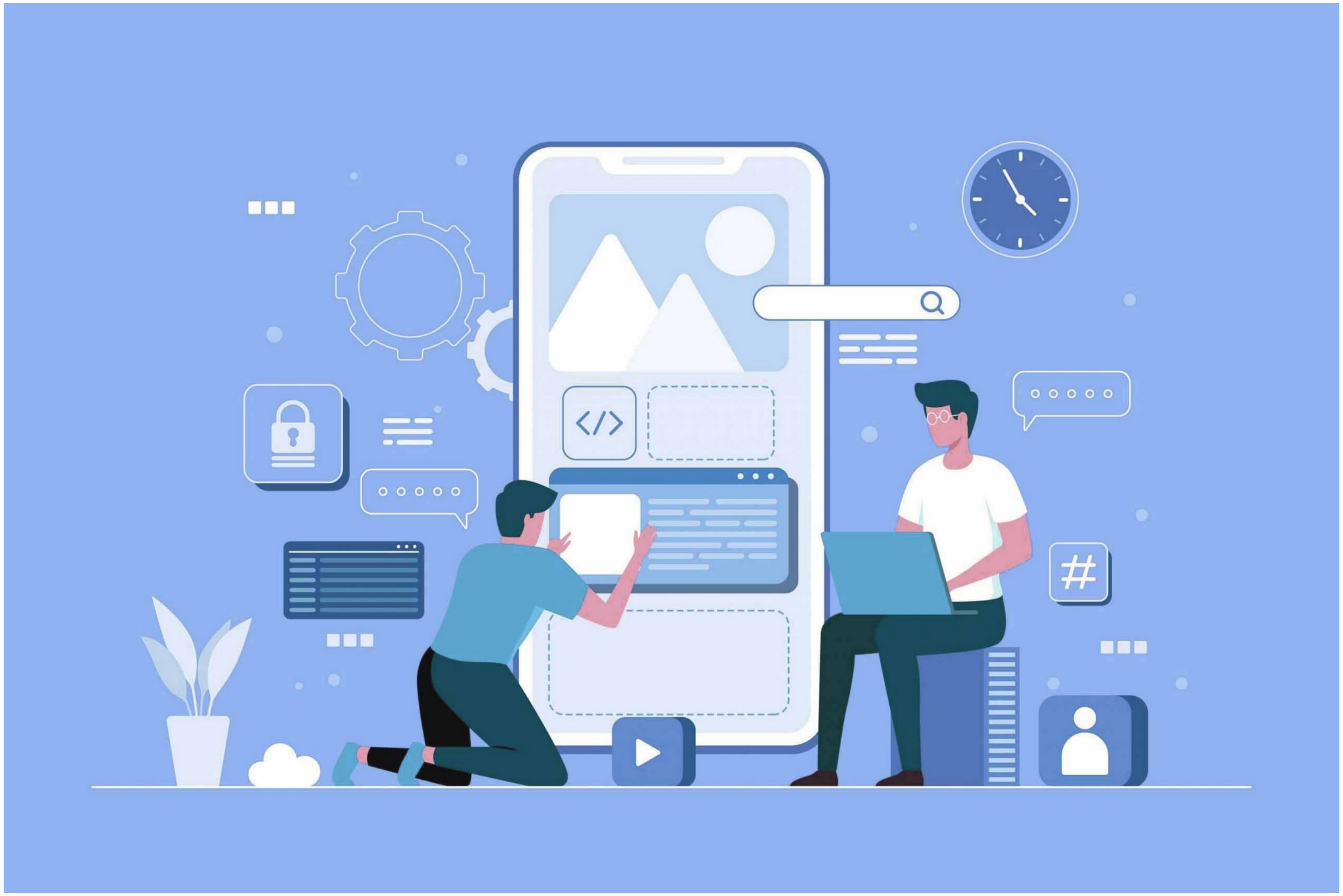
CS385 Mobile Application Development (Lecture 4)

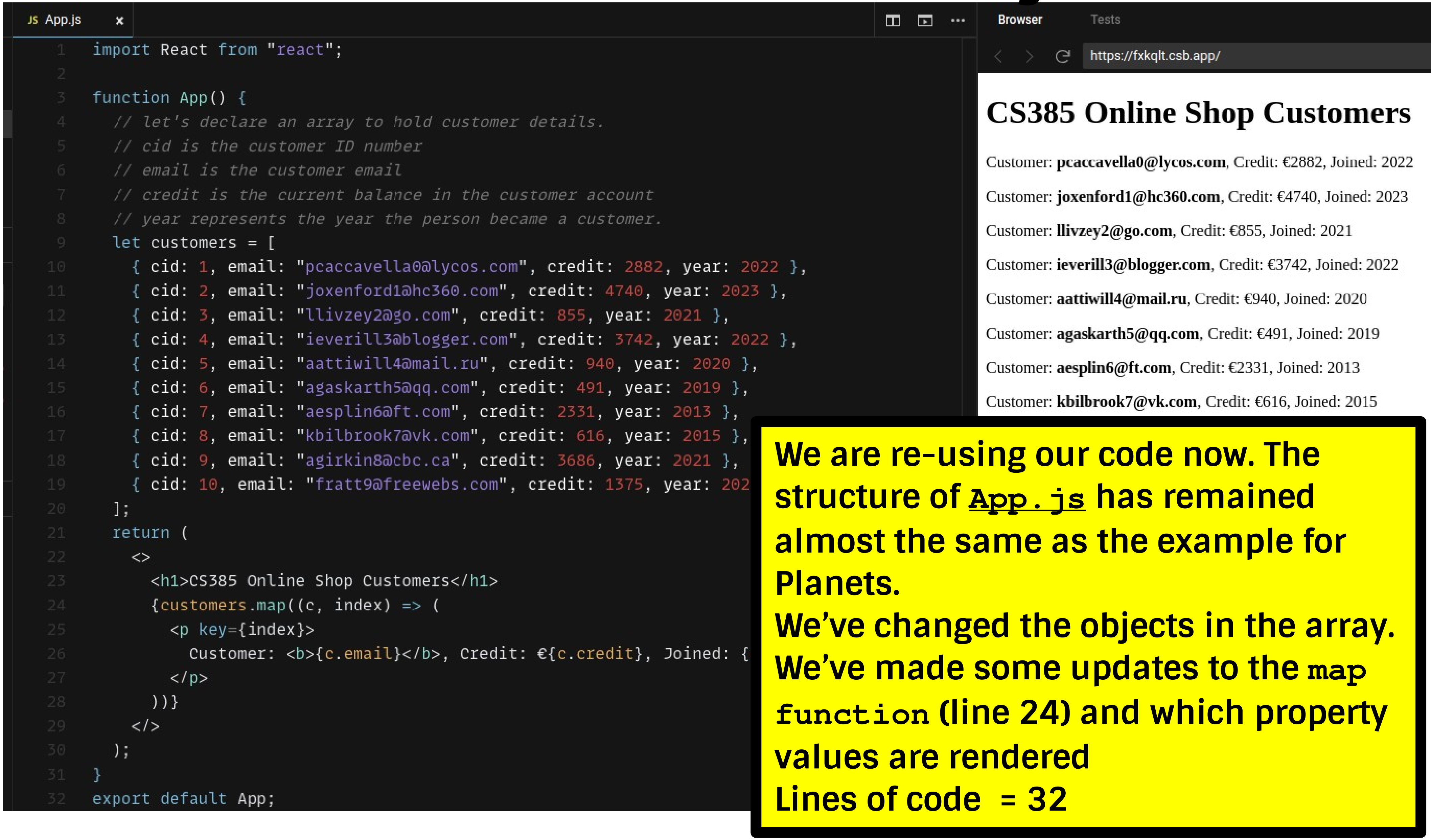


Peter Mooney

Suppose we have an online shop

- We want to store customer information in a very basic way (#2 on previous slide)
- What PROPERTIES should each customer object have?
- We shall use:
  - Customer ID (**`cid`**)
  - Customer Email (**`email`**)
  - Customer Credit (**`credit`**)
  - Customer Year joined (**`year`**)

Our online shop – 10 customers (with 10 customer objects)



```
1   import React from "react";
2
3   function App() {
4     // let's declare an array to hold customer details.
5     // cid is the customer ID number
6     // email is the customer email
7     // credit is the current balance in the customer account
8     // year represents the year the person became a customer.
9     let customers = [
10      { cid: 1, email: "pcaccavella0@lycos.com", credit: 2882, year: 2022 },
11      { cid: 2, email: "joxenford1@hc360.com", credit: 4740, year: 2023 },
12      { cid: 3, email: "llivzey2@go.com", credit: 855, year: 2021 },
13      { cid: 4, email: "ieverill3@blogger.com", credit: 3742, year: 2022 },
14      { cid: 5, email: "aattiwill4@mail.ru", credit: 940, year: 2020 },
15      { cid: 6, email: "agaskarth5@qq.com", credit: 491, year: 2019 },
16      { cid: 7, email: "aesplin6@ft.com", credit: 2331, year: 2013 },
17      { cid: 8, email: "kbilbrook7@vk.com", credit: 616, year: 2015 },
18      { cid: 9, email: "agirkin8@cbc.ca", credit: 3686, year: 2021 },
19      { cid: 10, email: "fratt9@freewebs.com", credit: 1375, year: 202
20    ];
21    return (
22      <>
23        <h1>CS385 Online Shop Customers</h1>
24        {customers.map((c, index) => (
25          <p key={index}>
26            Customer: <b>{c.email}</b>, Credit: €{c.credit}, Joined: {
27          </p>
28        ))}
29      </>
30    );
31  }
32  export default App;
```

