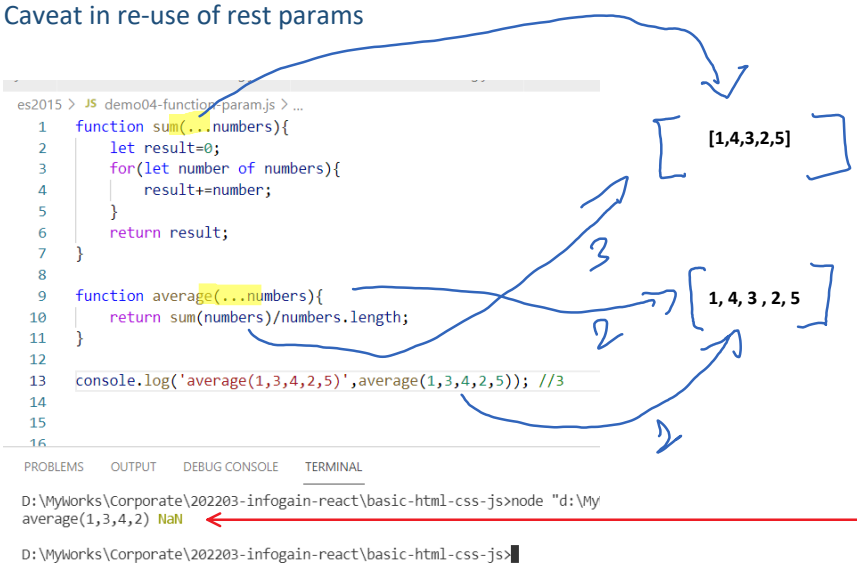
ES2015 Feature Set02

Monday, March 14, 2022 3:34 PM

Caveat in re-use of rest params




```
es2015 > JS demo04-function-param.js > ...
1   function sum(...numbers){
2       let result=0;
3       for(let number of numbers){
4           result+=number;
5       }
6       return result;
7   }
8
9   function average(...numbers){
10      return sum(numbers)/numbers.length;
11  }
12
13  console.log('average(1,3,4,2,5)',average(1,3,4,2,5)); //3
14
15
16
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

D:\MyWorks\Corporate\202203-infogain-react\basic-html-css-js>node "d:\My
average(1,3,4,2) NaN

D:\MyWorks\Corporate\202203-infogain-react\basic-html-css-js>
```

- Second … put my arguments in another array
- Now sum receives
  - An array which contains a single item
    - Array of 5 values
- **result+=number**
  - Actually adds **0 + [1,3,4,2,5]**

4. Javascript Spread Operator

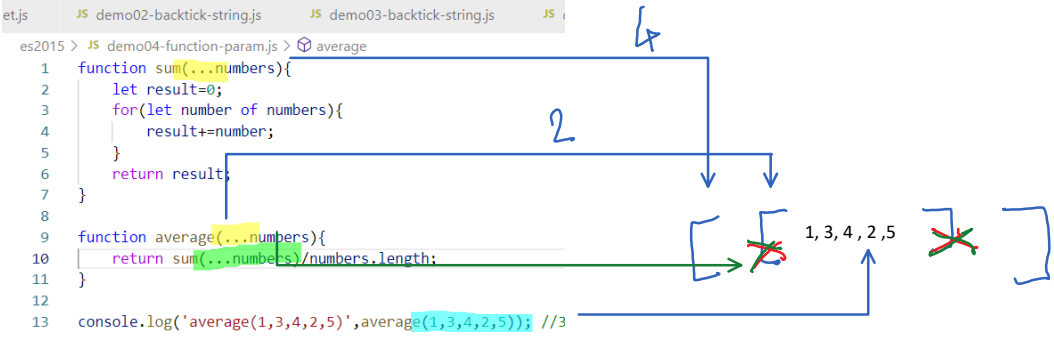
- JavaScript ES2015 defines a spread operator
- Spread operator is opposite of rest param
- It spreads out the content of an array as individual values.
- Ironically this operator looks exactly like params operator …

How do I differentiate between spread operator and rest parameter?

- rest parameter syntax works only when defining a function signat
- Spread operator is used everywhere else.

```
let  array = [2,3,4,6] ;


function callMe( x, y ){

    //logic doesn't matter
}


callMe( array );   // x-> 2   y->3

callMe( array );   // x-> [2,3,4,6]   y->undefined


callMe( ... array );   // callMe(2,3,4,6) // x-> 2   y->3    4, 6 (not assigned)
```

Avearge Function revisited




```
1   function sum(...numbers){
2       let result=0;
3       for(let number of numbers){
4           result+=number;
5       }
6       return result;
7   }
8
9   function average(...numbers){
10      return sum(...numbers)/numbers.length;
11  }
12
13  console.log('average(1,3,4,2,5)',average(1,3,4,2,5)); //3
```

- Spread operator
  an array to comm
  separated indepe
  valuess

```
14
15
16
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

D:\MyWorks\Corporate\202203-infogain-react\basic-html-css-js>node "
average(1,3,4,2,5) 3
```

es2015 > JS demo05-spread-operator.js > ...

