Basic Syntax Typescript

Variables:

```
let variableName: string = "value";
const constantName: string = "value"; // for constants
```

Data Types:

```
type ObjectType = {
  name: string;
};

let number: number = 10;
let string: string = "Hello, World!";
let boolean: boolean = true;
let array: number[] = [1, 2, 3];
let object: ObjectType = { key: "value" };
```

Functions:

```
function functionName(parameter1, parameter2) {
  // Function body
  return someValue;
}
```

Loops:

```
for (let i = 0; i < 5; i++) {
  // Loop body
}

while (condition) {
  // Loop body
}
```

Conditional Statements:

```
if (condition) {
  // Code block to execute if condition is true
} else {
```

```
  // Code block to execute if condition is false
}
```

Basic string manipulation

1. Concatenation:

```
let str1: string = "Hello";
let str2: string = "World";
let result: string = str1.concat(", ", str2); // Result: "Hello, World"
```

2. Length:

```
let message: string = "This is a message.";
let length: number = message.length; // Result: 18
```

3. Substring:

```
let text: string = "TypeScript";
let part: string = text.substring(0, 4); // Result: "Type"
```

4. Uppercase & Lowercase:

```
let lowerCase: string = "hello".toLowerCase(); // Result: "hello"
let upperCase: string = "world".toUpperCase(); // Result: "WORLD"
```

5. Replace:

```
let sentence: string = "I like apples.";
let newSentence: string = sentence.replace("apples", "oranges"); // Result: "I
like oranges."
```

6. Template Literals:

```
let name: string = "John";
let greeting: string = `Hello, ${name}!`; // Result: "Hello, John!"
```

7. Trim:

```
let paddedString: string = "   Trim me!   ";
let trimmedString: string = paddedString.trim(); // Result: "Trim me!"
```

Basic array manipulation

Certainly! TypeScript, being a superset of JavaScript, inherits all the array manipulation functions that JavaScript provides. Here are some examples:

1. Map:

```
let numbers: number[] = [1, 2, 3, 4, 5];
let squaredNumbers: number[] = numbers.map((num) => num * num); // Result: [1, 4, 9, 16, 25]
```

2. ForEach:

```
let names: string[] = ["Alice", "Bob", "Charlie"];
names.forEach((name) => {
  console.log(name);
});
// Output: Alice, Bob, Charlie (printed to the console)
```

3. Find:

```
let numbers: number[] = [10, 20, 30, 40, 50];
let foundNumber: number | undefined = numbers.find((num) => num > 25); // Result: 30
```

4. Filter:

```
let ages: number[] = [25, 30, 18, 40, 16];
let adults: number[] = ages.filter((age) => age >= 18); // Result: [25, 30, 18, 40]
```

5. Reduce:

```
let numbers: number[] = [1, 2, 3, 4, 5];
let sum: number = numbers.reduce((acc, curr) => acc + curr, 0); // Result: 15
```

6. Slice:

```
let arr: number[] = [1, 2, 3, 4, 5];
let sliced: number[] = arr.slice(2); // Result: [3, 4, 5]
```

7. Concat:

```
let arr1: number[] = [1, 2];
let arr2: number[] = [3, 4];
let merged: number[] = arr1.concat(arr2); // Result: [1, 2, 3, 4]
```

8. Filter and Map Combined:

```
let numbers: number[] = [1, 2, 3, 4, 5];
let squaredEvenNumbers: number[] = numbers
  .filter((num) => num % 2 === 0)
  .map((num) => num * num); // Result: [4, 16]
```

Basic object manipulation

1. Object.keys():

```
let car = { brand: "Toyota", model: "Corolla", year: 2022 };
let keys: string[] = Object.keys(car); // Result: ['brand', 'model', 'year']
```

2. Object.values():

```
let car = { brand: "Toyota", model: "Corolla", year: 2022 };
let values: any[] = Object.values(car); // Result: ['Toyota', 'Corolla', 2022]
```

3. Object.entries():

```
let car = { brand: "Toyota", model: "Corolla", year: 2022 };
let entries: [string, any][] = Object.entries(car); // Result: [['brand',
'Toyota'], ['model', 'Corolla'], ['year', 2022]]
```

4. Object.assign():

```
let obj1 = { a: 1, b: 2 };
let obj2 = { b: 3, c: 4 };
let mergedObj: { [key: string]: number } = Object.assign({}, obj1, obj2); //
Result: { a: 1, b: 3, c: 4 }
```

5. Object.hasOwnProperty():

```
let person = { name: "Alice", age: 25 };
let hasName: boolean = person.hasOwnProperty("name"); // Result: true
```

6. Object.fromEntries():

```
let entries: [string, number][] = [
  ["a", 1],
  ["b", 2],
  ["c", 3],
];
let obj: { [key: string]: number } = Object.fromEntries(entries); // Result: { a:
1, b: 2, c: 3 }
```

7. Mapping over Object Properties:

```
let car = { brand: "Toyota", model: "Corolla", year: 2022 };
let uppercasedCar = Object.fromEntries(
  Object.entries(car).map(([key, value]) => [key, String(value).toUpperCase()])
);
// Result: { brand: 'TOYOTA', model: 'COROLLA', year: '2022' }
```

Basic json manipulation

1. Parsing JSON:

`JSON.parse()`

```
let jsonString: string = '{"name": "John", "age": 30}';
let obj: { name: string; age: number } = JSON.parse(jsonString);
// Result: obj is { name: 'John', age: 30 }
```

2. Stringifying JavaScript Object:

`JSON.stringify()`

```
let person: { name: string; age: number } = { name: "Alice", age: 25 };
let jsonString: string = JSON.stringify(person);
// Result: jsonString is '{"name":"Alice","age":25}'
```

Higher order function

1. map:

```
let numbers: number[] = [1, 2, 3, 4, 5];

// Using map to transform each element
let squaredNumbers: number[] = numbers.map((num) => num * num);
// Result: [1, 4, 9, 16, 25]
```

2. filter:

```
let ages: number[] = [25, 30, 18, 40, 16];

// Using filter to get elements that satisfy a condition
let adults: number[] = ages.filter((age) => age >= 18);
// Result: [25, 30, 18, 40]
```

3. reduce:

```
let numbers: number[] = [1, 2, 3, 4, 5];

// Using reduce to calculate the sum of elements
let sum: number = numbers.reduce((acc, curr) => acc + curr, 0);
// Result: 15
```

4. forEach:

```
let names: string[] = ["Alice", "Bob", "Charlie"];

// Using forEach to perform an action on each element
names.forEach((name) => {
  console.log(name);
});
// Output: Alice, Bob, Charlie (printed to the console)
```

5. Combining Higher-Order Functions:

```
let numbers: number[] = [1, 2, 3, 4, 5];

// Using filter and map together
let squaredEvenNumbers: number[] = numbers
  .filter((num) => num % 2 === 0)
  .map((num) => num * num);
// Result: [4, 16]
```