CS385 Online Shop Customers

Customer: **pcaccavella0@lycos.com**, Credit: €2882, Joined: 2022

Customer: **joxenford1@hc360.com**, Credit: €4740, Joined: 2023

Customer: **llivzey2@go.com**, Credit: €855, Joined: 2021

Customer: **ieverill3@blogger.com**, Credit: €3742, Joined: 2022

Customer: **aattiwill4@mail.ru**, Credit: €940, Joined: 2020

Customer: **agaskarth5@qq.com**, Credit: €491, Joined: 2019

Customer: **aesplin6@ft.com**, Credit: €2331, Joined: 2013

Customer: **kbilbrook7@vk.com**, Credit: €616, Joined: 2015

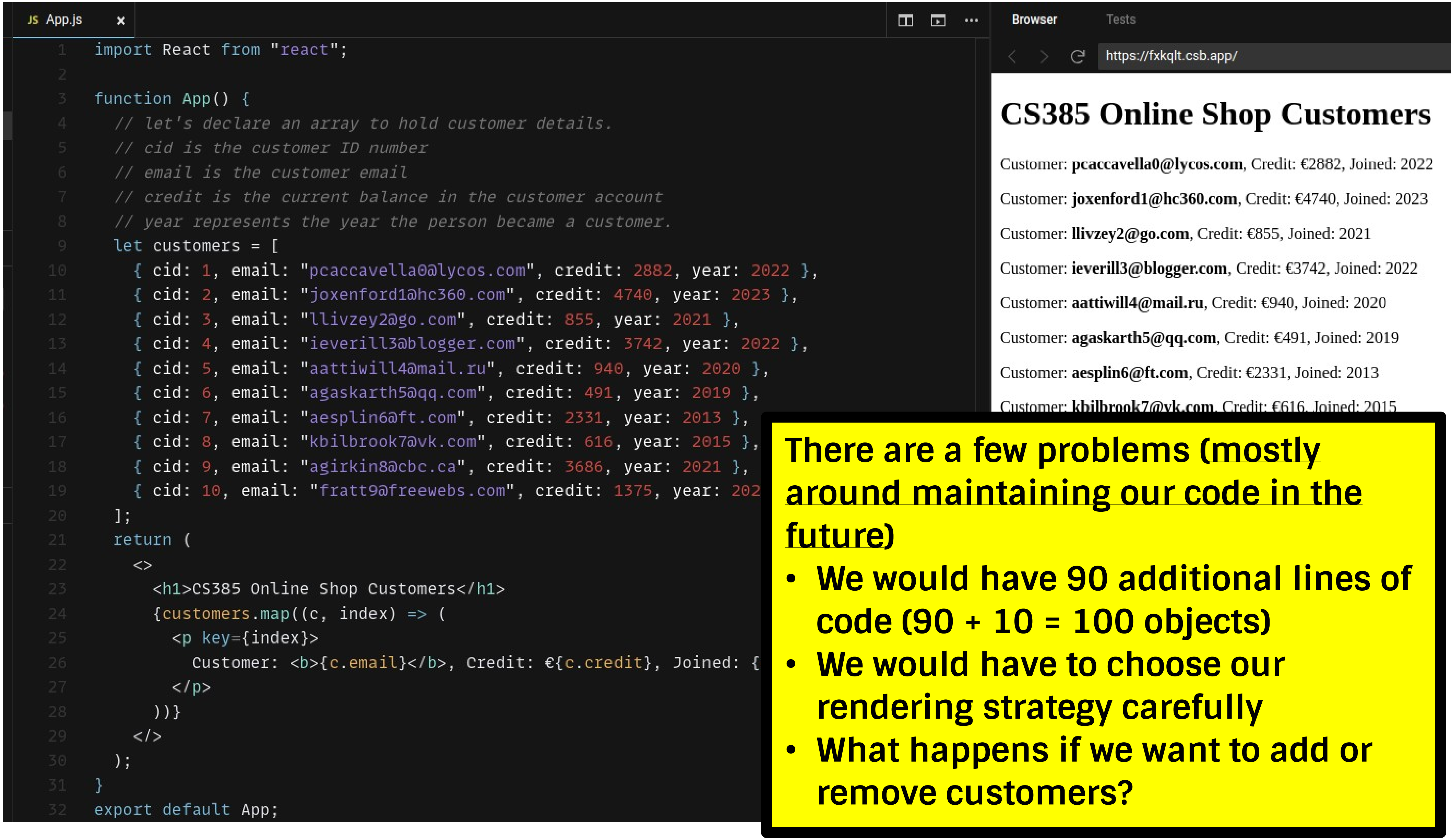
We are re-using our code now. The structure of `App.js` has remained almost the same as the example for Planets.

We’ve changed the objects in the array.

We’ve made some updates to the `map function` (line 24) and which property values are rendered

Lines of code = 32

Question for you – can you name any problems if we had 100 customers?



```
import React from "react";

function App() {
  // let's declare an array to hold customer details.
  // cid is the customer ID number
  // email is the customer email
  // credit is the current balance in the customer account
  // year represents the year the person became a customer.
  let customers = [
    { cid: 1, email: "pcaccavella0@lycos.com", credit: 2882, year: 2022 },
    { cid: 2, email: "joxenford1@hc360.com", credit: 4740, year: 2023 },
    { cid: 3, email: "llivzey2@go.com", credit: 855, year: 2021 },
    { cid: 4, email: "ieverill3@blogger.com", credit: 3742, year: 2022 },
    { cid: 5, email: "aattiwill4@mail.ru", credit: 940, year: 2020 },
    { cid: 6, email: "agaskarth5@qq.com", credit: 491, year: 2019 },
    { cid: 7, email: "aesplin6@ft.com", credit: 2331, year: 2013 },
    { cid: 8, email: "kbilbrook7@vk.com", credit: 616, year: 2015 },
    { cid: 9, email: "agirkin8@cbc.ca", credit: 3686, year: 2021 },
    { cid: 10, email: "fratt9@freewebs.com", credit: 1375, year: 202
  ];
  return (
    <>
      <h1>CS385 Online Shop Customers</h1>
      {customers.map((c, index) => (
        <p key={index}>
          Customer: <b>{c.email}</b>, Credit: €{c.credit}, Joined: {
        </p>
      ))}
    </>
  );
}
export default App;
```

CS385 Online Shop Customers

Customer: **pcaccavella0@lycos.com**, Credit: €2882, Joined: 2022

Customer: **joxenford1@hc360.com**, Credit: €4740, Joined: 2023

Customer: **llivzey2@go.com**, Credit: €855, Joined: 2021

Customer: **ieverill3@blogger.com**, Credit: €3742, Joined: 2022

Customer: **aattiwill4@mail.ru**, Credit: €940, Joined: 2020

Customer: **agaskarth5@qq.com**, Credit: €491, Joined: 2019

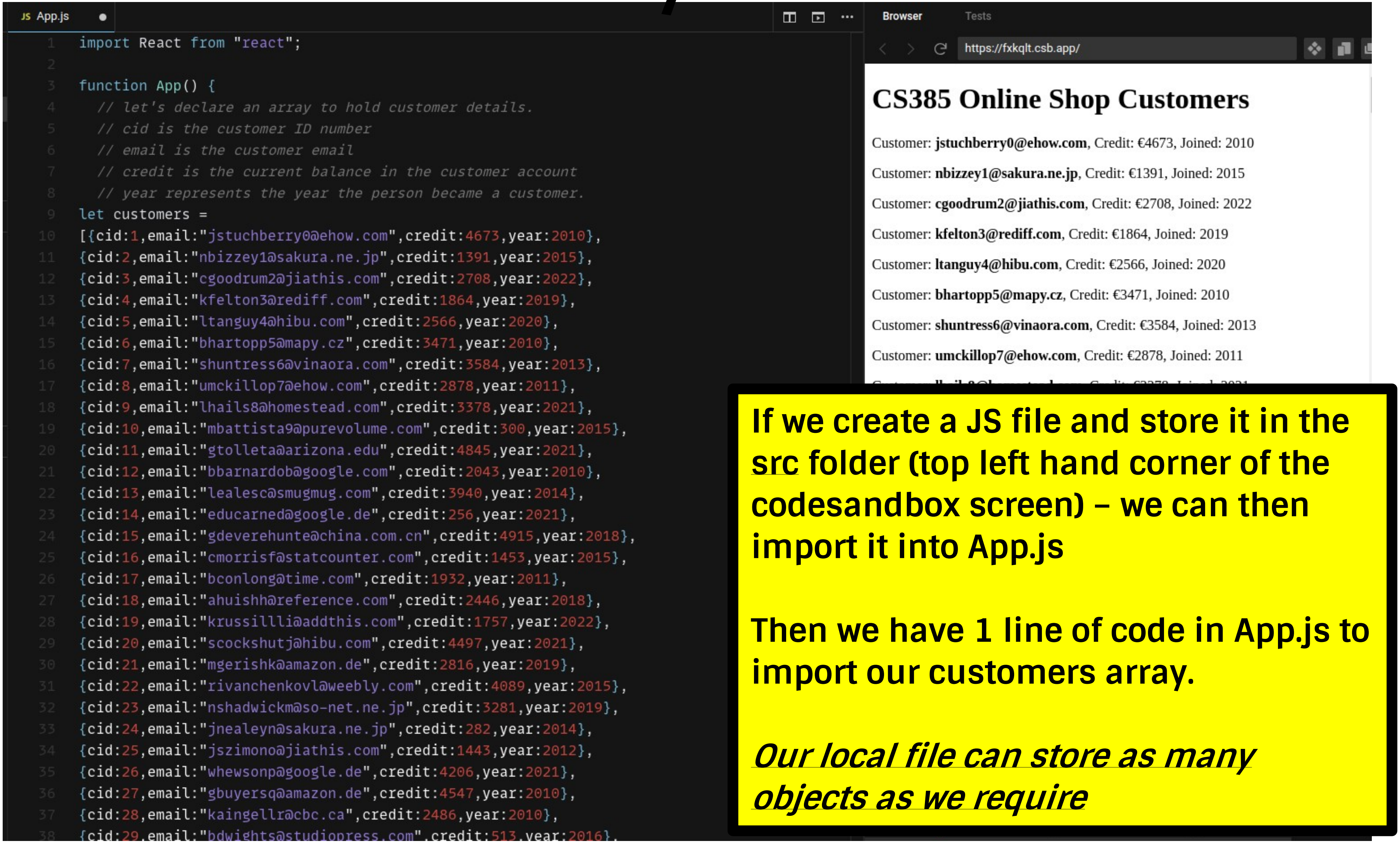
Customer: **aesplin6@ft.com**, Credit: €2331, Joined: 2013

Customer: **kbilbrook7@vk.com**, Credit: €616, Joined: 2015

There are a few problems (mostly around maintaining our code in the future)

- We would have 90 additional lines of code (90 + 10 = 100 objects)
- We would have to choose our rendering strategy carefully
- What happens if we want to add or remove customers?

