The main difference between TypeScript and JavaScript is the inclusion of static typing in TypeScript. Here are some key differences in syntax:

1. Static Typing:

TypeScript supports static typing, allowing developers to define variable types. For instance:

```
let age: number = 25;
let name: string = "John";
```

2. Interfaces and Type Annotations:

TypeScript allows the use of interfaces and type annotations for objects, enabling better code documentation and type checking:

```
interface Person {
  name: string;
  age: number;
}

function greet(person: Person) {
  console.log(`Hello, ${person.name}`);
}
```

3. Compiled Language:

TypeScript needs to be compiled to JavaScript before it can be executed in a browser or a JavaScript runtime. The TypeScript compiler translates TypeScript code into JavaScript code.

4. ES6+ Features:

TypeScript includes all the features of ECMAScript 6 (ES6) and later versions. So, any JavaScript code that is compliant with ES6 will work in TypeScript.

5. Optional Parameters and Default Values:

TypeScript allows defining function parameters as optional and specifying default values:

```
function greet(name: string, message: string = "Hello") {
  console.log(`${message}, ${name}`);
}
```

6. Union and Intersection Types:

TypeScript provides features like union types (a value can be of multiple types) and intersection types (combining multiple types):

```
let val: string | number; // Union type
type First = { a: number };
type Second = { b: number };
let result: First & Second; // Intersection type
```

These additional features in TypeScript offer more control, better error checking, and improved development experience by catching potential errors during development rather than runtime, thereby enhancing code quality and maintainability.