Grace_Xu Homework 4 Js basic 2

How to Traverse an Array

For Loop: Traditional way to iterate through an array.

```
const arr = [1, 2, 3];
for (let i = 0; i < arr.length; i++) {
   console.log(arr[i]);
}
```

1. forEach Method: Executes a callback for each array element.

   ```
   arr.forEach(element => console.log(element));
   ```

2. for...of Loop: Iterates over values in the array.

   ```
   for (const element of arr) {
   console.log(element);
   }
   ```

What Does Array.prototype.map Do?

- Purpose: Creates a new array by applying a callback function to each element.

```
const numbers = [1, 2, 3];
const doubled = numbers.map(num => num * 2);
console.log(doubled); // [2, 4, 6]
```

- Returns a new array, doesn't modify the original.

What is Destructuring?

- Destructuring extracts values from arrays or properties from objects into distinct variables.

Example:

Array Destructuring:

```
const arr = [1, 2, 3];
const [a, b] = arr;
console.log(a, b); // 1, 2
```

Object Destructuring:

```
const obj = { name: "John", age: 25 };
const { name, age } = obj;
console.log(name, age); // "John", 25
```

Bonus: Renaming While Destructuring

```
const obj = { name: “John", age: 25 };
const { name: fullName, age: userAge } = obj;
console.log(fullName, userAge); // "Grace", 25
```

Difference Between Array and Object Destructuring

Array: Uses positional assignment.

```
const arr = [1, 2, 3];
const [first, second] = arr;
```

1. Object: Uses property names, order doesn’t matter.

   ```
   const obj = { name: "John", age: 25 };

   const { age, name } = obj;
   ```

2. ## Rest Operator

The rest operator (...) collects remaining items into a single variable.

Array:

```
const [first, ...rest] = [1, 2, 3, 4];
console.log(rest); // [2, 3, 4]
```

Object:

```
const { name, ...other } = { name: "John", age: 25, city: "NY" };
console.log(other); // { age: 25, city: "NY" }
```

What is Shallow vs Deep Copy?

1. Shallow Copy:
    - Copies only the first level of an object/array.
    - Changes in nested objects reflect in the copy.

Example:

```
const original = { name:John", details: { age: 25 } };
const shallowCopy = { ...original };
shallowCopy.details.age = 30;
console.log(original.details.age); // 30
```

2. Deep Copy:
    - Copies all levels, creating independent copies of nested objects.

Example:

```
const original = { name: "Grace", details: { lastname: Xu } };
const deepCopy = JSON.parse(JSON.stringify(original));
deepCopy.details.lastname = Xu;
console.log(original.details.lastname); // Xu
```

3. How to Perform Shallow and Deep Copies:

- Shallow Copy:

Use the spread operator:

```
const shallowCopy = { ...originalObject };
const shallowArrayCopy = [...originalArray];
```

- Deep Copy:

Use JSON.stringify and JSON.parse (works for JSON-safe objects):

```
const deepCopy = JSON.parse(JSON.stringify(originalObject));
```

Use libraries like Lodash:

```
const _ = require('lodash');
```

    - const deepCopy = _.cloneDeep(originalObject);

Summary of Note:

Summary Notes: JavaScript Basics

Traversing an Array

1. For Loop:
    - Traditional way to iterate through an array.

```
for (let i = 0; i < arr.length; i++) {
   console.log(arr[i]);
}
```

2. forEach Method:
    - Executes a callback for each element.

```
arr.forEach(element => console.log(element));
```

3. for...of Loop:
    - Iterates over values in the array.

```
for (const element of arr) {
   console.log(element);
}
```

4. Array.prototype.map

- Purpose: Creates a new array by applying a callback function to each element.

```
const numbers = [1, 2, 3];
const doubled = numbers.map(num => num * 2);
console.log(doubled); // [2, 4, 6]
```

- Key Points:
    - Returns a new array.
    - Does not modify the original array.

Destructuring

1. Array Destructuring:
    - Extracts values from an array into variables.

```
const [a, b] = [1, 2, 3];

console.log(a, b); // 1, 2
```

2. Object Destructuring:
    - Extracts properties from an object into variables.

```
const { name, age } = { name: "John", age: 25 };

console.log(name, age); // "John", 25
```

3. Renaming While Destructuring:

```
const { name: fullName, age: userAge } = { name: "John", age: 25 };

console.log(fullName, userAge); // "John", 25
```

4. Difference Between Array and Object Destructuring

1. Array Destructuring:
    - Uses positional assignment.

```
const [first, second] = [1, 2, 3];
```

2. Object Destructuring:
    - Uses property names; order doesn't matter.

```
const { age, name } = { name: "John", age: 25 };
```

3. Rest Operator

Collects remaining items into a single variable.

With Arrays:

```
const [first, ...rest] = [1, 2, 3, 4];
```

```
console.log(rest); // [2, 3, 4]
```

1. With Objects:

```
const { name, ...other } = { name: "John", age: 25, city: "NY" };

console.log(other); // { age: 25, city: "NY" }
```

2. Shallow vs Deep Copy

1. Shallow Copy:
    - Copies only the first level of an object/array.
    - Changes in nested objects reflect in the copy.
      const shallowCopy = { ...original };

2. Deep Copy:
    - Creates independent copies of all nested objects.
      const deepCopy = JSON.parse(JSON.stringify(original));

3. How to Perform Copies

1. Shallow Copy:

Use the spread operator:
javascript
Copy code
```
const shallowCopy = { ...originalObject };

const shallowArrayCopy = [...originalArray];
```

2. Deep Copy:

Use JSON.stringify and JSON.parse:

```
const deepCopy = JSON.parse(JSON.stringify(originalObject));
```

Use libraries like Lodash:

```
const _ = require('lodash');
```

    - const deepCopy = _.cloneDeep(originalObject);