Solution: Let’s store our customers array in a local JS file



```
import React from "react";

function App() {
  // let's declare an array to hold customer details.
  // cid is the customer ID number
  // email is the customer email
  // credit is the current balance in the customer account
  // year represents the year the person became a customer.
let customers =
[{cid:1,email:"jstuchberry0@ehow.com",credit:4673,year:2010},
{cid:2,email:"nbizzey1@sakura.ne.jp",credit:1391,year:2015},
{cid:3,email:"cgoodrum2@jiathis.com",credit:2708,year:2022},
{cid:4,email:"kfelton3@rediff.com",credit:1864,year:2019},
{cid:5,email:"ltanguy4@hibu.com",credit:2566,year:2020},
{cid:6,email:"bhartopp5@mapy.cz",credit:3471,year:2010},
{cid:7,email:"shuntress6@vinaora.com",credit:3584,year:2013},
{cid:8,email:"umckillop7@ehow.com",credit:2878,year:2011},
{cid:9,email:"lhails8@homestead.com",credit:3378,year:2021},
{cid:10,email:"mbattista9@purevolume.com",credit:300,year:2015},
{cid:11,email:"gtolleta@arizona.edu",credit:4845,year:2021},
{cid:12,email:"bbarnardob@google.com",credit:2043,year:2010},
{cid:13,email:"lealesc@smugmug.com",credit:3940,year:2014},
{cid:14,email:"educarned@google.de",credit:256,year:2021},
{cid:15,email:"gdeverehunte@china.com.cn",credit:4915,year:2018},
{cid:16,email:"cmorrisf@statcounter.com",credit:1453,year:2015},
{cid:17,email:"bconlong@time.com",credit:1932,year:2011},
{cid:18,email:"ahuishh@reference.com",credit:2446,year:2018},
{cid:19,email:"krussillli@addthis.com",credit:1757,year:2022},
{cid:20,email:"scockshutj@hibu.com",credit:4497,year:2021},
{cid:21,email:"mgerishk@amazon.de",credit:2816,year:2019},
{cid:22,email:"rivanchenkovl@weebly.com",credit:4089,year:2015},
{cid:23,email:"nshadwickm@so-net.ne.jp",credit:3281,year:2019},
{cid:24,email:"jnealeyn@sakura.ne.jp",credit:282,year:2014},
{cid:25,email:"jszimono@jiathis.com",credit:1443,year:2012},
{cid:26,email:"whewsonp@google.de",credit:4206,year:2021},
{cid:27,email:"gbuyersq@amazon.de",credit:4547,year:2010},
{cid:28,email:"kaingellr@cbc.ca",credit:2486,year:2010},
{cid:29,email:"bdwights@studiopress.com",credit:513,year:2016},
```

CS385 Online Shop Customers

Customer: **jstuchberry0@ehow.com**, Credit: €4673, Joined: 2010

Customer: **nbizzey1@sakura.ne.jp**, Credit: €1391, Joined: 2015

Customer: **cgoodrum2@jiathis.com**, Credit: €2708, Joined: 2022

Customer: **kfelton3@rediff.com**, Credit: €1864, Joined: 2019

Customer: **ltanguy4@hibu.com**, Credit: €2566, Joined: 2020

Customer: **bhartopp5@mapy.cz**, Credit: €3471, Joined: 2010

Customer: **shuntress6@vinaora.com**, Credit: €3584, Joined: 2013

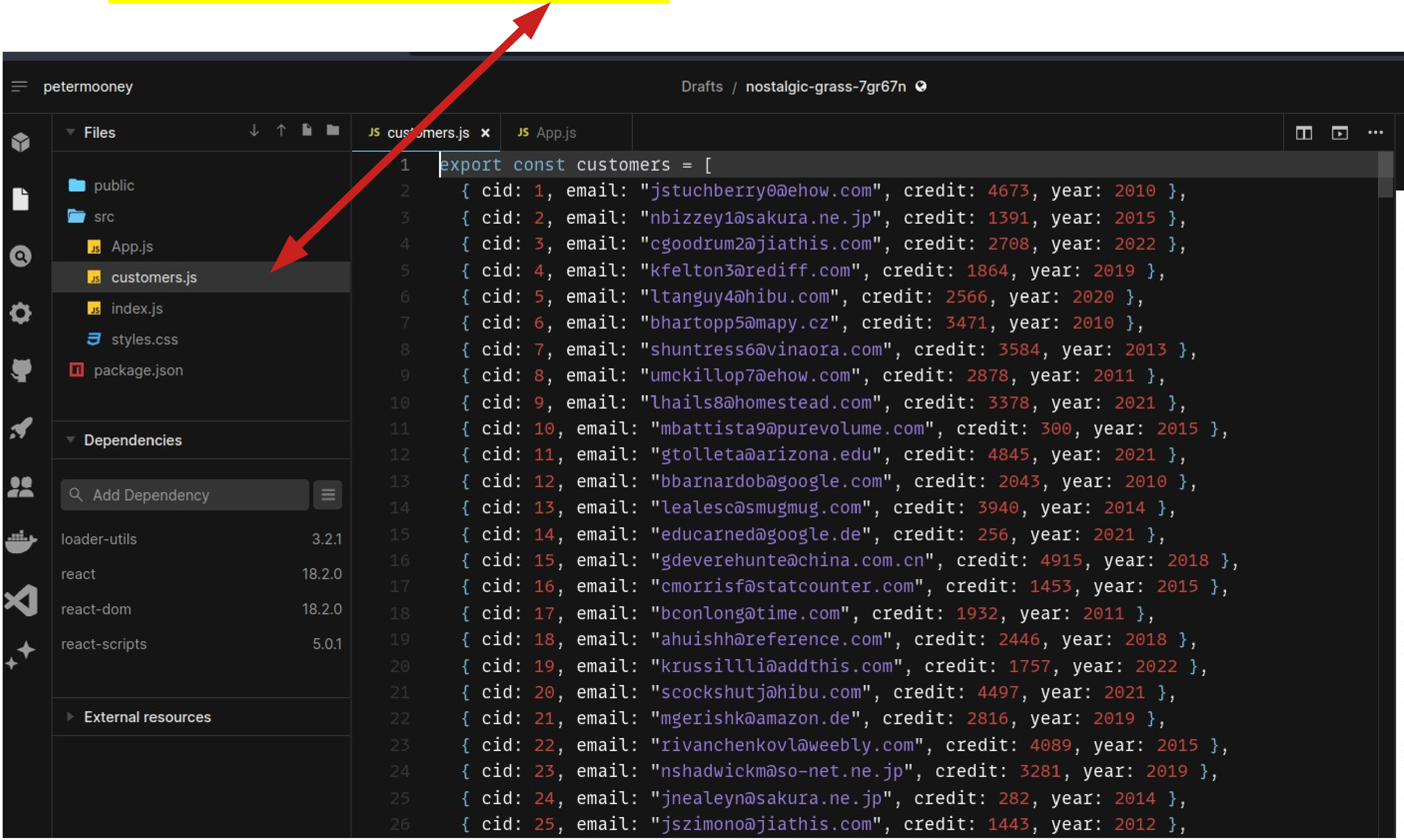
Customer: **umckillop7@ehow.com**, Credit: €2878, Joined: 2011

If we create a JS file and store it in the src folder (top left hand corner of the codesandbox screen) – we can then import it into App.js

Then we have 1 line of code in App.js to import our customers array.

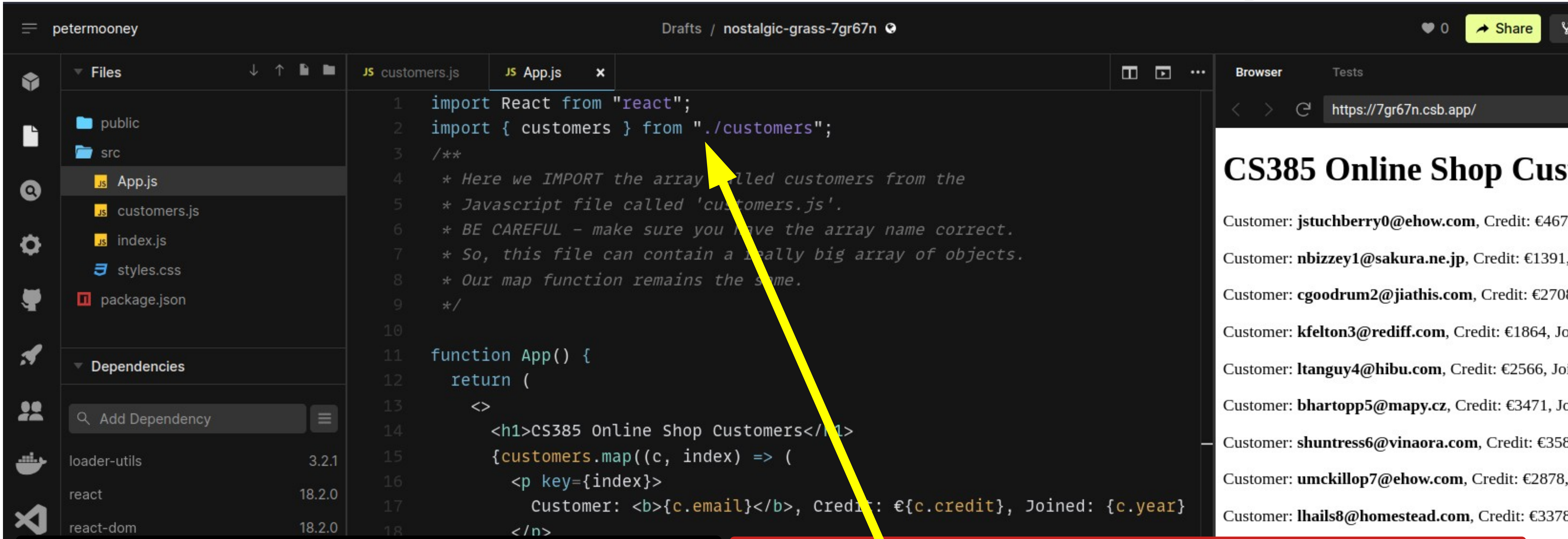
Our local file can store as many objects as we require

