Front-end development: Methods

Javascript - Destructure

The following exercise contains the following subjects:

- Array and Object Destructure

1. What does the following code return/print?

```
let numbers = [10, 20, 30];
[numbers[1], numbers[2]] = [numbers[2], numbers[1]]

console.log(numbers) // ?
```

2. Refactor the following exercise

```
var obj = {
  numbers: {
    a: 1,
    b: 2
  }
};

var a = obj.numbers.a;
var b = obj.numbers.b;
```

3. Swap with array destructuring:

```
var arr = [1, 2];
var temp = arr[0];
arr[0] = arr[1];
arr[1] = temp;
```

4. Destructure an object with computed property names:

```
const propName = 'name';
const person = {
  [propName]: 'John',
  age: 30
};
```

5. Destructure a nested object with computed property names:

```
const propName = 'address';
const person = {
  name: 'John',
  age: 30,
  [propName]: {
    street: '123 Main St',
    city: 'Anytown',
    state: 'CA'
  }
};
```

6. Destructure a complex object with default values, renaming, and computed property names:

```
const propName = 'address';
const person = {
  firstName: 'John',
  lastName: 'Doe',
  [propName]: {
    street: '123 Main St',
    city: 'Anytown'
  }
};
```

7. Destructure an object with deeply nested properties and a function:

```
const person = {
  name: 'John',
  age: 30,
  address: {
    street: '123 Main St',
    city: 'Anytown',
    state: 'CA',
    coordinates: {
      latitude: 37.7749,
      longitude: -122.4194
    }
  },
  getFullName() {
    return `${this.name} Doe`;
  }};
```

8. Destructure an array of complex objects with renamed properties and default values:

```
const people = [
  { firstName: 'John', lastName: 'Doe', age: 30 },
  { firstName: 'Jane', lastName: 'Doe' }
];
```

9. Destructure an object with computed property names and a function:

```
const propName = 'person';
const obj = {
  [propName]: {
    name: 'John',
    age: 30,
    getAddress() {
      return {
        street: '123 Main St',
        city: 'Anytown',
        state: 'CA'
      };
    }
  }
};
```