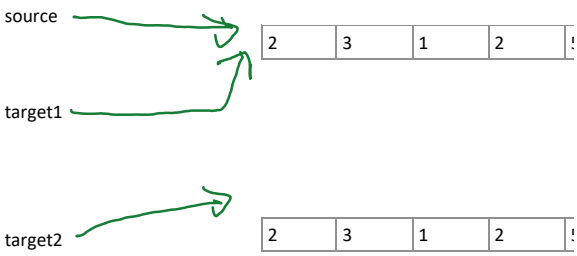
```
1
2
3   function callMe(x,y){
4       console.log(`x=${x}\ty=${y}\targuments=${JSON.stringify(arguments)}`);
5   }
6
7   callMe(12,31);
8
9   const arr=[2,3,9,2,5];
10
11  callMe(arr);
12
13  callMe(...arr);
```

How to merge multiple arrays

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

D:\MyWorks\Corporate\202203-infogain-react\basic-html-css-js>node "d:\MyWorks\Corporate\202203
\demo05-spread-operator.js"
x=12    y=31    arguments={"0":12,"1":31}
x=2,3,9,2,5     y=undefined     arguments={"0":[2,3,9,2,5]}
x=2     y=3     arguments={"0":2,"1":3,"2":9,"3":2,"4":5}
```

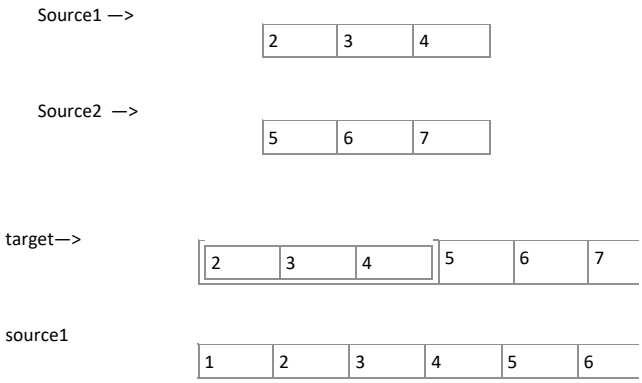
How to copy an array into another array

```
const  source=[2,3,1,2,5];

const target1 = source;    //refers to same memory of source

const target2 = [ ...source ]; //creates new duplicate array
```



How to merge two array in a third array

```
const  source1= [2,3,4] ;

const source2=[5,6,7];

//how to get the array [ 1, 2, 3, 4, 5, 6 , 7 ]

~~const  target = [   source1,  … source2 ];~~

const   target =  [ 1, …source1, …source2 ]
```



Spreading an object properties

- Just like we can spread an array, we can also spread an object properties.
- The object properties are spread as key value pair
- It only make sense to spread an object inside another object

```
let person={name:'Sanjay', email:'sanjay@gmail.com', age:50, phone:"1223332"};
```

```
//ES2015 SHORTCUT:

let emp2={
    id:1,
    ...person,    //gets all field of person including name, age, phone and email (personal)

    salary:50000,
    email:'Sanjay@company.com'  //this email value will replace the personal email value
}
```

Array DE structuring

- Sometimes we need to copy the values from an array to local variables
- Example

```
const  array = [2,3,9, 7];

const x = array [0 ];   //2

const y = array [1] ;  // 3
```

ES2015 Shortcut

```
const  [ a, b ] = array ;   // a--> 2    b—> 3
```

How to get first and third item 2 and 9

- We can leave empty spaces by comma

```
const [ i ,, j ] =array;  // i—> 2, 3 (ignored) , j—>9
```

Destructuring an Object

- Just like an array we can destructure and object also
- We can take out some fields from that object

```
let  person= {
    name:"Sanjay",
    email:"sanjay@email.com",
    phone:"9035949494",
    age:50
}
```

How do I get the name and phone field of this person?

ES5 Approach

```
const  name= person.name;
const phone = person.phone;
```

ES2015 approach

```
const  { name, phone  }  =  person;
```

- Create varaiable name and phone
- Assign the value of the two fields to corresponding variable

How do I get 98th and 99th item?

- Theoretically you can do it by giving 98 commas
- In practice this approach is used only for
  - Short arrays
  - When we want to access first few items only

- If you need 98th and 99th item
  - Don't waste time in counting commans
  - Use ES5 approach

```
let a = array[98];

let b= array[99];
```

How do I get phone as 'cell'?

```
const  { name, phone : cell }  =  person;
```

- Create variables name and cell
- name—> person.name
- cell—> person . phone

Destructuring Method parameter

- Suppose we have a person with many details

```
let  person= {
    name:"Sanjay",
    email:"sanjay@email.com",
    phone:"9035949494",
    age:50
}
```

- We want to create a function that can send email to some email address of somebody
  - Person
  - Company
  - Server

ES 5 Approach #1

```
function  sendEmail( email ) {

    console.log('sending email to ',email);
}

sendEmail( person.email);
```

ES 2015 Approach #1

```
function  sendEmail( person ) {

    console.log('sending email to ',person.email);
}

sendEmail( person);
```

Manual extraction of value

ES 2015

```
function  sendEmail( {email} ) {

    console.log('sending email to ',email);
}

sendEmail( person);
```

Manual extraction of value

- Get only the email property of the supplied object
- Ignore rest of the functions