Chapter 3 – Complex Types

1. Introduction
   1. TypeScript makes it very easy to keep track of element types in arrays
   2. Manual type-checking is needlessly difficult and adds complications

|  |
| --- |
| let customersArray = ['Custy Stomer', 'C. Oostomar', 'C.U.S. Tomer', 3432434, 'Custo Mer', 'Custopher Ustomer', 3432435, 'Kasti Yastimeur'];  //Write Your Code here:  function checkCustomersArray() {  for (el of customersArray) {  if (typeof el != 'string') {  console.log(`Type error: ${el} should be a string!`);  }  }  }  function stringPush(val) {  if (typeof val != 'string') {  return;  }  customersArray.push(val);  } |

1. Array Type Annotations
   1. Type annotation for array types is fairly straightforward: [] after the element type
   2. Throw errors when elements of the wrong type are added

|  |
| --- |
| let names: string[] = ['Danny', 'Samantha'];  let names: string[] = ['Damien'];  names.push(666) // Type Error! |

* 1. Alternate - use the Array<T> syntax, where T stands for the type.

|  |
| --- |
| let names: Array<string> = ['Danny', 'Samantha']; |

Exercise

|  |
| --- |
| // Arrays:  let bestNumbers: number[] = [7,77,4];  let bestLunches: string[] = ['chicken soup', 'non-chicken soup'];  let bestBreakfasts: string[] = ['fasting', 'oatmeal', 'tamago kake gohan', 'any kind of soup'];  let bestBooleans: boolean[] = [true, false]; |