# 30 Days Of JavaScript: Destructuring and Spreading

LinkedIn Follow @asabeneh

Author: Asabeneh Yetayeh

January, 2020

<< Day 10 | Day 12>>



- Day 11
  - Destructuring and Spread
    - Destructing Arrays
    - Destructuring during iteration
    - Destructuring Object
    - Renaming during structuring
    - Object parameter without destructuring
    - Object parameter with destructuring
    - Destructuring object during iteration
    - Spread or Rest Operator
    - Spread operator to get the rest of array elements
    - Spread operator to copy array
    - Spread operator to copy object
      - Spread operator with arrow function
  - Exercises
    - Exercises: Level 1
    - Exercises: Level 2
    - Exercises: Level 3


# Day 11

## Destructuring and Spread

Destructuring is a way to unpack arrays, and objects and assigning to a distinct variable.

### Destructing Arrays

```
  const numbers = [1, 2, 3]
  let [numOne, numTwo, numThree] = numbers

  console.log(numOne, numTwo, numThree)
```

```
  1 2 3
```

```
  const names = ['Asabeneh', 'Brook', 'David', 'John']
  let [firstPerson, secondPerson, thirdPerson, fourthPerson] = names

  console.log(firstPerson, secondPerson,thirdPerson, fourthPerson)
```

```
Asabeneh Brook David John
```

```
  const scientificConstants = [2.72, 3.14, 9.81, 37, 100]
  let [e, pi, gravity, bodyTemp, boilingTemp] = scientificConstants

  console.log(e,pi,gravity, bodyTemp, boilingTemp)
```

```
2.72 3.14 9.81 37 100
```

```
const fullStack = [
  ['HTML', 'CSS', 'JS', 'React'],
  ['Node', 'Express', 'MongoDB']
]
const [frontEnd, backEnd] = fullStack

console.log(frontEnd)
console.log(backEnd)
```

```
["HTML", "CSS", "JS", "React"]
["Node", "Express", "MongoDB"]
```

If we like to skip on of the values in the array we use additional comma. The comma helps to omit the value at that specific index

```
  const numbers = [1, 2, 3]
  let [numOne, , numThree] = numbers //2 is omitted
```

```
  console.log(numOne, numThree)
```

```
1 3
```

```
  const names = ['Asabeneh', 'Brook', 'David', 'John']
  let [, secondPerson, , fourthPerson] = names // first and third person is omitted

  console.log(secondPerson, fourthPerson)
```

```
Brook John
```

We can use default value in case the value of array for that index is undefined:

```
const names = [undefined, 'Brook', 'David']
let [
  firstPerson = 'Asabeneh',
  secondPerson,
  thirdPerson,
  fourthPerson = 'John'
] = names

console.log(firstPerson, secondPerson, thirdPerson, fourthPerson)
```

```
Asabeneh Brook David John
```

We can not assign variable to all the elements in the array. We can destructure few of the first and we can get the remaining as array using spread operator(...).

```
const nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
let [num1, num2, num3, ...rest] = nums

console.log(num1, num2, num3)
console.log(rest)
```

```
1 2 3
[4, 5, 6, 7, 8, 9, 10]
```

## Destructuring during iteration

```
const countries = [['Finland', 'Helsinki'], ['Sweden', 'Stockholm'], ['Norway', 'Oslo']]

for (const [country, city] of countries) {
console.log(country, city)
}
```

```
Finland Helsinki
Sweden Stockholm
Norway Oslo
```

```
const fullStack = [
  ['HTML', 'CSS', 'JS', 'React'],
  ['Node', 'Express', 'MongoDB']
]

for(const [first, second, third] of fullStack) {
console.log(first, second, third)
}
```

```
HTML CSS JS
Node Express MongoDB
```

## Destructuring Object

When we destructure the name of the variable we use to destructure should be exactly the same as the key or property of the object. See the example below.

```
const rectangle = {
  width: 20,
  height: 10,
  area: 200
}
let { width, height, area, perimeter } = rectangle

console.log(width, height, area, perimeter)
```

```
20 10 200 undefined
```

## Renaming during structuring

```
const rectangle = {
  width: 20,
```

```
  height: 10,
  area: 200
}
let { width: w, height: h, area: a, perimeter: p } = rectangle

console.log(w, h, a, p)



20 10 200 undefined
```

If the key is not found in the object the variable will be assigned to undefined. Sometimes the key might not be in the object, in that case we can give a default value during declaration. See the example.

```
const rectangle = {
  width: 20,
  height: 10,
  area: 200
}
let { width, height, area, perimeter = 60 } = rectangle

console.log(width, height, area, perimeter) //20 10 200 60
//Let us modify the object:width to 30 and perimeter to 80



const rectangle = {
  width: 30,
  height: 10,
  area: 200,
  perimeter: 80
}
let { width, height, area, perimeter = 60 } = rectangle
console.log(width, height, area, perimeter) //30 10 200 80
```

Destructuring keys as a function parameters. Let us create a function which takes a rectangle object and it returns a perimeter of a rectangle.

