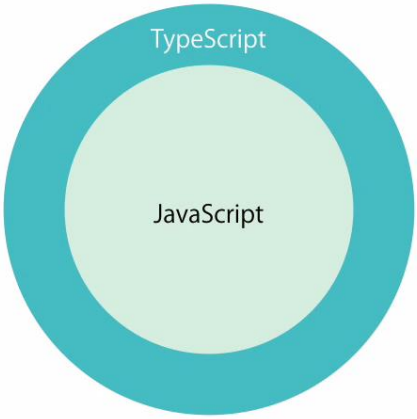
TypeScript Fundamentals

In this section...

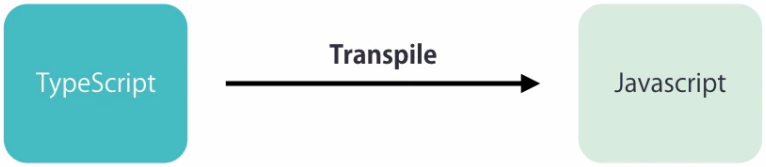
- Type annotations
- Arrow functions
- Interfaces
- Classes
- Constructors
- Access modifiers
- Properties
- Modules

What is TypeScript?


TypeScript
JavaScript

TypeScript

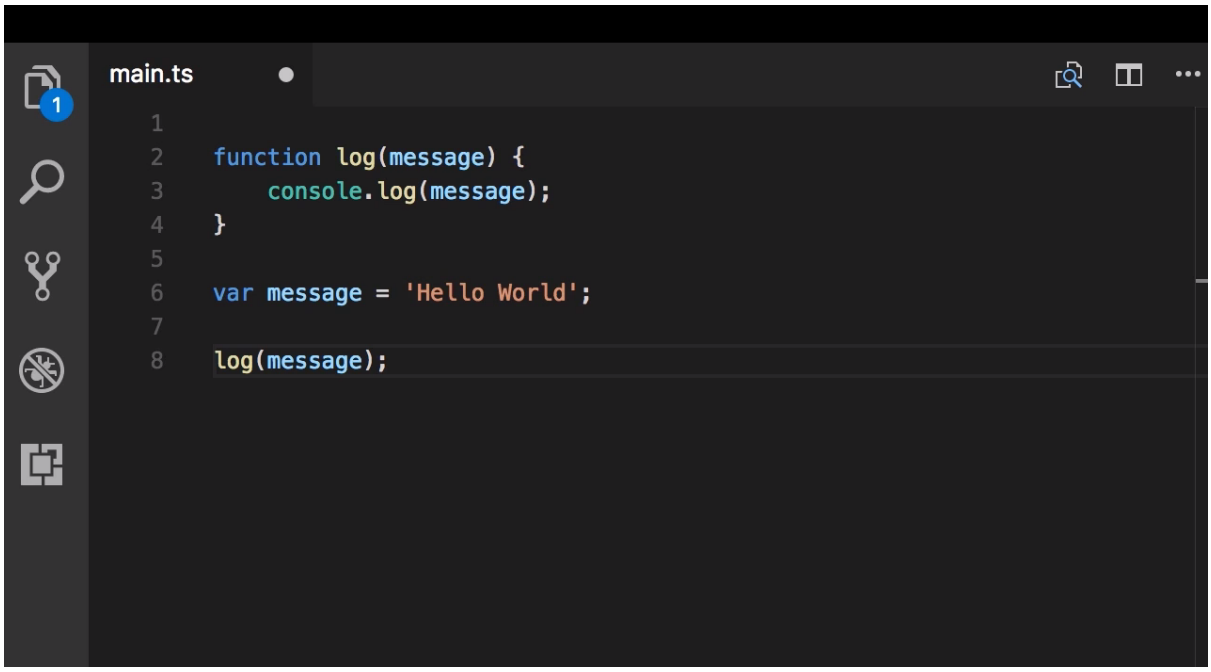
- Strong typing
- Object-oriented features
- Compile-time errors
- Great tooling

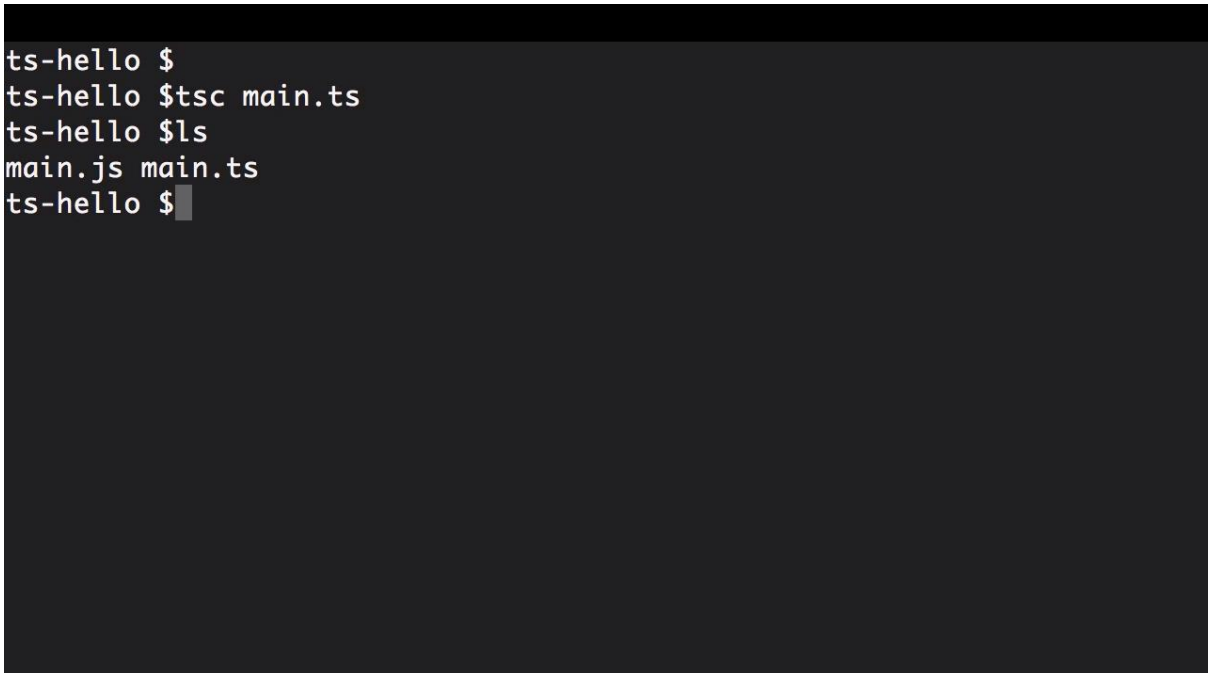

TypeScript
Transpile
Javascript

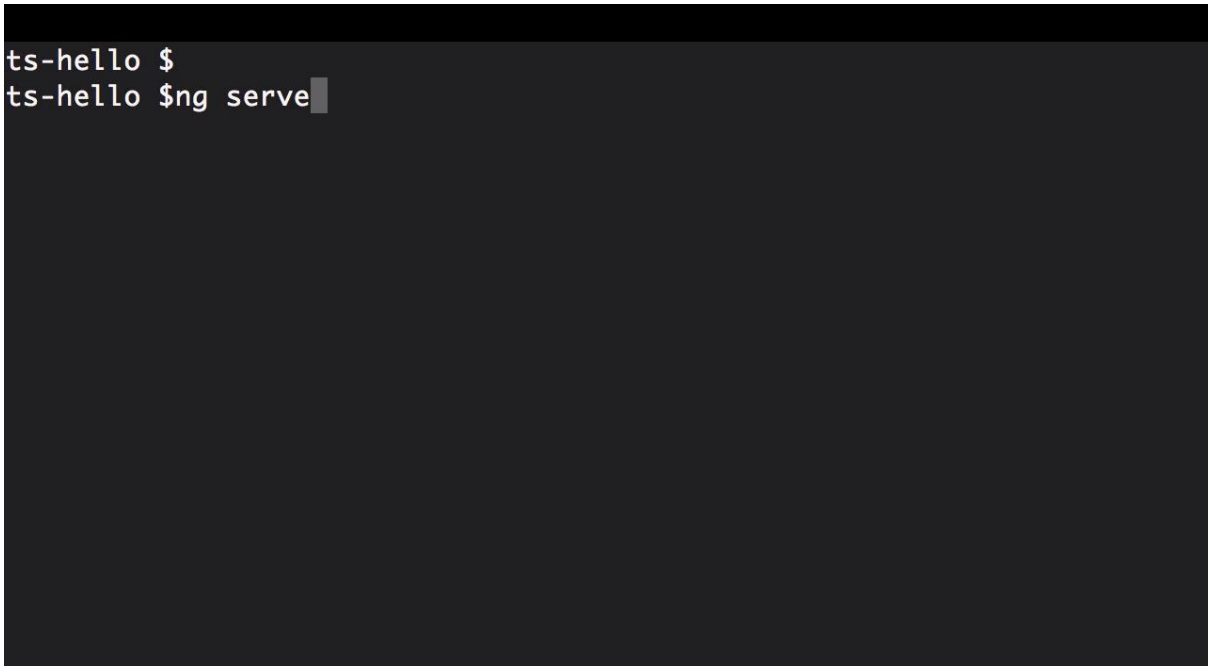
```
code $
code $sudo npm install -g typescript
Password:
/usr/local/bin/tsserver -> /usr/local/lib/node_modules/typescript/bin/t
sserver
/usr/local/bin/tsc -> /usr/local/lib/node_modules/typescript/bin/tsc
/usr/local/lib
└── typescript@2.3.4

code $
```

```
code $
code $tsc --version
Version 2.3.4
code $
```

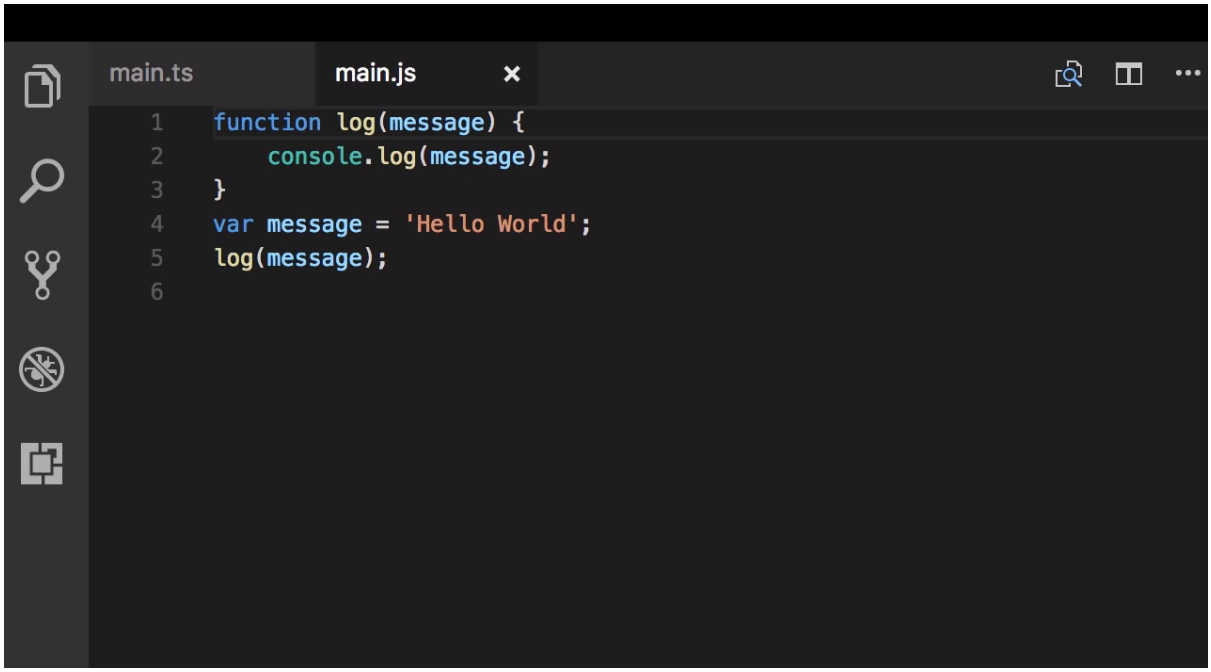
```
code $
code $mkdir ts-hello
code $cd ts-hello/
ts-hello $code main.ts
```

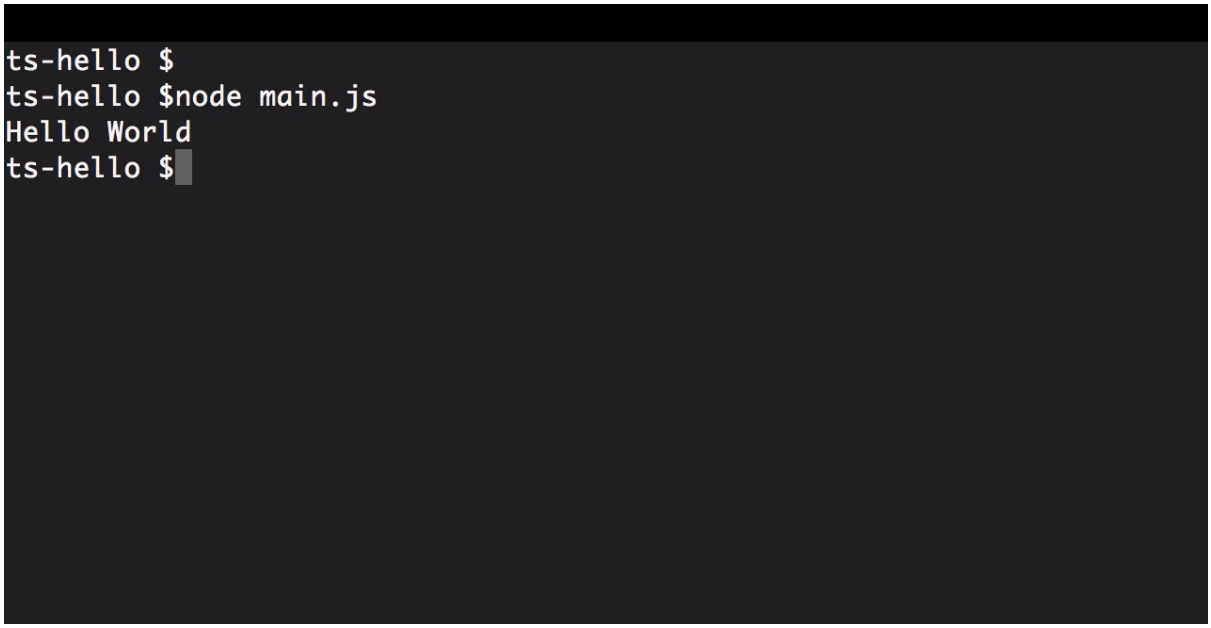




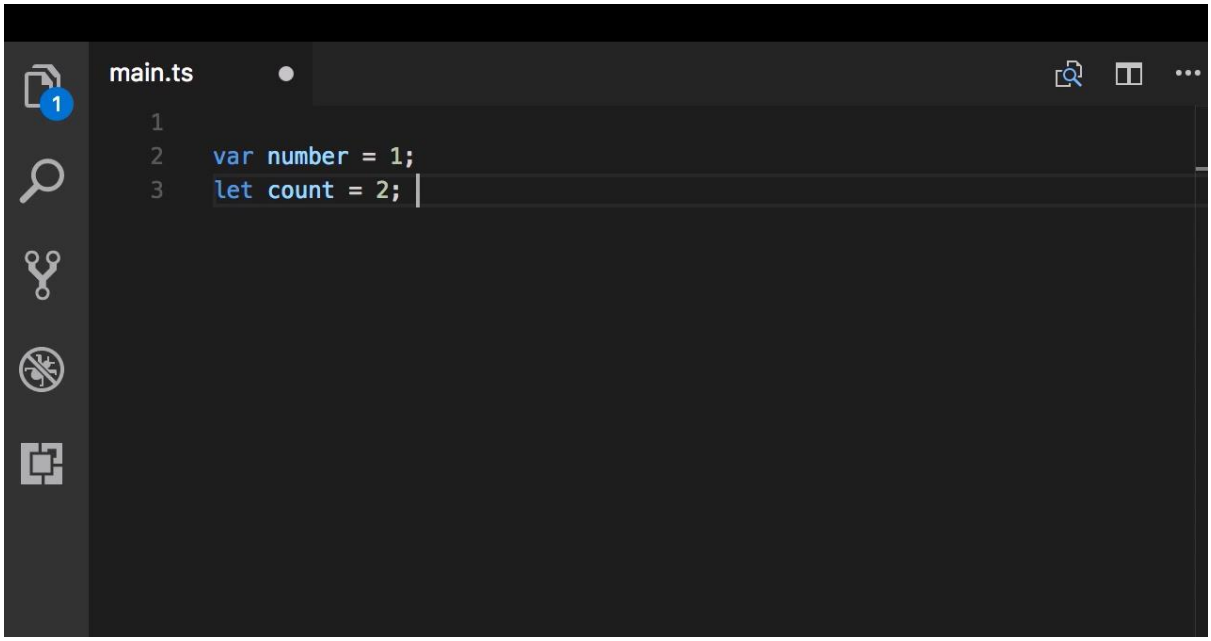


Transpiling is automatic in the above case.


main.ts
main.js
1 function log(message) {
2 console.log(message);
3 }
4 var message = 'Hello World';
5 log(message);
6


ts-hello $
ts-hello $node main.js
Hello World
ts-hello $

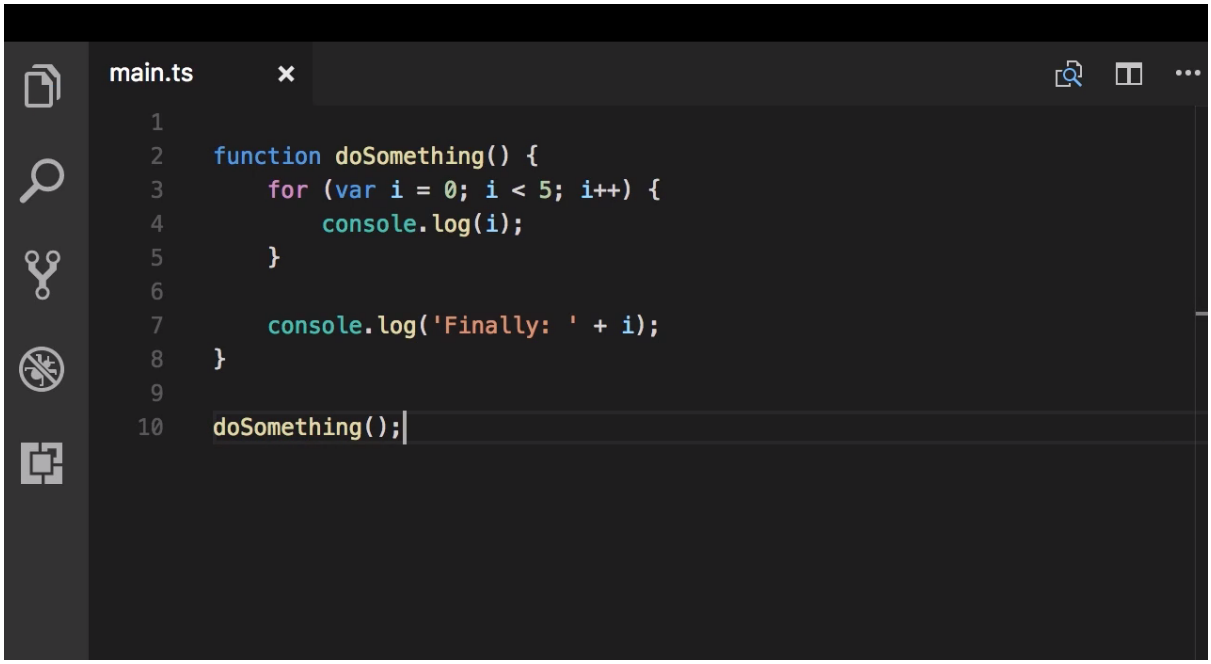
Declaring Variables

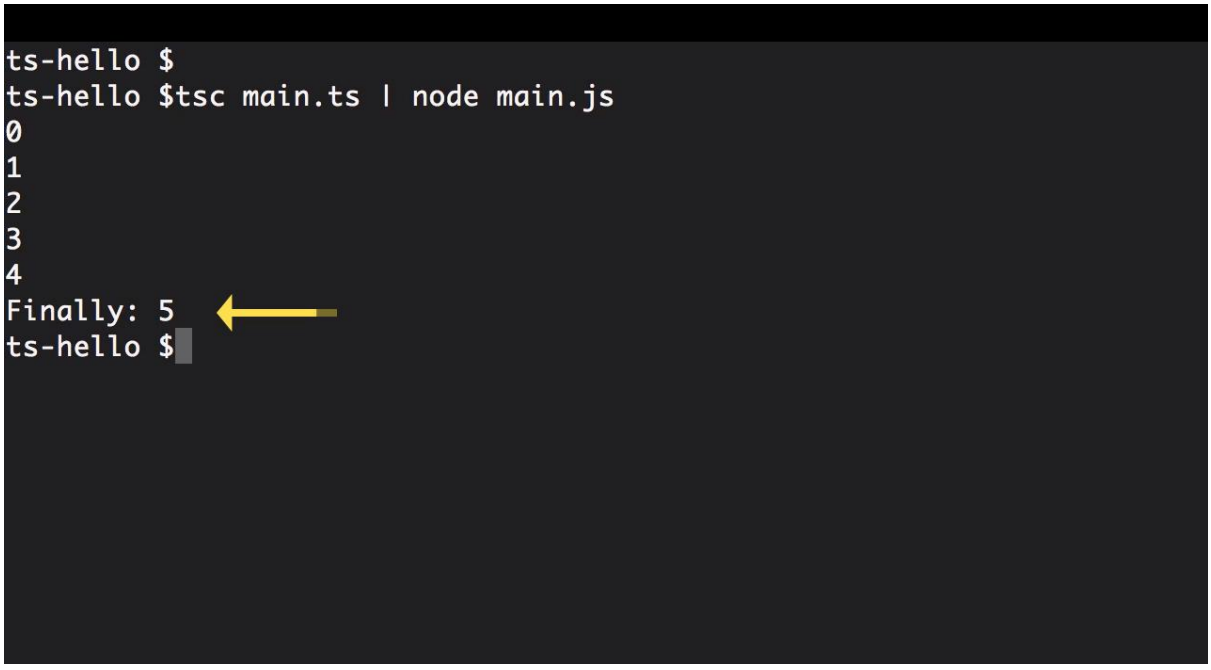


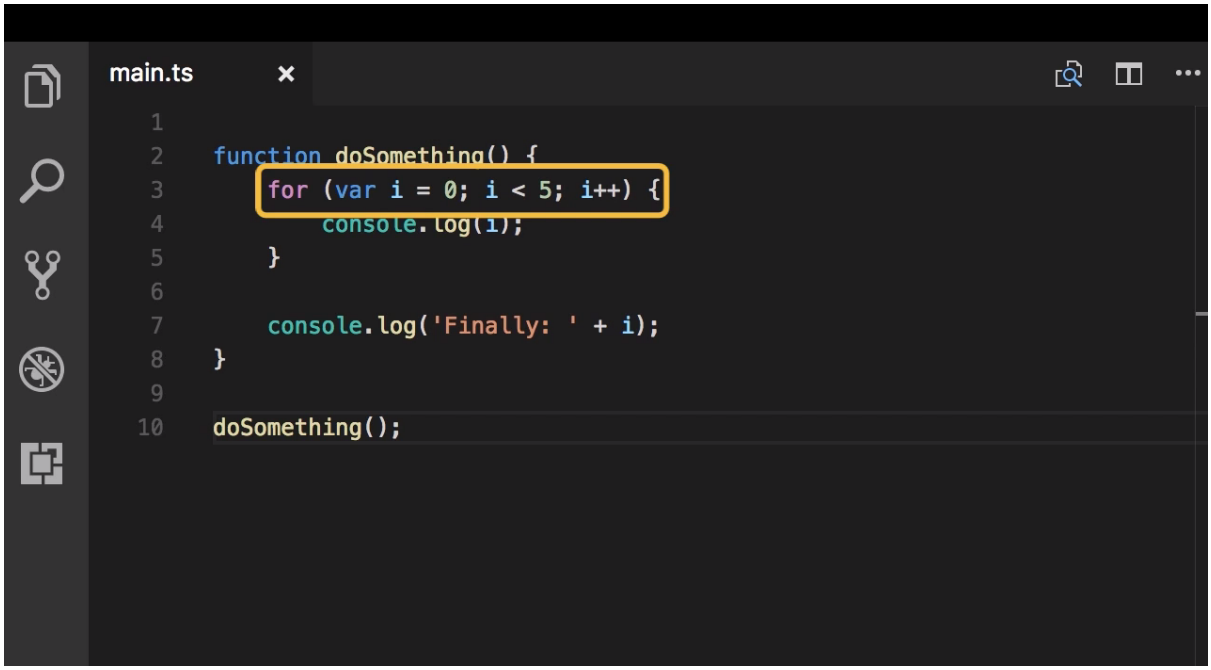
JavaScript Versions

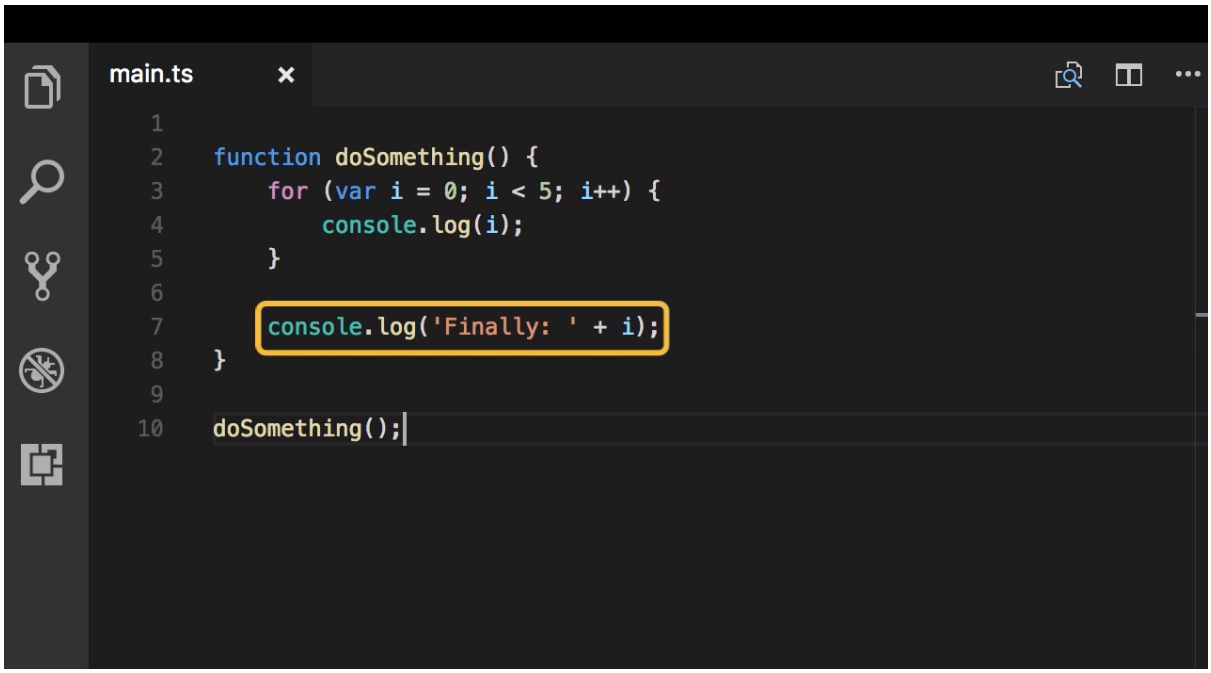
- **ES5 (ECMAScript 5):** supported by all browsers
- **ES6 (2015)**
- **ES2016**
- **ES2017**

