Day 11 - Destructuring and Spreading

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 = ['Cinthia', 'Chris', 'Alex']
let [oldest, middle, youngest] = names
console.log(middle, youngest, oldest)
// Chris Alex Cinthia
const scientificConstants = [2.72, 3.14, 9.81, 98.6, 100]
let [e, pi, gravity, bodyTemp, boilingTemp] = scientificConstants
console.log(e, pi, gravity, bodyTemp, boilingTemp)
// 2.72 3.14 9.81 98.6 100

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"]
```

We can skip a value by adding an additional comma:

```
const numbers = [1, 2, 3]
let [numOne,,numThree] = numbers
console.log(numOne, numThree)
// 1 3

const names = ['Cinthia', 'Chris', 'Alex']
let [, middle,] = names
console.log(middle)
// Chris
```

We can use default values in case a value of an array for that index is undefined:

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

console.log(firstPerson, secondPerson, thirdPerson, fourthPerson)
// Chris Brook David John
```

If there is a lot of elements in an array, we can destructure the first few and get the rest as an array using the spread operator (…).

```
const nums = [1,2,3,4,5,6,7,8,9,10]
let [num1, num2, num3, num4, ...rest] = nums
console.log(num1, num2, num3, num4) // 1 2 3 4
console.log(rest) // [4,5,6,7,8,9,10]
```

Destructuring during iteration

```
const states = [['Arizona','Phoenix'], ['Alaska','Juneau'], ['California', 'Sacramento']]
for (const [state, capital] of states) {
  console.log(state, city)
}
```

Destructuring Object

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

```
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 destructuring

```
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 a key isn't found in the object the variable will be assigned to undefined. In the case a key isn't in the object, we can give a default value during declaration.

```
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
```

Modifying the object:

```
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
```

We can create a function that takes a rectangle object and returns the perimeter of a rectangle

Object parameter with destructuring

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

```
}
console.log(calculatePerimeter(rect))

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

  personInfo = `${firstName} ${lastName} lives in ${country}. He is ${age} years old. He is an ${job}. He teaches ${formattedSkills} and ${
  return personInfo
}
```

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 of the elements as an array. Additionally, we can use the 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) // 1 2 3
console.log(rest) // [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) // Germany
console.log(fra) // France
console.log(nordicCountries) //["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) // [0, 2, 4, 6, 8, 10]
console.log(oddNumbers) // [1, 3, 5, 7, 9]
console.log(wholeNumbers) // [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

```
const user = {
  name:'Chris',
  title:'Programmer',
  country:'USA',
  city:'San Tan Valley'
}

const copiedUser = {...user}
console.log(copiedUser)
// {name: "Chris", title: "Programmer", country: "USA", city: "San Tan Valley"}
```

Modifying or changing the object while copying

```
const user = {
  name:'Chris',
  title:'Programmer',
  country:'USA',
  state:'Arizona'
}

const copiedUser = {...user, title:'student'}
console.log(copiedUser)
// {name: 'Chris', title:'student', country:'USA', state:'Arizona'}
```

Spread operator with arrow function

Whenever we want to write an arrow function that takes an unlimited number of arguments, we use a spread operator. When a spread operator is used as a parameter, the argument passed when the function is invoked is changed into 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
```