

U64MK7215 : I want to replicate ajax code in elm

U3SJEDR96 : Or even `Expect.equal (Macro { name = "foo", args = [] })`

U3SJEDR96 : <@U64MK7215> - Right. So that example I posted makes HTTP requests, receives JSON, and uses that JSON to drive the application state.

U64MK7215 : \$(document).ready(function () {     hidebody();

```
$.ajax({
  type: 'POST',
  url: "/load",
  contentType: false,
  data: null,
  processData: false,
  success: function (data) {
    $("#loader").hide();
    showbody();
  },
  error: function (data) {
    console.log(data);
  }
});
```

U4872964V : <@U3SJEDR96> yes, for the special case of checking full equality, that works too :slightly\_smiling\_face:

U3SJEDR96 : Exactly. Just make sure we cover all bases and don't miss the "obvious" :wink:

U3LUC6SNS : <@U4872964V>, <@U3SJEDR96>, thankou!!! I have my first test working, so should be able to do what I need.

U29JSAR9S : I'm trying to make the following function point free and can't figure out why I'm getting an error when I do so:``

whoAteThePies : List Int -&gt; List Int -&gt; ( Int, Int )

whoAteThePies a b =

```
List.map2 (,) a b
  |&gt; List.indexedMap flipFlopSort
  |&gt; List.map (mapTuple evenToZero)
  |&gt; List.foldr sumTuples ( 0, 0 )
```

...

Anyone have any ideas?

U4872964V : well, you should look at what the error says :slightly\_smiling\_face:

U4872964V : hard to say without knowing what those other functions are

U29JSAR9S : ``whoAteThePies : List Int -&gt; List Int -&gt; ( Int, Int )

whoAteThePies =

```
List.map2 (,)
  |&gt; List.indexedMap flipFlopSort
  |&gt; List.map (mapTuple evenToZero)
  |&gt; List.foldr sumTuples ( 0, 0 )
```

...

gives me

...

The right side of (|&gt;) is causing a type mismatch.

```
8| List.map2 (,)
```

```
9|&gt;      |&gt; List.indexedMap flipFlopSort
```

(|&gt;) is expecting the right side to be a:

```
(List Int -&gt; List Int -&gt; List ( Int, Int )) -&gt; a
```

But the right side is:

```
(List ( Int, Int )) -&gt; List ( Int, Int )
```

...

and

...

```
whoAteThePies : List Int -> List Int -> ( Int, Int )
```

```
whoAteThePies =
```

```
  List.map2 (,)
    >>> List.indexedMap flipFlopSort
```

```
    >>> List.map (mapTuple evenToZero)
```

```
    >>> List.foldr sumTuples ( 0, 0 )
```

...

gives me

...

The right side of (>>>) is causing a type mismatch.

```
8|   List.map2 (,)
```

```
9|>>>   >>> List.indexedMap flipFlopSort
```

(>>>) is expecting the right side to be a:

```
(List Int -> List ( Int, Int )) -> c
```

But the right side is:

```
(List ( Int, Int )) -> List ( Int, Int )
```

...

U29JSAR9S : mainly I think I'm misunderstanding something because I thought

...

```
whoAteThePies a b =
```

```
  List.map2 (,) a b
```

...

was essentially the same as

...

```
whoAteThePies =
```

```
  List.map2 (,)
```

...

U4872964V : it is, but not if you put stuff after it

U4872964V : you can't do "point free style" with two arguments though

U29JSAR9S : if I'm piping things onwards you mean?

U4872964V : yes

U29JSAR9S : ah, ok

U4872964V : you can only `|>` and `>>>` one value

U29JSAR9S : right, ok - that makes sense

U3SJEDR96 : you could make a `>>>>` but that seems symbol-heavy and won't get the special treatment

`>>>>` and `|>` do :slightly\_smiling\_face:

U4872964V : it would also induce headache