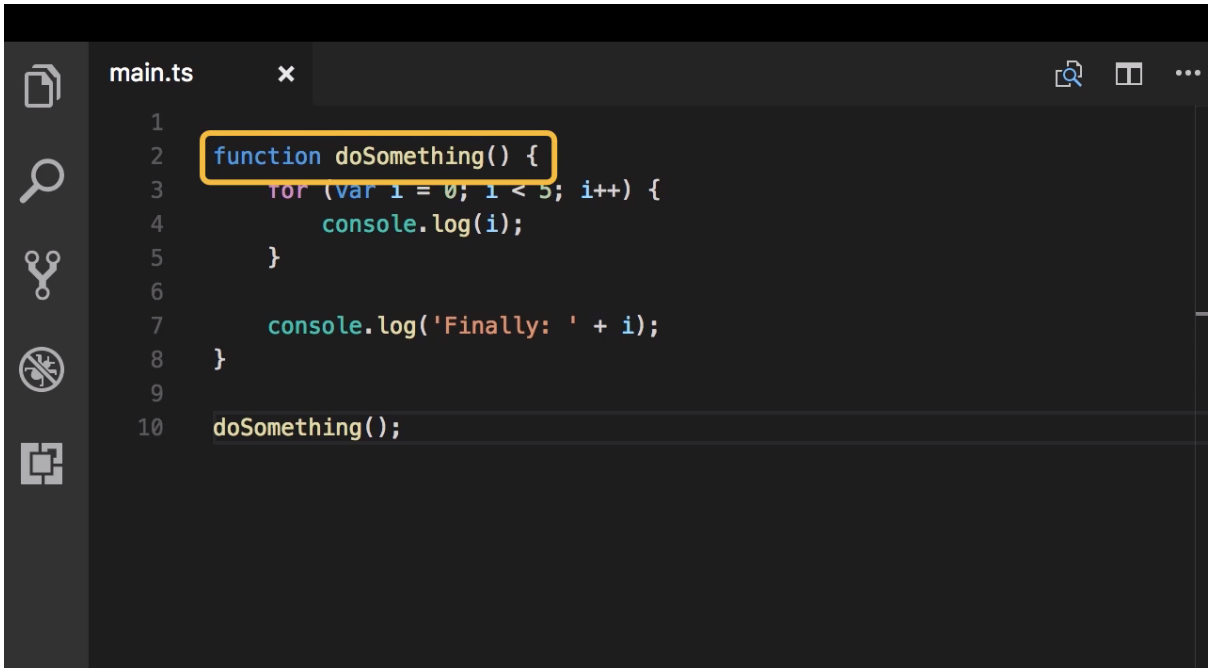
Let keyword

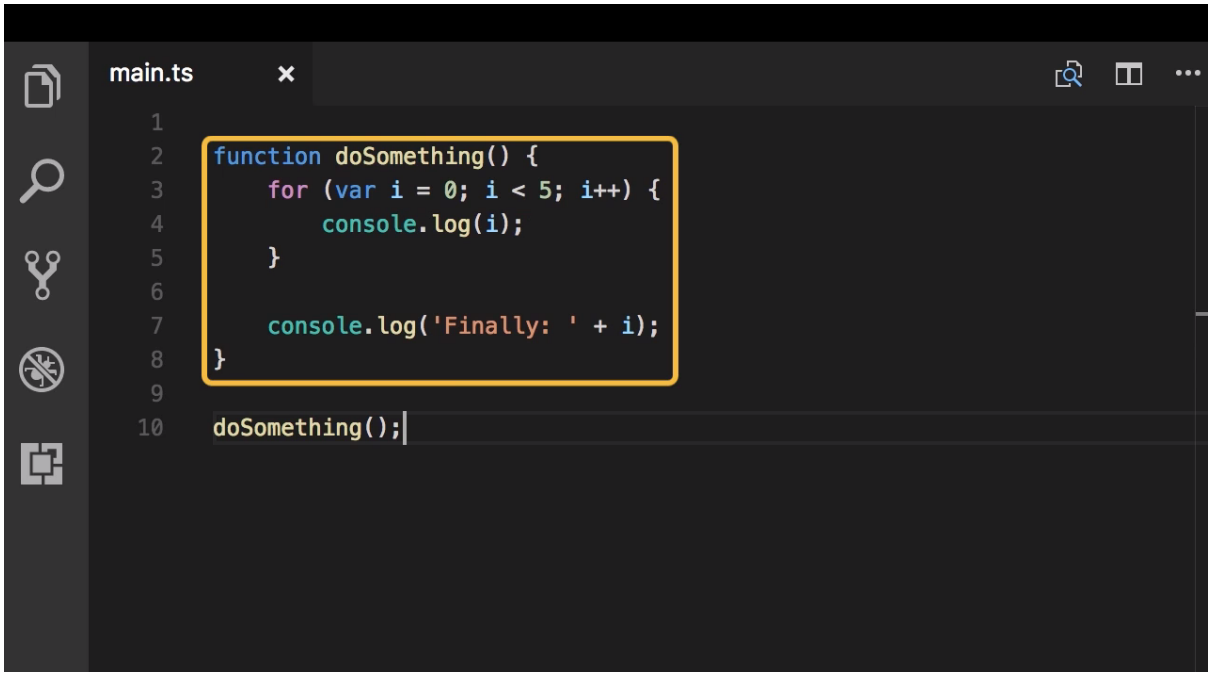

main.ts
function doSomething() {
for (var i = 0; i < 5; i++) {
console.log(i);
}
console.log('Finally: ' + i);
}
doSomething();

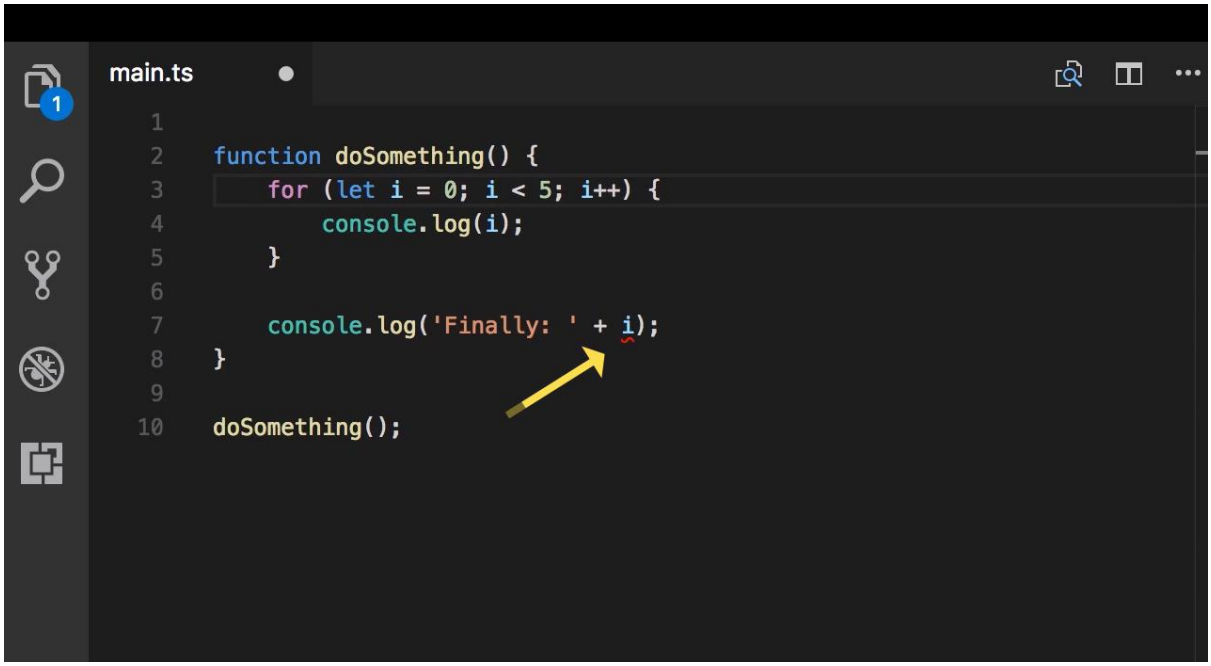

ts-hello $
ts-hello $tsc main.ts | node main.js
0
1
2
3
4
Finally: 5
ts-hello $

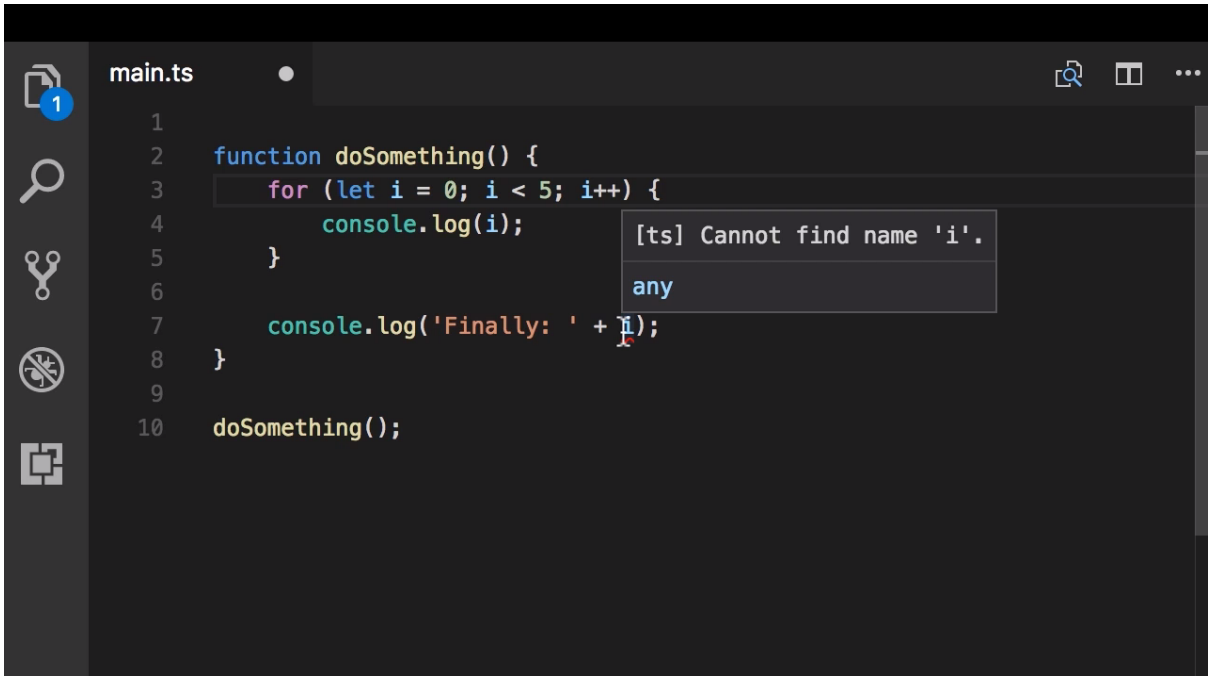

main.ts
function doSomething() {
for (var i = 0; i < 5; i++) {
console.log(i);
}
console.log('Finally: ' + i);
}
doSomething();


main.ts
function doSomething() {
for (var i = 0; i < 5; i++) {
console.log(i);
}
console.log('Finally: ' + i);
}
doSomething();

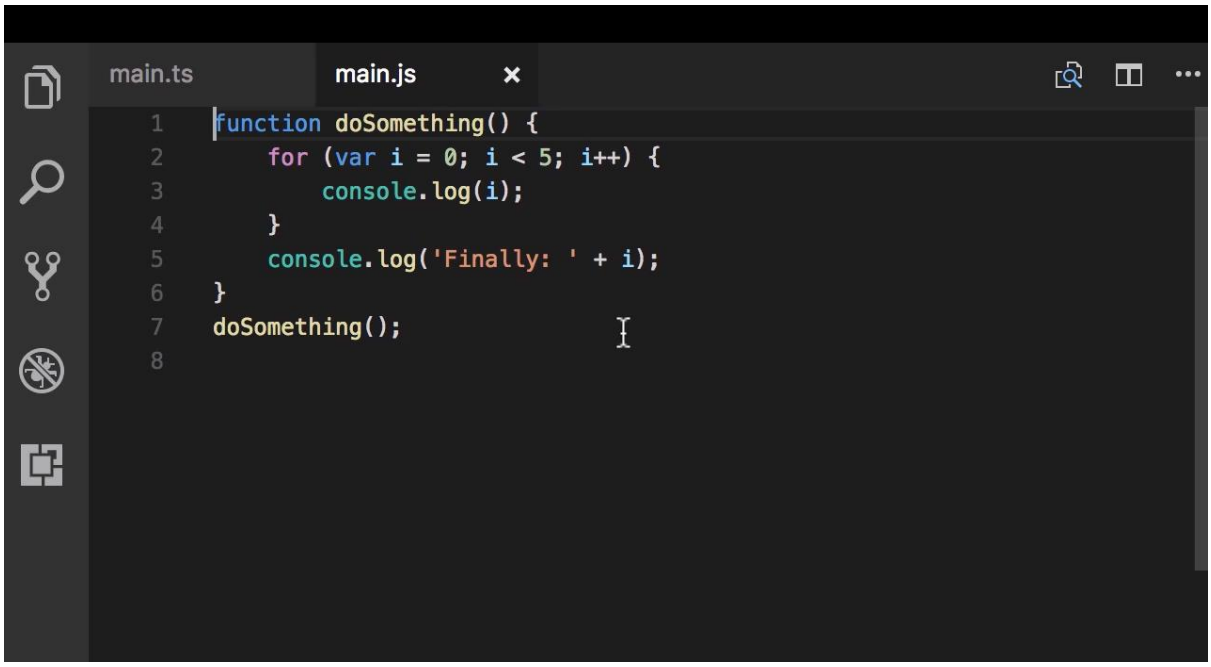

main.ts
function doSomething() {
for (var i = 0; i < 5; i++) {
console.log(i);
}
console.log('Finally: ' + i);
}
doSomething();

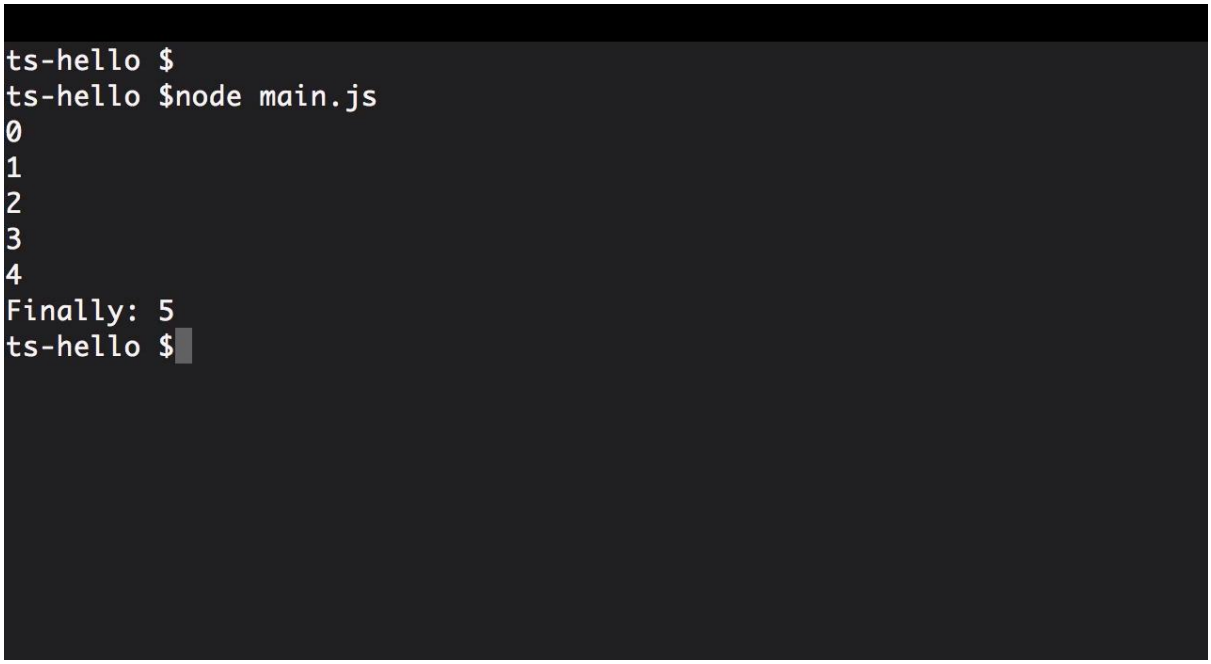

main.ts
function doSomething() {
for (var i = 0; i < 5; i++) {
console.log(i);
}
console.log('Finally: ' + i);
}
doSomething();


main.ts
1
function doSomething() {
for (let i = 0; i < 5; i++) {
console.log(i);
}
console.log('Finally: ' + i);
}
doSomething();

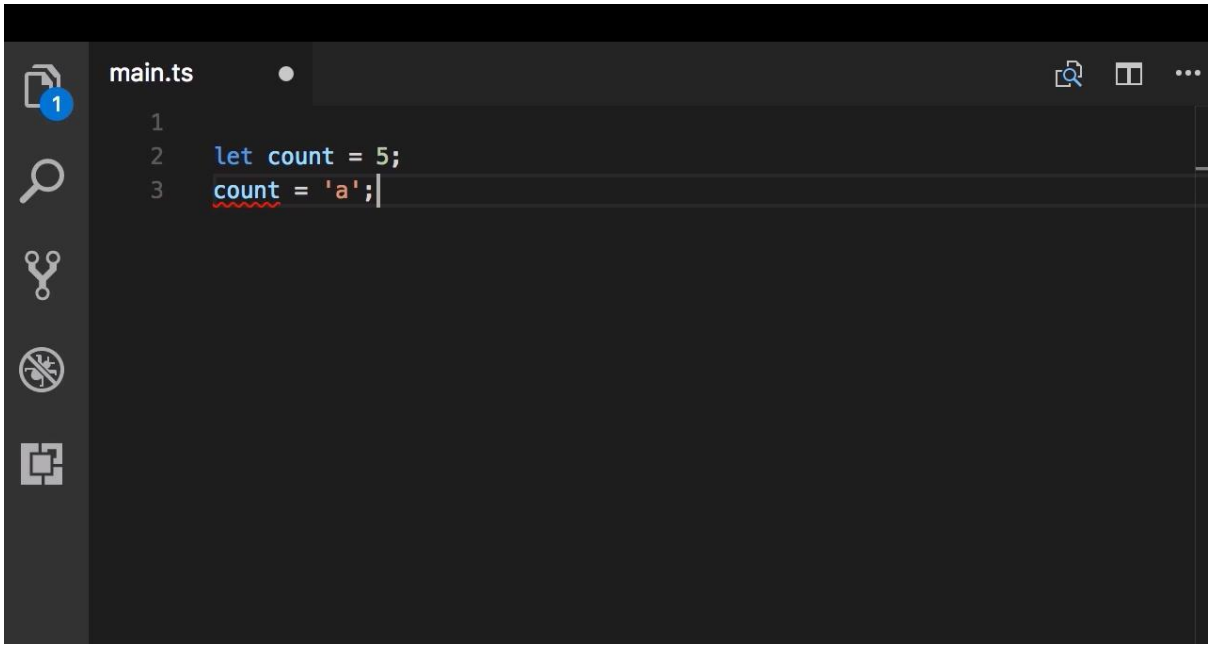

main.ts
1
function doSomething() {
for (let i = 0; i < 5; i++) {
console.log(i);
}
console.log('Finally: ' + i);
}
doSomething();
[ts] Cannot find name 'i'.
any

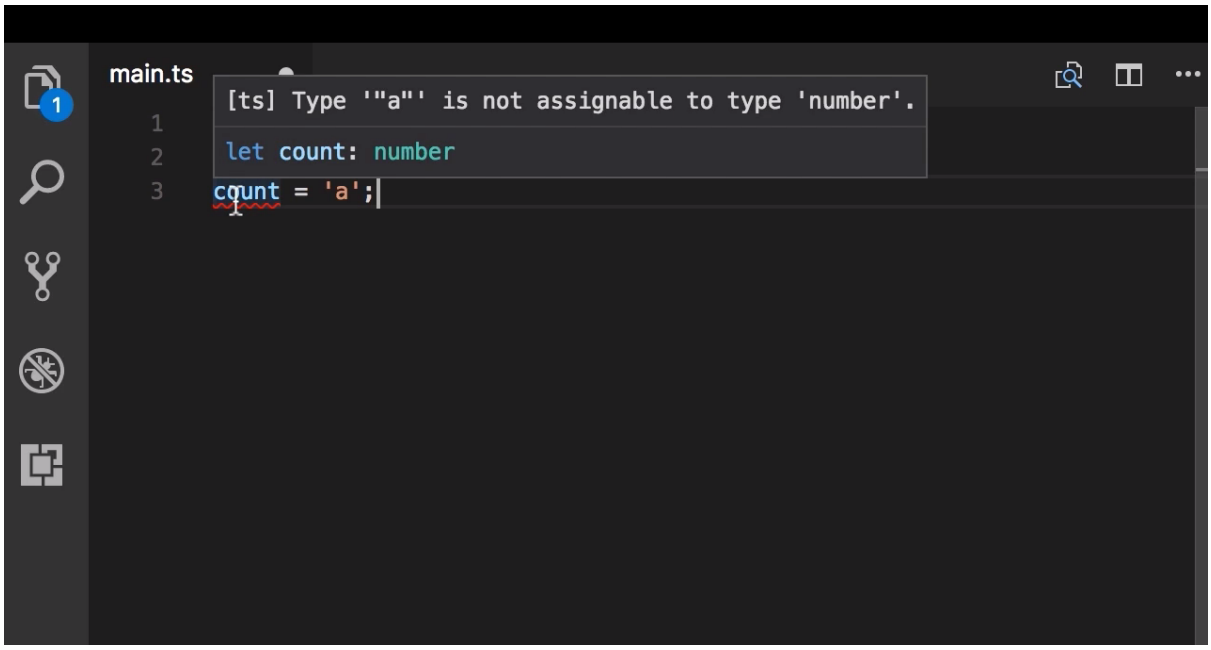
```
ts-hello $
ts-hello $rm main.js
ts-hello $tsc main.ts
main.ts(7,31): error TS2304: Cannot find name 'i'.
ts-hello $ls
main.js main.ts
ts-hello $
```



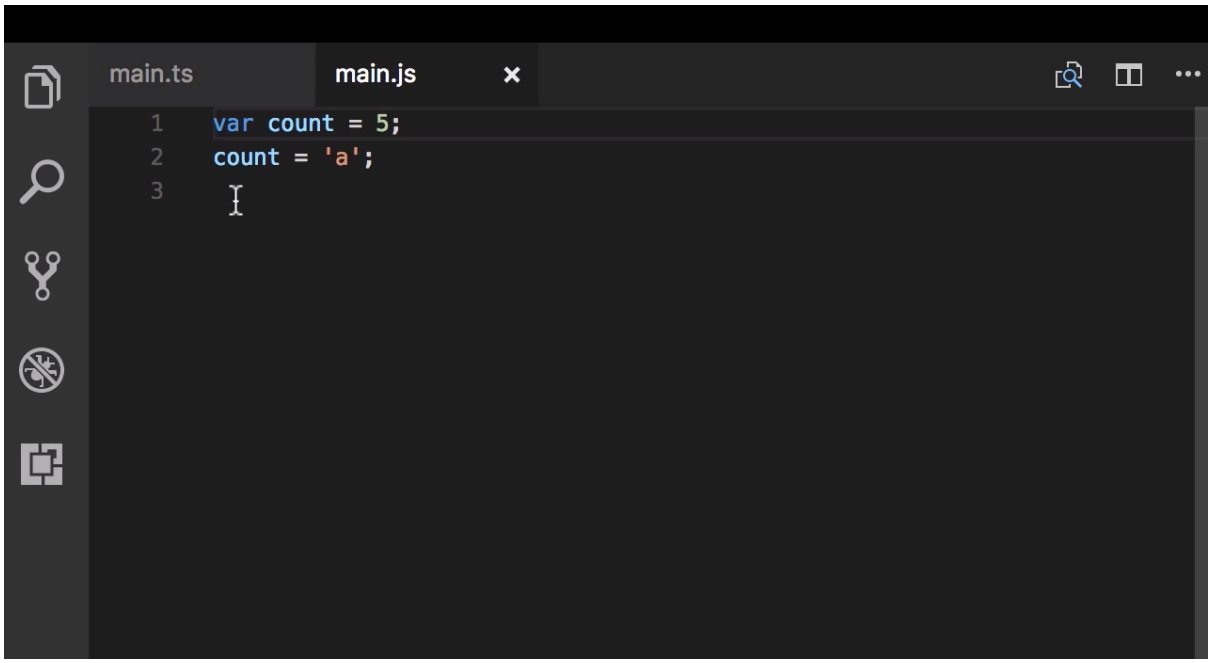

ts-hello $
ts-hello $node main.js
0
1
2
3
4
Finally: 5
ts-hello $

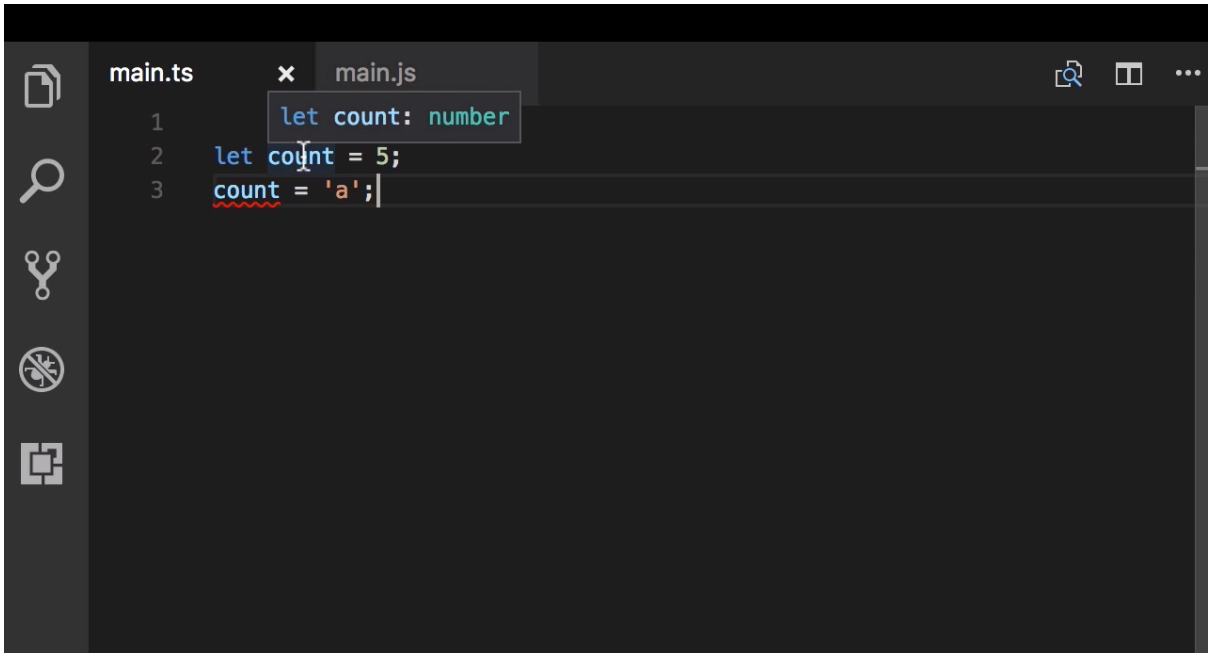
Types

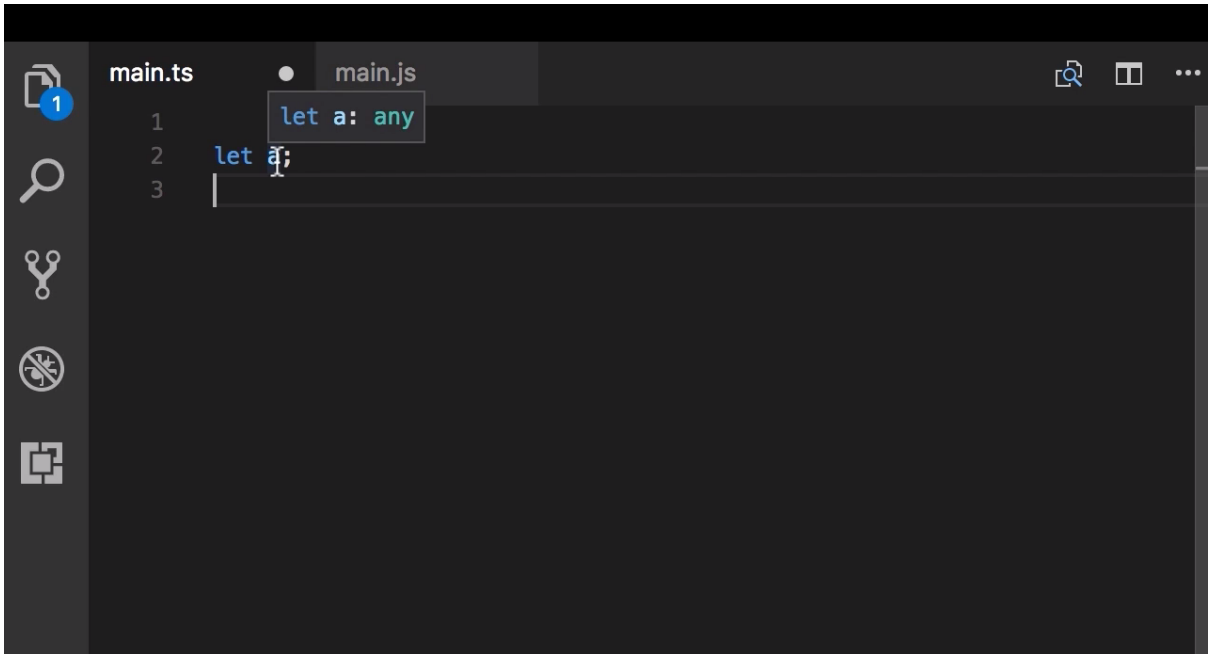

main.ts
1
2 let count = 5;
3 count = 'a';

