Javascript Fundamentals

My Technical Notes:

```
function changeName(firstname, lastname) {
  firstname = "something";
  lastname = "else";
}
let firstname = "First";
let lastname = "Last";
changeName(firstname, lastname);
console.log(firstname, lastname);
```

https://javascript.plainenglish.io/javascript-concepts-every-programmer-should-know-v1-0-2-cc87f541e05

1.Javascript Basics

Sum two numbers:

// Write a function that takes two numbers (a and b) as argument
// Sum a and b
// Return the result

```
function myFunction(a, b) {
  return a + b;
}
```

Test cases:

```
myFunction(1,2) Expected: 3
myFunction(1,10) Expected: 11
myFunction(99,1) Expected: 100
```

Comparison operators, strict equality:

// Write a function that takes two values, say a and b, as arguments
// Return true if the two values are equal and of the same type

```
function myFunction(a, b) {
 return a === b;
}
```

Test cases:

```
myFunction(2, 3) Expected: false
myFunction(3, 3)  Expected: true
myFunction(1, '1') Expected: false
myFunction('10', '10')  Expected: true
```

Get type of value:

// Write a function that takes a value as argument
// Return the type of the value

```
function myFunction(a) {
  return typeof a;
}
```

Test cases:

```
myFunction(1)  Expected: 'number'
myFunction(false) Expected: 'boolean'
myFunction({})  Expected: 'object'
myFunction(null)  Expected: 'object'
myFunction('string')  Expected:'string'
myFunction(['array']) Expected: 'object'
```

Get nth character of string:

// Write a function that takes a string (a) and a number (n) as argument
// Return the nth character of 'a'

```
function myFunction(a, n){
  return a[n-1];
}
```

Test cases:

```
myFunction('abcd',1)     Expected: 'a'
myFunction('zyxbwpl',5)     Expected: 'w'
```

Remove first n characters of string:

// Write a function that takes a string (a) as argument
// Remove the first 3 characters of a
// Return the result

```
function myFunction(a) {
  return a.slice(3);
}
```

Test cases:

```
myFunction('abcdefg')Expected: 'defg'
myFunction('1234') Expected: '4'
myFunction('fgedcba') Expected: 'dcba'
```

Get last n characters of string:

// Write a function that takes a string as argument
// Extract the last 3 characters from the string
// Return the result

```
function myFunction(str) {
  return str.substr(str.length - 3);
}
function myFunction(str) {
  return str.slice(-3);
}
```

Test Cases:

```
myFunction('abcdefg')Expected: 'efg'
myFunction('1234') Expected: '234'
myFunction('fgedcba') Expected: 'cba'
```

Get first n characters of string:

// Write a function that takes a string (a) as argument
// Get the first 3 characters of a
// Return the result

```
function myFunction(a) {
  return a.slice(0, 3);
}
function myFunction(a) {
  return a.substr(0, 3);
}
```

Test cases:

```
myFunction('abcdefg') Expected: 'abc'
myFunction('1234') Expected: '123'
myFunction('fgedcba') Expected: 'fge'
```

Extract first half of string:

// Write a function that takes a string (a) as argument
// Extract the first half a
// Return the result

```
function myFunction(a) {
  return a.slice(0, a.length / 2);
}
```

Test cases:

```
myFunction('abcdefgh') Expected: 'abcd'
myFunction('1234')  Expected: '12'
myFunction('gedcba')  Expected: 'ged'
```

Remove last n characters of string:

// Write a function that takes a string (a) as argument
// Remove the last 3 characters of a
// Return the result

```
function myFunction(a) {
  return a.slice(0, -3);
}
```

Test cases:

```
myFunction('abcdefg') Expected: 'abcd'
myFunction('1234') Expected: '1'
myFunction('fgedcba') Expected: 'fged'
```

Return the percentage of a number:

// Write a function that takes two numbers (a and b) as argument
// Return b percent of a

```
function myFunction(a, b) {
 return b / 100 * a
}
```

Test cases:

```
myFunction(100,50)  Expected: 50
myFunction(10,1)  Expected: 0.1
myFunction(500,25)  Expected: 125
```

Multiplication, division, and comparison operators:

// Write a function that takes two numbers (a and b) as arguments
// If a is smaller than b, divide a by b
// Otherwise, multiply both numbers

// Return the resulting value

```
function myFunction(a, b) {
  if (a < b) {
  return a / b;
  }
  return a * b;
}
function myFunction(a, b) {
 return a < b ? a / b : a * b
}
```

Test Cases:

```
myFunction(10, 100) Expected: 0.1
myFunction(90, 45) Expected: 4050
myFunction(8, 20) Expected: 0.4
myFunction(2, 0.5) Expected: 1
```