We create a new file called “customers.js” in the src folder



```
export const customers = [
  { cid: 1, email: "jstuchberry0@ehow.com", credit: 4673, year: 2010 },
  { cid: 2, email: "nbizzey1@sakura.ne.jp", credit: 1391, year: 2015 },
  { cid: 3, email: "cgoodrum2@jiathis.com", credit: 2708, year: 2022 },
  { cid: 4, email: "kfelton3@rediff.com", credit: 1864, year: 2019 },
  { cid: 5, email: "ltanguy4@hibu.com", credit: 2566, year: 2020 },
  { cid: 6, email: "bhartopp5@mapy.cz", credit: 3471, year: 2010 },
  { cid: 7, email: "shuntress6@vinaora.com", credit: 3584, year: 2013 },
  { cid: 8, email: "umckillop7@ehow.com", credit: 2878, year: 2011 },
  { cid: 9, email: "lhails8@homestead.com", credit: 3378, year: 2021 },
  { cid: 10, email: "mbattista9@purevolume.com", credit: 300, year: 2015 },
  { cid: 11, email: "gtolleta@arizona.edu", credit: 4845, year: 2021 },
  { cid: 12, email: "bbarnardob@google.com", credit: 2043, year: 2010 },
  { cid: 13, email: "lealesc@smugmug.com", credit: 3940, year: 2014 },
  { cid: 14, email: "educarned@google.de", credit: 256, year: 2021 },
  { cid: 15, email: "gdeverehunte@china.com.cn", credit: 4915, year: 2018 },
  { cid: 16, email: "cmorrisf@statcounter.com", credit: 1453, year: 2015 },
  { cid: 17, email: "bconlong@time.com", credit: 1932, year: 2011 },
  { cid: 18, email: "ahuishh@reference.com", credit: 2446, year: 2018 },
  { cid: 19, email: "krussillli@addthis.com", credit: 1757, year: 2022 },
  { cid: 20, email: "scockshutj@hibu.com", credit: 4497, year: 2021 },
  { cid: 21, email: "mgerishk@amazon.de", credit: 2816, year: 2019 },
  { cid: 22, email: "rivanchenkovl@weebly.com", credit: 4089, year: 2015 },
  { cid: 23, email: "nshadwickm@so-net.ne.jp", credit: 3281, year: 2019 },
  { cid: 24, email: "jnealeyn@sakura.ne.jp", credit: 282, year: 2014 },
  { cid: 25, email: "jszimono@jiathis.com", credit: 1443, year: 2012 },
```

Finished: By using a special ‘`import`’ statement (line 7) we render `customers.js`



loader-utils	3.2.1
react	18.2.0
react-dom	18.2.0

```
1   import React from "react";
2   import { customers } from "./customers";
3   /**
4    * Here we IMPORT the array called customers from the
5    * Javascript file called 'customers.js'.
6    * BE CAREFUL - make sure you have the array name correct.
7    * So, this file can contain a really big array of objects.
8    * Our map function remains the same.
9    */
10
11  function App() {
12    return (
13      <>
14        <h1>CS385 Online Shop Customers</h1>
15        {customers.map((c, index) => (
16          <p key={index}>
17            Customer: <b>{c.email}</b>, Credit: €{c.credit}, Joined: {c.year}
18          </p>
```

CS385 Online Shop Cus

Customer: **jstuchberry0@ehow.com**, Credit: €467

Customer: **nbizzey1@sakura.ne.jp**, Credit: €1391,

Customer: **cgoodrum2@jiathis.com**, Credit: €270

Customer: **kfelton3@rediff.com**, Credit: €1864, Jo

Customer: **ltanguy4@hibu.com**, Credit: €2566, Jo

Customer: **bhartopp5@mapy.cz**, Credit: €3471, Jo

Customer: **shuntress6@vinaora.com**, Credit: €358

Customer: **umckillop7@ehow.com**, Credit: €2878,

Customer: **lhails8@homestead.com**, Credit: €3378

redit:

845, J

t: €20

€3940,

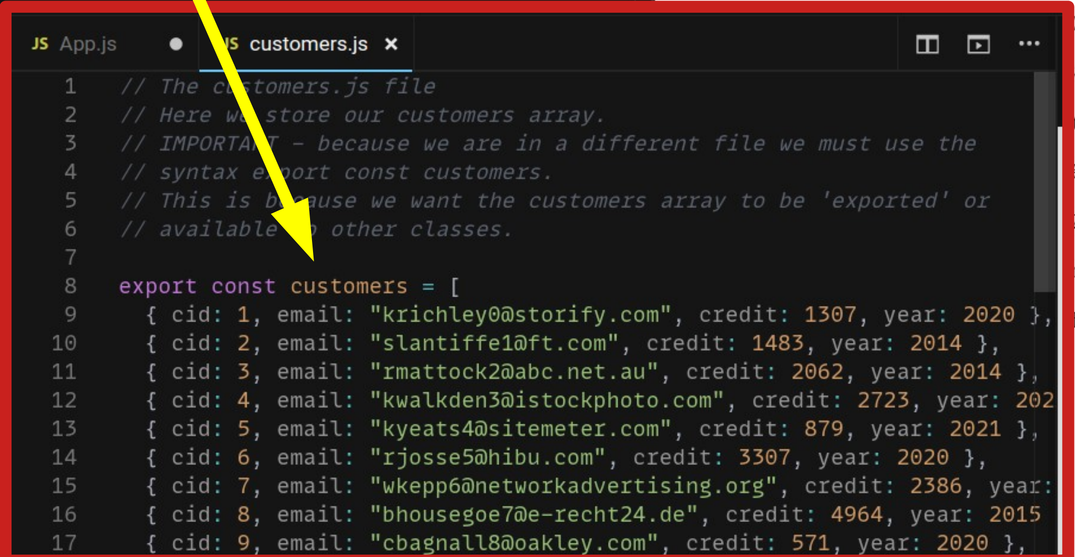
256, Jo

redit: €

lit: €1

WARNING – watch for errors with the filename AND the name of the array in the export const statement (line 8) of the local JS file.

These are one of the most common errors people make when creating these examples.



```
1   // The customers.js file
2   // Here we store our customers array.
3   // IMPORTANT - because we are in a different file we must use the
4   // syntax export const customers.
5   // This is because we want the customers array to be 'exported' or
6   // available to other classes.
7
8   export const customers = [
9     { cid: 1, email: "krichley0@storify.com", credit: 1307, year: 2020 },
10    { cid: 2, email: "slantiffe1@ft.com", credit: 1483, year: 2014 },
11    { cid: 3, email: "rmattock2@abc.net.au", credit: 2062, year: 2014 },
12    { cid: 4, email: "kwalkden3@istockphoto.com", credit: 2723, year: 202
13    { cid: 5, email: "kyeats4@sitemeter.com", credit: 879, year: 2021 },
14    { cid: 6, email: "rjosse5@hibu.com", credit: 3307, year: 2020 },
15    { cid: 7, email: "wkepp6@networkadvertising.org", credit: 2386, year:
16    { cid: 8, email: "bhousegoe7@e-recht24.de", credit: 4964, year: 2015
17    { cid: 9, email: "cbagnall8@oakley.com", credit: 571, year: 2020 },
```

In-lecture DEMO

or

Try it out yourself on Codesandbox.io (code is available on Moodle)

Using the local JS file has many advantages

- **Data manipulation** – we can easily add, remove or edit customers in our object array without changing code in `App.js`

- **Data scale** – as the JS file is local it can scale to many hundreds or thousands of customers

- **Reproducibility** – we can reuse our code easily

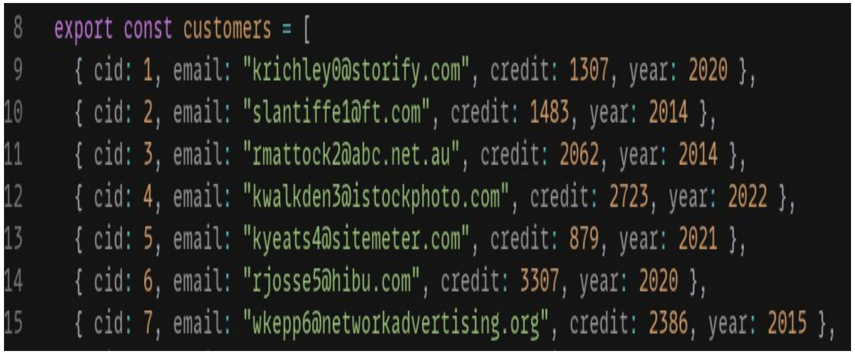
- **Decoupling of data and code** – we can change our rendering code (App.js) independently of our array of objects

But in reality – we rarely want to display the entire contents of an array or collection of objects

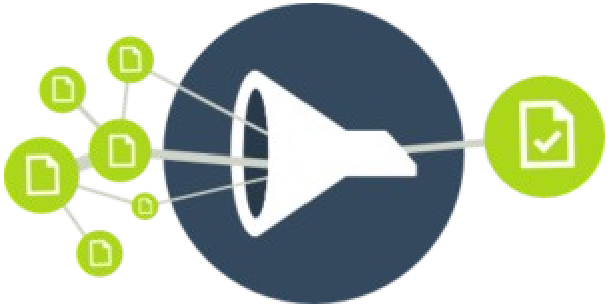
Filtering using the map function

- In the next example we will learn out to write our own functions in Javascript which can be used by the map function to control which objects are rendered

A filter function



```
export const customers = [
  { cid: 1, email: "krichley0@storify.com", credit: 1307, year: 2020 },
  { cid: 2, email: "slantiffe1@ft.com", credit: 1483, year: 2014 },
  { cid: 3, email: "rmattock2@abc.net.au", credit: 2062, year: 2014 },
  { cid: 4, email: "kwalkden3@istockphoto.com", credit: 2723, year: 2022 },
  { cid: 5, email: "kyeats4@sitemeter.com", credit: 879, year: 2021 },
  { cid: 6, email: "rjosse5@hibu.com", credit: 3307, year: 2020 },
  { cid: 7, email: "wkepp6@networkadvertising.org", credit: 2386, year: 2015 },
```