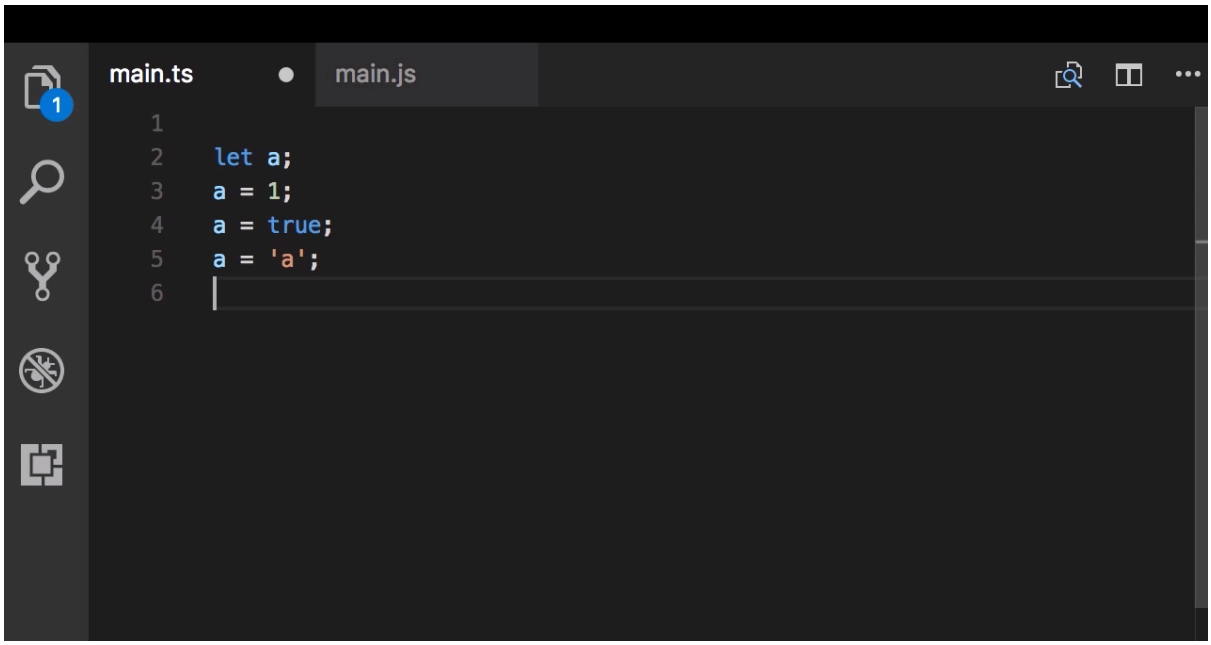

main.ts
[ts] Type '"a"' is not assignable to type 'number'.
let count: number
1
2
3 count = 'a';

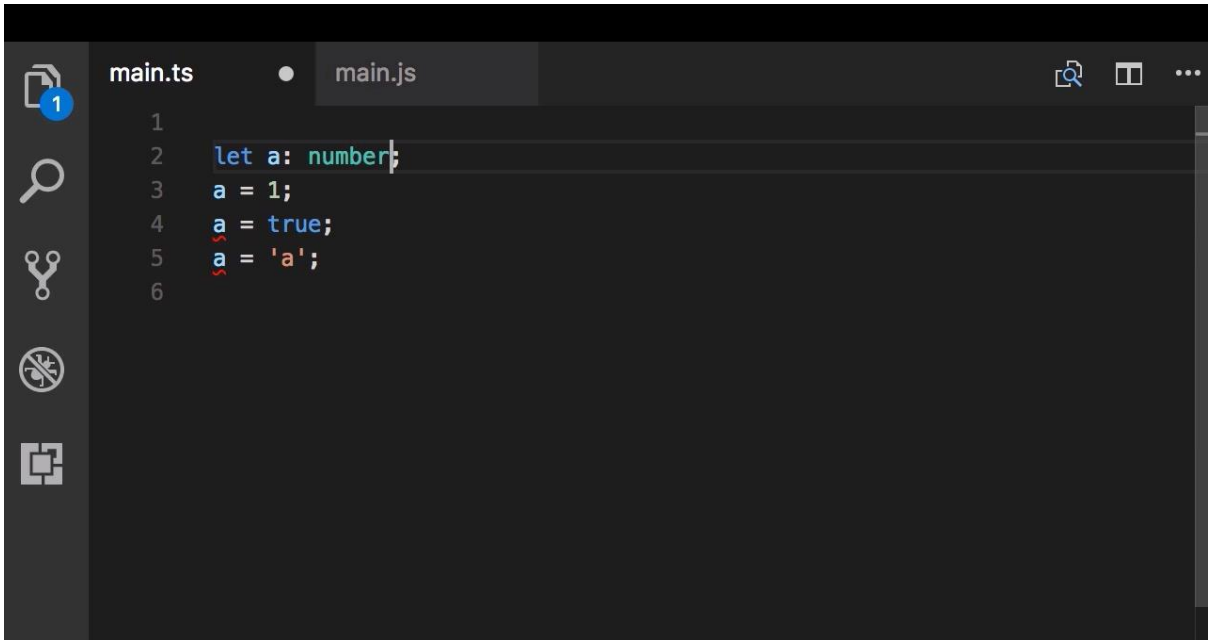
Coverts to

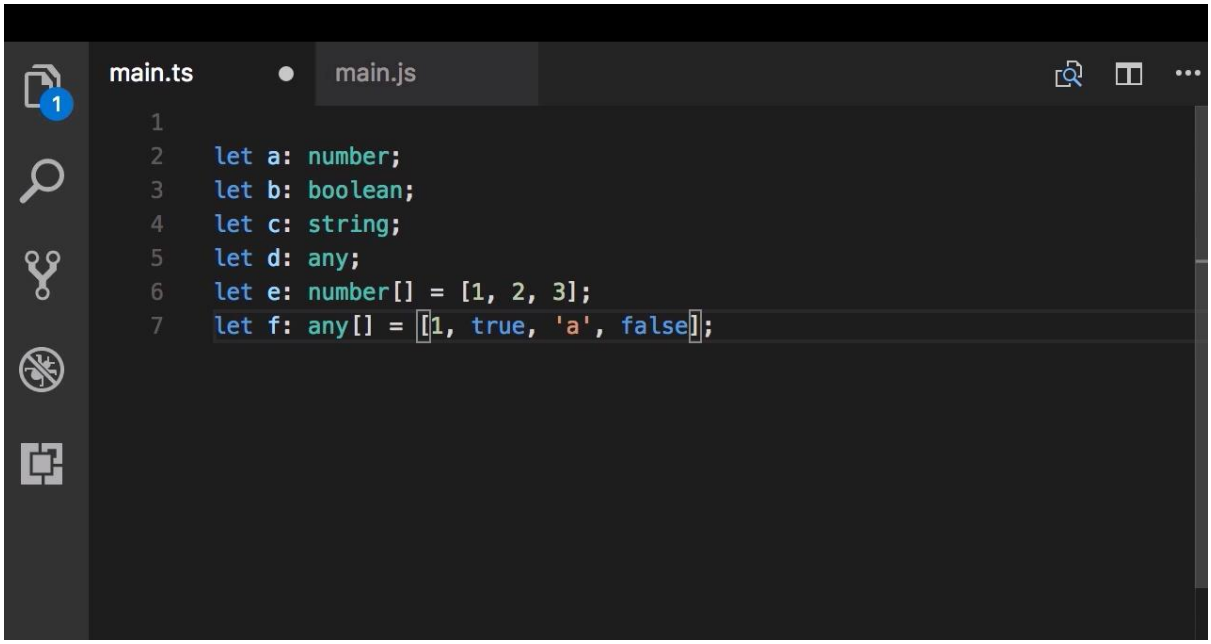

main.ts
main.js
1 var count = 5;
2 count = 'a';
3


main.ts
main.js
let count: number
1
2 let count = 5;
3 count = 'a';

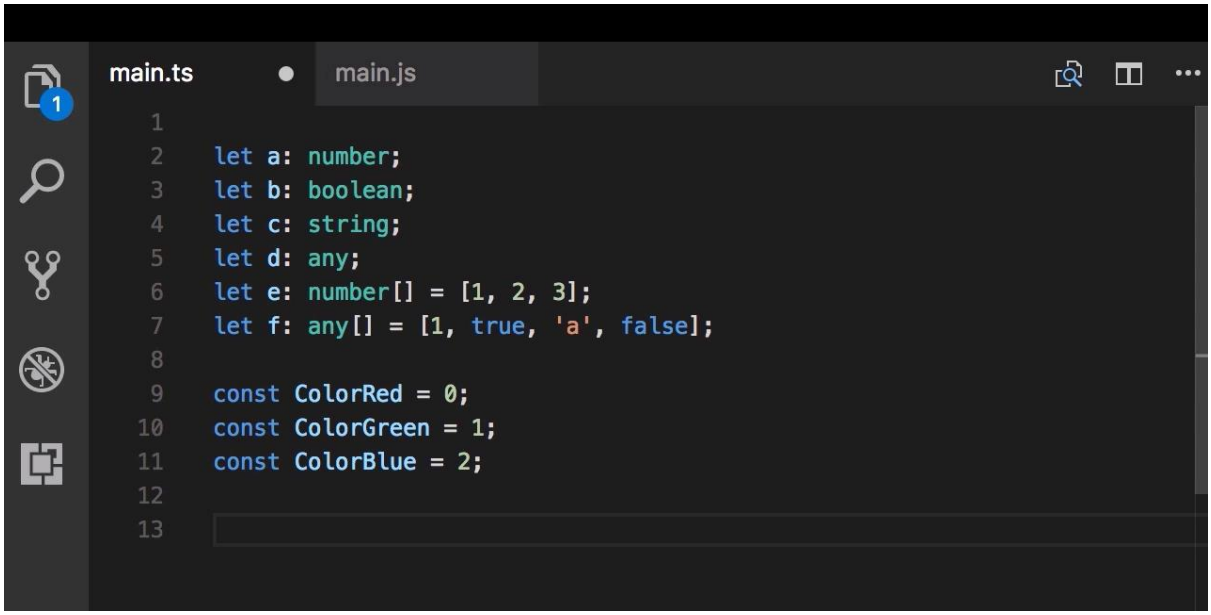

main.ts
main.js
let a: any
1
2 let a;
3

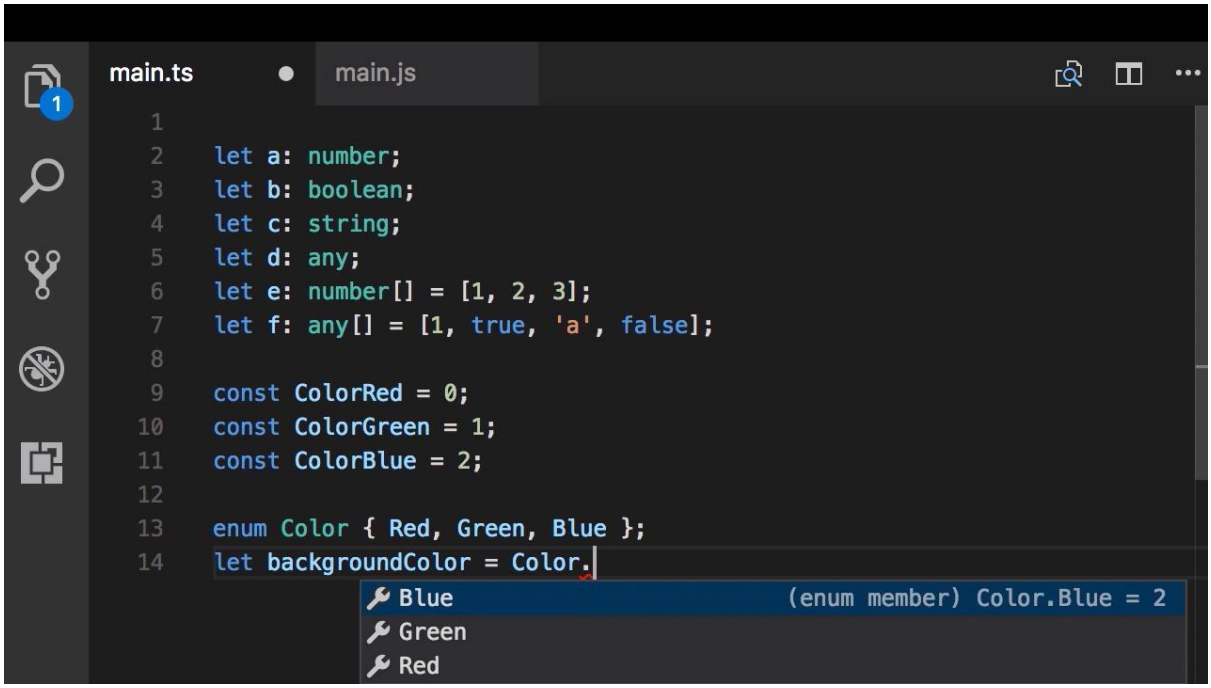

main.ts
main.js
1
2 let a;
3 a = 1;
4 a = true;
5 a = 'a';
6

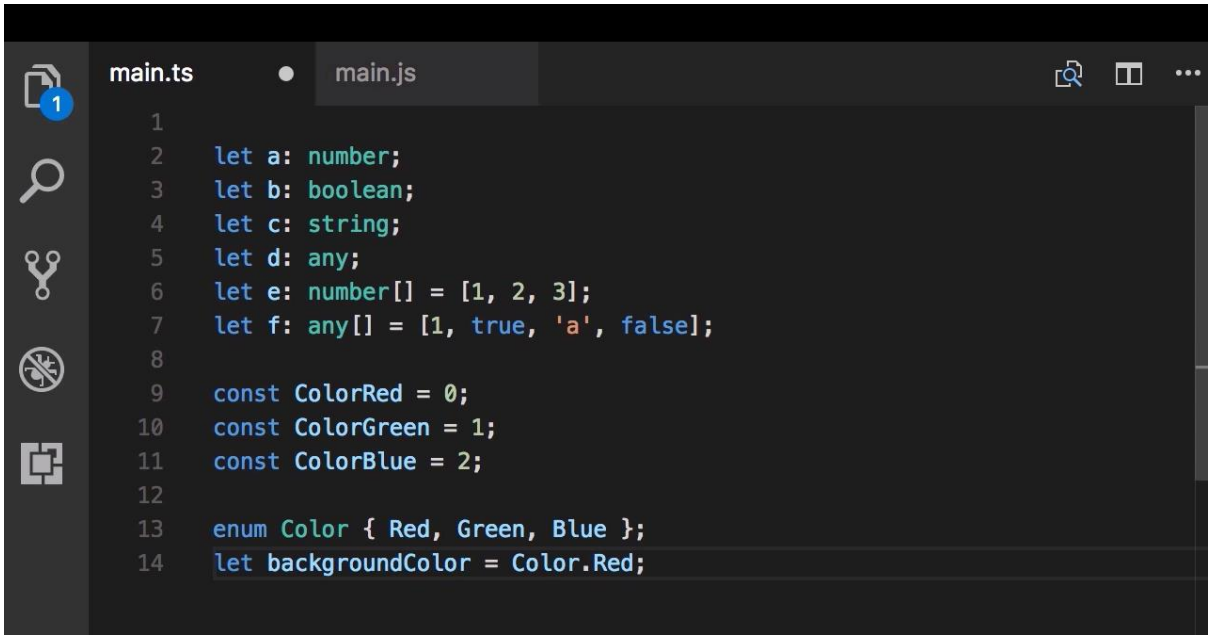

main.ts
main.js
1
let a: number;
a = 1;
a = true;
a = 'a';
6

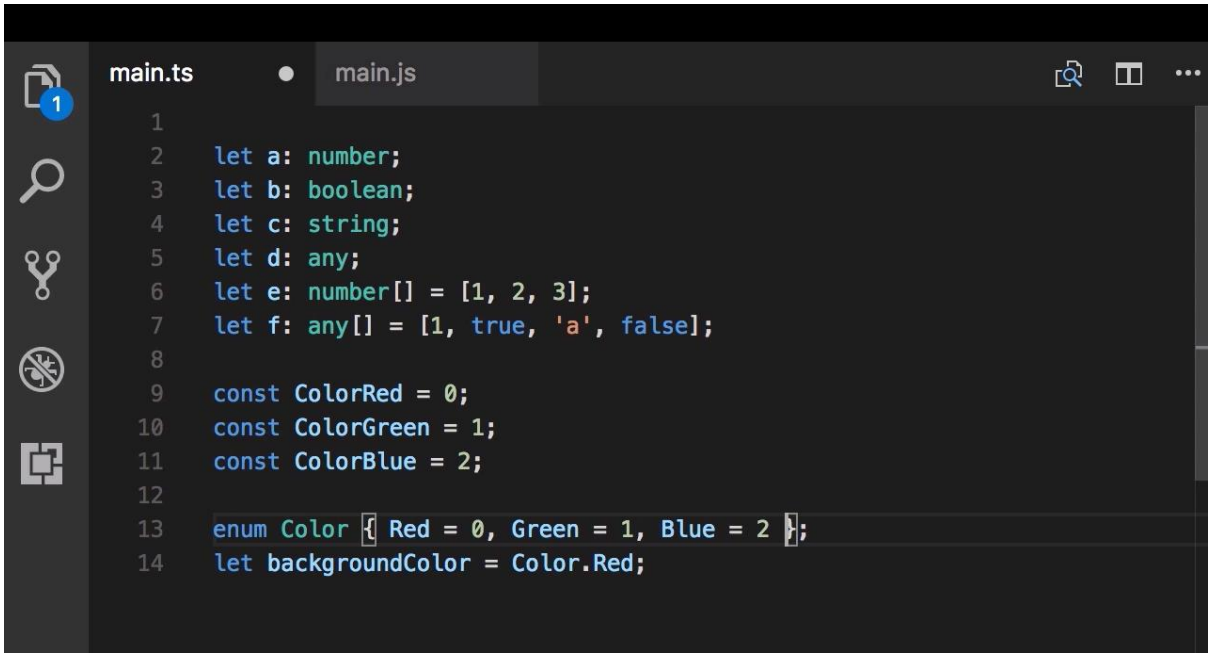

main.ts
main.js
1
let a: number;
let b: boolean;
let c: string;
let d: any;
let e: number[] = [1, 2, 3];
let f: any[] = [1, true, 'a', false];

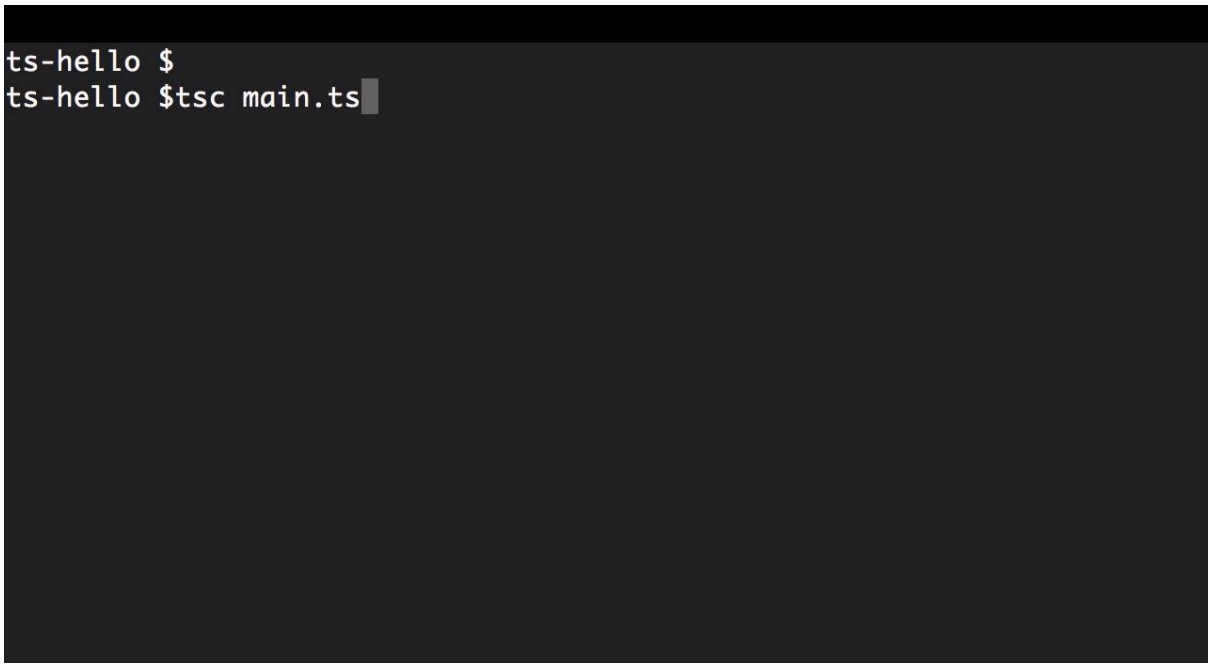
enums

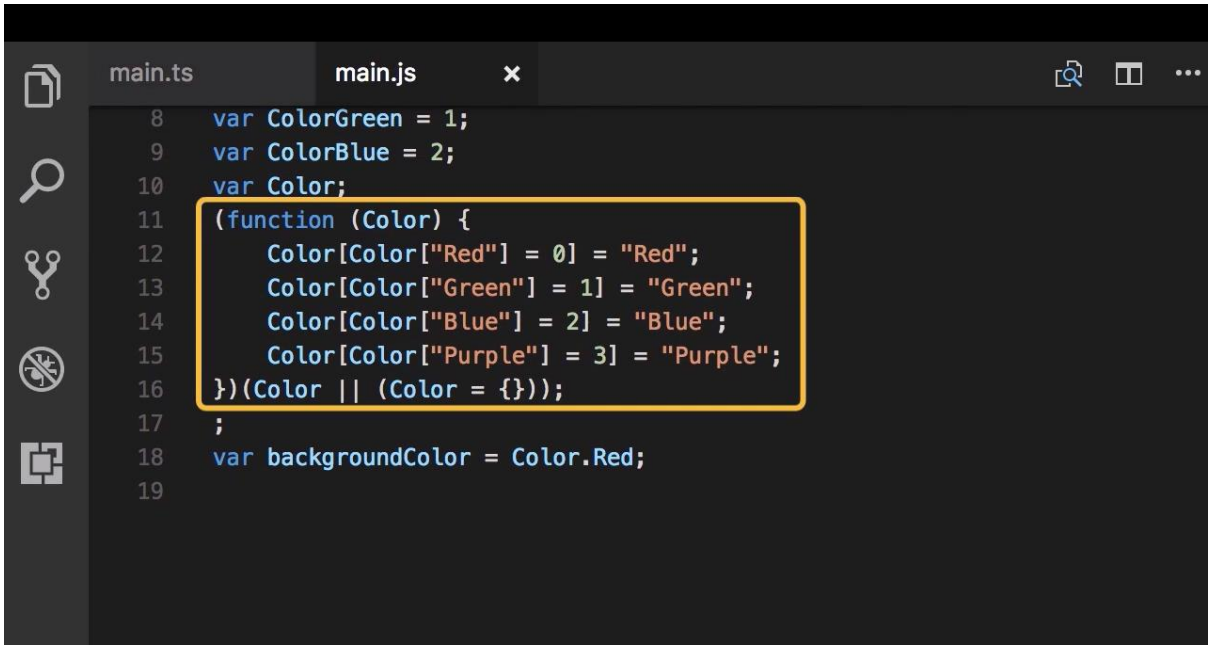

main.ts
main.js
1
let a: number;
let b: boolean;
let c: string;
let d: any;
let e: number[] = [1, 2, 3];
let f: any[] = [1, true, 'a', false];
const ColorRed = 0;
const ColorGreen = 1;
const ColorBlue = 2;

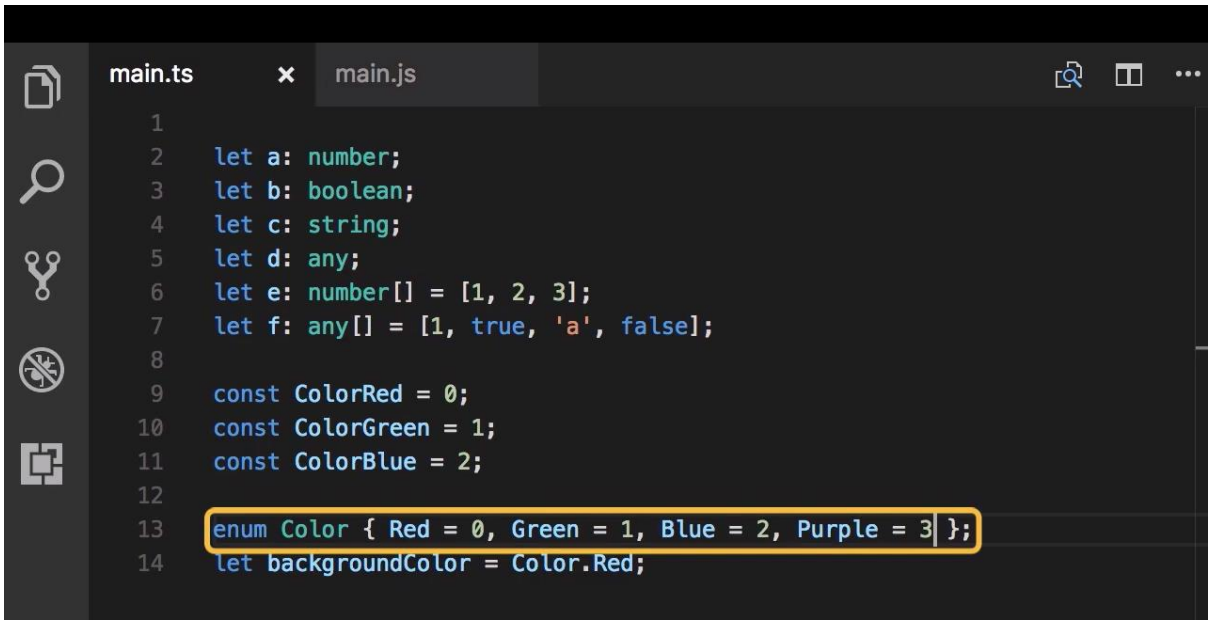

main.ts
main.js
1
let a: number;
let b: boolean;
let c: string;
let d: any;
let e: number[] = [1, 2, 3];
let f: any[] = [1, true, 'a', false];
const ColorRed = 0;
const ColorGreen = 1;
const ColorBlue = 2;
enum Color { Red, Green, Blue };
let backgroundColor = Color.
Blue
(enum member) Color.Blue = 2
Green
Red


main.ts
main.js
1
let a: number;
let b: boolean;
let c: string;
let d: any;
let e: number[] = [1, 2, 3];
let f: any[] = [1, true, 'a', false];
const ColorRed = 0;
const ColorGreen = 1;
const ColorBlue = 2;
enum Color { Red, Green, Blue };
let backgroundColor = Color.Red;