Check whether a string contains another string and concatenate:

// Write a function that takes two strings (a and b) as arguments
// If a contains b, append b to the beginning of a
// If not, append it to the end
// Return the concatenation

```
function myFunction(a, b) {
   return a.includes(b) ? (b + a) : (a + b);
}
function myFunction(a, b) {
 return a.indexOf(b) === -1 ? a + b : b + a
}
```

Test cases:

```
myFunction('cheese', 'cake') Expected: 'cheesecake'
myFunction('lips', 's') Expected: 'slips'
myFunction('Java', 'script') Expected: 'Javascript'
myFunction(' think, therefore I am', 'I') Expected: 'I think,
therefore I am'
```

Basic JavaScript math operators:

// Write a function that takes 6 values (a,b,c,d,e,f) as arguments
// Sum a and b
// Then substract by c
// Then multiply by d and divide by e
// Finally raise to the power of f and return the result
// Tipp: mind the order

```
function myFunction(a, b, c, d, e, f) {
  return (((a + b - c) * d) / e) ** f;
}
```

```
myFunction(6,5,4,3,2,1) Expected: 10.5
myFunction(6,2,1,4,2,3) Expected: 2744
myFunction(2,3,6,4,2,3) Expected: -8
```

Check if a number is even:

// Write a function that takes a number as argument
// If the number is even, return true
// Otherwise, return false

```
function myFunction(a) {
   return a % 2 === 0 ? true :false;
}
function myFunction(a) {
 return a % 2 === 0
}
```

```
myFunction(10) Expected: true
myFunction(-4) Expected: true
myFunction(5) Expected: false
myFunction(-111)Expected: false
```

2.Javascript Arrays:

Get nth element of array:

// Write a function that takes an array (a) and a value (n) as argument
// Return the nth element of 'a'

```
function myFunction(a, n) {
  return a[n - 1];
}
```

Test cases:

```
myFunction([1,2,3,4,5],3) Expected: 3
myFunction([10,9,8,7,6],5) Expected: 6
myFunction([7,2,1,6,3],1) Expected: 7
```

Remove first n elements of an array:

// Write a function that takes an array (a) as argument
// Remove the first 3 elements of 'a'
// Return the result

```
function myFunction(a) {
   return a.splice(3);
}
function myFunction(a) {
  return a.slice(3);
}
```

Test cases:

```
myFunction([1,2,3,4]) Expected: [4]
myFunction([5,4,3,2,1,0]) Expected: [2,1,0]
myFunction([99,1,1]) Expected: []
```

Get last n elements of an array:

// Write a function that takes an array (a) as argument
// Extract the last 3 elements of a
// Return the resulting array

```
function myFunction(a) {
  return a.slice(a.length - 3)
}
function myFunction(a) {
  return a.slice(-3);
}
```

Test cases:

```
myFunction([1,2,3,4]) Expected: [2,3,4]
myFunction([5,4,3,2,1,0]) Expected: [2,1,0]
myFunction([99,1,1]) Expected: [99,1,1]
```

Get first n elements of an array:

// Write a function that takes an array (a) as argument
// Extract the first 3 elements of a
// Return the resulting array

```
function myFunction(a) {
  return a.slice(0, 3);
}
```

Test cases:

```
myFunction([1,2,3,4]) Expected: [1,2,3]
myFunction([5,4,3,2,1,0]) Expected: [5,4,3]
myFunction([99,1,1]) Expected: [99,1,1]
```

Return last n array elements:

// Write a function that takes an array (a) and a number (n) as arguments
// It should return the last n elements of a

```
function myFunction(a, n) {
 return a.slice(-n);
}
```

Test cases:

```
myFunction([1, 2, 3, 4, 5], 2) Expected: [ 4, 5 ]
myFunction([1, 2, 3], 6) Expected: [ 1, 2, 3 ]
myFunction([1, 2, 3, 4, 5, 6, 7, 8], 3) Expected: [ 6, 7, 8 ]
```

Count number of elements in JavaScript array:

// Write a function that takes an array (a) as argument
// Return the number of elements in a

```
function myFunction(a) {
  return a.length;
}
```

Test cases:

```
myFunction([1,2,2,4]) Expected: 4
myFunction([9,9,9]) Expected: 3
myFunction([4,3,2,1,0]) Expected: 5
```

Count number of negative values in array:

// Write a function that takes an array of numbers as argument
// Return the number of negative values in the array

```

function myFunction(a) {
  let count = 0;
  a.forEach((value) => {
    if (value < 0) {
       count++
    }
  });
  return count;
}

function myFunction(a) {
  return a.filter((el) => el < 0).length;
}
```

