```
addPointsToScore(player); // error TS2346
// Supplied parameters do not match any signature of call target.
```

To show that a parameter is optional in a function (or a property in an interface), you can add `?` after the parameter. Here, the `points` parameter could be optional:

```
function addPointsToScore(player: HasScore, points?: number): void {
  points = points || 0;
  player.score += points;
}
```

## 5.6. Functions as property

You may also be interested in describing a parameter that must have a specific function instead of a property:

```
function startRunning(pony) {
  pony.run(10);
}
```

The interface definition will be:

```
interface CanRun {
  run(meters: number): void;
}
    function startRunning(pony: CanRun): void {
      pony.run(10);
    }

    const ponyOne = {
      run: meters => logger.log(`pony runs ${meters}m`)
    };
    startRunning(ponyOne);
```

## 5.7. Classes

A class can implement an interface. For us, the `Pony` class should be able to run, so we can write:

```
class Pony implements CanRun {
  run(meters) {
    logger.log(`pony runs ${meters}m`);
  }
}
```