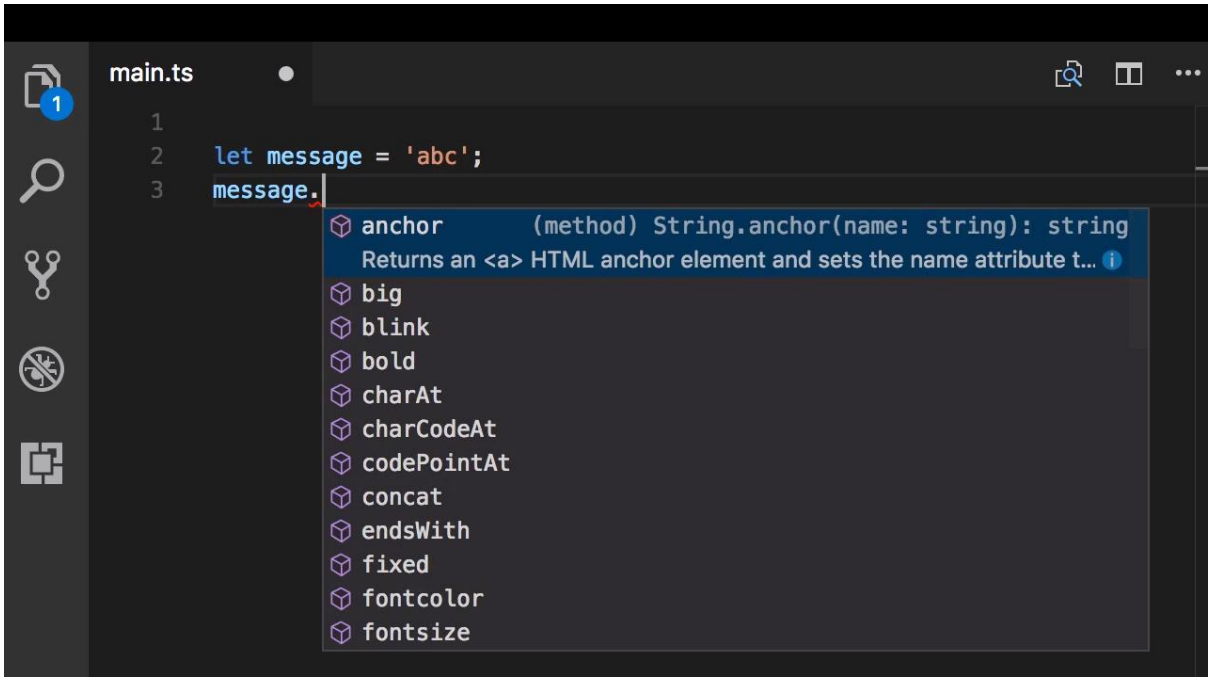

main.ts
main.js
let a: number;
let b: boolean;
let c: string;
let d: any;
let e: number[] = [1, 2, 3];
let f: any[] = [1, true, 'a', false];
const ColorRed = 0;
const ColorGreen = 1;
const ColorBlue = 2;
enum Color { Red = 0, Green = 1, Blue = 2 };
let backgroundColor = Color.Red;

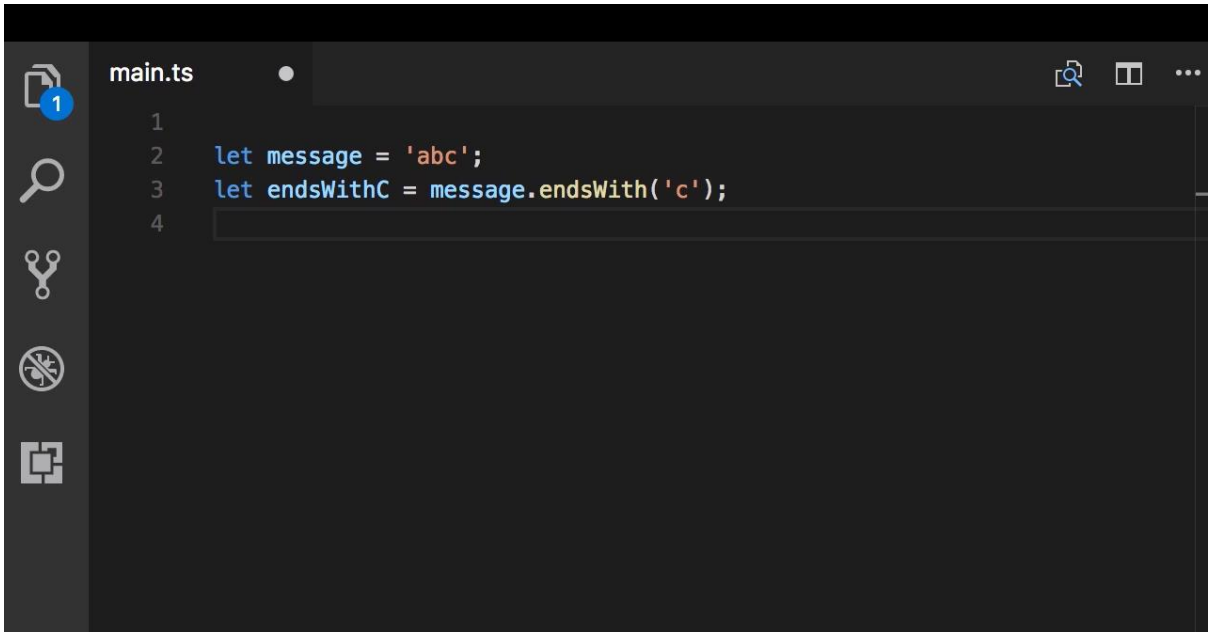

ts-hello $
ts-hello $tsc main.ts


main.ts
main.js
var ColorGreen = 1;
var ColorBlue = 2;
var Color;
(function (Color) {
Color[Color["Red"] = 0] = "Red";
Color[Color["Green"] = 1] = "Green";
Color[Color["Blue"] = 2] = "Blue";
Color[Color["Purple"] = 3] = "Purple";
})(Color || (Color = {}));
;
var backgroundColor = Color.Red;

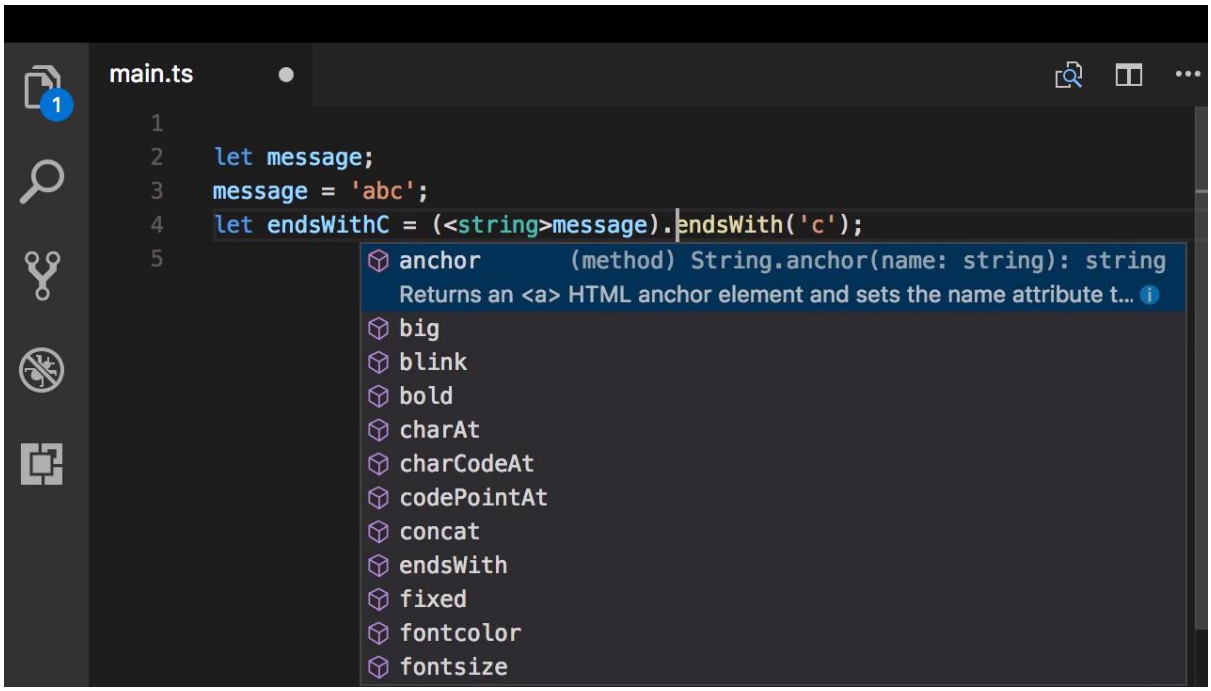

main.ts
main.js
let a: number;
let b: boolean;
let c: string;
let d: any;
let e: number[] = [1, 2, 3];
let f: any[] = [1, true, 'a', false];
const ColorRed = 0;
const ColorGreen = 1;
const ColorBlue = 2;
enum Color { Red = 0, Green = 1, Blue = 2, Purple = 3 };
let backgroundColor = Color.Red;

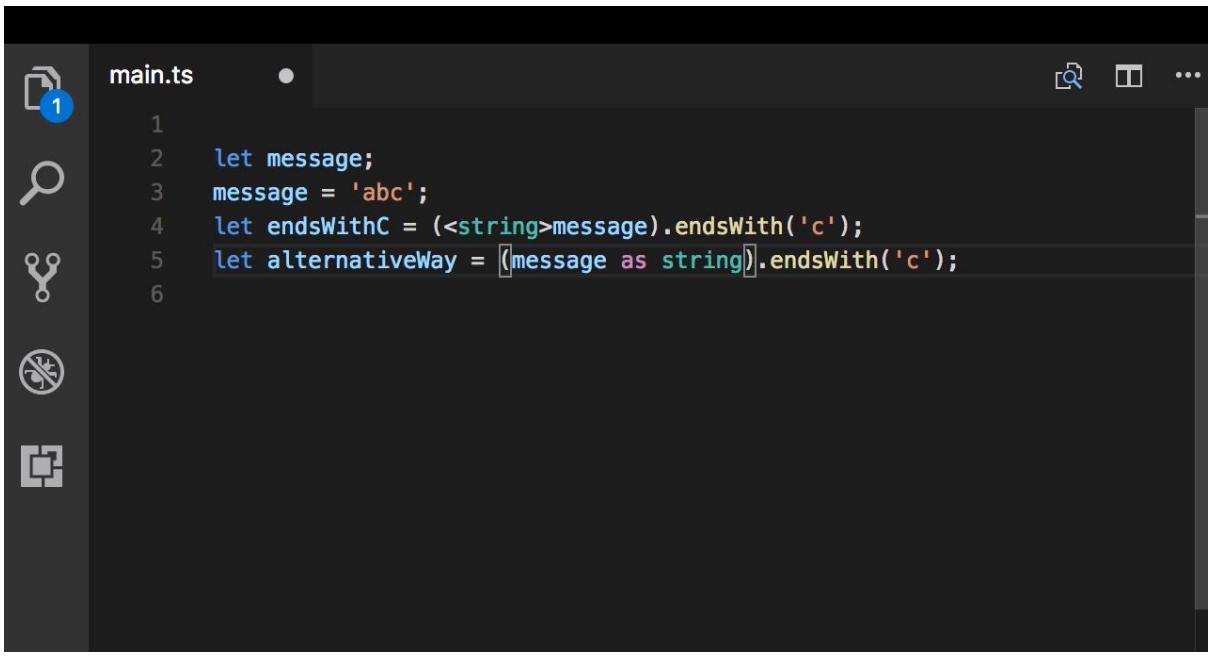
Type Assertions



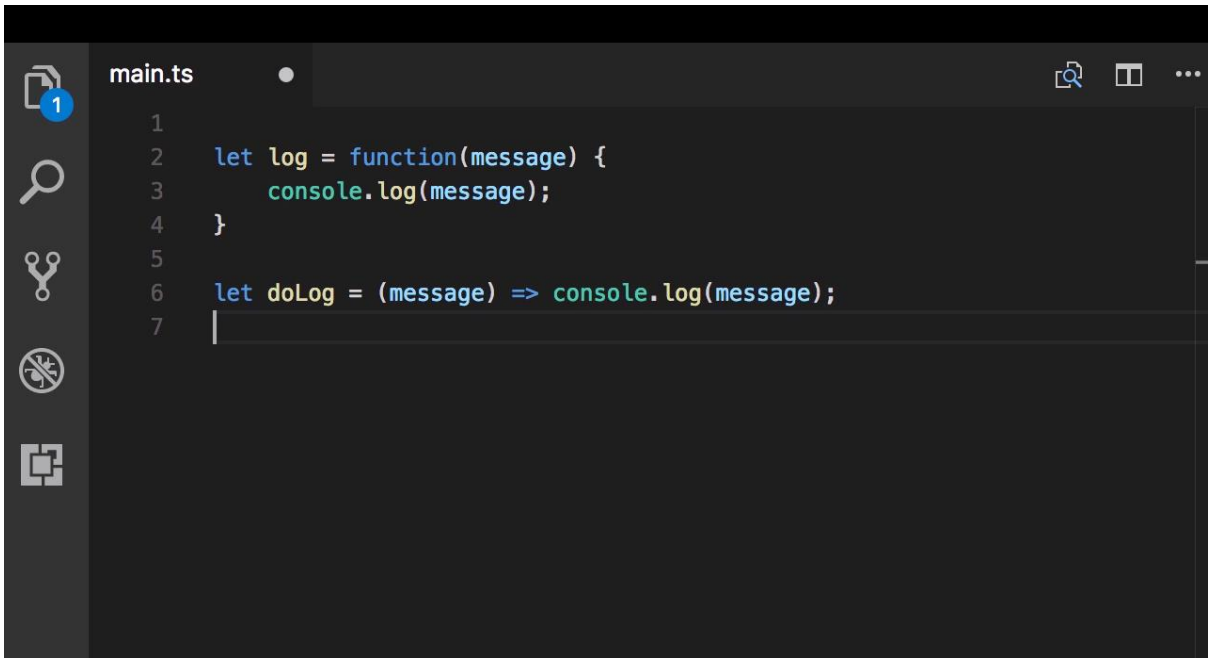


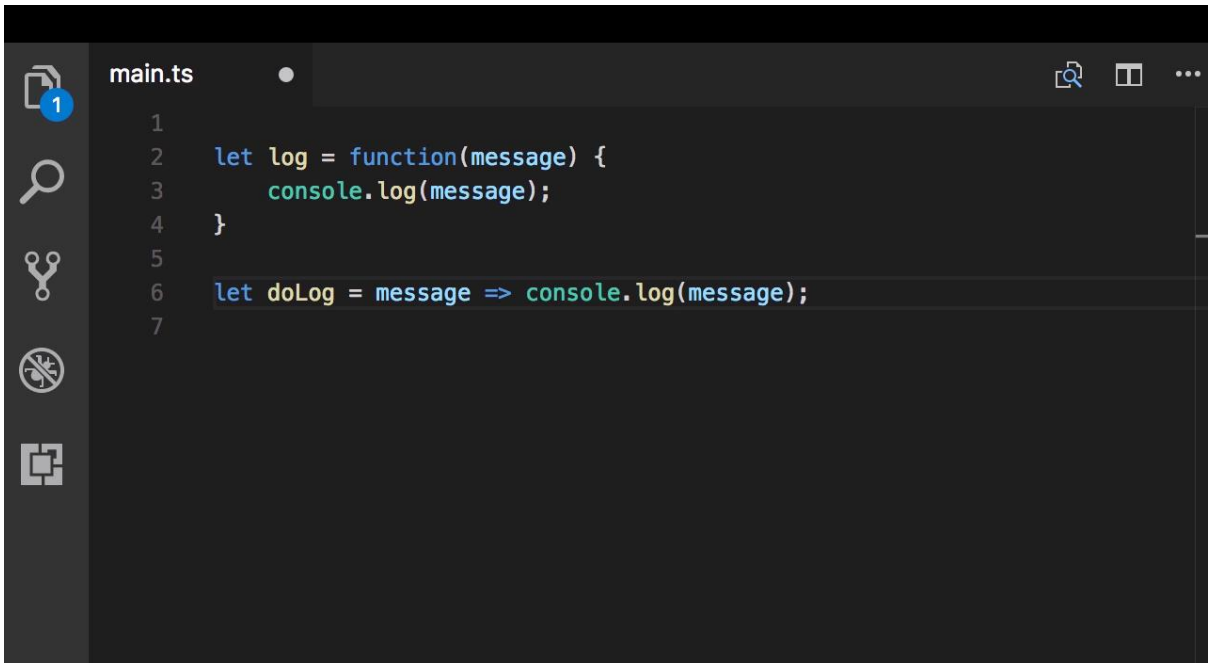
Type assertion in action:





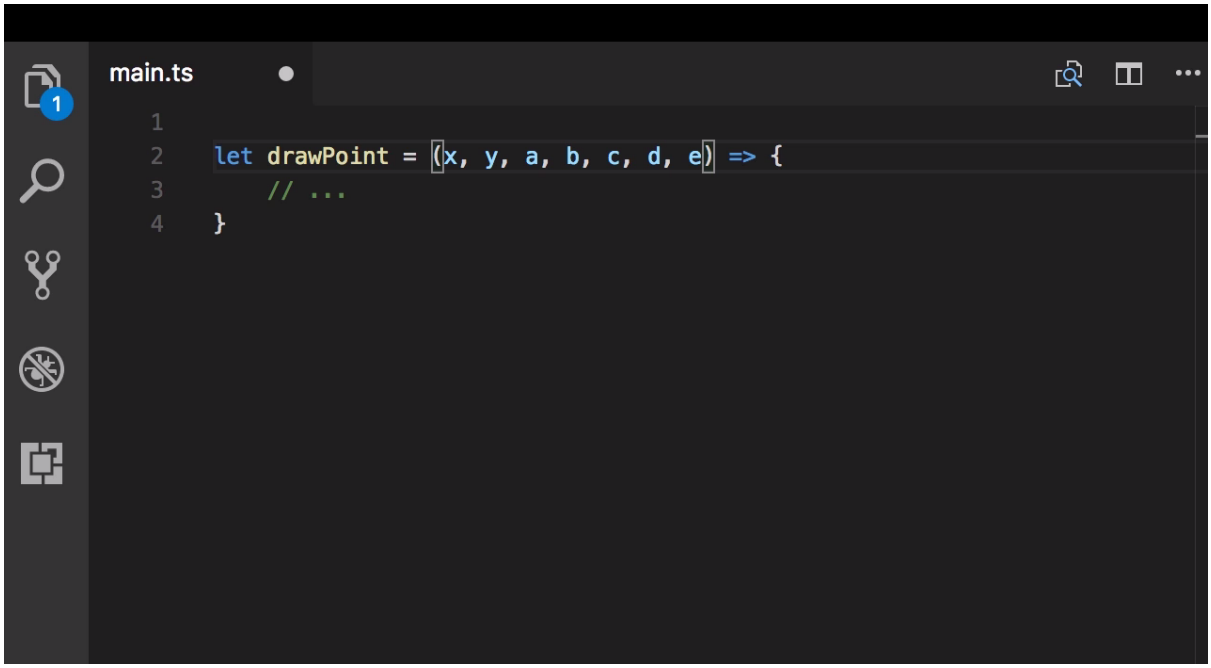
Arrow Functions



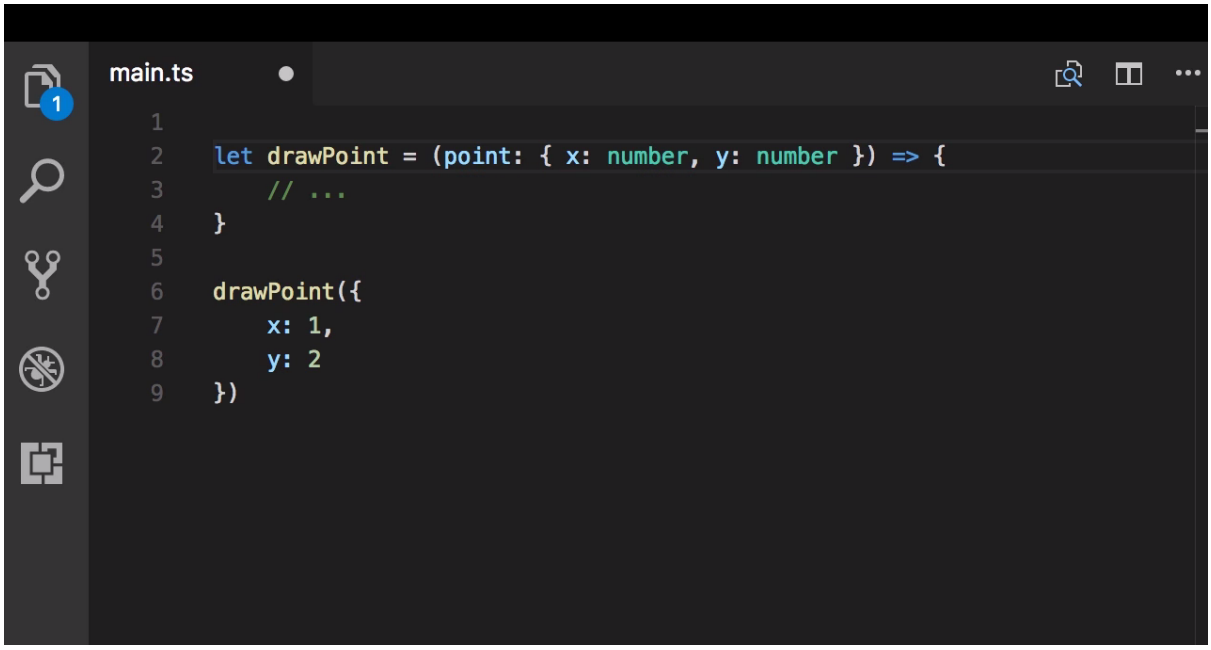


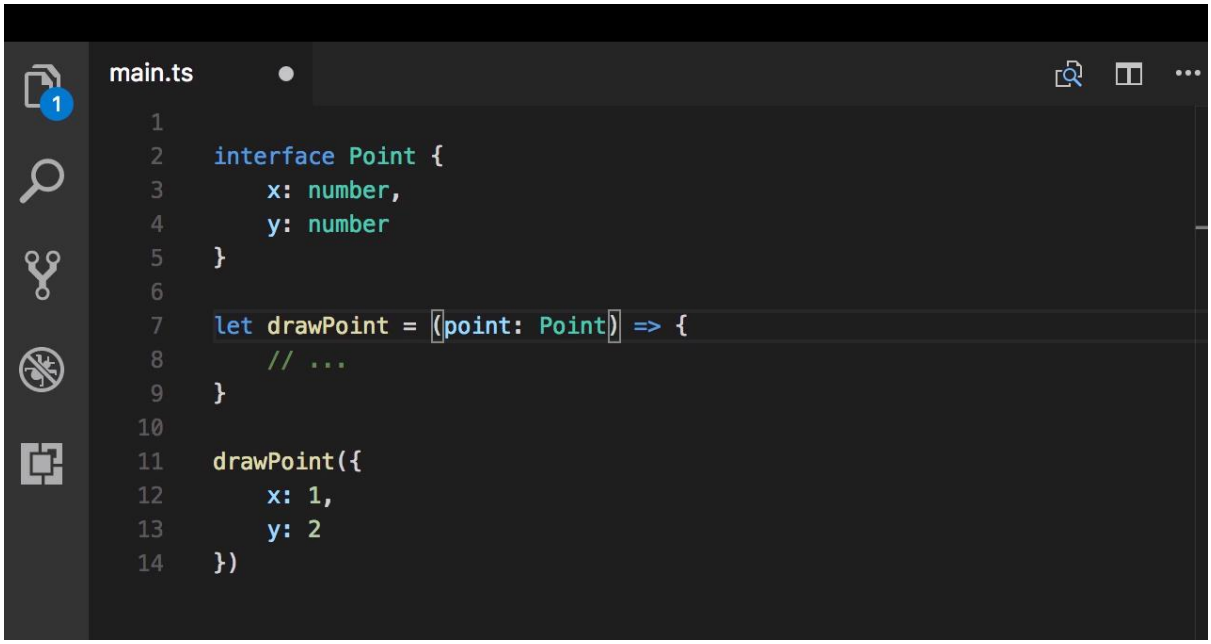

main.ts
1
2 let log = function(message) {
3 console.log(message);
4 }
5
6 let doLog = () => console.log();
7

Interfaces



Inline annotation


main.ts
let drawPoint = (point: { x: number, y: number }) => {
// ...
}
drawPoint({
x: 1,
y: 2
})


main.ts
interface Point {
x: number,
y: number
}
let drawPoint = (point: Point) => {
// ...
}
drawPoint({
x: 1,
y: 2
})

Classes

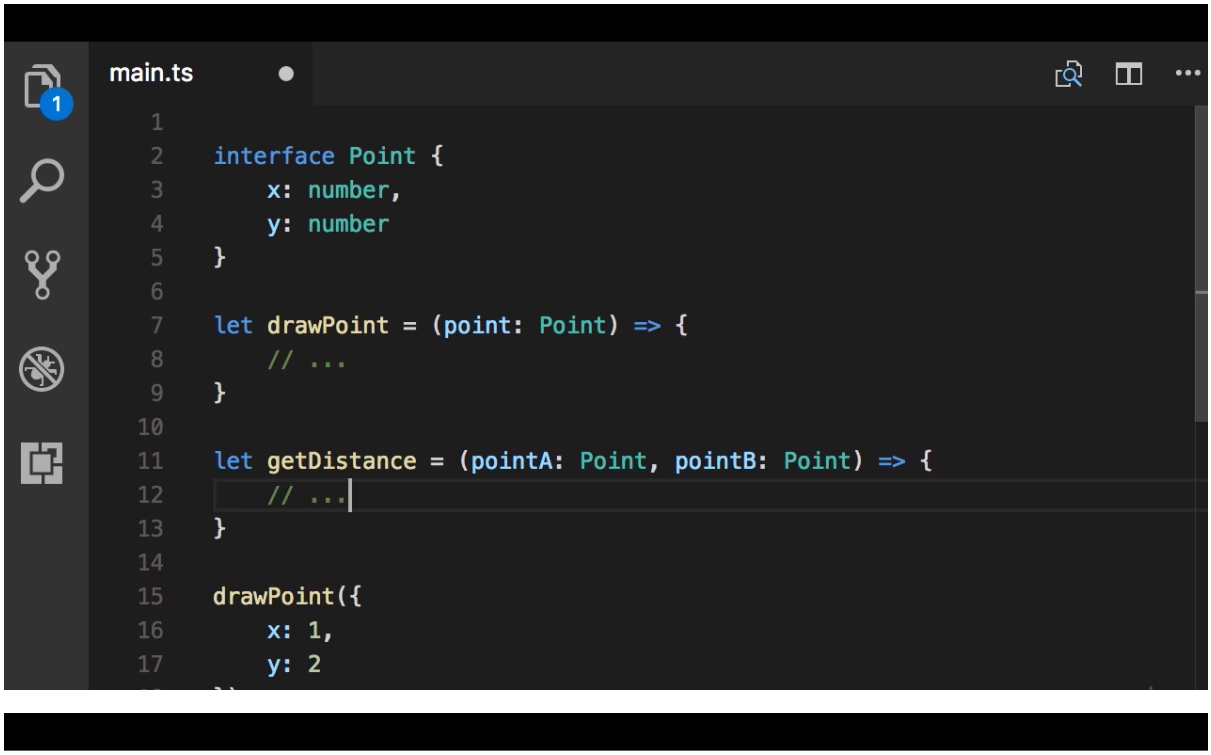
main.ts

```
1
2   interface Point {
3       x: number,
4       y: number
5   }
6
7   let drawPoint = (point: Point) => {
8       // ...
9   }
10
11  drawPoint({
12      x: 1,
13      y: 2
14  })
15
16
```

Cohesion

All the related things should be at one place. Data and Functions

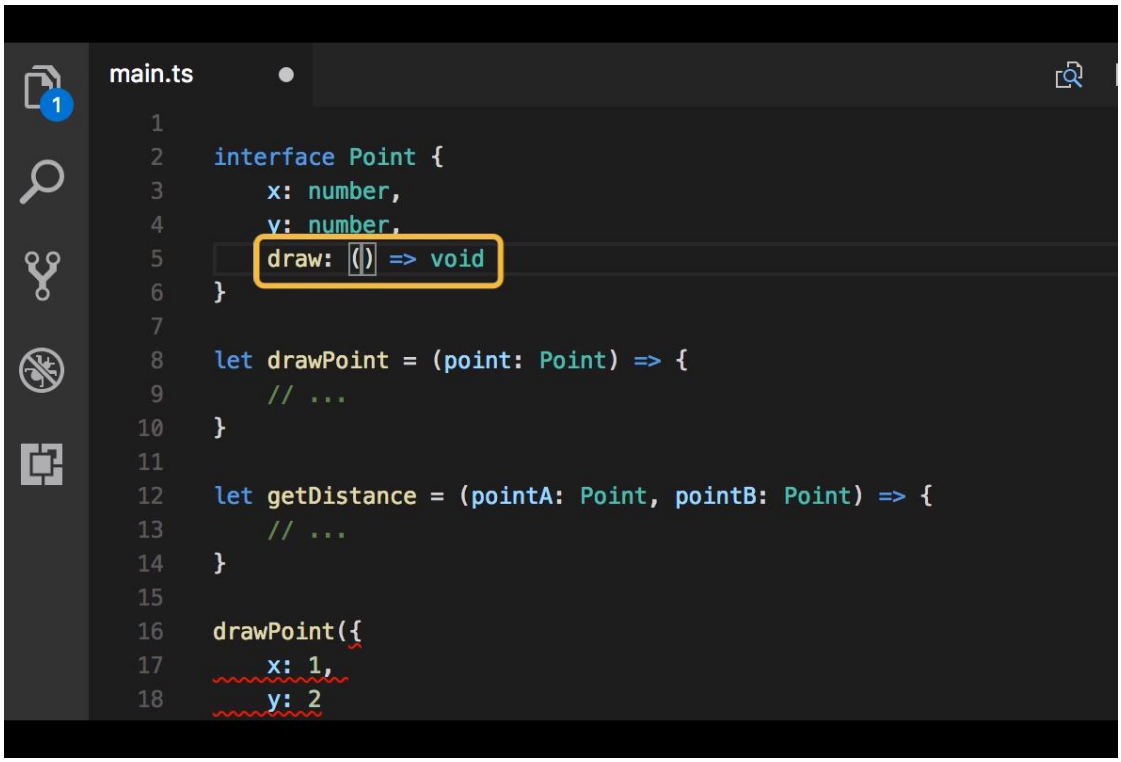
This code violates the principle of cohesion.

