh kedua menghasilkan [“expressions”].

Untuk mengestrak nilai lebih dari satu kali, kamu bisa menambahkan *global search flag* yaitu “g”



**Match Anything with Wildcard Period**

Kamu juga bisa menggunakan *The wildcard character (.)* tanpa perlu tau kata apa yang tepat didalam sebuah *petterns . wildcard character*  juga disebut *dot* atau *period* .kamu bisa mencocokan kata *hug, huh, hut,* dan *hum* menggunakan “ /hu./”



Dua nilai tersebut akan menghasilkan *true.*

**Match Letters of the Alphabet**

 Kamu bisa menetapkan beberapa *range* dari sebuah karakter menggunakan *hyphen character :* “-“

Secara berurutan , dari ketiga string itu akan menghasilkan [“cat”], [“bat”] dan null. Kenapa yang satunya *null* sementara yang lain tidak, karena urutan alfabet dari “a” ke “e” tidak ditemukan huruf “m” maka nilai yang dikembalikan adalah *null* .

Kamu juga bisa mencocokan sebuah karakter yang selain ditentukan dengan menggunakan *negated character sets:* ^*.* Cara penulisannya dengan meletakkan ^ setelah *opening bracket* dan sebelum karakter yang akan dicocokan .

Contoh /[^aeiou]/gi maka dia mencocokan semua karakter yang selain huruf vowel (a,I,u,e,o) termasuk spasi, ., !, [, @, / .

**Match Characters that Occur One or More Times**

**Match Characters that Occur Zero or More Times**

**Find Characters with Lazy Matching**

**Find One or More Criminals in a Hunt**

**Match Beginning String Patterns**

**Match Ending String Patterns**

**Match All Letters and Numbers (\w)**

**Match Everything But Letters and Numbers (\W)**

**Match All Numbers**

**Match All Non-Numbers**

1. <https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/accessing-object-properties-with-variables> [↑](#footnote-ref-1)