Test cases:

```
myFunction([1,-2,2,-4]) Expected: 2
myFunction([0,9,1]) Expected: 0
myFunction([4,-3,2,1,0]) Expected: 1
```

Sort an array of strings alphabetically:

// Write a function that takes an array of strings as argument
// Sort the array elements alphabetically
// Return the result

```
function myFunction( arr ) {
return arr.sort()
}
```

Test cases:

```
myFunction(['b', 'c', 'd', 'a']) Expected: ['a', 'b', 'c', 'd']
myFunction(['z', 'c', 'd', 'a', 'y', 'a', 'w']) Expected:
['a', 'a', 'c', 'd', 'w', 'y', 'z']
```

Remove a specific array element:

// Write a function that takes an array (a) and a value (b) as argument
// The function should clean a from all occurrences of b
// Return the filtered array

```
function myFunction( a, b ) {
 return a.filter(cur => cur !== b)
}
```

Test cases:

```
myFunction([1,2,3], 2) Expected: [1, 3]
myFunction([1,2,'2'], '2') Expected: [1, 2]
myFunction([false,'2',1], false) Expected: ['2', 1]
myFunction([1,2,'2',1], 1) Expected: [2, '2']
```

Sort an array of numbers in descending order:

// Write a function that takes an array of numbers as argument
// It should return an array with the numbers sorted in descending order

```
function myFunction(arr) {
  arr.sort();
  return arr.reverse();
}
function myFunction( arr ) {
 return arr.sort((a, b) => b - a)
}
```

```
myFunction([1,3,2]) Expected: [3,2,1]
myFunction([4,2,3,1]) Expected: [4,3,2,1]
```

Calculate the sum of an array of numbers:

// Write a function that takes an array of numbers as argument
// It should return the sum of the numbers

```
function myFunction(a) {
  return a.reduce((acc, cur) => acc + cur, 0);
}
```

```
myFunction([10,100,40]) Expected: 150
myFunction([10,100,1000,1]) Expected: 1111
myFunction([-50,0,50,200])  Expected: 200
```

3.Javascript Objects:

Accessing object properties one:

// Write a function that takes an object with two properties as argument
// It should return the value of the property with key country

```
function myFunction(obj) {
 return obj.country
}
```

Test case:

```
myFunction({  continent: 'Asia',  country: 'Japan'}): Expected:
'Japan'
myFunction({  country: 'Sweden',  continent: 'Europe'}) Expected:
'Sweden'
```

Accessing object properties two:

// Write a function that takes an object with two properties as argument
// It should return the value of the property with key 'prop-2'
// Tipp: you might want to use the square brackets property accessor

Accessing object properties three:

// Write a function that takes an object with two properties and a string as arguments
// It should return the value of the property with key equal to the value of the string

```
function myFunction(obj, key) {
 return obj[key]
}
```

Test Cases:

```
myFunction({  continent: 'Asia',  country: 'Japan'}, 'continent')
Expected :'Asia'
myFunction({  country: 'Sweden',  continent: 'Europe'}, 'country')
Expected: 'Sweden'
```

Check if property exists in object:

// Write a function that takes an object (a) and a string (b) as argument
// Return true if a has a property with key b
// Return false otherwise

```
function myFunction(a, b) {
  return b in a;
}
```

Test cases:

```
myFunction({a:1,b:2,c:3},'b'); Expected: true
myFunction({x:'a',y:'b',z:'c'},'a'); Expected: false
myFunction({x:'a',y:'b',z:'c'},'z'); Expected: true
```

Creating Javascript objects one:

// Write a function that a string (a) as argument
// Create an object that has a property with key 'key' and a value of a
// Return the object

```
function myFunction(a) {
  return { key: a };
}
```

Test cases:

```
myFunction('a')Expected: {key:'a'}
myFunction('z') Expected: {key:'z'}
myFunction('b') Expected: {key:'b'}
```

Extract keys from Javascript object:

```
// Write a function that takes an object (a) as argument
// Return an array with all object keys
```

```
function myFunction(a) {
  return Object.keys(a);
}
```

```
Test cases:
```

```
myFunction({a:1,b:2,c:3}) Expected: ['a','b','c']
myFunction({j:9,i:2,x:3,z:4}) Expected: ['j','i','x','z']
myFunction({w:15,x:22,y:13}) Expected: ['w','x','y']
```

Creating Javascript objects two:

// Write a function that takes two strings (a and b) as arguments
// Create an object that has a property with key 'a' and a value of 'b'
// Return the object

```
function myFunction(a, b) {
  return { [a]: b };
}
```

```
myFunction('a','b') Expected: {a:'b'}
myFunction('z','x') Expected: {z:'x'}
myFunction('b','w') Expected: {b:'w'}
```

