# Exercise Hints

Here are some hints that hopefully help you solve this exercise. If not - no problem, we'll solve it together over the next lectures! :-)

- The **project structure** of this section has **one important difference** compared to the structure from the previous section: Images (and statically served assets in general) are now stored in the `public/` folder - NOT in a nested `assets/` folder!
- To reference images stored in the `public/` folder you would use a path like this: `<img src="some-image.png">` - i.e., the `public` folder name is NOT part of that path (it's NOT `<img src="public/some-image.png">`)
- The `investment-results.ts` file contains code that you can use to calculate annual investment results - you might need to move that code into a different place though (e.g., into some component or service method or anything like that) => Use it as a starting point & help but don't be afraid to change it!
- To output the calculated investment results, a `<table>` will be rendered (in my solution, over the next lectures) => If you're not feeling confident with using the HTML `<table>` element, you can instead also render regular `<div>`s (or any other element) and set up basic styling on your own
- If you do go for a `<table>`, keep in mind that you'll likely need to work with `<thead>` & `<tbody>` (as well as `<tr>` etc.) to achieve the desired look
- When using a `<table>` also make sure to NOT split the table elements into multiple components => this would break styling since extra DOM element (=> your component elements) would be introduced into the markup
- You might need to define some TypeScript types - e.g., for describing the shapes of objects or arrays

    - Object types are defined like this: `type MyObj = { someProp1: string; someProp2: number};`
    - Array types are defined like this: `string[]`, `number[]`, `MyObj[]` etc.
    - To define properties or parameters as optional, you can add a `?` after the name: e.g., `someProp?: string;`
    - To allow for multiple types in a single value you can use Union Types: `myVal: number | string`
- To format the currency values appropriately, you could consider using Angular's built-in CurrencyPipe