Here's an overview of TypeScript arrays, including syntax, methods, and types:

1. Array Syntax:
In TypeScript, arrays can be defined using square brackets `[]`. Arrays can hold elements of the same or different types.

```typescript
// Define an array of numbers
let numbers: number[] = [1, 2, 3, 4, 5];

// Define an array of strings
let fruits: string[] = ["apple", "banana", "orange"];

// Define an array of mixed types
let mixedArray: (number | string)[] = [1, "hello", 3, "world"];
```

2. Array Methods:
Arrays in TypeScript come with built-in methods to manipulate and interact with their elements. Some common array methods include `push`, `pop`, `shift`, `unshift`, `slice`, `splice`, `concat`, `indexOf`, `forEach`, `map`, `filter`, `reduce`, etc.

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

// Add an element to the end of the array
numbers.push(6);

// Remove and return the last element of the array
let lastElement = numbers.pop();

// Remove and return the first element of the array
let firstElement = numbers.shift();

// Add an element to the beginning of the array
numbers.unshift(0);

// Remove elements from the array and/or insert new elements
numbers.splice(2, 1); // Removes one element at index 2
```

3. Array Types:
In TypeScript, you can specify the type of elements that an array can hold. This helps in catching type errors during development.

```typescript
// Array of numbers
let numbers: number[] = [1, 2, 3, 4, 5];

// Array of strings
let fruits: string[] = ["apple", "banana", "orange"];

// Array of mixed types
let mixedArray: (number | string)[] = [1, "hello", 3, "world"];

// Array of custom types
interface Person {
    name: string;
    age: number;
}

let people: Person[] = [
    { name: "Alice", age: 30 },
    { name: "Bob", age: 25 },
    { name: "Charlie", age: 35 }
];
```

Using arrays in TypeScript allows for type-safe manipulation and iteration over collections of elements, enhancing code readability and maintainability.