1) what are Higher Order function

Higher Order function (HOF)s are function that take one or more function as arguments or return a function as their result

In JavaScript HOF are a powerful tool for functional programming because they allow you to abstract away repetitive task and create more flexible and reusable code

Taking function as Arguments

A function can accept another function as an argument to customize its behaviour

Returning function

A function can return another function creating a more flexible and modular design

2. Differences b/w map, forEach, reduce, filter, sort

map
creates a new array by applying a provided function to each element of the original array
Use map when you need to transform the elements of an array.

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

o/p - (2, 4, 6, 8)

forEach
~~what~~ + Executes the provided function once for each array element Does not return a value
when to use : forEach when you need to iterate over an array and perform side effects but don't need to return

a new array

```
const numbers = [1,2,3];
numbers.forEach(x => console.log(x))
```

o/p → 1 2 3

reduce

Applies a function against an accumulator and each element to reduce it to a single value

Use reduce when you need to aggregate or accumulate values into a single result

```
const numbers = [1, 2, 3, 4];
const sum = numbers.reduce((acc, x) =>
    acc + x, 0);
console.log(sum);
```

Filter
Creates a new array with all elements that pass the test implemented by the provided function
Use filter when you want to select certain elements from an array based on a condition.

Sort
Sort the elements of an array in a place and return the sorted array
Use sort when you need to order the elements in an array

3 Readability
HOFs like map filter reduce abstract away the need for explicit loops
This results in cleaner, more readable code.
They allow you to express operations on arrays in a declarative way focusing on what you want to achieve instead

of how to achieve it.

Reusability
HOFs encourage reusable functions: you can define smaller reusable functions that can be passed to different higher order functions.

4) Real world Scenario Examples
Transforming Data in an E commerce platform

```
const product = [
    { name: "Laptop", price: 1000 },
    { name: "phone", price: 500 }
    { name: "Tablet", price: 300 }
];
const discountedPrices = products.map(product
    => {{ ({ ...product, price:
    product.price * 0.9
}));
```

6

```
const total = discountedPrices.reduce((acc,
    product) => acc + product.price, 0);
console.log(total);
```