## Object parameter without destructuring

```
// Without destructuring
const rect = {
  width: 20,
  height: 10
}
const calculatePerimeter = rectangle => {
  return 2 * (rectangle.width + rectangle.height)
}
```

```
console.log(calculatePerimeter(rect)) // 60
//with destructuring


//Another Example
const person = {
  firstName: 'Asabeneh',
  lastName: 'Yetayeh',
  age: 250,
  country: 'Finland',
  job: 'Instructor and Developer',
  skills: [
    'HTML',
    'CSS',
    'JavaScript',
    'React',
    'Redux',
    'Node',
    'MongoDB',
    'Python',
    'D3.js'
  ],
  languages: ['Amharic', 'English', 'Suomi(Finnish)']
}
// Let us create a function which give information about the person object without destructuri

const getPersonInfo = obj => {
  const skills = obj.skills
  const formattedSkills = skills.slice(0, -1).join(', ')
  const languages = obj.languages
  const formattedLanguages = languages.slice(0, -1).join(', ')

  personInfo = `${obj.firstName} ${obj.lastName} lives in ${obj.country}. He is  ${
    obj.age
  } years old. He is an ${obj.job}. He teaches ${formattedSkills} and ${
    skills[skills.length - 1]
  }. He speaks ${formattedLanguages} and a little bit of ${languages[2]}.`

  return personInfo
}

console.log(getPersonInfo(person))
```

## Object parameter with destructuring

```
const calculatePerimeter = ({ width, height }) => {
  return 2 * (width + height)
}
```

```
console.log(calculatePerimeter(rect)) // 60


// Let us create a function which give information about the person object with destructuring
const getPersonInfo = ({
  firstName,
  lastName,
  age,
  country,
  job,
  skills,
  languages
}) => {
  const formattedSkills = skills.slice(0, -1).join(', ')
  const formattedLanguages = languages.slice(0, -1).join(', ')

  personInfo = `${firstName} ${lastName} lives in ${country}. He is ${age} years old. He is an
    skills[skills.length - 1]
  }. He speaks ${formattedLanguages} and a little bit of ${languages[2]}.`

  return personInfo
}
console.log(getPersonInfo(person))
/*
Asabeneh Yetayeh lives in Finland. He is  250 years old. He is an Instructor and Developer. He
*/
```

## Destructuring object during iteration

```
const todoList = [
{
  task:'Prepare JS Test',
  time:'4/1/2020 8:30',
  completed:true
},
{
  task:'Give JS Test',
  time:'4/1/2020 10:00',
  completed:false
},
{
  task:'Assess Test Result',
  time:'4/1/2020 1:00',
  completed:false
}
]

for (const {task, time, completed} of todoList){
```

```
  console.log(task, time, completed)
}


Prepare JS Test 4/1/2020 8:30 true
Give JS Test 4/1/2020 10:00 false
Assess Test Result 4/1/2020 1:00 false
```

## Spread or Rest Operator

When we destructure an array we use the spread operator(...) to get the rest elements as array. In addition to that we use spread operator to spread array elements to another array.

## Spread operator to get the rest of array elements

```
const nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
let [num1, num2, num3, ...rest] = nums

console.log(num1, num2, num3)
console.log(rest)


1 2 3
[4, 5, 6, 7, 8, 9, 10]


const countries = [
  'Germany',
  'France',
  'Belgium',
  'Finland',
  'Sweden',
  'Norway',
  'Denmark',
  'Iceland'
]

let [gem, fra, , ...nordicCountries] = countries

console.log(gem)
console.log(fra)
console.log(nordicCountries)


Germany
France
["Finland", "Sweden", "Norway", "Denmark", "Iceland"]
```

## Spread operator to copy array

```
const evens = [0, 2, 4, 6, 8, 10]
const evenNumbers = [...evens]

const odds = [1, 3, 5, 7, 9]
const oddNumbers = [...odds]

const wholeNumbers = [...evens, ...odds]

console.log(evenNumbers)
console.log(oddNumbers)
console.log(wholeNumbers)
```

```
[0, 2, 4, 6, 8, 10]
[1, 3, 5, 7, 9]
[0, 2, 4, 6, 8, 10, 1, 3, 5, 7, 9]
```

```
const frontEnd = ['HTML', 'CSS', 'JS', 'React']
const backEnd = ['Node', 'Express', 'MongoDB']
const fullStack = [...frontEnd, ...backEnd]

console.log(fullStack)
```

```
["HTML", "CSS", "JS", "React", "Node", "Express", "MongoDB"]
```

## Spread operator to copy object

We can copy an object using a spread operator

```
const user = {
  name:'Asabeneh',
  title:'Programmer',
  country:'Finland',
  city:'Helsinki'
}

const copiedUser = {...user}
console.log(copiedUser)
```

```
{name: "Asabeneh", title: "Programmer", country: "Finland", city: "Helsinki"}
```

Modifying or changing the object while copying

```
const user = {
  name:'Asabeneh',
  title:'Programmer',
  country:'Finland',
  city:'Helsinki'
}

const copiedUser = {...user, title:'instructor'}
console.log(copiedUser)
```

```
{name: "Asabeneh", title: "instructor", country: "Finland", city: "Helsinki"}
```

