code examples for the features and benefits of TypeScript compared to JavaScript.

Features:

1. **Type Annotations**:
   ```typescript
   // Type annotations for variables and function arguments
   let message: string = "Hello, TypeScript!";

   function greet(name: string): void {
       console.log("Hello, " + name);
   }

   greet("Alice");
   ```

2. **Interfaces**:
   ```typescript
   interface Person {
       name: string;
       age: number;
   }

   function introduce(person: Person): void {
       console.log(`My name is ${person.name} and I am ${person.age} years old.`);
   }

   const alice: Person = { name: "Alice", age: 30 };
   introduce(alice);
   ```

3. **Classes**:
   ```typescript
   class Car {
       private model: string;
       protected speed: number;

       constructor(model: string, speed: number) {
           this.model = model;
           this.speed = speed;
       }

       public accelerate(): void {
           this.speed += 10;
   ```

            console.log(`${this.model} is accelerating. Speed: ${this.speed}`);
        }
    }

    const myCar = new Car("Toyota", 50);
    myCar.accelerate();

4. **Generics**:
    ```typescript
    // A generic function that works with arrays of any type
    function reverseArray<T>(arr: T[]): T[] {
        return arr.reverse();
    }

    const numbers = [1, 2, 3, 4];
    const reversedNumbers = reverseArray(numbers);
    console.log(reversedNumbers);

    const strings = ["a", "b", "c"];
    const reversedStrings = reverseArray(strings);
    console.log(reversedStrings);
    ```

5. **Type Inference**:
    ```typescript
    // TypeScript can infer the types of variables and function return types
    let inferredString = "This is a string";
    let inferredNumber = 42;

    function add(x: number, y: number) {
        return x + y; // TypeScript infers the return type as `number`
    }
    ```

6. **Advanced Type System**:
    ```typescript
    // Union types allow a variable to be one of several types
    let value: string | number;
    value = "Hello";
    value = 42;

    // Intersection types combine multiple types
    interface Bird {
    ```

~~~
        fly(): void;
    }

    interface Fish {
        swim(): void;
    }

    type FlyingFish = Bird & Fish;

    const flyingFish: FlyingFish = {
        fly: () => console.log("Flying"),
        swim: () => console.log("Swimming")
    };
    flyingFish.fly();
    flyingFish.swim();
    ```

### Benefits Compared to JavaScript:

1. **Error Prevention**:
    ```typescript
    // TypeScript catches errors at compile time
    function sum(a: number, b: number): number {
        return a + b;
    }

    // This would cause a compile-time error because 'c' is not a number
    // let c = "10";
    // sum(10, c);
    ```

2. **Enhanced Readability**:
    ```typescript
    // Type annotations help clarify the expected types
    function getUserInfo(name: string, age: number): string {
        return `${name} is ${age} years old.`;
    }

    const userInfo = getUserInfo("Alice", 30);
    console.log(userInfo);
    ```

3. **Refactoring**:
    ```typescript
~~~

```
// With type annotations, you can safely refactor code
class Person {
  name: string;

  constructor(name: string) {
    this.name = name;
  }
}

const person = new Person("Alice");

// If you rename 'name' to 'firstName', the compiler will find all uses of 'name' that need updating
```

4. **Interoperability**:
   ```typescript
   // You can use JavaScript libraries in TypeScript without any changes
   import * as _ from "lodash"; // JavaScript library

   const numbers = [1, 2, 3, 4, 5];
   const shuffledNumbers = _.shuffle(numbers); // Using lodash
   console.log(shuffledNumbers);
   ```

5. **Community and Ecosystem**:
   - TypeScript's popularity and support in modern IDEs provide a rich development experience.
   - The type system enables tooling such as auto-completion, refactoring, and type checking, leading to faster and more efficient development.

These examples highlight the core features and benefits of using TypeScript compared to plain JavaScript.