In TypeScript, variables work similarly to variables in JavaScript, but with the added benefit of type annotations and type inference. Here's a breakdown of how variables work in TypeScript:

1. **Variable Declaration**:
   You can declare variables in TypeScript using the `let`, `const`, or `var` keywords.

   ```typescript
   let myVariable: number;
   const myConstant: string = "Hello";
   var anotherVariable: boolean = true;
   ```

   - `let`: Declares a mutable variable.
   - `const`: Declares an immutable variable (constant). Once assigned, its value cannot be changed.
   - `var`: Declares a variable with function-scoped or globally-scoped access (avoid using `var` because of its scoping quirks).

2. **Type Annotations**:
   TypeScript allows you to specify the type of a variable using type annotations. This helps in catching type-related errors during development.

   ```typescript
   let myNumber: number = 10;
   let myString: string = "Hello, TypeScript!";
   let myBoolean: boolean = true;
   ```

3. **Type Inference**:
   TypeScript can often infer the type of a variable based on its initialization value.

   ```typescript
   let inferredString = "This is a string"; // TypeScript infers the type as string
   let inferredNumber = 42; // TypeScript infers the type as number
   ```

4. **Dynamic Typing**:
   TypeScript allows dynamic typing just like JavaScript, where variables can hold values of different types.

   ```typescript
   let dynamicVariable: any;
   dynamicVariable = 42; // Now it's a number
   dynamicVariable = "TypeScript"; // Now it's a string
   ```

5. **Type Annotations vs Type Inference**:
   While TypeScript can infer types, it's often beneficial to specify types explicitly to improve code clarity and catch errors early.

6. **Type Guards and Assertion**:
   TypeScript provides mechanisms like type guards and type assertion to work with variables whose types may not be known at compile-time.

   ```typescript
   let maybeString: unknown = "Hello";
   if (typeof maybeString === "string") {
       // In this block, maybeString is treated as a string
       console.log(maybeString.toUpperCase());
   }

   let definitelyNumber: any = "42";
   let parsedNumber: number = parseInt(definitelyNumber); // Type assertion
   ```

7. **Null and Undefined**:
   TypeScript has special types `null` and `undefined` which can be assigned to variables.

   ```typescript
   let nullable: null = null;
   let undefinedValue: undefined = undefined;
   ```

   You can also use `strictNullChecks` option to enforce stricter null checks.