### Spread operator with arrow function

Whenever we like to write an arrow function which takes unlimited number of arguments we use a spread operator. If we use a spread operator as a parameter, the argument passed when we invoke a function will change to an array.

```
const sumAllNums = (...args) => {
  console.log(args)
}

sumAllNums(1, 2, 3, 4, 5)
```

```
[1, 2, 3, 4, 5]
```

```
const sumAllNums = (...args) => {
  let sum = 0
  for (const num of args){
    sum += num
  }
  return sum

}

console.log(sumAllNums(1, 2, 3, 4, 5))
```

```
15
```

🌝 You achieved quite a lot so far. Now, your level of JavaScript is upper intermediate. Keep going! You have just completed day 11 challenges and you are 11 steps a head in to your way to greatness. Now do some exercises for your brain and for your muscle.

# Exercises

## Exercises: Level 1

```
const constants = [2.72, 3.14, 9.81, 37, 100]
const countries = ['Finland', 'Estonia', 'Sweden', 'Denmark', 'Norway']
const rectangle = {
  width: 20,
  height: 10,
  area: 200,
  perimeter: 60
}
const users = [
{
  name:'Brook',
  scores:75,
  skills:['HTM', 'CSS', 'JS'],
  age:16
},
{
  name:'Alex',
  scores:80,
  skills:['HTM', 'CSS', 'JS'],
  age:18
},
{
  name:'David',
  scores:75,
  skills:['HTM', 'CSS'],
  age:22
},
{
  name:'John',
  scores:85,
  skills:['HTML'],
  age:25
},
{
  name:'Sara',
  scores:95,
  skills:['HTM', 'CSS', 'JS'],
```

```
    age: 26
  },
  {
    name:'Martha',
    scores:80,
    skills:['HTM', 'CSS', 'JS'],
    age:18
  },
  {
    name:'Thomas',
    scores:90,
    skills:['HTM', 'CSS', 'JS'],
    age:20
  }
  ]
```

1. Destructure and assign the elements of constants array to e, pi, gravity, humanBodyTemp, waterBoilingTemp.
2. Destructure and assign the elements of countries array to fin, est, sw, den, nor
3. Destructure the rectangle object by its properties or keys.

## Exercises: Level 2

1. Iterate through the users array and get all the keys of the object using destructuring
2. Find the persons who have less than two skills

## Exercises: Level 3

1. Destructure the countries object print name, capital, population and languages of all countries
2. A junior developer structure student name, skills and score in array of arrays which may not easy to read. Destructure the following array name to name, skills array to skills, scores array to scores, JavaScript score to jsScore and React score to reactScore variable in one line.

```
const student = ['David', ['HTM', 'CSS', 'JS', 'React'], [98, 85, 90, 95]]
console.log(name, skills, jsScore, reactScore)
```

```
David (4) ["HTM", "CSS", "JS", "React"] 90 95
```

3. Write a function called *convertArrayToObject* which can convert the array to a structure object.

```
  const students = [
      ['David', ['HTM', 'CSS', 'JS', 'React'], [98, 85, 90, 95]],
      ['John', ['HTM', 'CSS', 'JS', 'React'], [85, 80, 85, 80]]
    ]
```

```
console.log(convertArrayToObject(students))
[
  {
    name: 'David',
    skills: ['HTM','CSS','JS','React'],
    scores: [98,85,90,95]
  },
  {
    name: 'John',
    skills: ['HTM','CSS','JS','React'],
    scores: [85, 80,85,80]
  }
]
```

4. Copy the student object to newStudent without mutating the original object. In the new object add the following ?

- Add Bootstrap with level 8 to the front end skill sets
- Add Express with level 9 to the back end skill sets
- Add SQL with level 8 to the data base skill sets
- Add SQL without level to the data science skill sets

```
const student = {
  name: 'David',
  age: 25,
  skills: {
    frontEnd: [
      { skill: 'HTML', level: 10 },
      { skill: 'CSS', level: 8 },
      { skill: 'JS', level: 8 },
      { skill: 'React', level: 9 }
    ],
    backEnd: [
      { skill: 'Node',level: 7 },
      { skill: 'GraphQL', level: 8 },
    ],
    dataBase:[
      { skill: 'MongoDB', level: 7.5 },
    ],
    dataScience:['Python', 'R', 'D3.js']
  }
}
```

The copied object output should look like this:

```
{
name: 'David',
age: 25,
```

```
    skills: {
      frontEnd: [
        {skill: 'HTML',level: 10},
        {skill: 'CSS',level: 8},
        {skill: 'JS',level: 8},
        {skill: 'React',level: 9},
        {skill: 'BootStrap',level: 8}
      ],
      backEnd: [
        {skill: 'Node',level: 7},
        {skill: 'GraphQL',level: 8},
        {skill: 'Express',level: 9}
      ],
      dataBase: [
        { skill: 'MongoDB',level: 7.5},
        { skill: 'SQL',level: 8}
      ],
      dataScience: ['Python','R','D3.js','SQL']
    }
  }
```

🎉 CONGRATULATIONS ! 🎉

<< Day 10 | Day 12 >>