Creating Javascript objects three:

// Write a function that takes two arrays (a and b) as arguments
// Create an object that has properties with keys 'a' and corresponding values 'b'
// Return the object

```
function myFunction(a, b) {
   return a.reduce((arr, value, i) => (arr[value] = b[i],
arr), {});
}

function myFunction(a, b) {
  return a.reduce((acc, cur, i) => ({ ...acc, [cur]: b[i] }),
{});
}
```

Test cases:

```
myFunction(['a','b','c'],[1,2,3]) Expected: {a:1,b:2,c:3}
myFunction(['w','x','y','z'],[10,9,5,2]) Expected: {w:10,x:9,y:5,z:2}
myFunction([1,'b'],['a',2]) Expected: {1:'a',b:2}
```

Sum object values:

// Write a function that takes an object (a) as argument
// Return the sum of all object values

```
function myFunction(a) {
   return Object.values(a).reduce((a, b) => a +b)
}
function myFunction(a) {
  return Object.values(a).reduce((sum, cur) => sum + cur, 0);
}
```

```
myFunction({a:1,b:2,c:3}) Expected: 6
myFunction({j:9,i:2,x:3,z:4}) Expected: 18
myFunction({w:15,x:22,y:13}) Expected: 50
```

Remove a property from an object:

// Write a function that takes an object as argument
// It should return an object with all original object properties
// except for the property with key 'b'

```
function myFunction(obj) {
  delete obj.b;
  return obj
}

function myFunction(obj) {
 const { b, ...rest } = obj;
 return rest;
}
```

```
myFunction({ a: 1, b: 7, c: 3 })Expected: { a: 1, c: 3 }
myFunction({ b: 0, a: 7, d: 8 }) Expected: { a: 7, d: 8 }
myFunction({ e: 6, f: 4, b: 5, a: 3 }) Expected: { e: 6, f: 4, a: 3 }
```

Check if property exists in object and is truthy:

// Write a function that takes an object (a) and a string (b) as argument
// Return true if the object has a property with key 'b', but only if it has a truthy value
// In other words, it should not be null or undefined or false
// Return false otherwise

```
function myFunction(a, b) {
   return !!a[b];
}
function myFunction(a, b) {
  return Boolean(a[b]);
}
```

```
myFunction({a:1,b:2,c:3},'b') Expected: true
myFunction({x:'a',y:null,z:'c'},'y') Expected: false
myFunction({x:'a',b:'b',z:undefined},'z')Expected: false
```

4.Javascript Dates

Check if one date is earlier than another:

// Write a function that takes two date instances (a and b) as arguments
// It should return true if a is earlier than b
// It should return false otherwise

```
function myFunction(a, b) {
  return a.getTime() < b.getTime();
}

function myFunction(a, b) {
  return a < b
}
```

Test cases:

```
myFunction(new Date('2000/01/01 08:00:00'), new Date('2000/01/01
08:45:00')) Expected: true
myFunction(new Date('2000/01/01 08:45:00'), new Date('2000/01/01
08:00:00')) Expected: false
myFunction(new Date('2000/01/01 08:00:00'), new Date('2000/01/01
08:00:00')) Expected: false
```

5.Javascript Sets

Check if value is present in Set:

```
// Write a function that takes a Set and a value as arguments
// Check if the value is present in the Set
```

```
function myFunction(set, val) {
 return set.has(val);
}
```

Test cases:

```
myFunction(new Set([1, 2, 3]), 2) Expected: true
myFunction(new Set([123]), 2) Expected: false
myFunction(new Set(['1', '2', '3']), '2') Expected: true
myFunction(new Set('123'), '2') Expected: true
```

Convert a Set to Array:

// Write a function that takes a Set as argument

// Convert the Set to an Array
// Return the Array

```
function myFunction(set) {
  return Array.from(set);
}

function myFunction(set) {
 return [...set];
}
```

Test cases:

```
myFunction(new Set([1, 2, 3])) Expected: [1, 2, 3]
myFunction(new Set([123])) Expected: [123]
myFunction(new Set(['1', '2', '3'])) Expected: ['1', '2', '3']
myFunction(new Set('123')) Expected: ['1', '2', '3']
```

Delete element from Set:

// Write a function that takes a Set and a value as argument
// If existing in the Set, remove the value from the Set
// Return the result

```
function myFunction(set, val) {
 set.delete(val);
 return set;
}
```

```
myFunction(new Set([1, 2, 3]), 1) Expected: new Set([2, 3])
myFunction(new Set('12345'), '3') Expected: new Set(['1', '2', '4',
'5'])
myFunction(new Set([1, 2, 3]), 4) Expected: new Set([1, 2, 3])
```