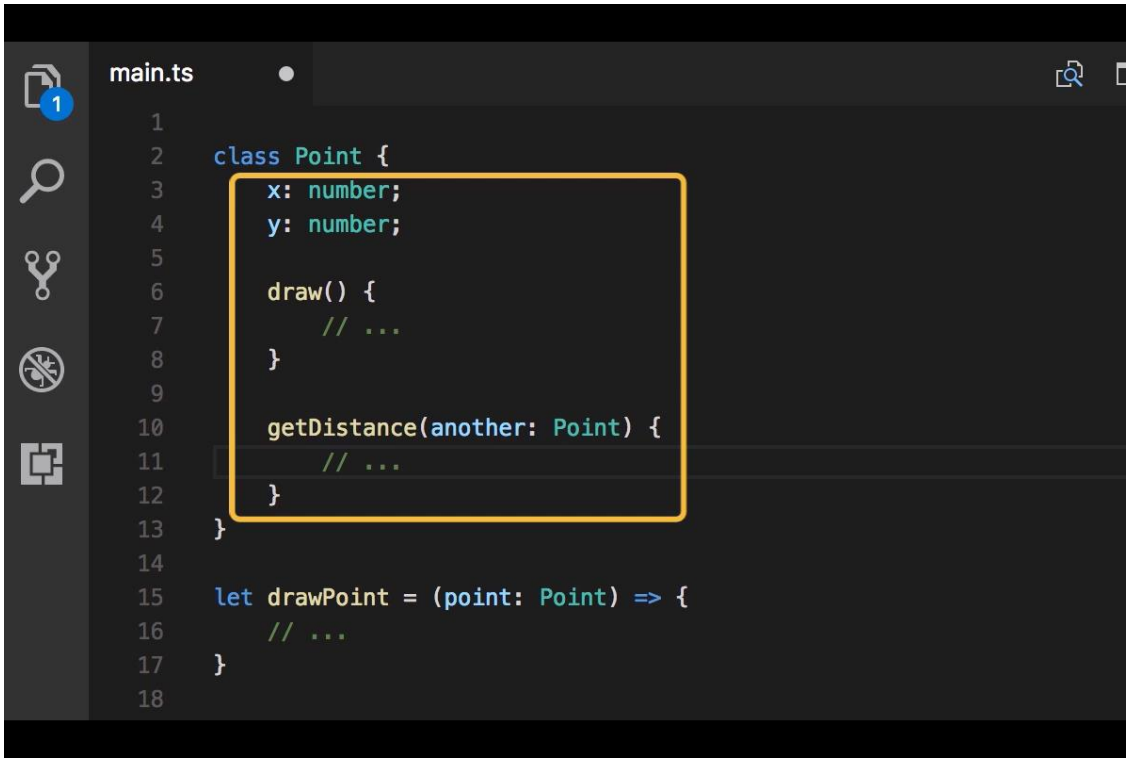


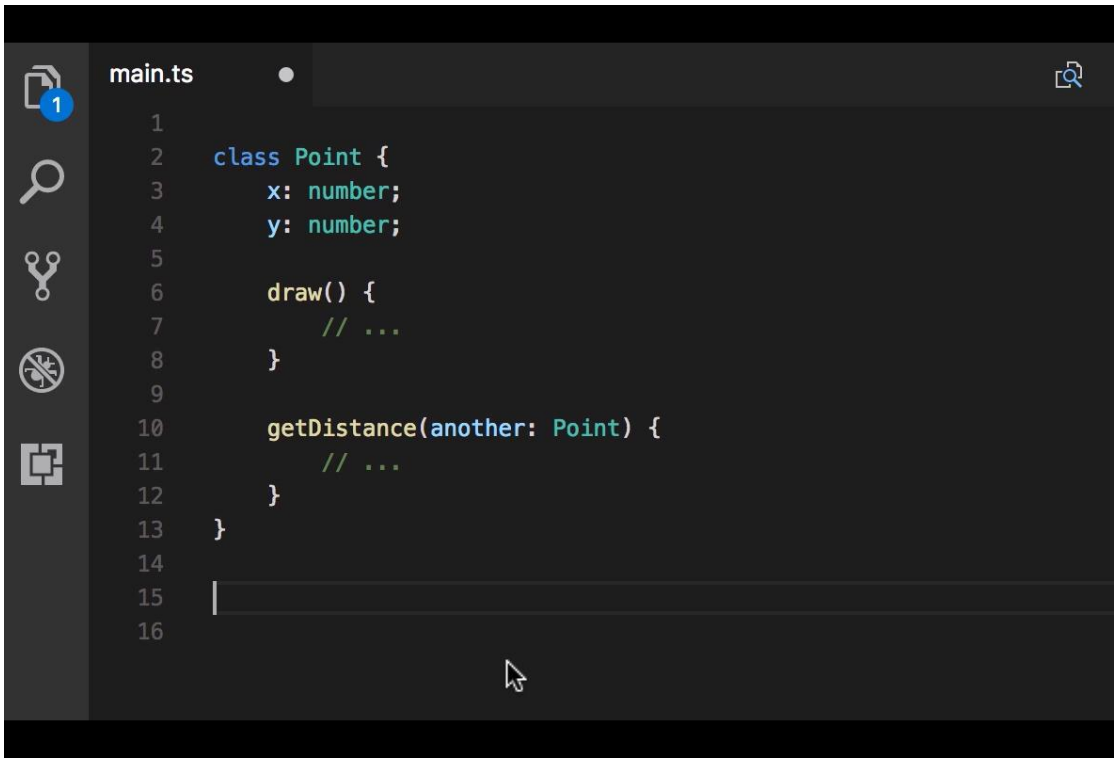
Class

Groups variables (properties) and functions (methods) that are highly related.

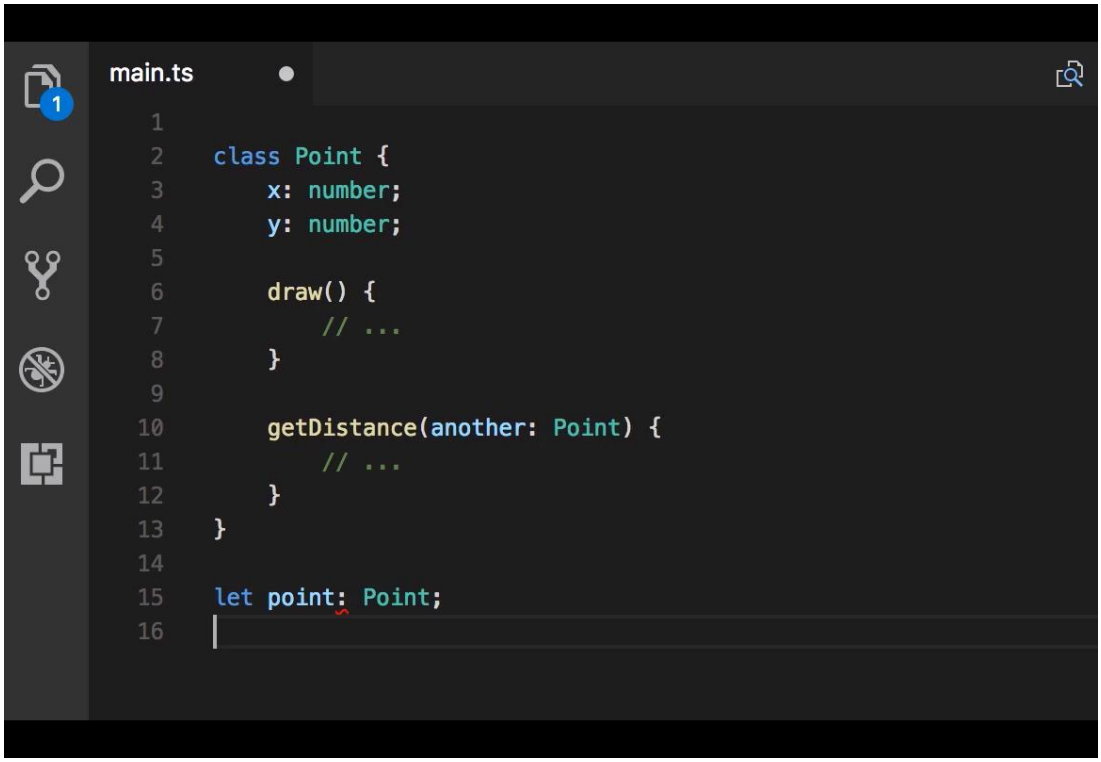
Interfaces have only declarations and not implementations

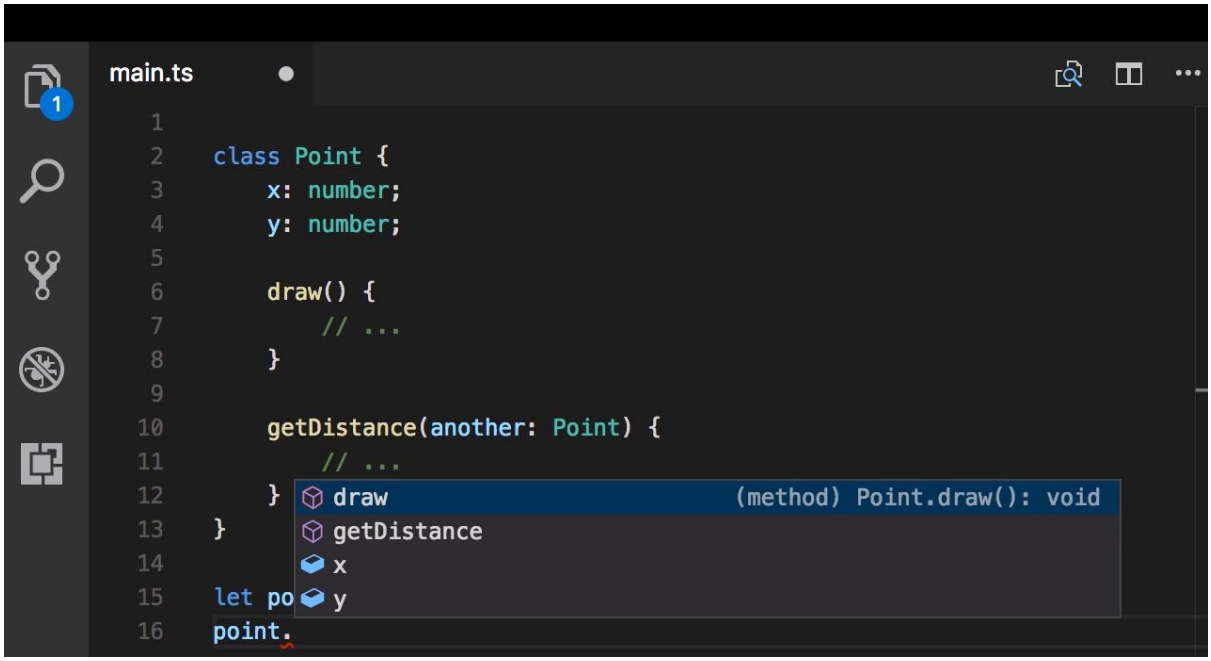


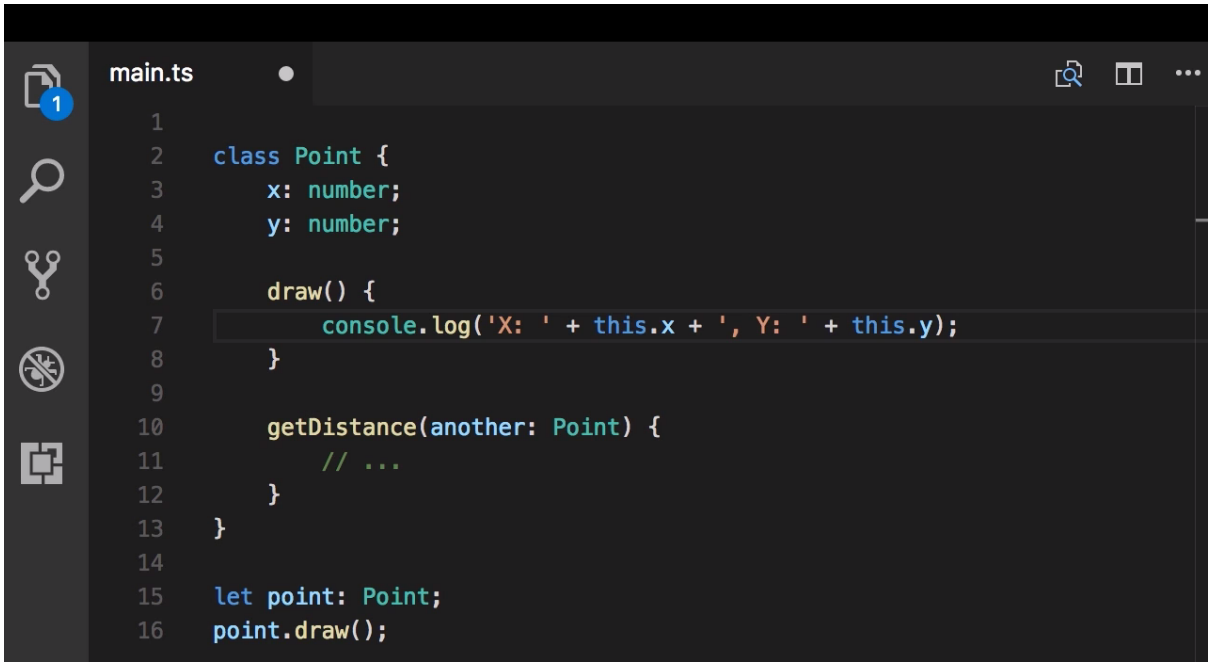

main.ts
1
class Point {
x: number;
y: number;
draw() {
// ...
}
getDistance(another: Point) {
// ...
}
}
let drawPoint = (point: Point) => {
// ...
}


main.ts
1
class Point {
x: number;
y: number;
draw() {
// ...
}
getDistance(another: Point) {
// ...
}
}

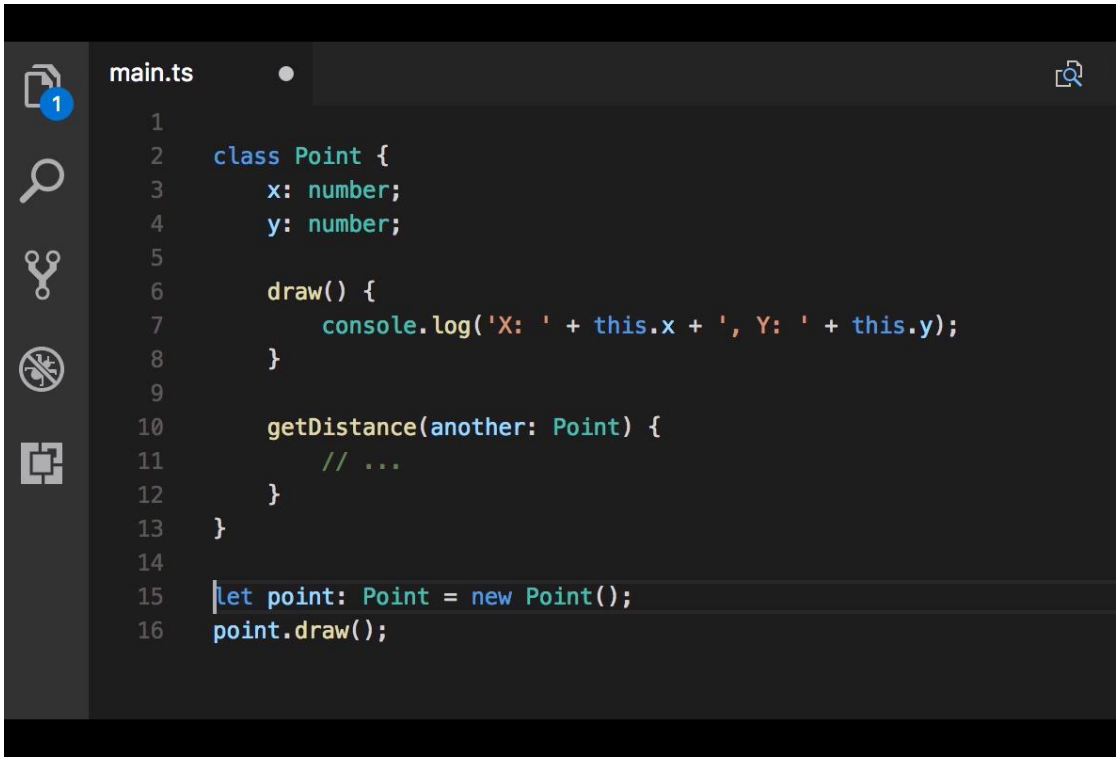
Objects

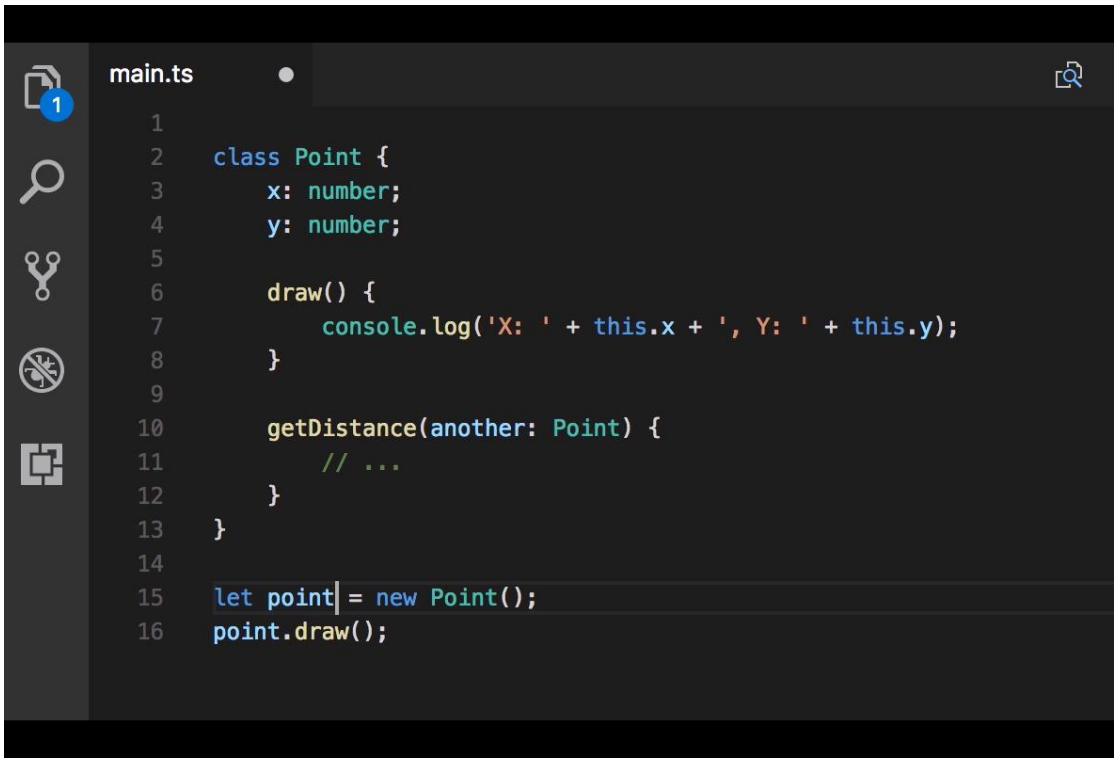

main.ts
1
class Point {
x: number;
y: number;
draw() {
// ...
}
getDistance(another: Point) {
// ...
}
}
let point: Point;


main.ts
1
class Point {
x: number;
y: number;
draw() {
// ...
}
getDistance(another: Point) {
// ...
}
}
let po
point.
draw
(method) Point.draw(): void
getDistance
x
y

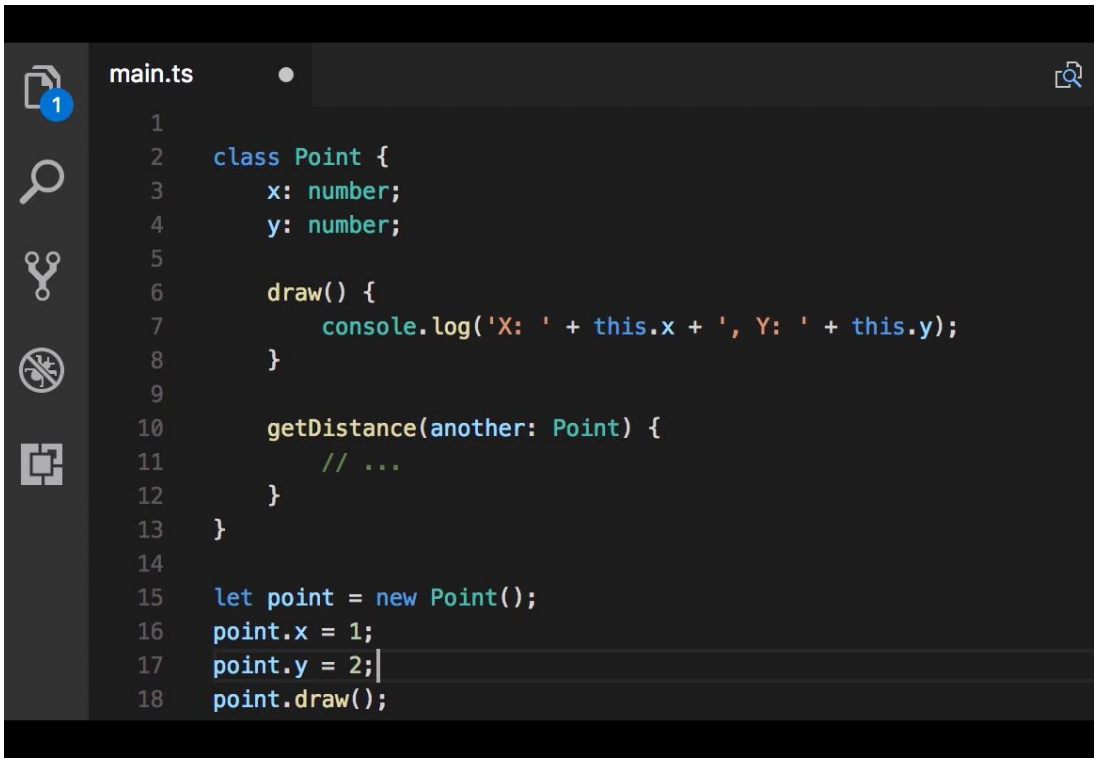



```
ts-hello $
ts-hello $tsc main.ts | node main.js
/Users/moshfeghhamedani/Desktop/Angular 4/code/ts-hello/main.js:13
point.draw();
     ^

TypeError: Cannot read property 'draw' of undefined
    at Object.<anonymous> (/Users/moshfeghhamedani/Desktop/Angular
de/ts-hello/main.js:13:6)
    at Module._compile (module.js:570:32)
    at Object.Module._extensions..js (module.js:579:10)
    at Module.load (module.js:487:32)
    at tryModuleLoad (module.js:446:12)
    at Function.Module._load (module.js:438:3)
    at Module.runMain (module.js:604:10)
    at run (bootstrap_node.js:390:7)
    at startup (bootstrap_node.js:150:9)
    at bootstrap_node.js:505:3
ts-hello $
```


main.ts
1
class Point {
x: number;
y: number;
draw() {
console.log('X: ' + this.x + ', Y: ' + this.y);
}
getDistance(another: Point) {
// ...
}
}
let point: Point = new Point();
point.draw();

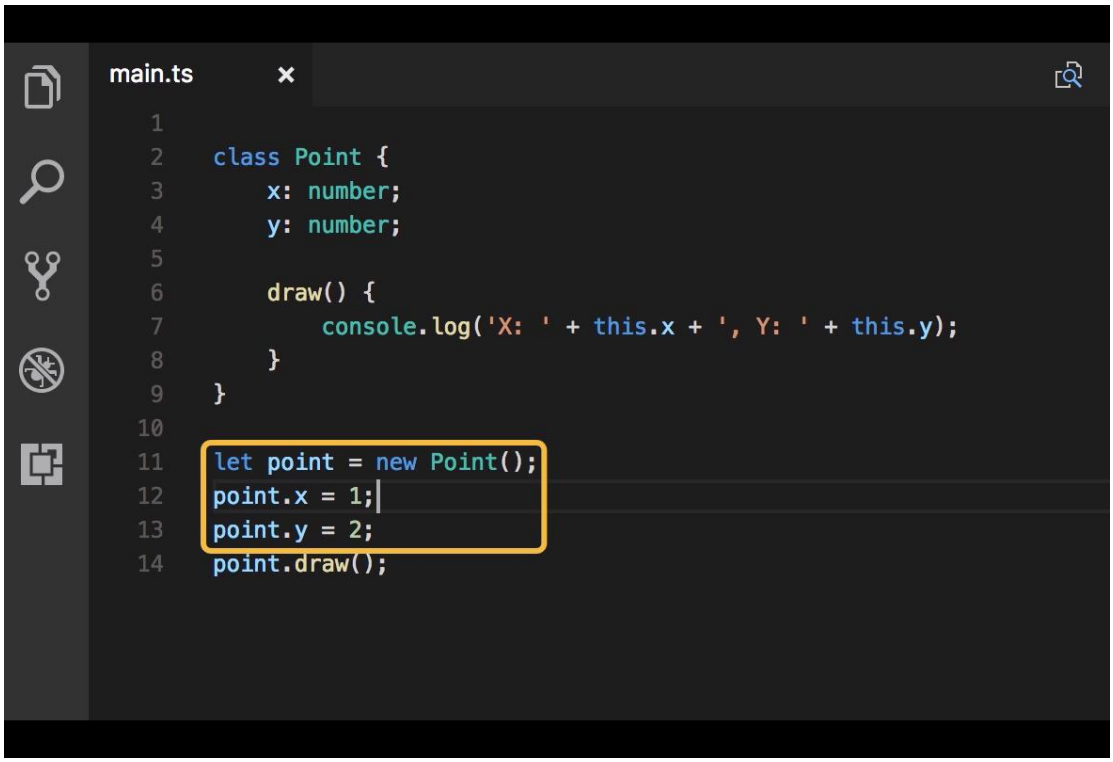

main.ts
1
class Point {
x: number;
y: number;
draw() {
console.log('X: ' + this.x + ', Y: ' + this.y);
}
getDistance(another: Point) {
// ...
}
}
let point = new Point();
point.draw();

