**Rest-Spread Operators. Destructuring Assignments**

1. **Rest Operators and Spreading Operators**

The … operator work with iterables (arrays, strings). As rest operators, it helps merging, while as spread operators, it helps spreading.

(…<newAraay>)

**Rest Operators**: **Merging any number of elements** into an array. Can help functions **receive any number of arguments**.

sum = (…newarr) => newarr.reduce((a,b) => a+b)

console.log( sum( (1,2,3) ) ); // 6

console.log( sum( (1,2,3,4,5) ) ); // 15

**Spreading Operators**: **“Spreading” all elements** of an iterable (arrays/strings). Can be declared as arguments while passing into the function, or to copy to a different array.

arr1 = [1,2,3,4,5];

arr2 = arr1 🡪 Aliasing

arr2 = […arr1] 🡪 Spreading out all elements of arr1, then put into an array.

// arr2 = [1,2,3,4,5]

1. **Destructuring Assignment**

**2.1. Objects**

**Destructuring technique (Property value shorthand)**

Instead of passing para1 value & assign it to para1 key (with the same name), we can instead merge them into one.

const <keyName> = <keyName (passed by parameter)>

Text

Description automatically generated

**Destructured Assignment**

If we want to extract key-value pairs and set them to variables (assign ‘value’ to ‘key’ variable), we can do

const <variable> = <object>.<keyName>

But instead, we can use the following syntax:

**const { key : variable } = <object>**

or to assign multiple property values to variables, or to assign values within nested objects to variables

**const { key1 : var1, key2 : var2, key3 : var3} = <object>**

**const { outerkey : { innerkey : variable } } = <outerobject>**

if key1 and var1 have the same name, we only need to state 1.

Passing objects into functions

Normally, we can easily pass the entire object to functions by passing in the object variable.

However, we can instead pass only the properties we need of the object, by using:

**const object = {min: 10, max: 50};**

**const returnAvg = ( { min, max} ) => (min+max)/2;**

**console.log(returnAvg(objects));** //30

**Note** how we only need use ‘min’ and ‘max’ not ‘object.min’ and ‘object.max’

**2.2. Arrays**

**Assign variables from arrays**

[z , x, , y] = [1, 2, 3, 4, 5, 6] 🡪 z = 1; x = 2, y = 4

**Switch**

a = 8, b = 6

[a , b] = [b, a] 🡪 a = 6, b = 8

**Reassign array elements**

arr1 = [1, 2, 3, 4, 5, 6];

arr2 = [ , , …arr1] 🡪 arr2 = [3, 4, 5, 6]