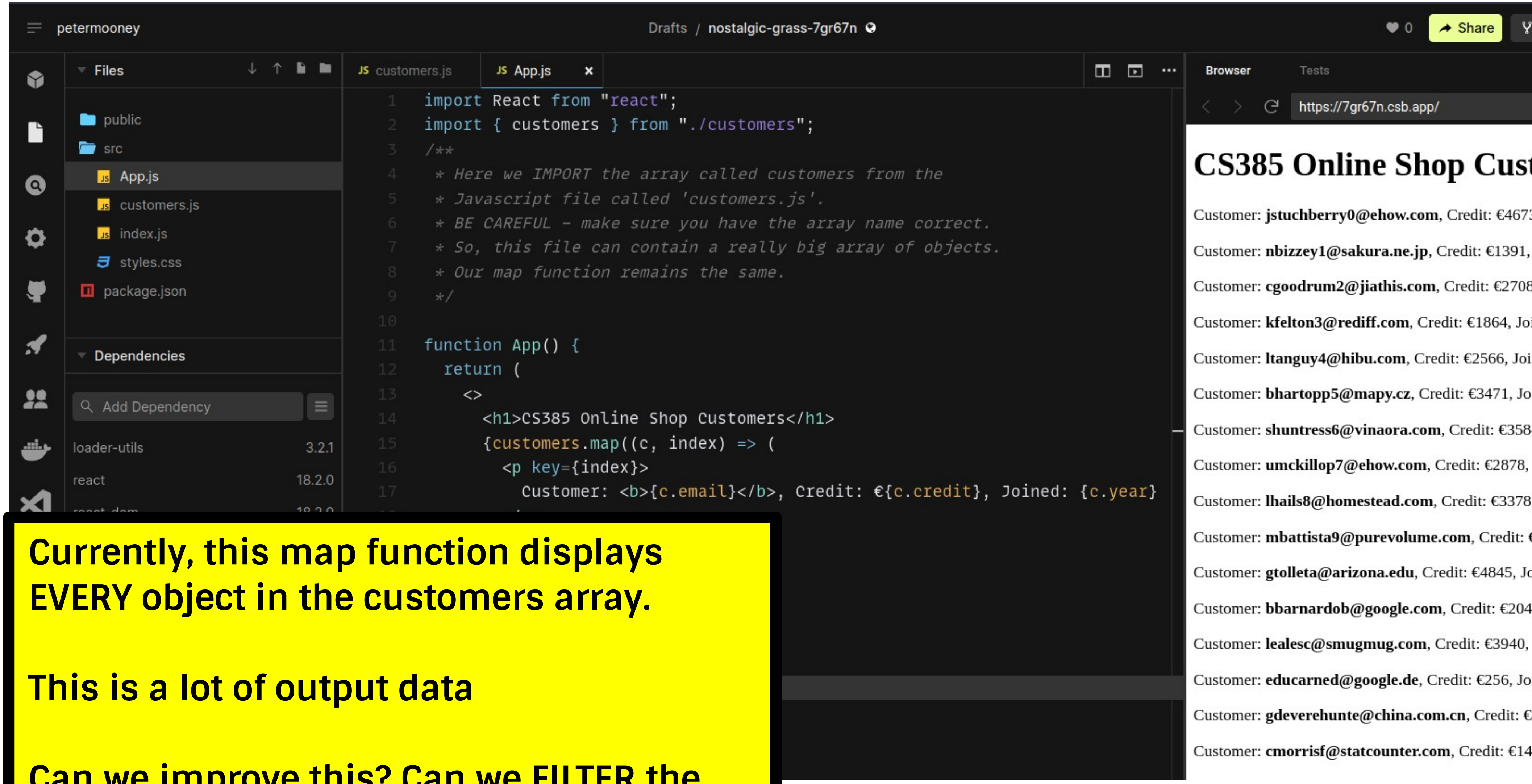
The map function

RENDER OUTPUT

Filtering with map functions

- So what will we need?
- We will try to write a function which can be as flexible as possible.
- We will use a very specific feature of Javascript – that is we will write a function which returns a function.
- This is a new concept and one you will probably not have seen before.
- **Let’s write a flexible filter function allowing us to restrict the objects rendered by the map function based on one of the properties of the objects**

Filtering with a map function – currently we display ALL objects in the array



loader-utils	3.2.1
react	18.2.0

```
import React from "react";
import { customers } from "./customers";
/**
 * Here we IMPORT the array called customers from the
 * Javascript file called 'customers.js'.
 * BE CAREFUL - make sure you have the array name correct.
 * So, this file can contain a really big array of objects.
 * Our map function remains the same.
 */

function App() {
  return (
    <>
      <h1>CS385 Online Shop Customers</h1>
      {customers.map((c, index) => (
        <p key={index}>
          Customer: <b>{c.email}</b>, Credit: €{c.credit}, Joined: {c.year}
```

CS385 Online Shop Cust

Customer: **jstuchberry0@ehow.com**, Credit: €467

Customer: **nbizzey1@sakura.ne.jp**, Credit: €1391,

Customer: **cgoodrum2@jiathis.com**, Credit: €2708

Customer: **kfelton3@rediff.com**, Credit: €1864, Jo

Customer: **ltanguy4@hibu.com**, Credit: €2566, Joi

Customer: **bhartopp5@mapy.cz**, Credit: €3471, Jo

Customer: **shuntress6@vinaora.com**, Credit: €358

Customer: **umckillop7@ehow.com**, Credit: €2878,

Customer: **lhails8@homestead.com**, Credit: €3378

Customer: **mbattista9@purevolume.com**, Credit: €

Customer: **gtolleta@arizona.edu**, Credit: €4845, Jo

Customer: **bbarnardob@google.com**, Credit: €204

Customer: **lealesc@smugmug.com**, Credit: €3940,

Customer: **educarned@google.de**, Credit: €256, Jo

Customer: **gdeverehunte@china.com.cn**, Credit: €

Customer: **cmorrisf@statcounter.com**, Credit: €14

Currently, this map function displays EVERY object in the customers array.

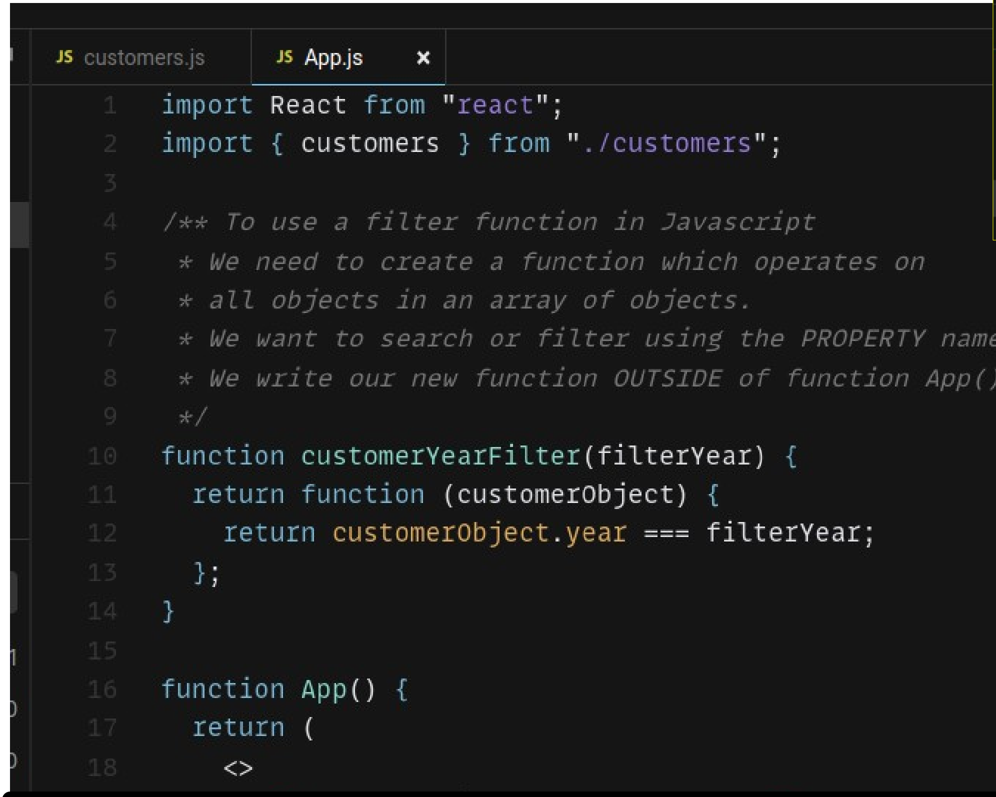
This is a lot of output data

Can we improve this? Can we FILTER the objects and restrict this to just specific objects (ex with certain year)

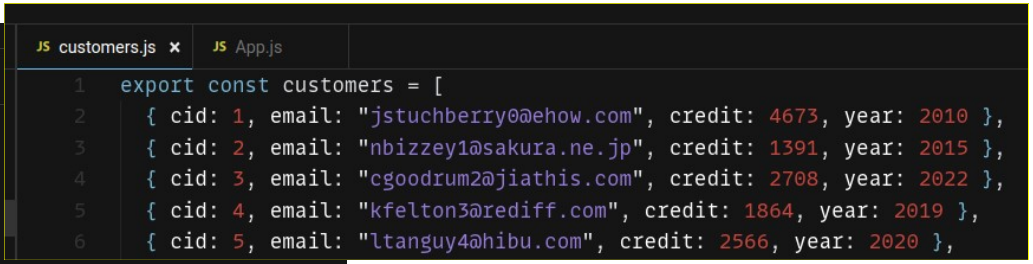
Filtering or filter functions

- The map function will render or display every object in an array.
- Javascript provides us with a means of FILTERING or reducing the number of objects rendered by the map function.
- **We must write our own function(s) to control the filtering – this means we decide which objects are rendered based on their property values**

Writing our own filter function called customerYearFilter



```
import React from "react";
import { customers } from "./customers";

/** To use a filter function in Javascript
 * We need to create a function which operates on
 * all objects in an array of objects.
 * We want to search or filter using the PROPERTY named 'year'
 * We write our new function OUTSIDE of function App().
 */
function customerYearFilter(filterYear) {
  return function (customerObject) {
    return customerObject.year === filterYear;
  };
}

function App() {
  return (
    <>
```