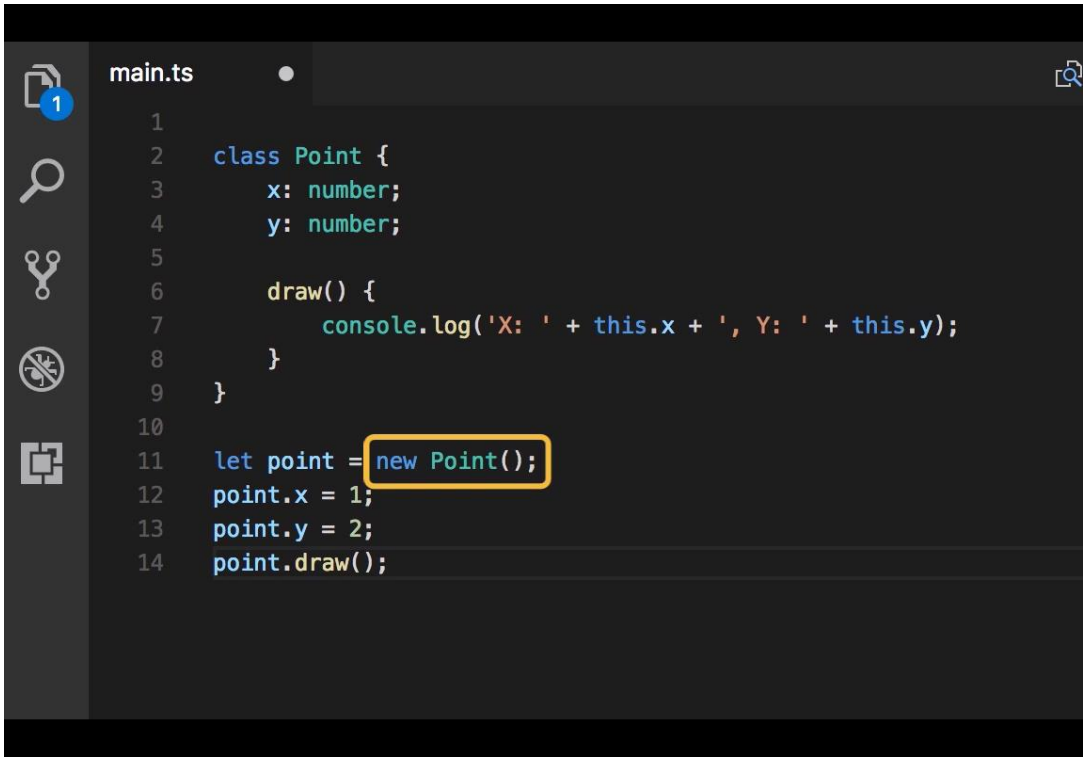
```
ts-hello $
ts-hello $tsc main.ts
ts-hello $node main.js
X: undefined, Y: undefined
ts-hello $
```

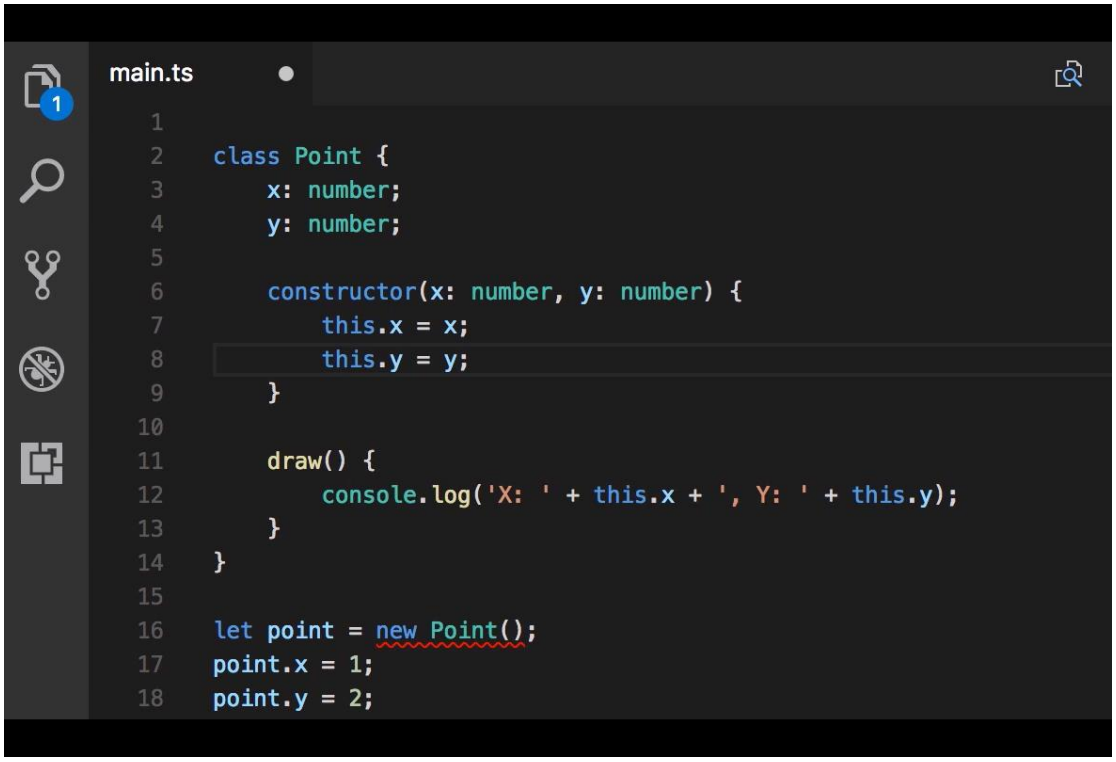


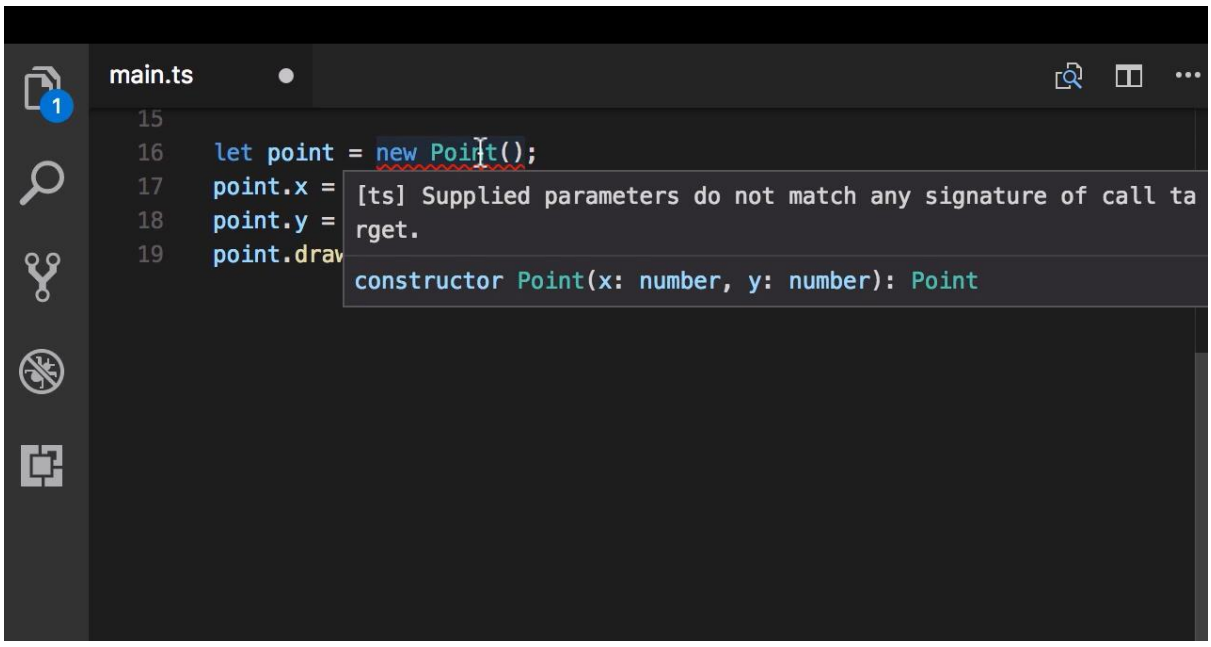

ts-hello $
ts-hello $tsc main.ts && node main.js
X: 1, Y: 2
ts-hello $

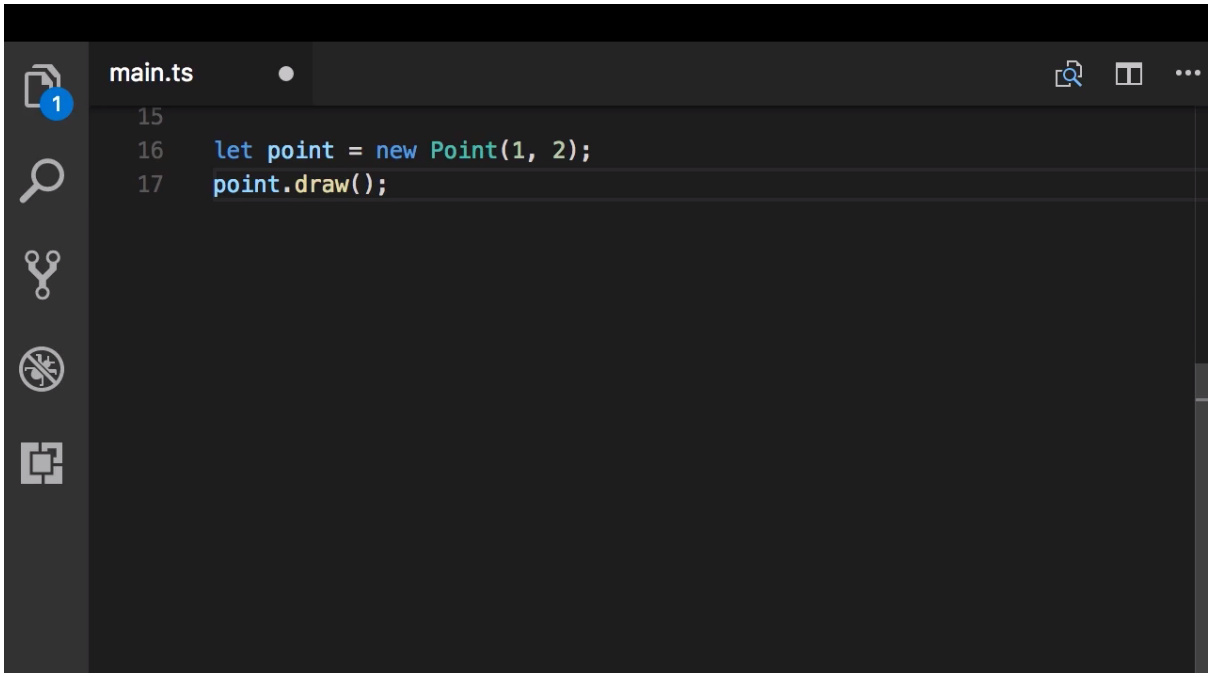
Constructors

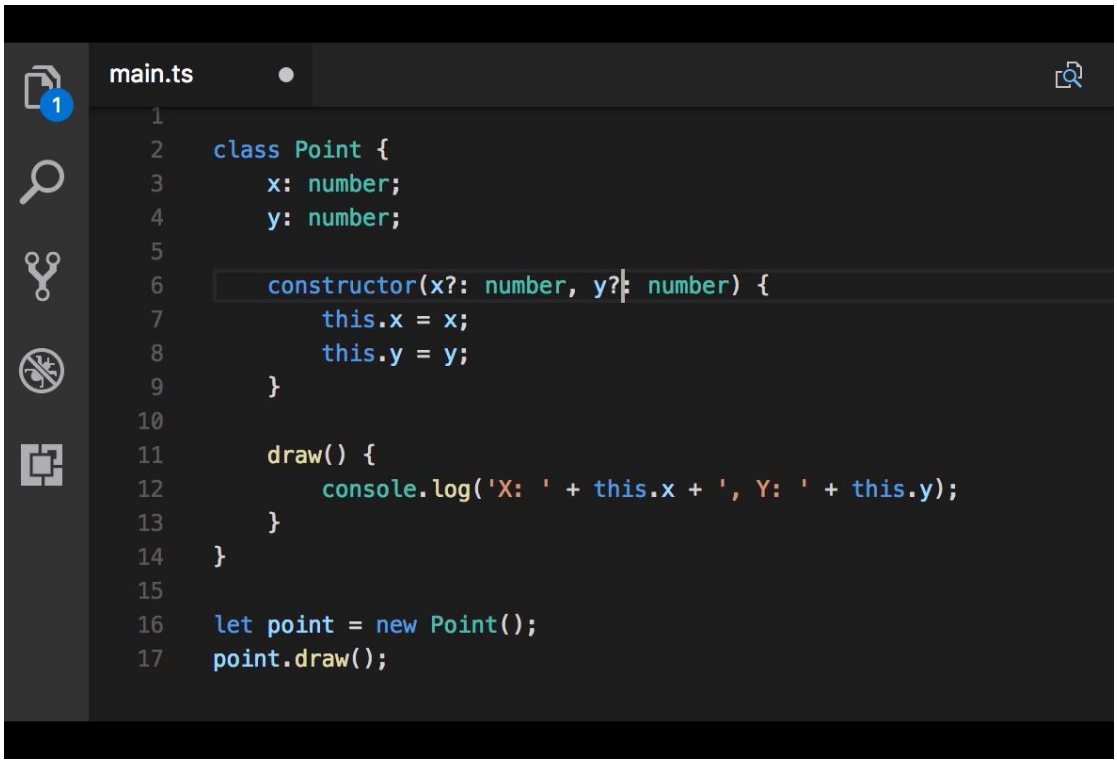




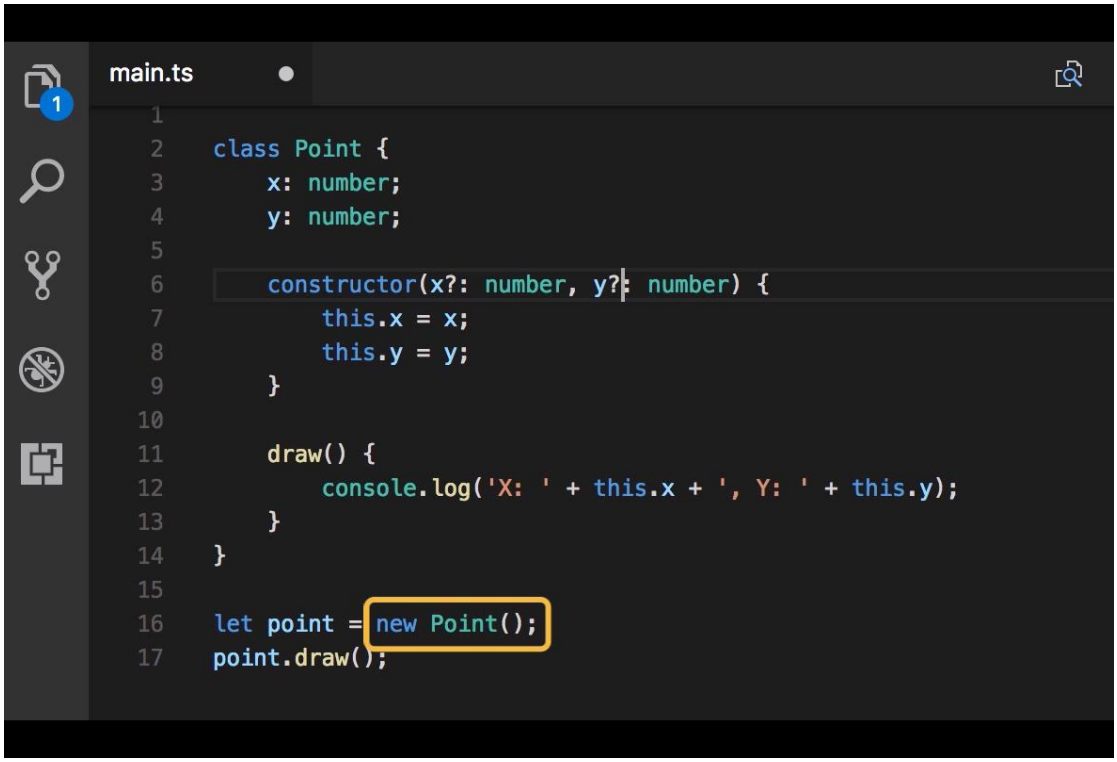

main.ts
1
class Point {
x: number;
y: number;
constructor(x: number, y: number) {
this.x = x;
this.y = y;
}
draw() {
console.log('X: ' + this.x + ', Y: ' + this.y);
}
}
let point = new Point();
point.x = 1;
point.y = 2;


main.ts
1
let point = new Point();
point.x =
point.y =
point.draw
[ts] Supplied parameters do not match any signature of call target.
constructor Point(x: number, y: number): Point

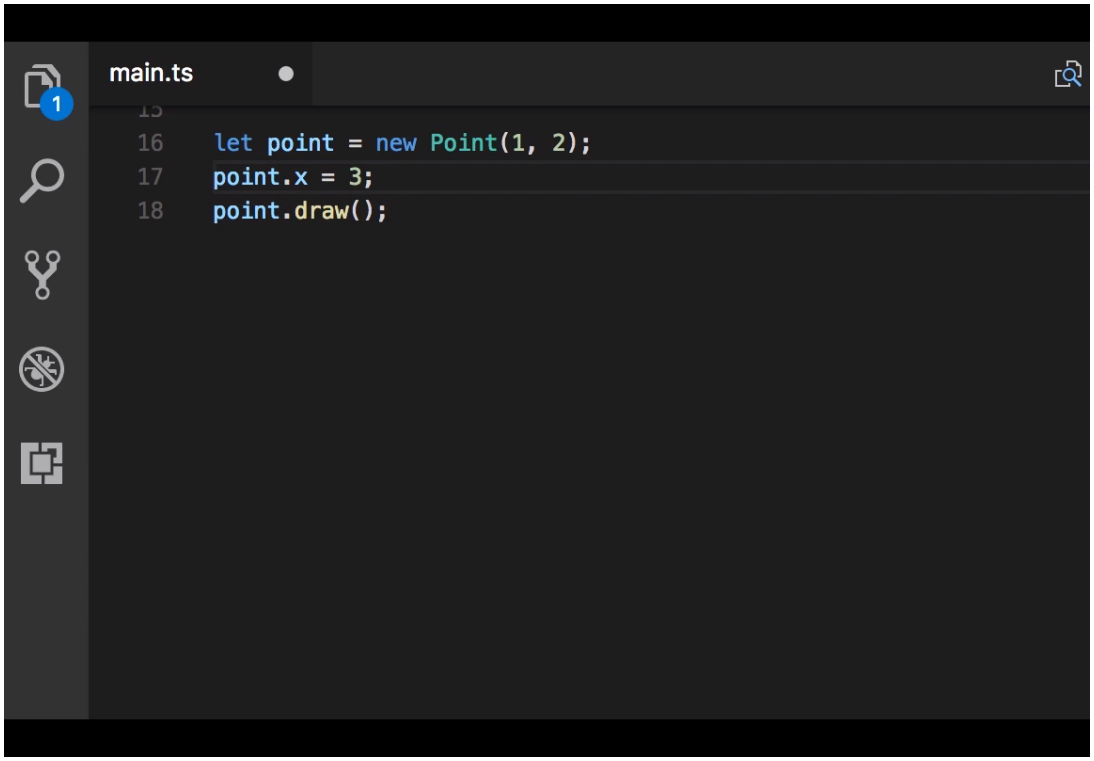




All the parameters on the right side of the first optional parameters should also be optional.


main.ts
1
class Point {
x: number;
y: number;
constructor(x?: number, y?: number) {
this.x = x;
this.y = y;
}
draw() {
console.log('X: ' + this.x + ', Y: ' + this.y);
}
}
let point = new Point();
point.draw();

