TypeScript and JavaScript are both popular programming languages used for web development, but they have some differences.

JavaScript:
1. JavaScript is a dynamic scripting language primarily used for front-end web development. It's supported by all major browsers.
2. It's dynamically typed, meaning you don't need to specify variable types explicitly.
3. JavaScript code is executed directly by the browser.
4. It's more prone to errors due to its dynamic nature, especially in large-scale applications.
5. It's flexible and allows for quick prototyping and development.

TypeScript:
1. TypeScript is a superset of JavaScript developed by Microsoft. It adds optional static typing to JavaScript.
2. It's statically typed, meaning you can specify variable types explicitly. This helps catch errors at compile time rather than runtime.
3. TypeScript code needs to be transpiled to JavaScript before running in the browser.
4. It offers better code maintainability and scalability, especially in large projects, due to its static typing.
5. It provides better tooling support, including features like code completion, refactoring, and navigation, thanks to its type system.

Example:
Sure, let's illustrate the differences between JavaScript and TypeScript with a simple example.

Let's say we have a function that calculates the area of a rectangle:

```javascript
// JavaScript code
function calculateArea(length, width) {
    return length * width;
}

console.log(calculateArea(5, 3)); // Output: 15
console.log(calculateArea("5", "3")); // Output: "53" (concatenation instead of multiplication)
```

In JavaScript, the `calculateArea` function takes two parameters, `length` and `width`, but it doesn't specify their types. This means you can pass any type of argument to the function, and JavaScript will try to perform the calculation. However, this can lead to unexpected behavior, such as when passing strings instead of numbers, as shown in the second `console.log` call.

Now let's rewrite the same function using TypeScript:

```typescript
// TypeScript code
function calculateArea(length: number, width: number): number {
    return length * width;
}

console.log(calculateArea(5, 3)); // Output: 15
// Error: Argument of type '"5"' is not assignable to parameter of type 'number'.
console.log(calculateArea("5", "3"));
```

In TypeScript, we've added type annotations to the function parameters (`length` and `width`) and to the return type of the function. This indicates that both parameters and the return value should be of type `number`. Now, if you try to pass a string as an argument, TypeScript will catch this error at compile time, preventing unexpected behavior at runtime.

So, the main difference here is that TypeScript provides static type checking, which helps catch errors early in the development process, while JavaScript relies on dynamic typing and may only catch errors at runtime.