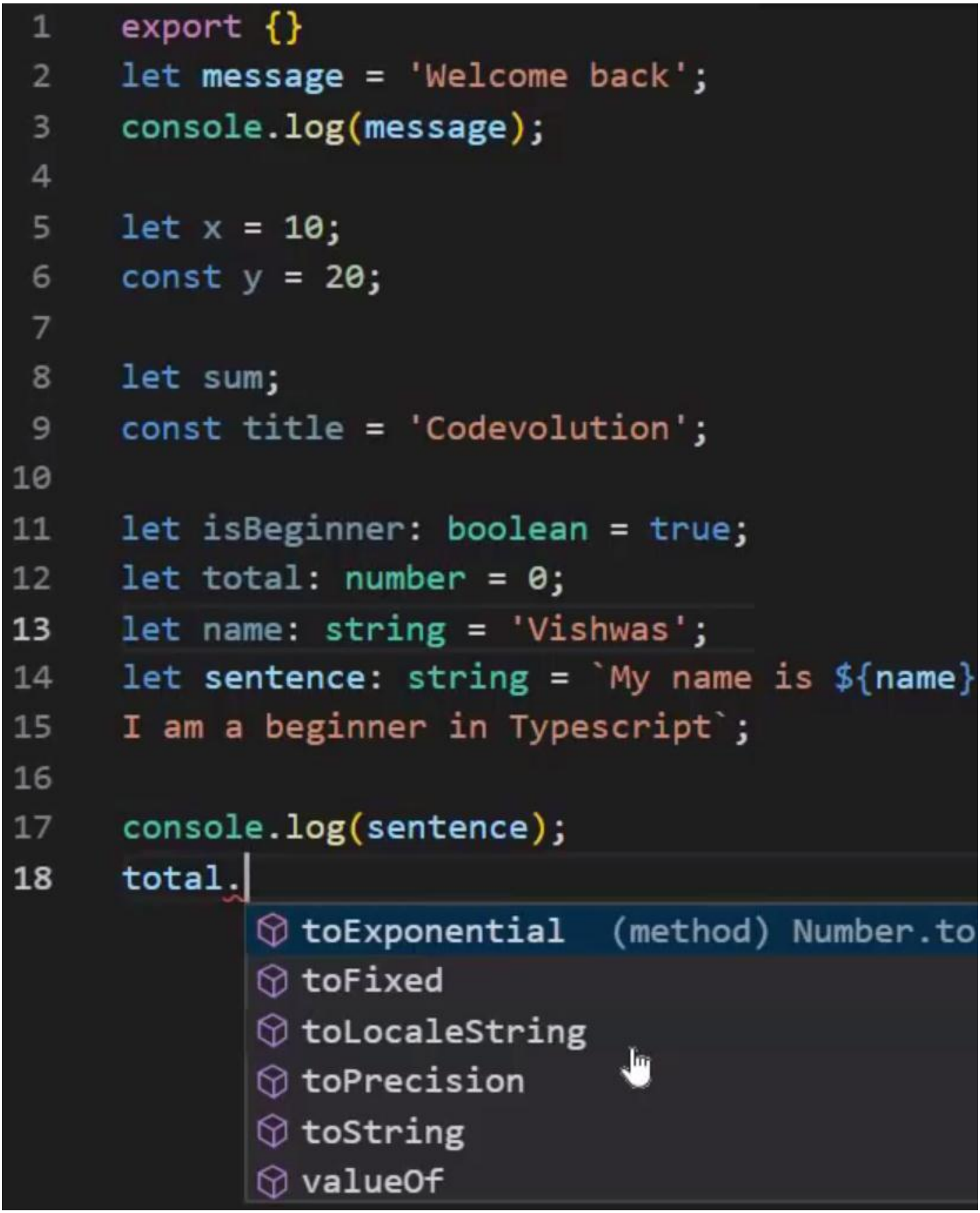
1. **Variable_DataType_TemplateString_Intellisense**



2. Variable_Null_Undefined_Array_Tuple_Enum_Any_Multitype

```
19    let n: null = null;
20    let u: undefined = undefined;
21
22    let isNew: boolean = null;
23    let myName: string = undefined;
24
25    let list1: number[] = [1,2,3];
26    let list2: Array<number> = [1,2,3];
27
28
29    let person1: [string, number] = ['Chris', 22];
30
31    enum Color {Red = 5, Green, Blue};
32
33    let c: Color = Color.Green;
34    console.log(c);
35
36    let randomValue: any = 10;
37    randomValue = true;
38    randomValue = 'Vishwas';
39
59    let multiType: number | boolean;
60    multiType = 20;
61    multiType = true;
```

3. Function_Basic

```
66  function add(num1: number, num2: number) {
67      return num1 + num2;
68  }
69  add(5, 10);
```

Function with Parameters with Data Type

```
66  function add(num1: number, num2: number): number {
67      return num1 + num2;
68  }
69  add(5, 10);
```

Function with Parameters with Data Type and Return Type

```
66  function add(num1: number, num2?: number): number {
67    if (num2)
68      return num1 + num2;
69    else
70      return num1;
71  }
72  add(5, 10);
73  add(5);
```

Function with Optional Parameters with Data Type and Return Type

```
66  function add(num1: number, num2: number = 10): number {
67    if (num2)
68      return num1 + num2;
69    else
70      return num1;
71  }
72  add(5, 10);
73  add(5);
```

Parameter Having Default Value, another way of making optional when calling

4. Function_WithObjectAndInterface

```
74  function fullName(person: {firstName: string, lastName: string}) {
75      console.log(`${person.firstName} ${person.lastName}`);
76  }
77
78  let p = {
79      firstName: 'Bruce',
80      lastName: 'Wayne'
81  };
82
83  fullName(p);
```

```
74  interface Person {
75      firstName: string;
76      lastName: string;
77  }
78
79  function fullName(person: Person) {
80      console.log(`${person.firstName} ${person.lastName}`);
81  }
82
83  let p = {
84      firstName: 'Bruce',
85      lastName: 'Wayne'
86  };
```

5. Class

```
90  class Employee {
91      employeeName: string;
92
93      constructor(name: string) {
94          this.employeeName = name;
95      }
96
97      greet() {
98          console.log(`Good Morning ${this.employeeName}`);
99      }
100 }
101
102 let emp1 = new Employee('Vishwas');
103 console.log(emp1.employeeName);
104 emp1.greet();
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: powersh

```
Bruce undefined
Vishwas
Good Morning Vishwas
```

6. Interface

```
106   class Manager extends Employee {
107       constructor(managerName: string) {
108           super(managerName);
109       }
110       delegateWork() {
111           console.log(`Manager delegating tasks`);
112       }
113   }
114
115   let m1 = new Manager('Bruce')
116   m1.delegateWork();
117   m1.greet();
118   console.log(m1.employeeName);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Manager delegating tasks
Good Morning Bruce
Bruce
PS E:\Codevolution\Course\Typescript>
```