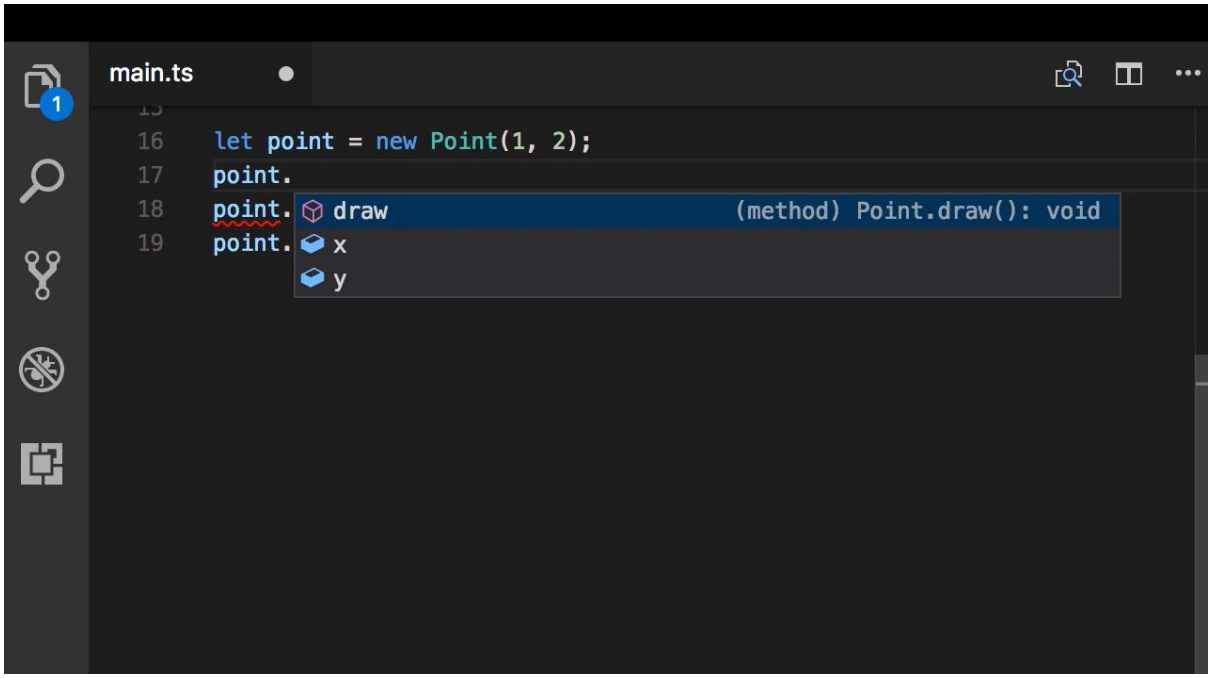
Access Modifiers

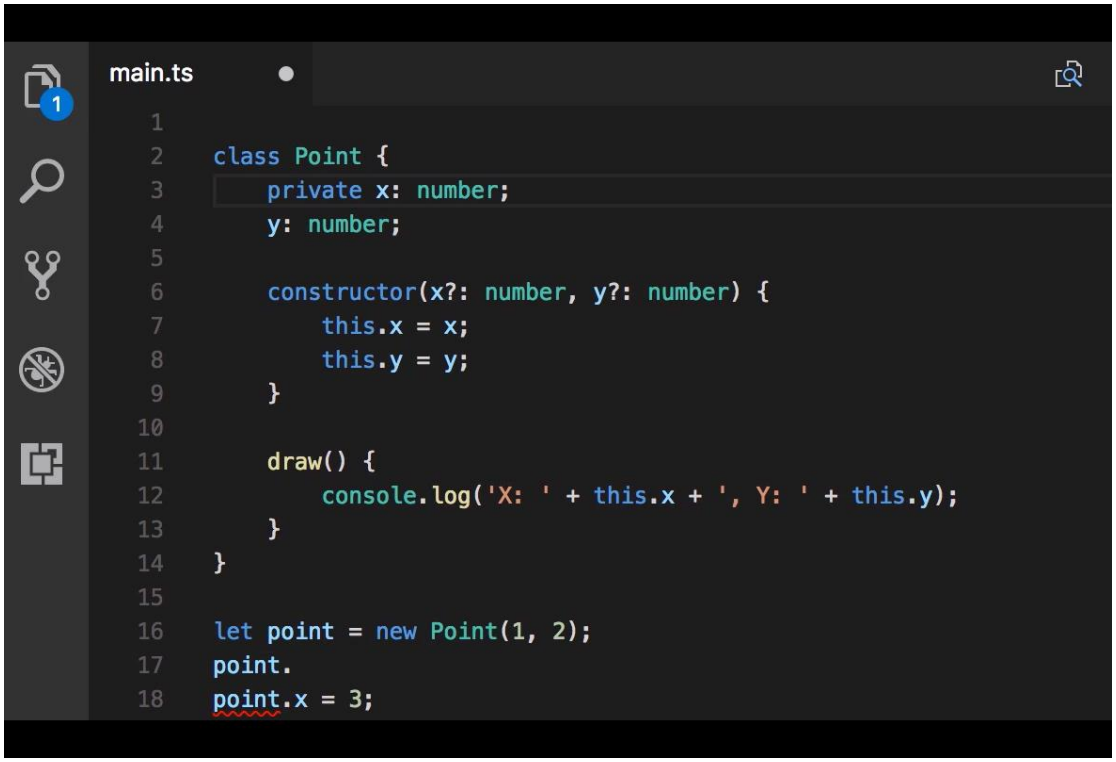


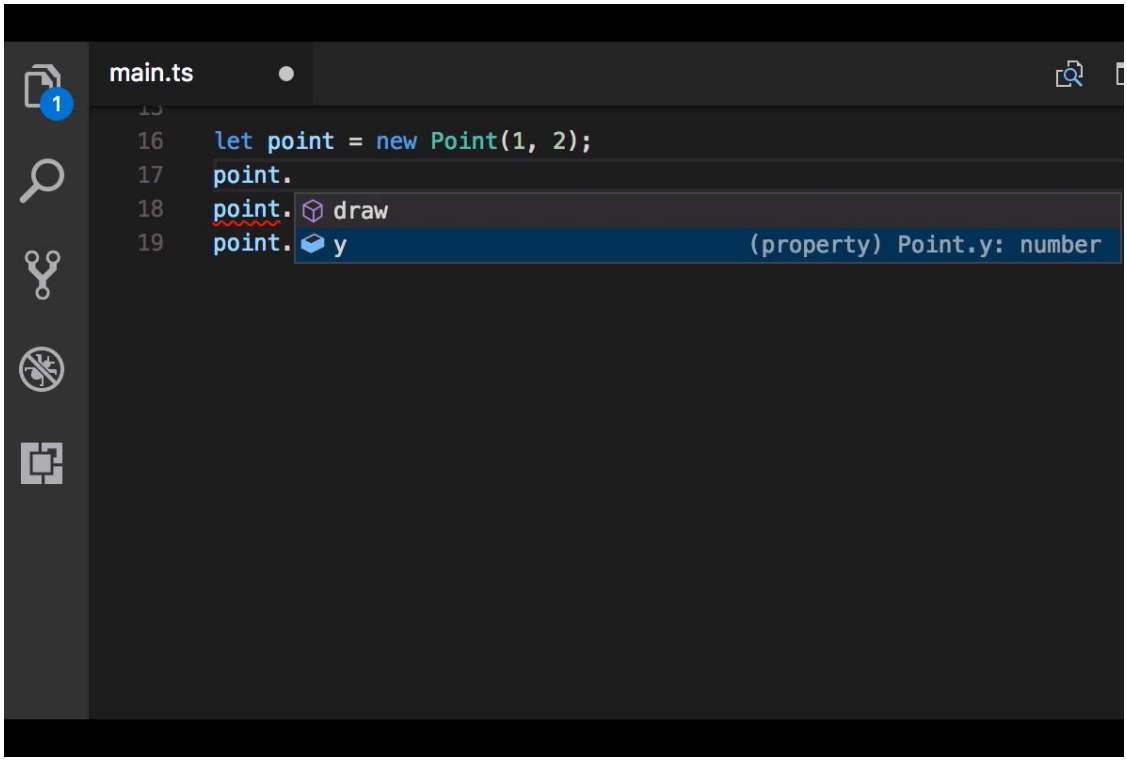
Access Modifiers

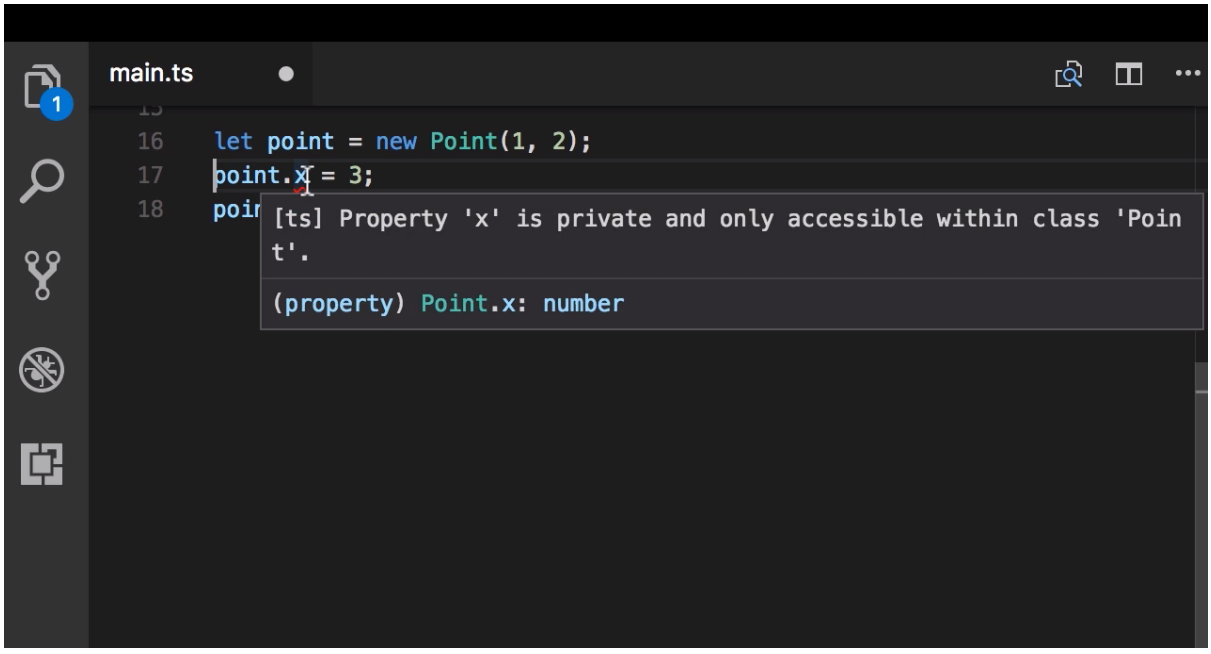
- public
- private
- protected

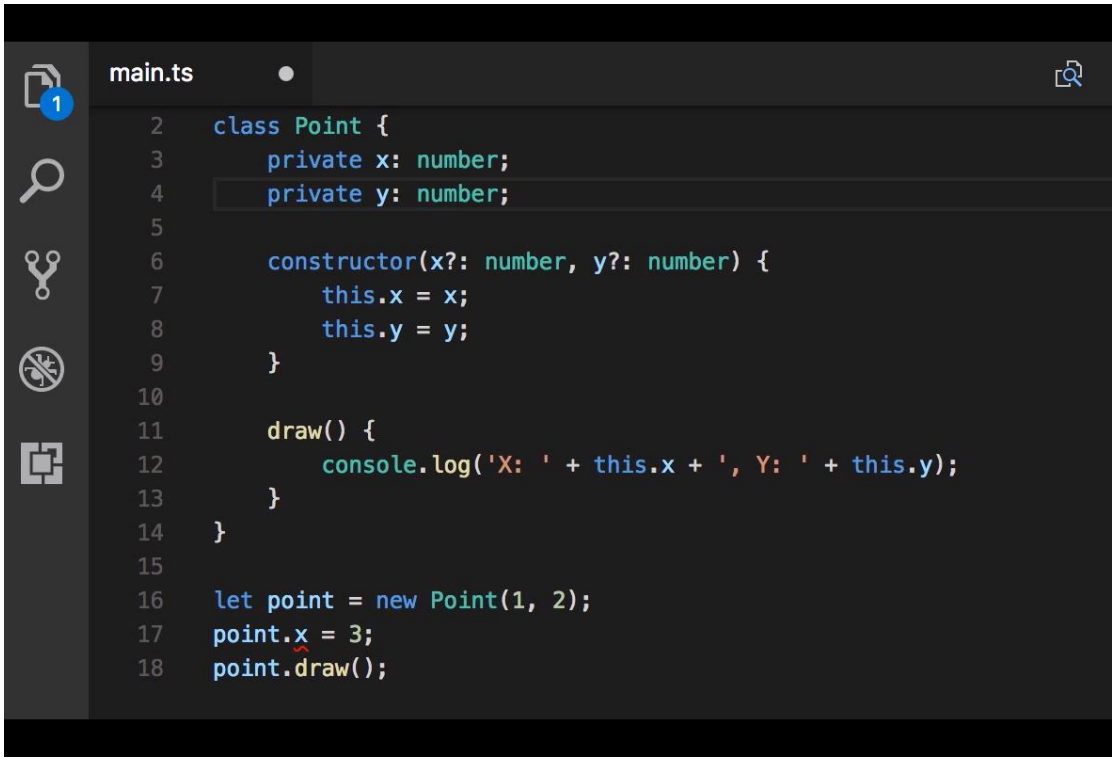
By default all public

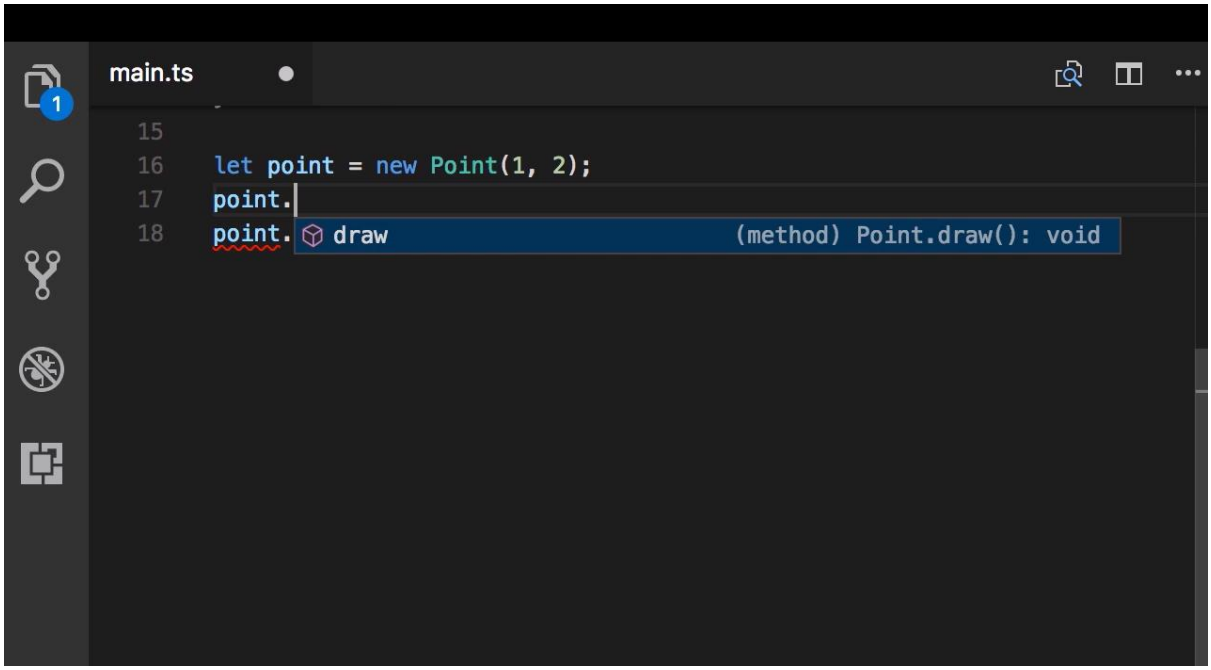

main.ts
let point = new Point(1, 2);
point.
point.
point.
draw
(method) Point.draw(): void
x
y

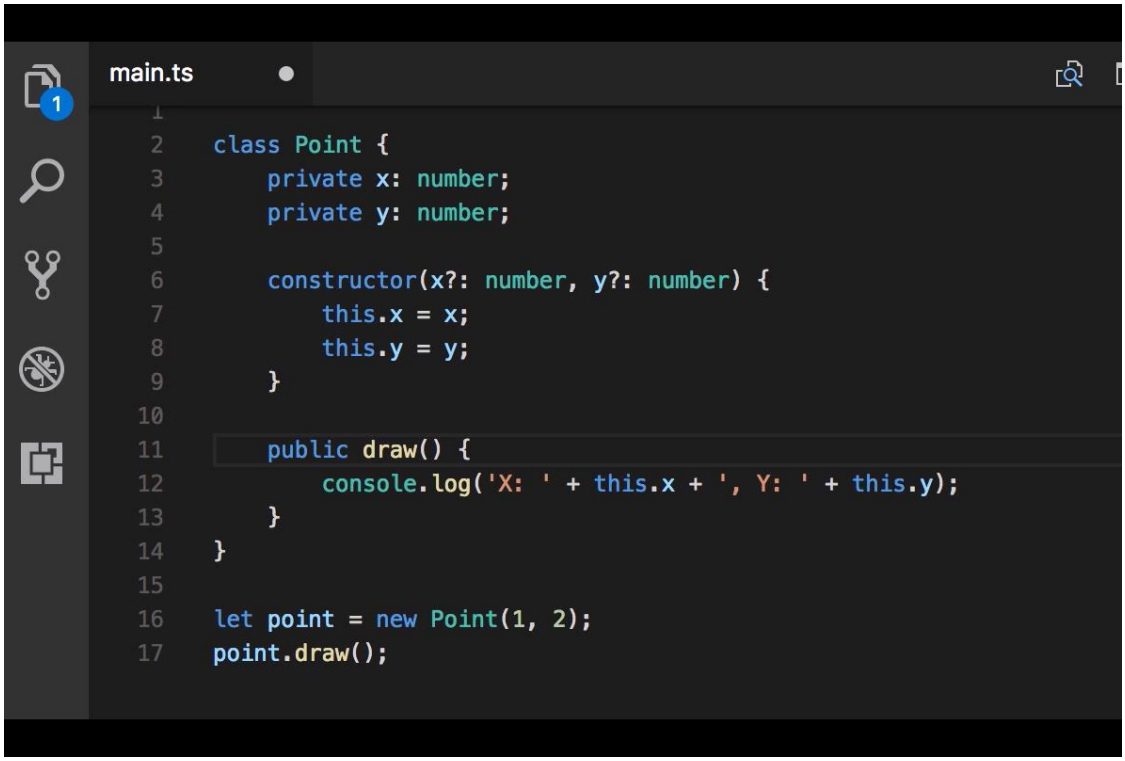

main.ts
class Point {
private x: number;
y: number;
constructor(x?: number, y?: number) {
this.x = x;
this.y = y;
}
draw() {
console.log('X: ' + this.x + ', Y: ' + this.y);
}
}
let point = new Point(1, 2);
point.
point.x = 3;


main.ts
let point = new Point(1, 2);
point.
point.
point.
draw
y
(property) Point.y: number

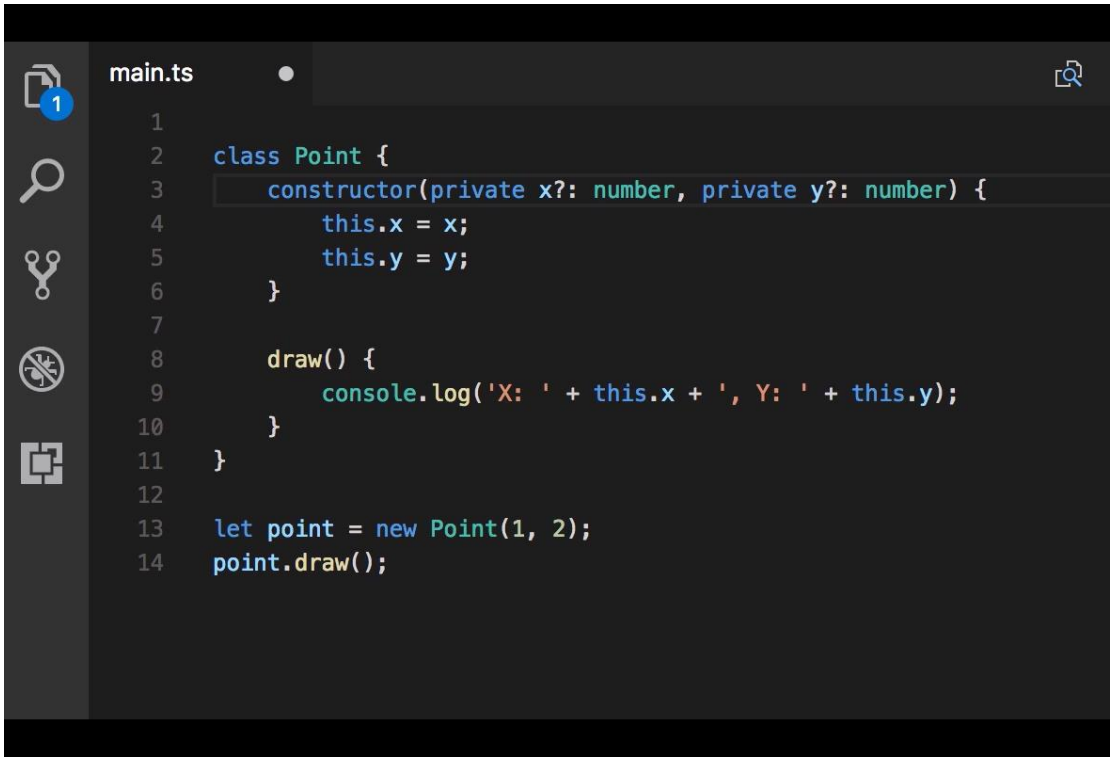

main.ts
let point = new Point(1, 2);
point.x = 3;
[ts] Property 'x' is private and only accessible within class 'Point'.
(property) Point.x: number

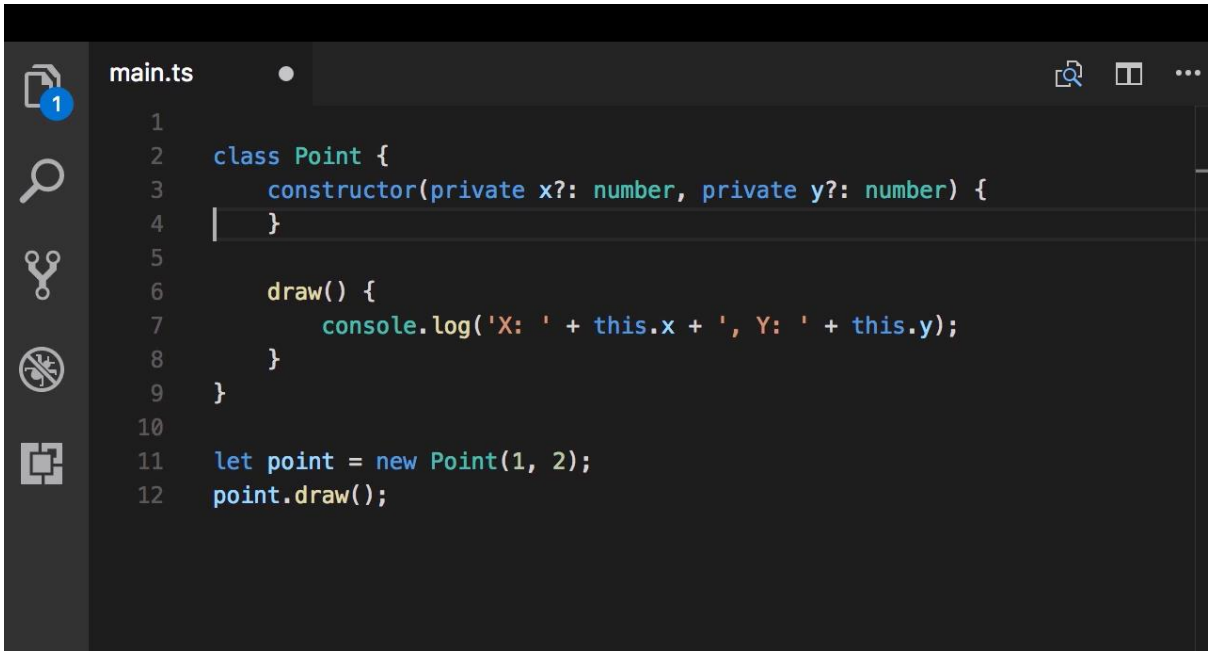

main.ts
class Point {
private x: number;
private y: number;
constructor(x?: number, y?: number) {
this.x = x;
this.y = y;
}
draw() {
console.log('X: ' + this.x + ', Y: ' + this.y);
}
}
let point = new Point(1, 2);
point.x = 3;
point.draw();

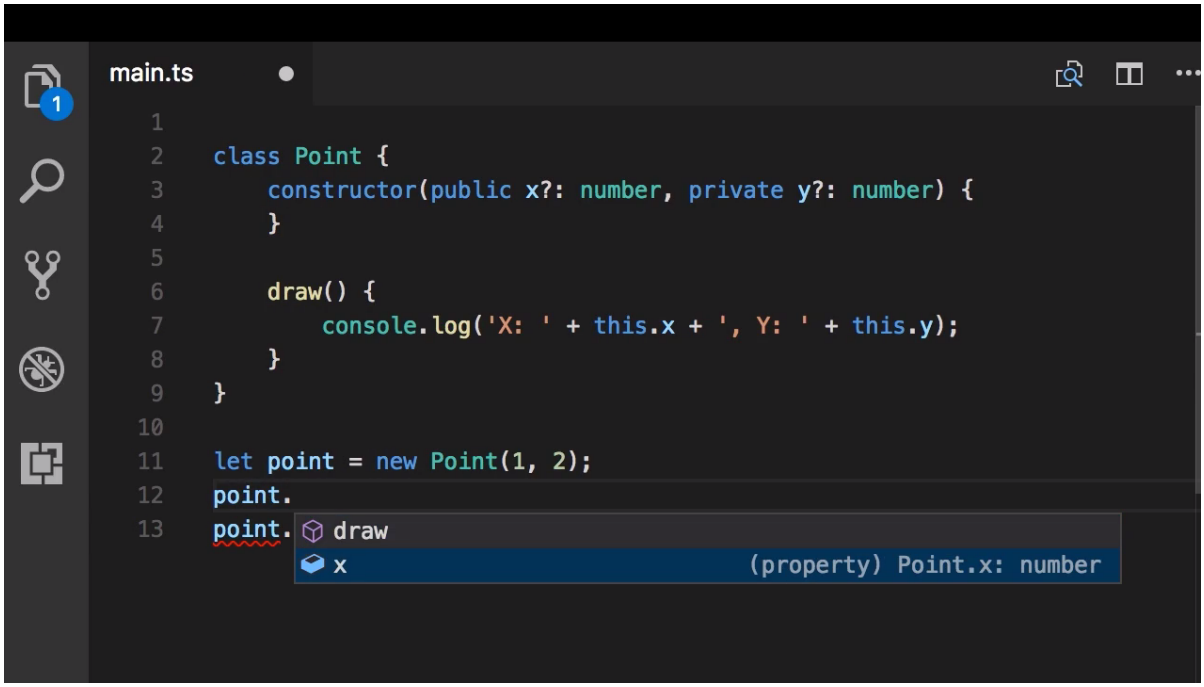

main.ts
let point = new Point(1, 2);
point.
point.
draw
(method) Point.draw(): void


main.ts
1
2 class Point {
3 private x: number;
4 private y: number;
5
6 constructor(x?: number, y?: number) {
7 this.x = x;
8 this.y = y;
9 }
10
11 public draw() {
12 console.log('X: ' + this.x + ', Y: ' + this.y);
13 }
14 }
15
16 let point = new Point(1, 2);
17 point.draw();

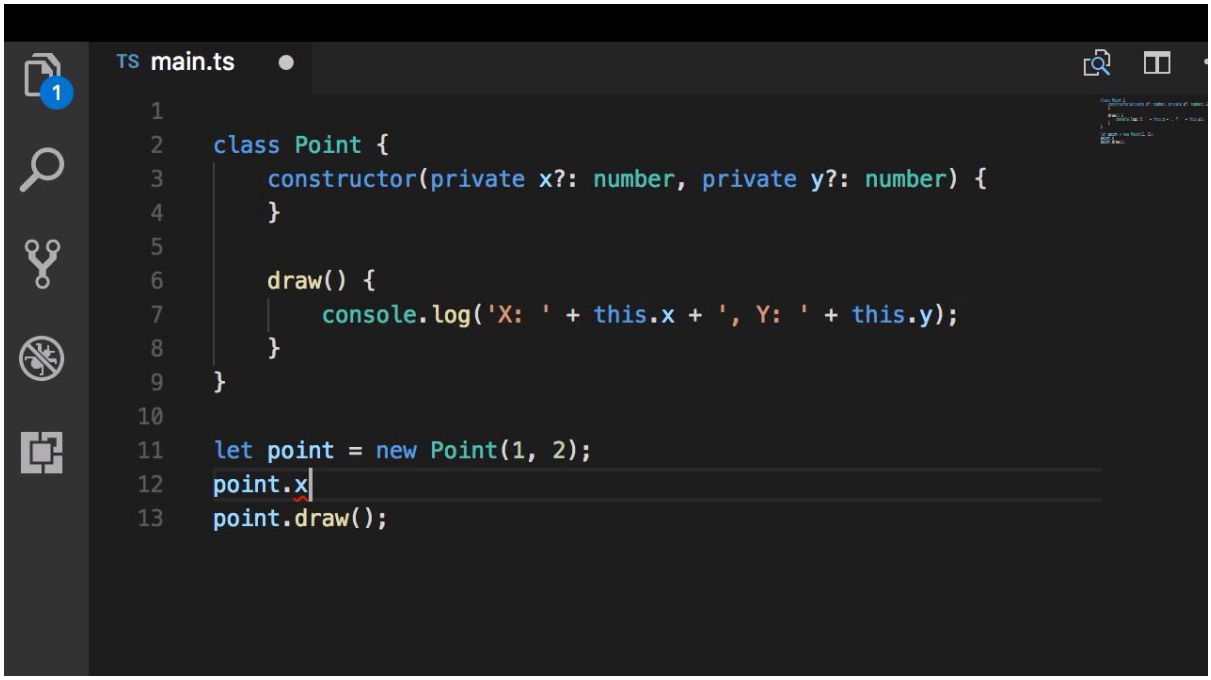
Access Modifiers in Constructor Parameters

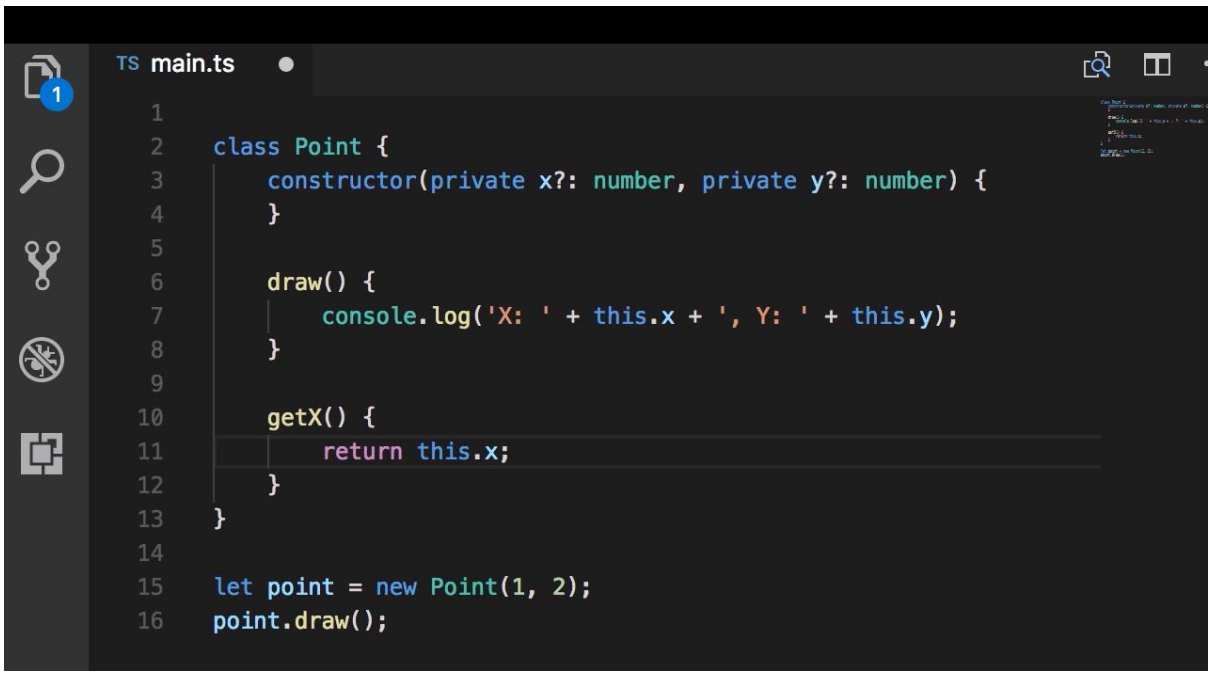


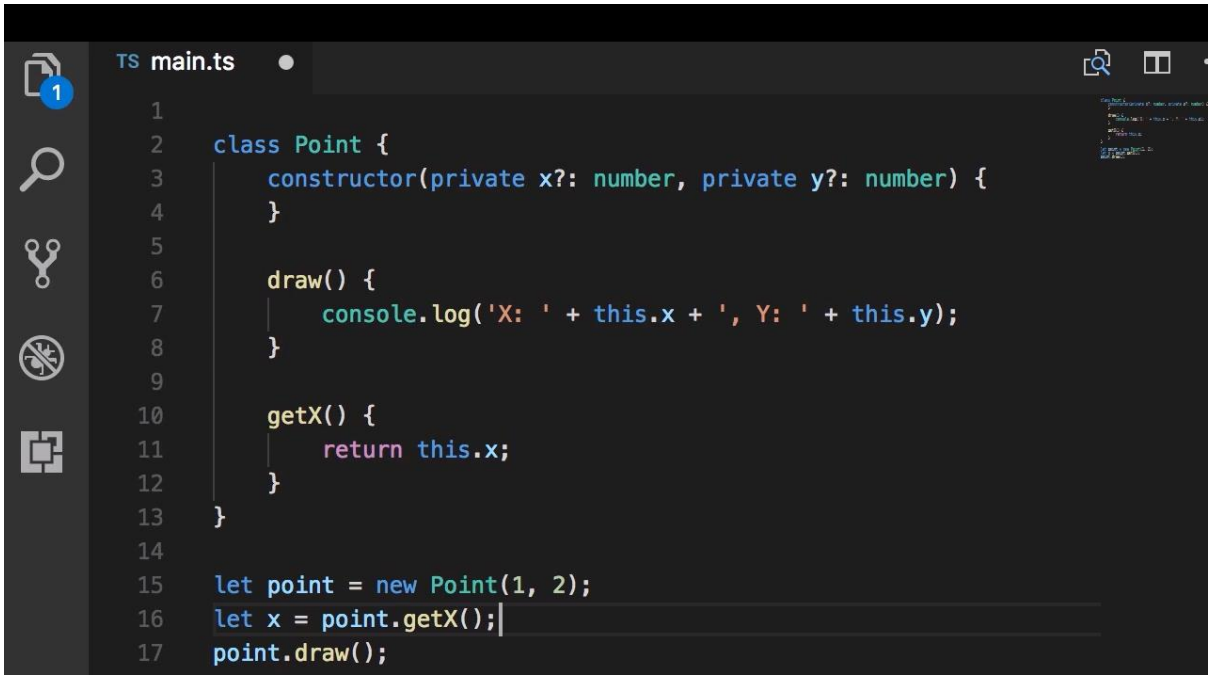


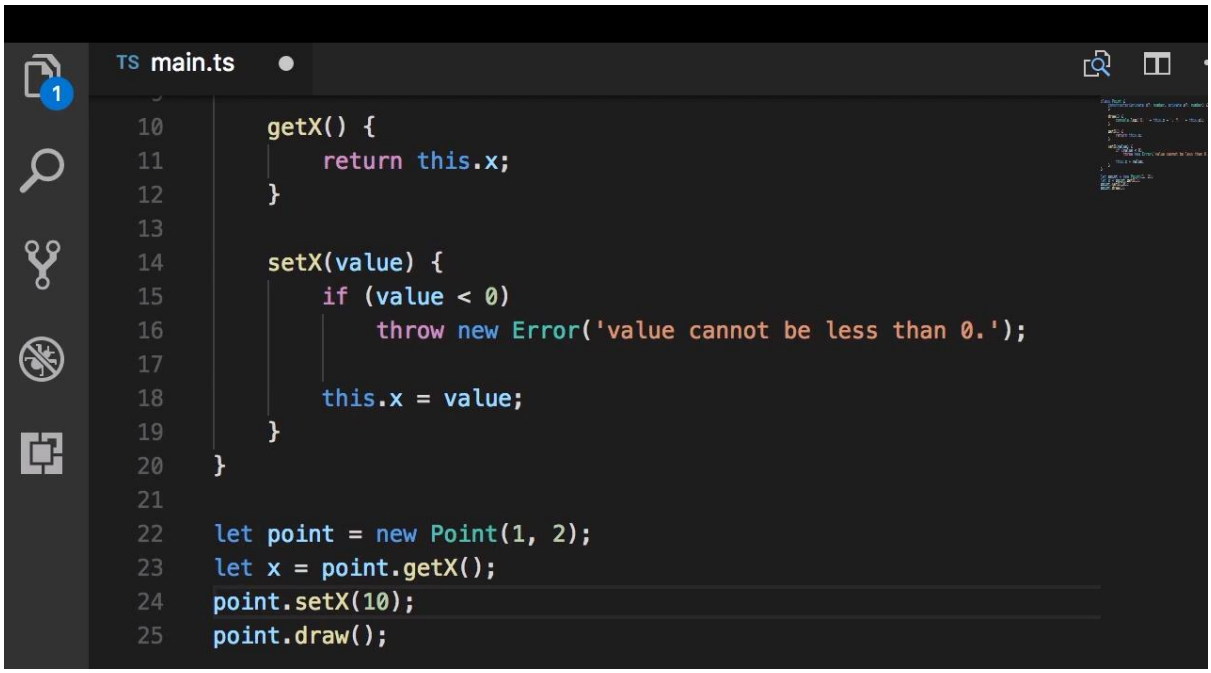

main.ts
1
class Point {
constructor(public x?: number, private y?: number) {
}
draw() {
console.log('X: ' + this.x + ', Y: ' + this.y);
}
}
let point = new Point(1, 2);
point.
point.
draw
x
(property) Point.x: number