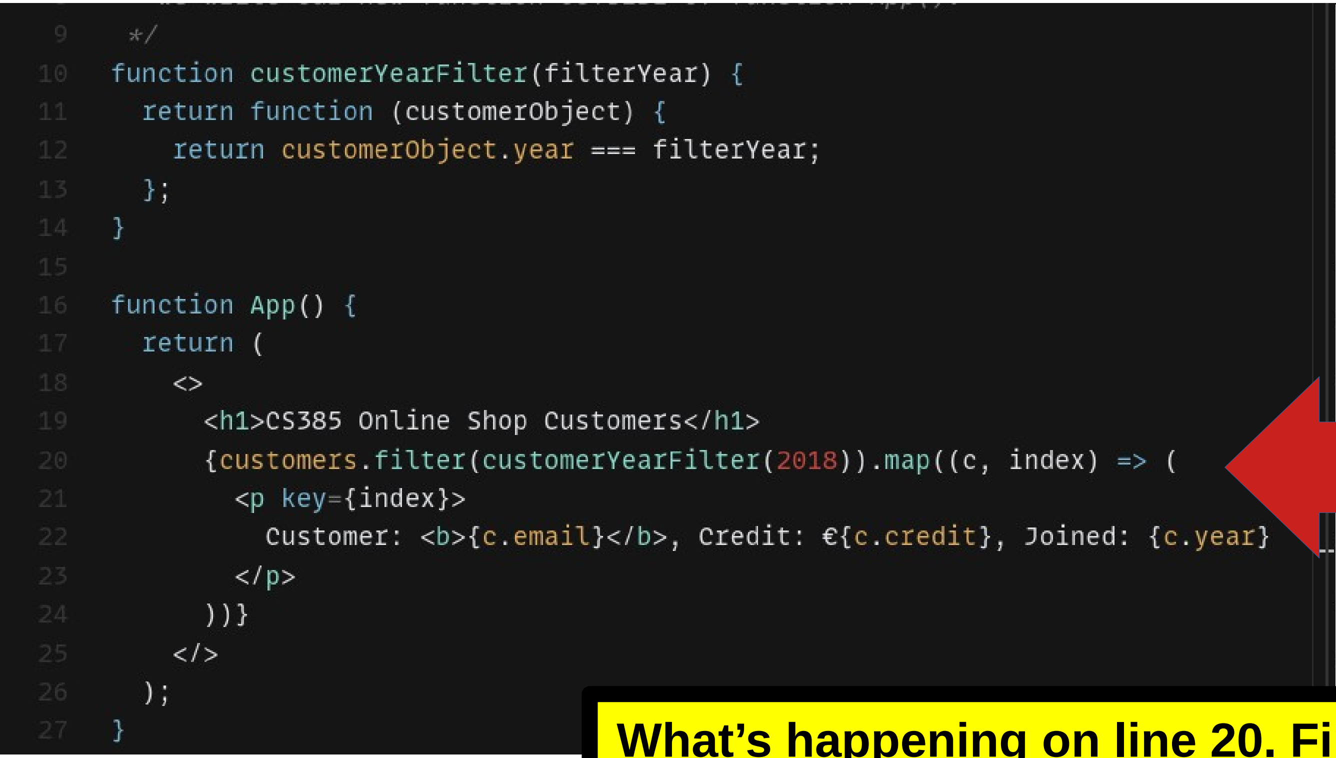
```
export const customers = [
  { cid: 1, email: "jstuchberry0@ehow.com", credit: 4673, year: 2010 },
  { cid: 2, email: "nbizzey1@sakura.ne.jp", credit: 1391, year: 2015 },
  { cid: 3, email: "cgoodrum2@jiathis.com", credit: 2708, year: 2022 },
  { cid: 4, email: "kfelton3@rediff.com", credit: 1864, year: 2019 },
  { cid: 5, email: "ltanguy4@hibu.com", credit: 2566, year: 2020 },
```

We specify the `year` we wish to filter as a parameter to the function called `customerYearFilter`

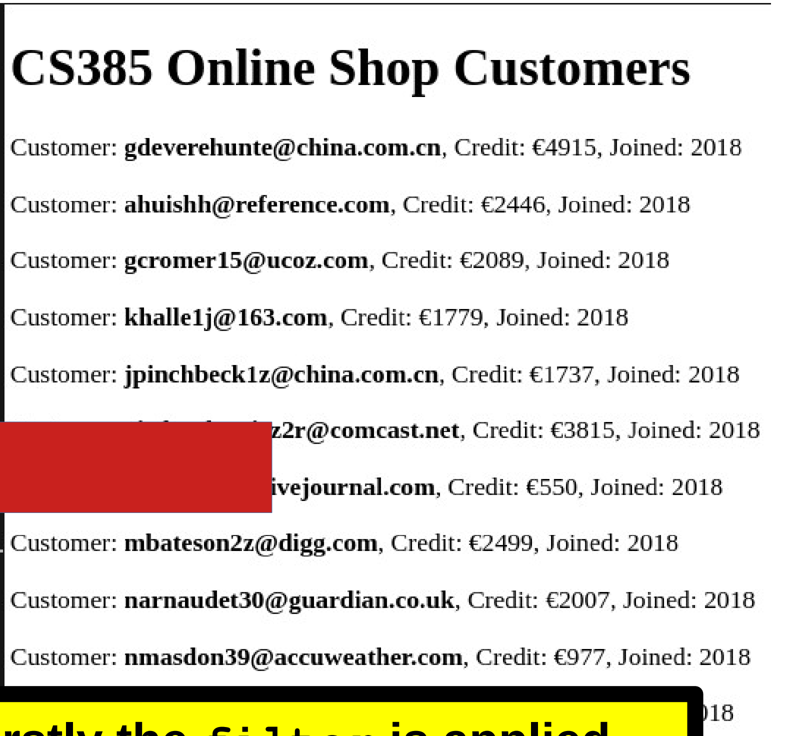
This function then returns `true` if one of the objects has a `year` property with the same value as `filterYear`

To help understand this. Think about the array of objects for our customers.
Every object in this array will be compared or checked using `customerYearFilter`.
If the current object has the same value in the `year` property as specified by the value in `filterYear` then we have a match! The function returns `true`

Finally, (line 28) we apply our filter function to the `customers` array



```
 9    */
10  function customerYearFilter(filterYear) {
11    return function (customerObject) {
12      return customerObject.year === filterYear;
13    };
14  }
15
16  function App() {
17    return (
18      <>
19        <h1>CS385 Online Shop Customers</h1>
20        {customers.filter(customerYearFilter(2018)).map((c, index) => (
21          <p key={index}>
22            Customer: <b>{c.email}</b>, Credit: €{c.credit}, Joined: {c.year}
23          </p>
24        ))}
25      </>
26    );
27  }
```



CS385 Online Shop Customers

Customer: **gdeverehunte@china.com.cn**, Credit: €4915, Joined: 2018

Customer: **ahuishh@reference.com**, Credit: €2446, Joined: 2018

Customer: **gcromer15@ucoz.com**, Credit: €2089, Joined: 2018

Customer: **khalle1j@163.com**, Credit: €1779, Joined: 2018

Customer: **jpinchbeck1z@china.com.cn**, Credit: €1737, Joined: 2018

z2r@comcast.net, Credit: €3815, Joined: 2018

ivejournal.com, Credit: €550, Joined: 2018

Customer: **mbateson2z@digg.com**, Credit: €2499, Joined: 2018

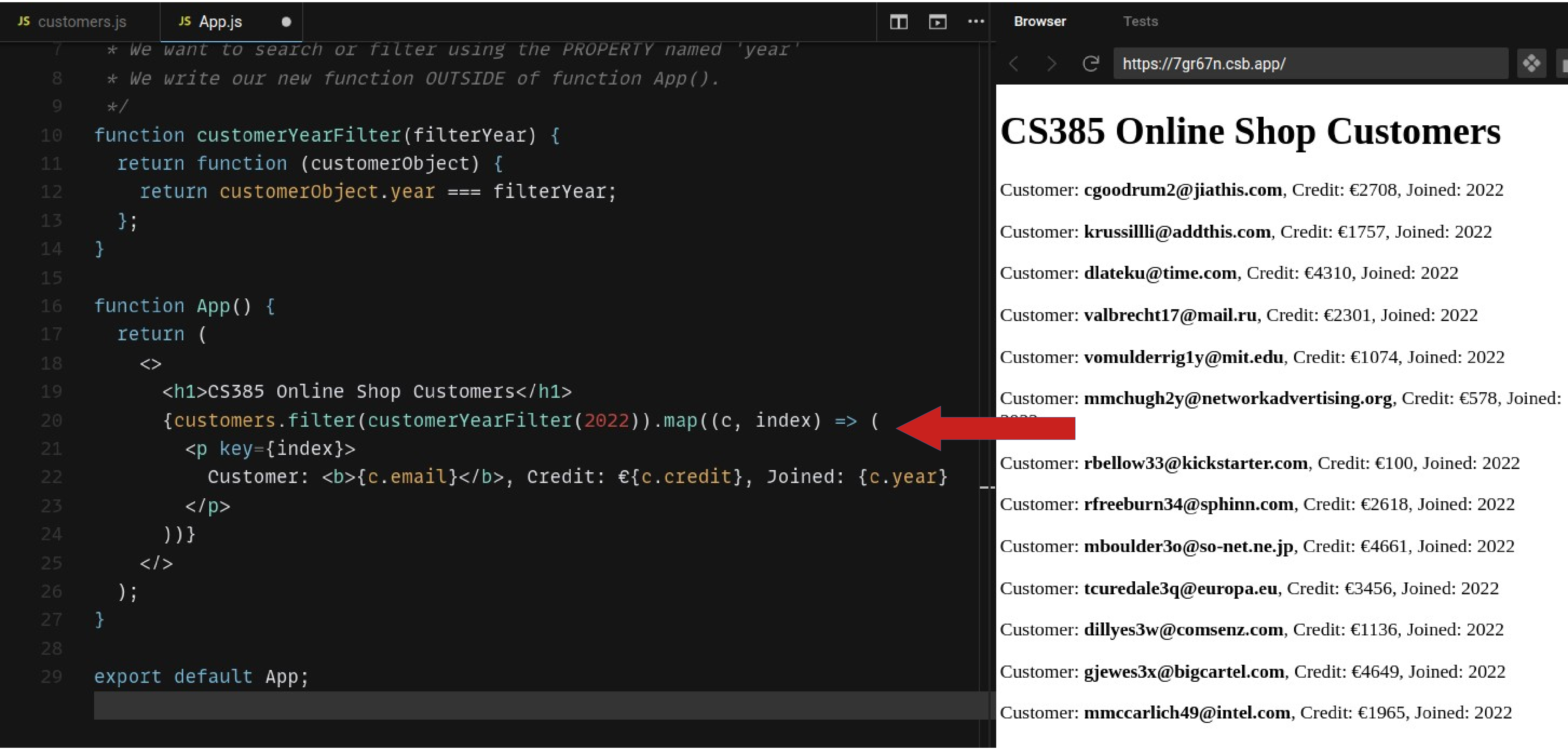
Customer: **narnaudet30@guardian.co.uk**, Credit: €2007, Joined: 2018

