Chapter 5 – Type Narrowing

1. Introduction
   1. Type narrowing is when TypeScript can infer more specific types based on the variable’s surrounding code.

|  |
| --- |
| function formatDate(date: string | number) {  // date can be a number or string here    if (typeof date === 'string') {  // date must be a string here  }  } |

1. Type Guards
   1. One way that TypeScript can narrow a type is with a conditional statement that checks if a variable is a specific type.
   2. TypeScript can recognize typeof type guards that check for these specific values: 'string', 'number', 'boolean', and 'symbol'.

Exercise

|  |
| --- |
| function formatStatistic(stat: string | number) {  if (typeof stat == 'number') {  return stat.toFixed(2);  } else if (typeof stat == 'string') {  return stat.toUpperCase();  }  }  console.log(formatStatistic('Win'));  console.log(formatStatistic(0.364)); |

1. Using in with Type Guards
   1. sometimes we want to see if a specific method exists on a type instead of a type like 'string'
   2. [The in operator](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/in) checks if a property exists on an object itself or anywhere within its prototype chain. Take a look at this example:

Example

|  |
| --- |
| type Tennis = {  serve: () => void;  }    type Soccer = {  kick: () => void;  }    function play(sport: Tennis | Soccer) {  if ('serve' in sport) {  return sport.serve();  }    if ('kick' in sport) {  return sport.kick();  }  } |

Exercise

|  |
| --- |
| type Cat = {  name: string;  run: () => string;  }  type Fish = {  name: string;  swim: () => string;  }  const siameseCat = {  name: 'Proxie',  run: () => 'pitter pat'  }  const bettaFish = {  name: 'Neptune',  swim: () => 'bubble blub'  }  function move(pet: Cat | Fish) {  if ('run' in pet) {  return pet.run();  }  if ('swim' in pet) {  return pet.swim();  }  }  console.log(move(siameseCat)) |

1. Narrowing with else