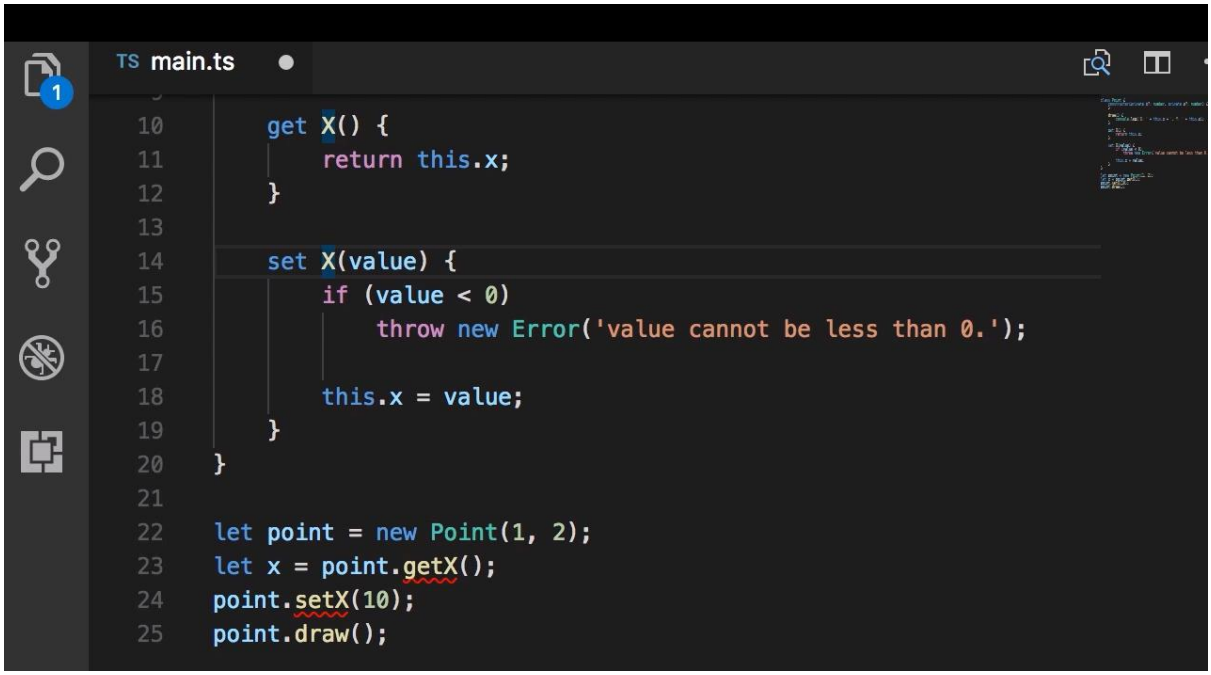
Properties

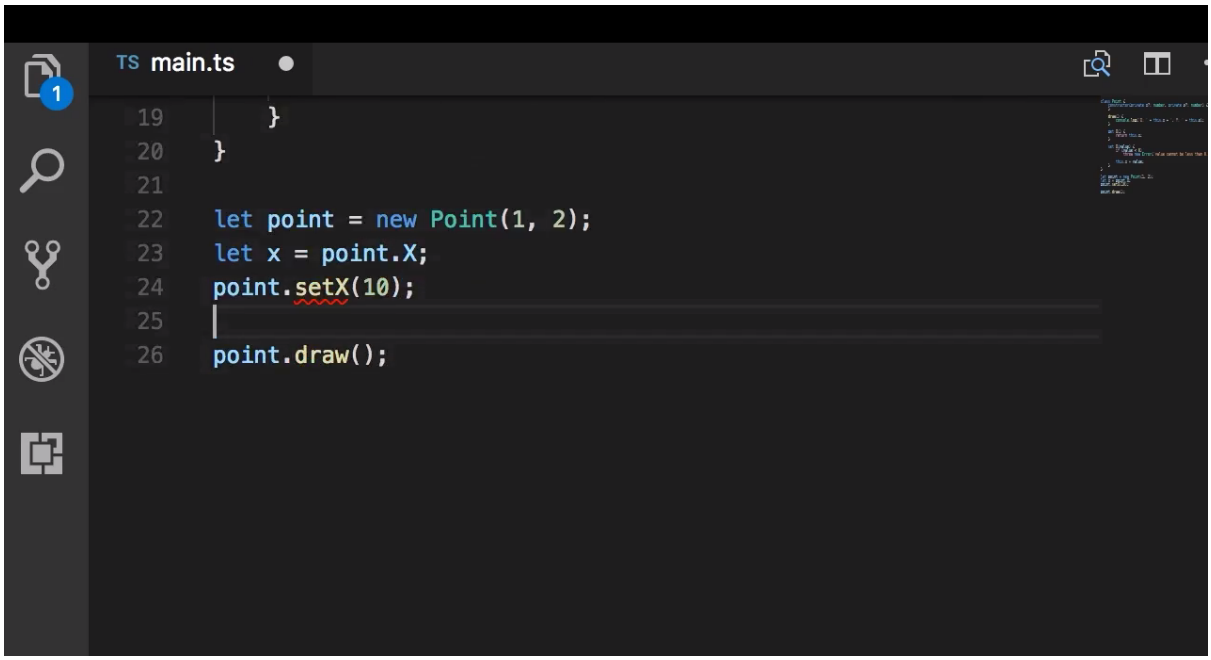

TS main.ts
1
class Point {
constructor(private x?: number, private y?: number) {
}
draw() {
console.log('X: ' + this.x + ', Y: ' + this.y);
}
}
let point = new Point(1, 2);
point.x
point.draw();

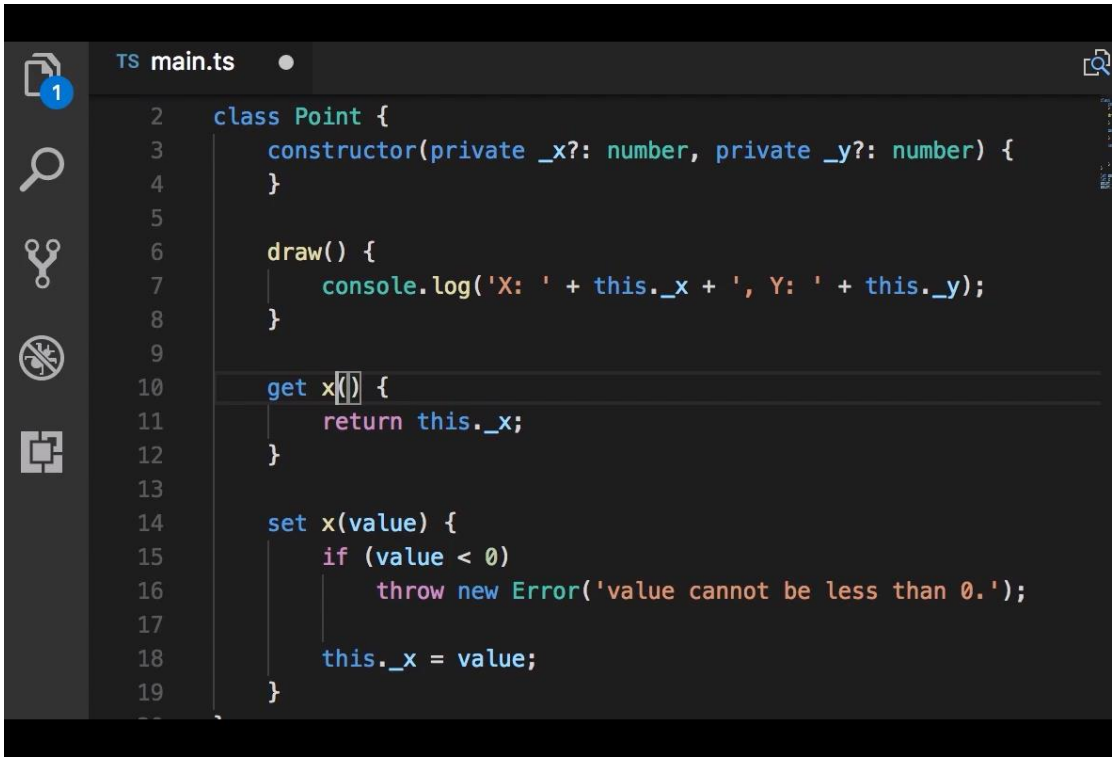

TS main.ts
1
class Point {
constructor(private x?: number, private y?: number) {
}
draw() {
console.log('X: ' + this.x + ', Y: ' + this.y);
}
getX() {
return this.x;
}
}
let point = new Point(1, 2);
point.draw();


TS main.ts
1
class Point {
constructor(private x?: number, private y?: number) {
}
draw() {
console.log('X: ' + this.x + ', Y: ' + this.y);
}
getX() {
return this.x;
}
}
let point = new Point(1, 2);
let x = point.getX();
point.draw();

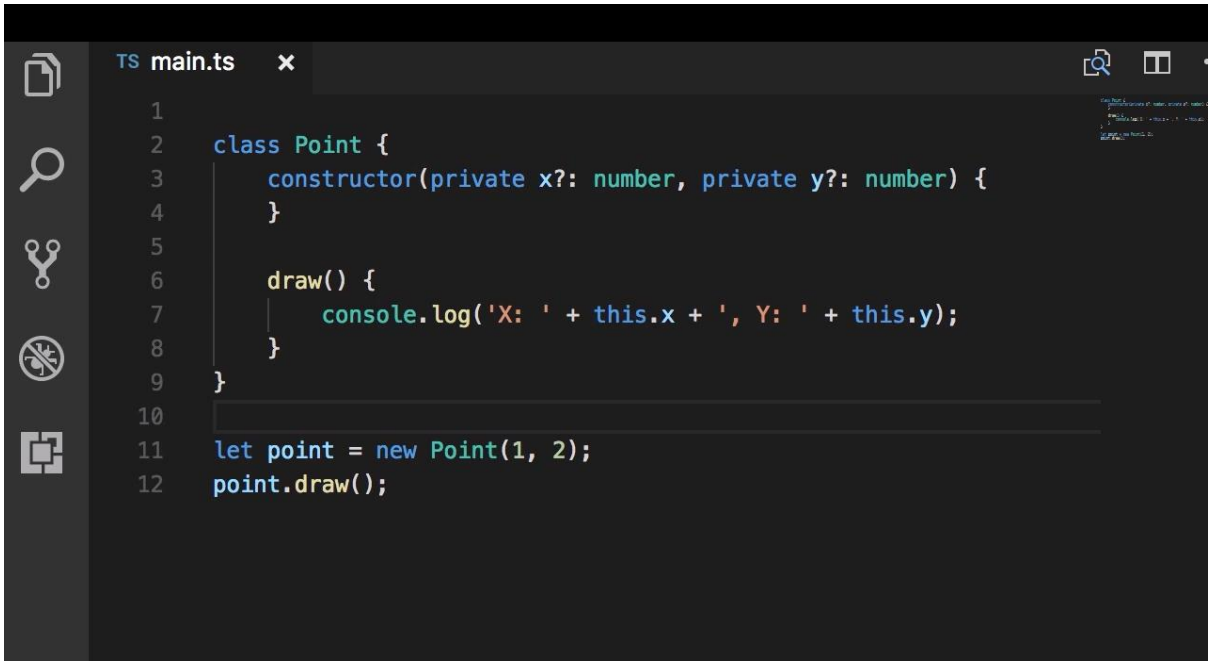

TS main.ts
getX() {
return this.x;
}
setX(value) {
if (value < 0)
throw new Error('value cannot be less than 0.');
this.x = value;
}
}
let point = new Point(1, 2);
let x = point.getX();
point.setX(10);
point.draw();

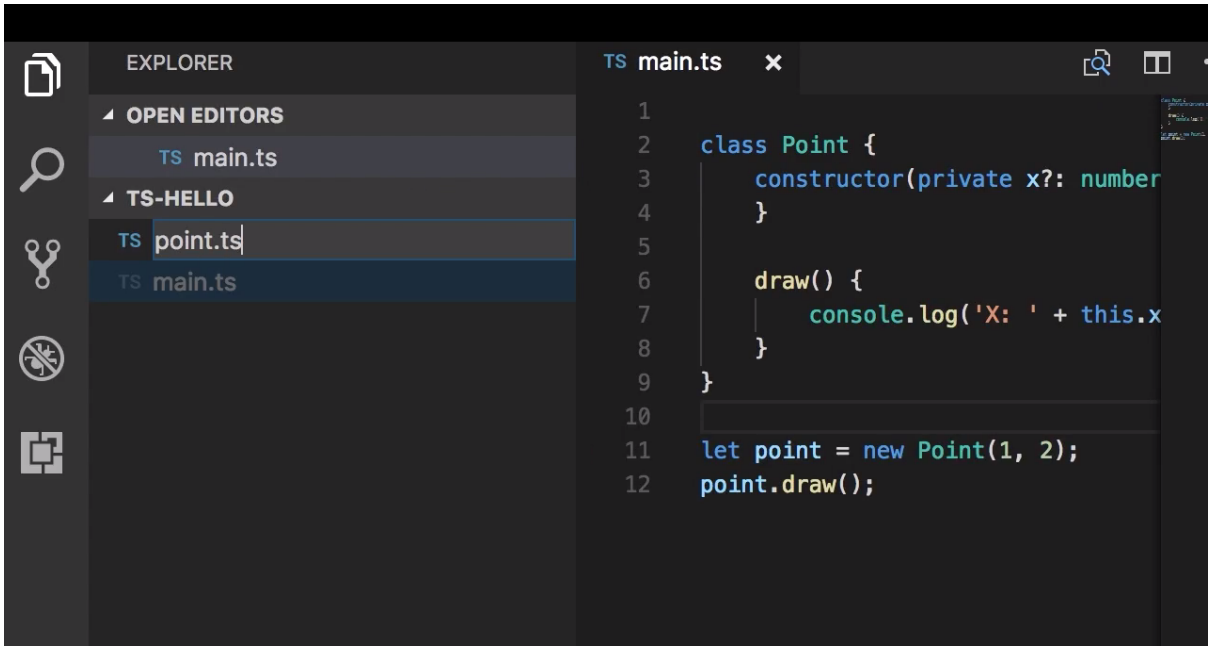

main.ts
get X() {
return this.x;
}
set X(value) {
if (value < 0)
throw new Error('value cannot be less than 0.');
this.x = value;
}
}
let point = new Point(1, 2);
let x = point.getX();
point.setX(10);
point.draw();

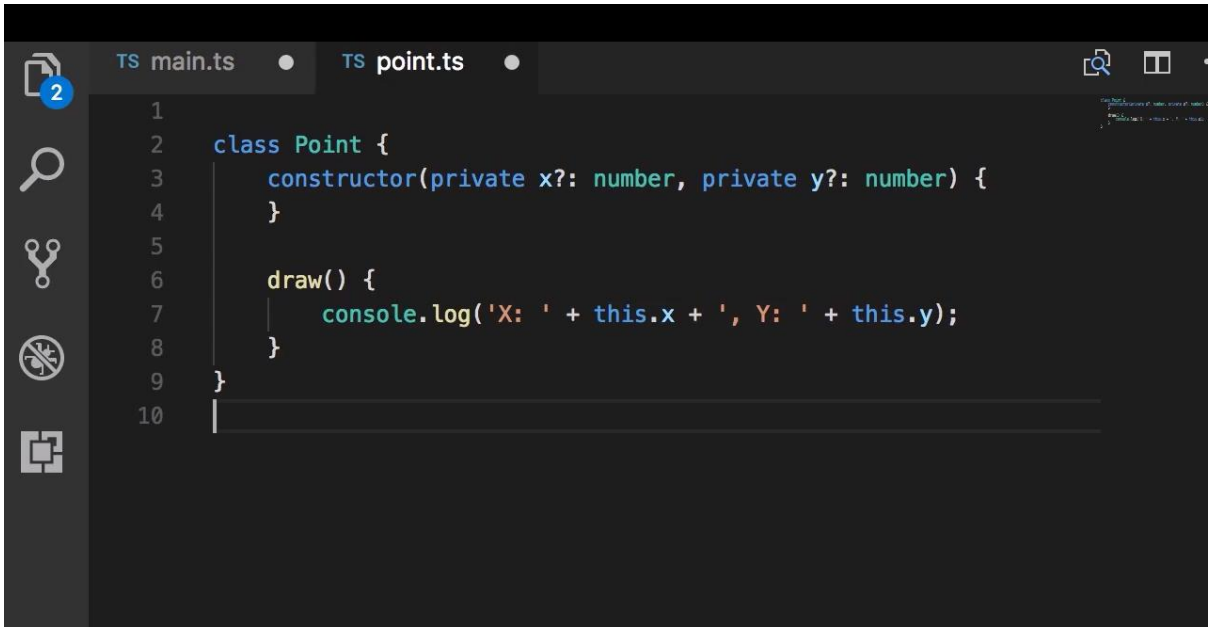

main.ts
}
}
let point = new Point(1, 2);
let x = point.X;
point.setX(10);
point.draw();

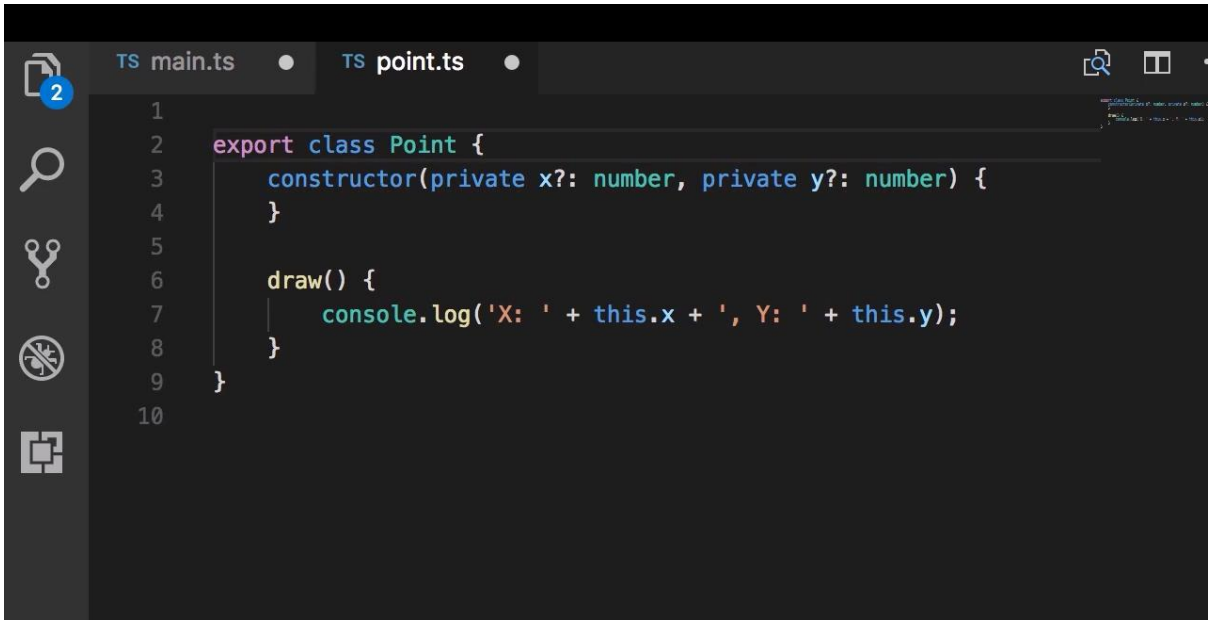

main.ts
class Point {
constructor(private _x?: number, private _y?: number) {
}
draw() {
console.log('X: ' + this._x + ', Y: ' + this._y);
}
get x() {
return this._x;
}
set x(value) {
if (value < 0)
throw new Error('value cannot be less than 0.');
this._x = value;
}

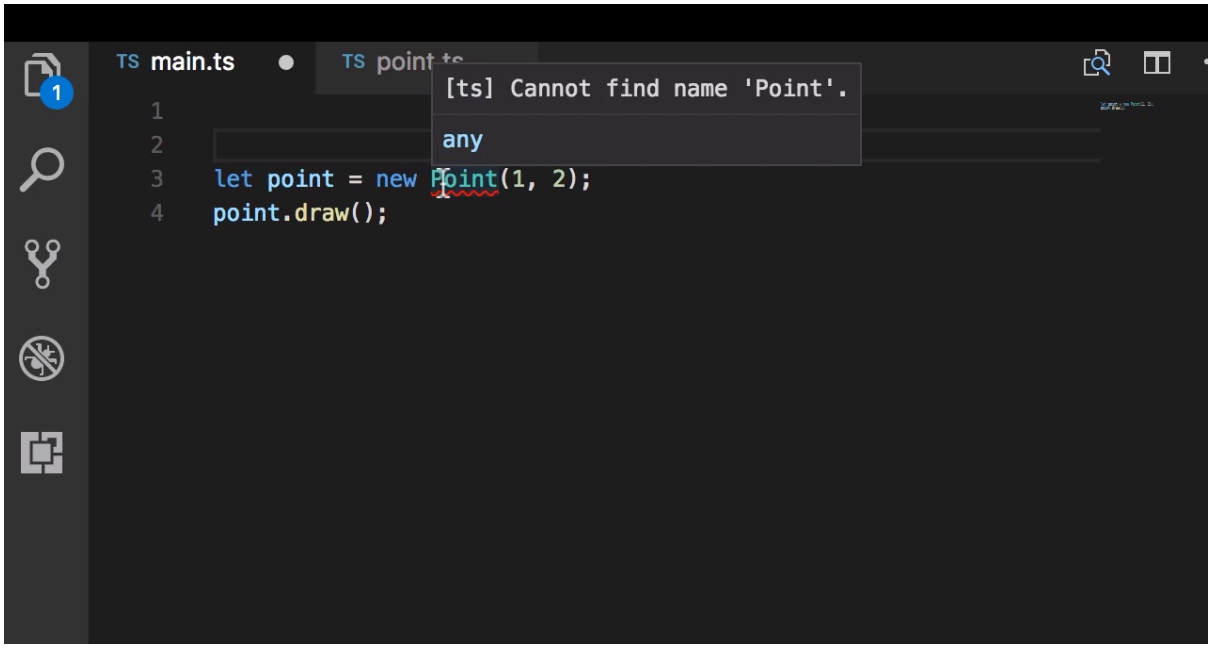
Modules

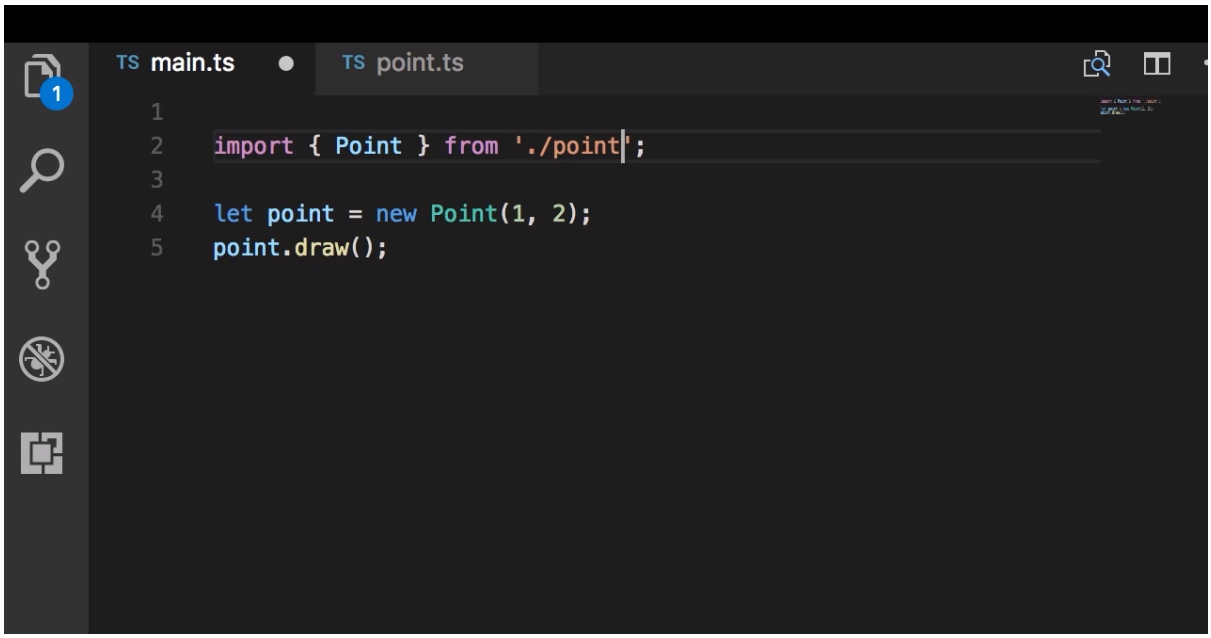

main.ts
class Point {
constructor(private x?: number, private y?: number) {
}
draw() {
console.log('X: ' + this.x + ', Y: ' + this.y);
}
}
let point = new Point(1, 2);
point.draw();

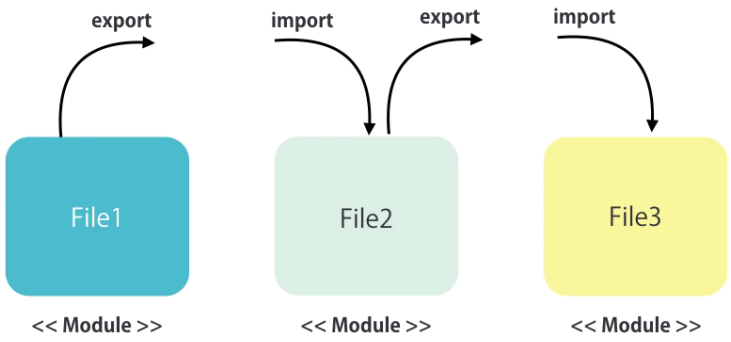

EXPLORER
OPEN EDITORS
main.ts
TS-HELLO
point.ts
main.ts
main.ts
class Point {
constructor(private x?: number
}
draw() {
console.log('X: ' + this.x
}
}
let point = new Point(1, 2);
point.draw();


TS main.ts
TS point.ts
1
2 class Point {
3 constructor(private x?: number, private y?: number) {
4 }
5
6 draw() {
7 console.log('X: ' + this.x + ', Y: ' + this.y);
8 }
9 }
10


TS main.ts
TS point.ts
1
2 export class Point {
3 constructor(private x?: number, private y?: number) {
4 }
5
6 draw() {
7 console.log('X: ' + this.x + ', Y: ' + this.y);
8 }
9 }
10


TS main.ts
TS point.ts
[ts] Cannot find name 'Point'.
any
1 2 3 4
let point = new Point(1, 2);
point.draw();


TS main.ts
TS point.ts
1 2 3 4 5
import { Point } from './point';
let point = new Point(1, 2);
point.draw();



Types

Classes

Simple variables

Objects