Customer: **nmasdon39@accuweather.com**, Credit: €977, Joined: 2018

018

What's happening on line 20. Firstly the `filter` is applied to the `customers` array. ONLY objects in this array where the `year` property is equal to 2018 are retained (placed in a temporary array of objects).
Then when all qualifying objects have been found by the filter the `map` function is then applied – the map function simply renders the results (as we can see the objects from all objects in our `customer` array)

We can easily change the filter value (below we see `year` 2022)



```
 * We want to search or filter using the PROPERTY named 'year'
 * We write our new function OUTSIDE of function App().
 */
function customerYearFilter(filterYear) {
  return function (customerObject) {
    return customerObject.year === filterYear;
  };
}

function App() {
  return (
    <>
      <h1>CS385 Online Shop Customers</h1>
      {customers.filter(customerYearFilter(2022)).map((c, index) => (
        <p key={index}>
          Customer: <b>{c.email}</b>, Credit: €{c.credit}, Joined: {c.year}
        </p>
      ))}
    </>
  );
}

export default App;
```

CS385 Online Shop Customers

Customer: **cgoodrum2@jiathis.com**, Credit: €2708, Joined: 2022

Customer: **krussillli@addthis.com**, Credit: €1757, Joined: 2022

Customer: **dlateku@time.com**, Credit: €4310, Joined: 2022

Customer: **valbrecht17@mail.ru**, Credit: €2301, Joined: 2022

Customer: **vomulderrig1y@mit.edu**, Credit: €1074, Joined: 2022

Customer: **mmchugh2y@networkadvertising.org**, Credit: €578, Joined: 2022

Customer: **rbellow33@kickstarter.com**, Credit: €100, Joined: 2022

Customer: **rfreeburn34@sphinn.com**, Credit: €2618, Joined: 2022

Customer: **mboulder3o@so-net.ne.jp**, Credit: €4661, Joined: 2022

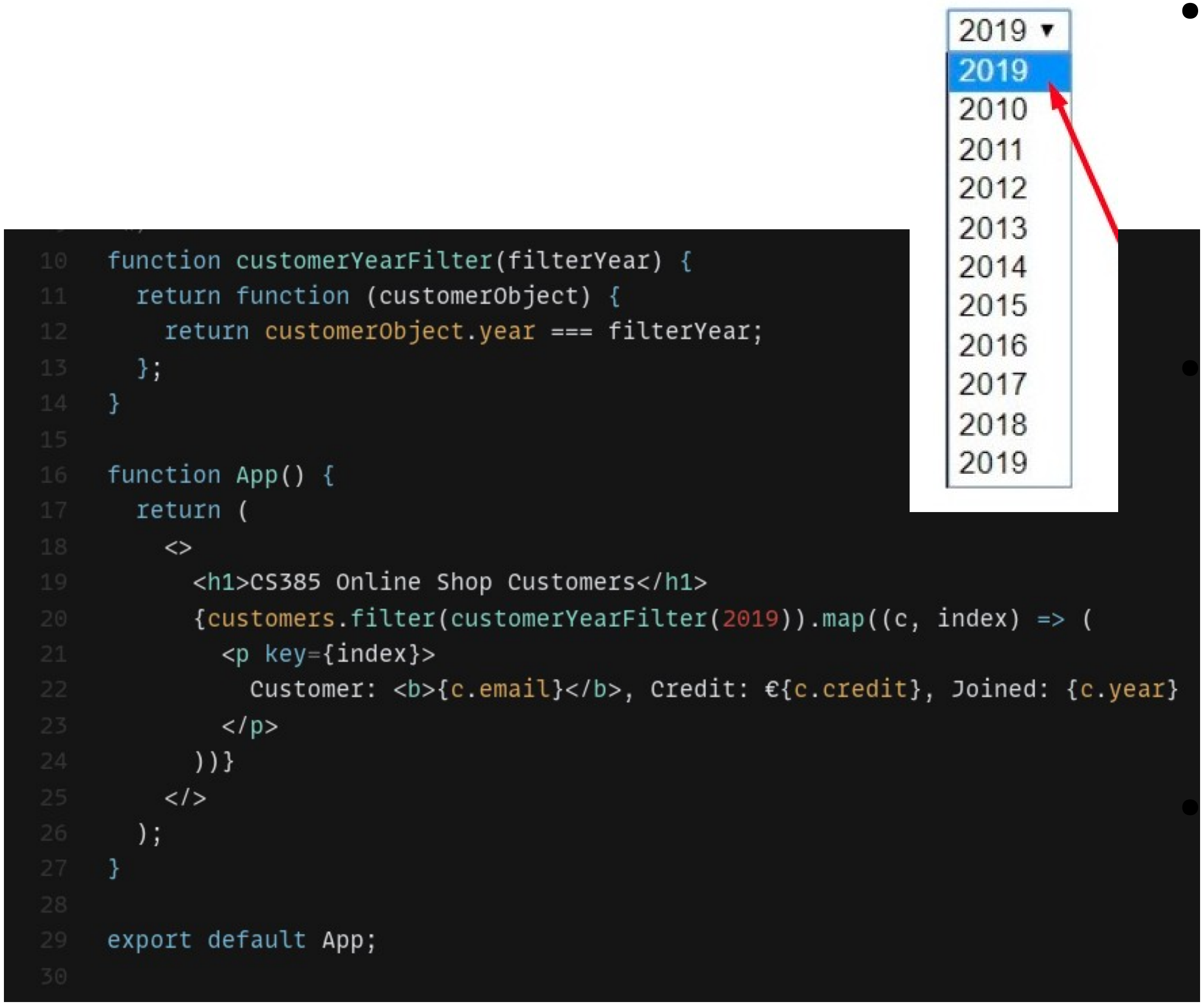
Customer: **tcuredale3q@europa.eu**, Credit: €3456, Joined: 2022

Customer: **dillyes3w@comsenz.com**, Credit: €1136, Joined: 2022

Customer: **gjewes3x@bigcartel.com**, Credit: €4649, Joined: 2022

Customer: **mmccarlich49@intel.com**, Credit: €1965, Joined: 2022

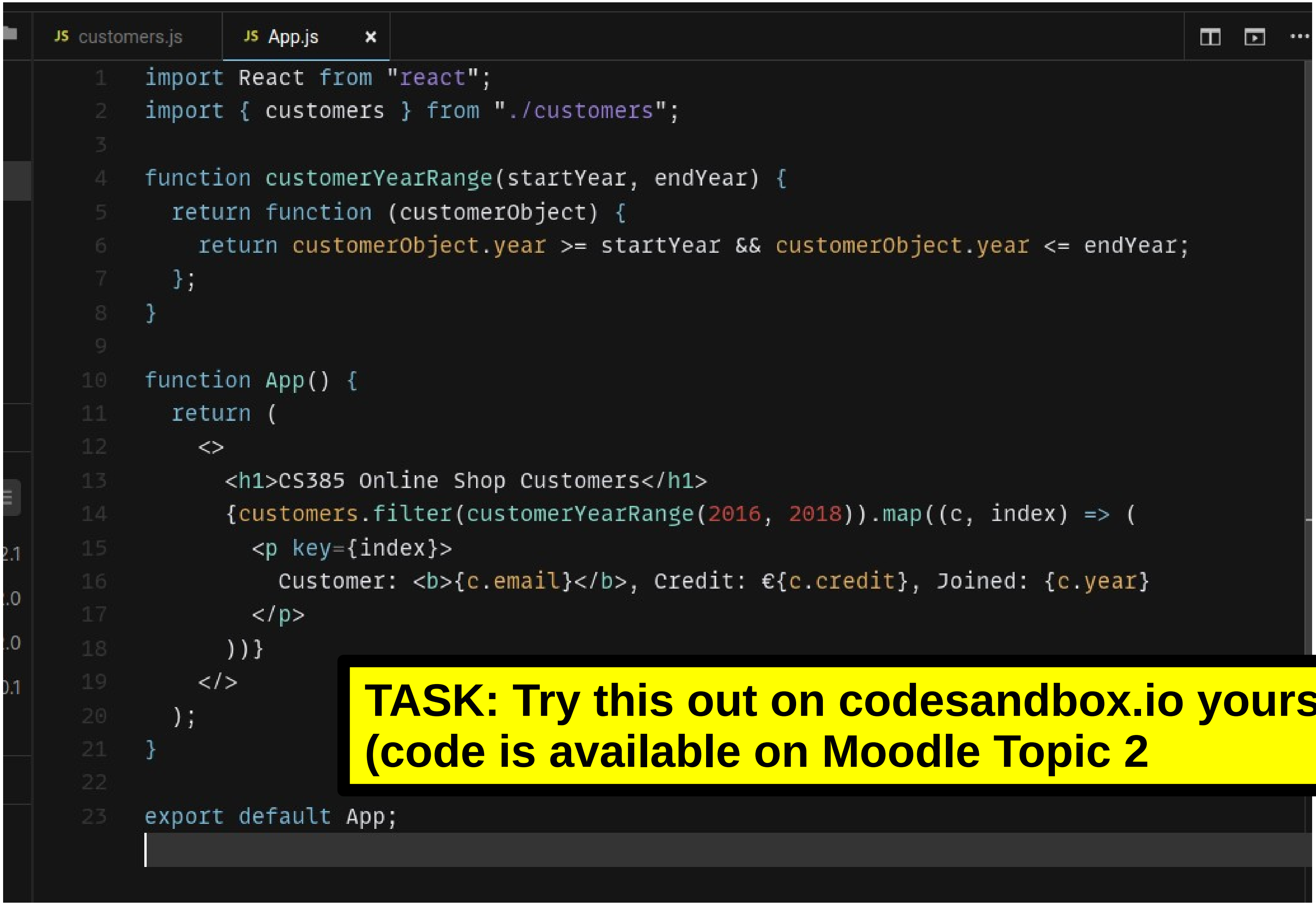
How does this flexible filter function help in real mobile apps?



```
function customerYearFilter(filterYear) {
  return function (customerObject) {
    return customerObject.year === filterYear;
  };
}

function App() {
  return (
    <>
      <h1>CS385 Online Shop Customers</h1>
      {customers.filter(customerYearFilter(2019)).map((c, index) => (
        <p key={index}>
          Customer: <b>{c.email}</b>, Credit: €{c.credit}, Joined: {c.year}
        </p>
      ))}
    </>
  );
}

export default App;
```

- Suppose you have a drop-down-list of years or an input box where a user enters a year value.
- The output from your drop-down-list or input box can be directed into the `customerYearFilter`
- **This is VERY USEFUL and works really really well. We'll see this working in a few lectures time.**

Question: Can you see how the `customerYearRange` filter function works?



```
import React from "react";
import { customers } from "./customers";

function customerYearRange(startYear, endYear) {
  return function (customerObject) {
    return customerObject.year >= startYear && customerObject.year <= endYear;
  };
}

function App() {
  return (
    <>
      <h1>CS385 Online Shop Customers</h1>
      {customers.filter(customerYearRange(2016, 2018)).map((c, index) => (
        <p key={index}>
          Customer: <b>{c.email}</b>, Credit: €{c.credit}, Joined: {c.year}
        </p>
      ))}
    </>
  );
}

export default App;
```

TASK: Try this out on codesandbox.io yourself (code is available on Moodle Topic 2

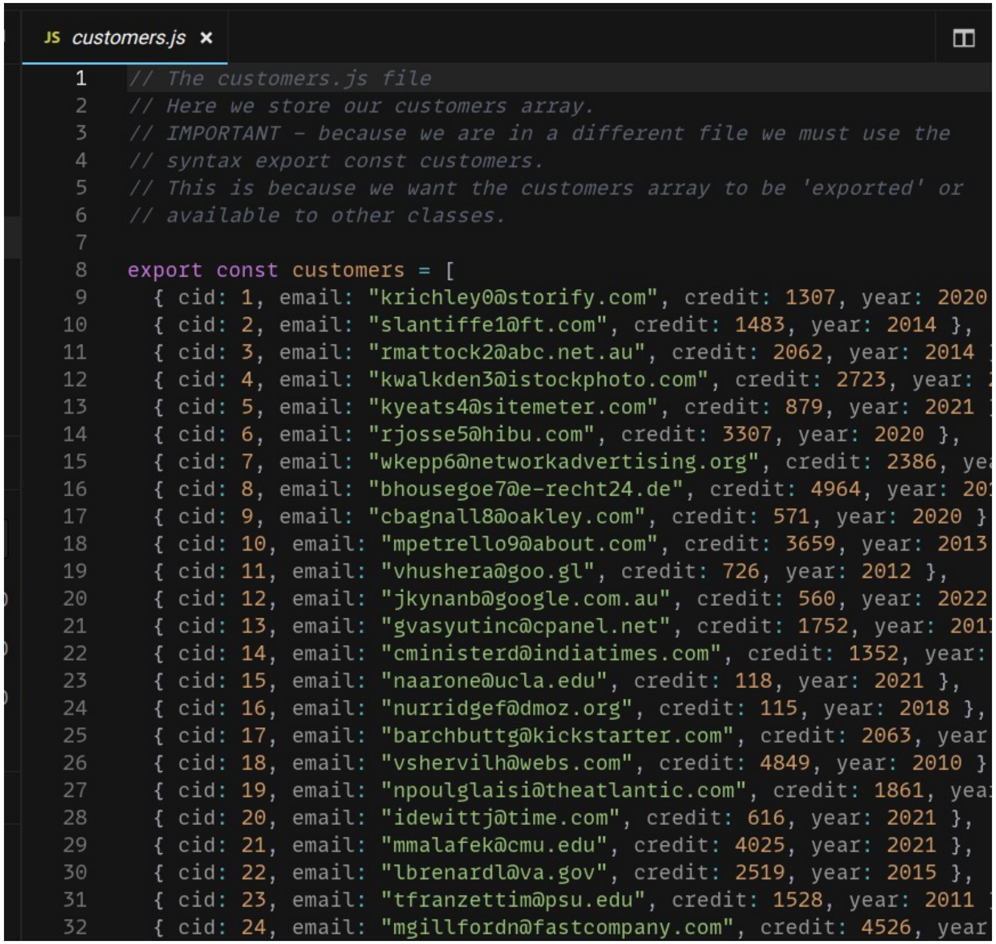
Filtering with string values

Working with strings

- **Javascript has very robust and extensive support for string manipulation** (working with strings). This includes: searching for characters in strings, string operations such as reversal, capitalisation, etc.
- In mobile applications a very large amount of the data submitted or entered by users is string-based data. Indeed a lot of the data passed between various parts of mobile applications is also string-based data.
- Therefore, it is very important to know how to use the functions provided by Javascript.

Example: Suppose we want to search for customer emails?

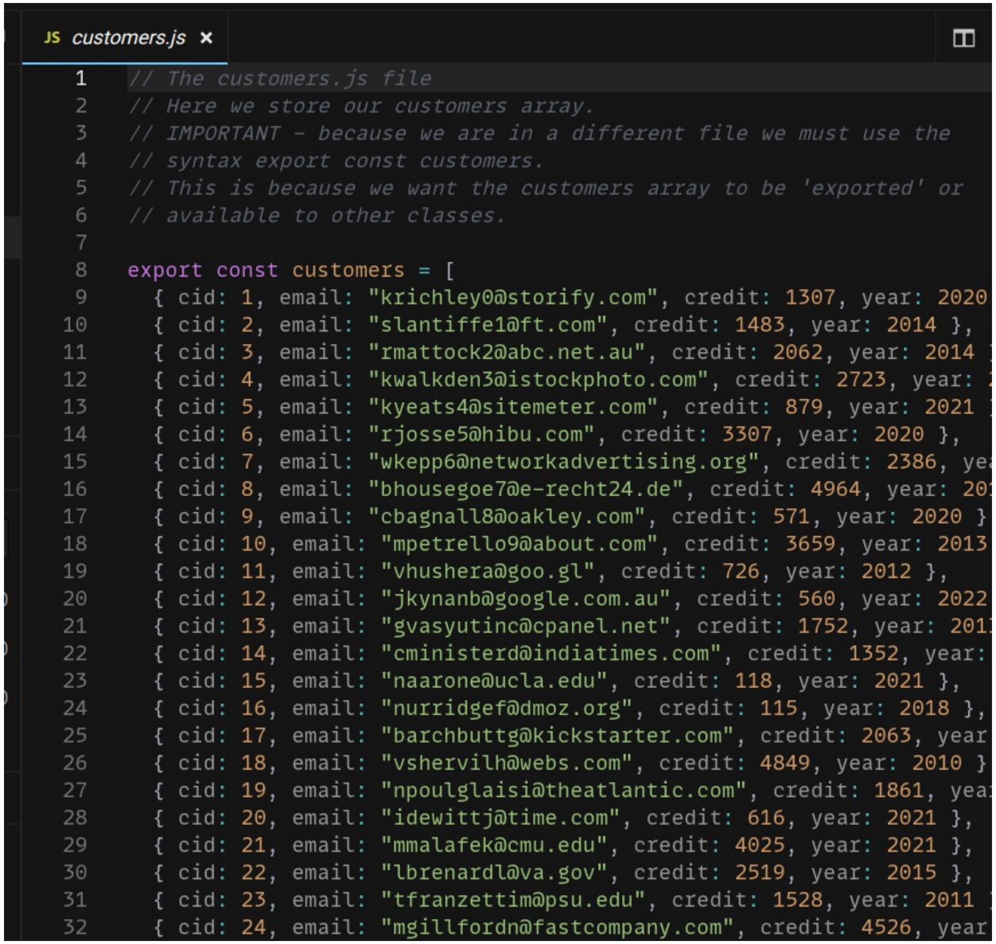
- **TASK:** Can we write our own filter function which can help us only render customers having an email address containing a specific string?
- For example – ending in '`.org`', containing the consecutive letters '`eta`'



```
// The customers.js file
// Here we store our customers array.
// IMPORTANT - because we are in a different file we must use the
// syntax export const customers.
// This is because we want the customers array to be 'exported' or
// available to other classes.

export const customers = [
  { cid: 1, email: "krichley0@storify.com", credit: 1307, year: 2020
  { cid: 2, email: "slantiffe1@ft.com", credit: 1483, year: 2014 },
  { cid: 3, email: "rmattock2@abc.net.au", credit: 2062, year: 2014
  { cid: 4, email: "kwalkden3@istockphoto.com", credit: 2723, year:
  { cid: 5, email: "kyeats4@sitemeter.com", credit: 879, year: 2021
  { cid: 6, email: "rjosse5@hibu.com", credit: 3307, year: 2020 },
  { cid: 7, email: "wkepp6@networkadvertising.org", credit: 2386, ye
  { cid: 8, email: "bhousegoe7@e-recht24.de", credit: 4964, year: 20
  { cid: 9, email: "cbagnall8@oakley.com", credit: 571, year: 2020 }
  { cid: 10, email: "mpetrello9@about.com", credit: 3659, year: 2013
  { cid: 11, email: "vhushera@goo.gl", credit: 726, year: 2012 },
  { cid: 12, email: "jkynanb@google.com.au", credit: 560, year: 2022
  { cid: 13, email: "gvasyutinc@cpanel.net", credit: 1752, year: 201
  { cid: 14, email: "cministerd@indiatimes.com", credit: 1352, year:
  { cid: 15, email: "naarone@ucla.edu", credit: 118, year: 2021 },
  { cid: 16, email: "nurridgef@dmoz.org", credit: 115, year: 2018 },
  { cid: 17, email: "barchbuttg@kickstarter.com", credit: 2063, year
  { cid: 18, email: "vshervilh@webs.com", credit: 4849, year: 2010 }
  { cid: 19, email: "npoulglaisi@theatlantic.com", credit: 1861, yea
  { cid: 20, email: "idewittj@time.com", credit: 616, year: 2021 },
  { cid: 21, email: "mmalafek@cmu.edu", credit: 4025, year: 2021 },
  { cid: 22, email: "lbrenardl@va.gov", credit: 2519, year: 2015 },
  { cid: 23, email: "tfranzettim@psu.edu", credit: 1528, year: 2011
  { cid: 24, email: "mgillfordn@fastcompany.com", credit: 4526, year
```

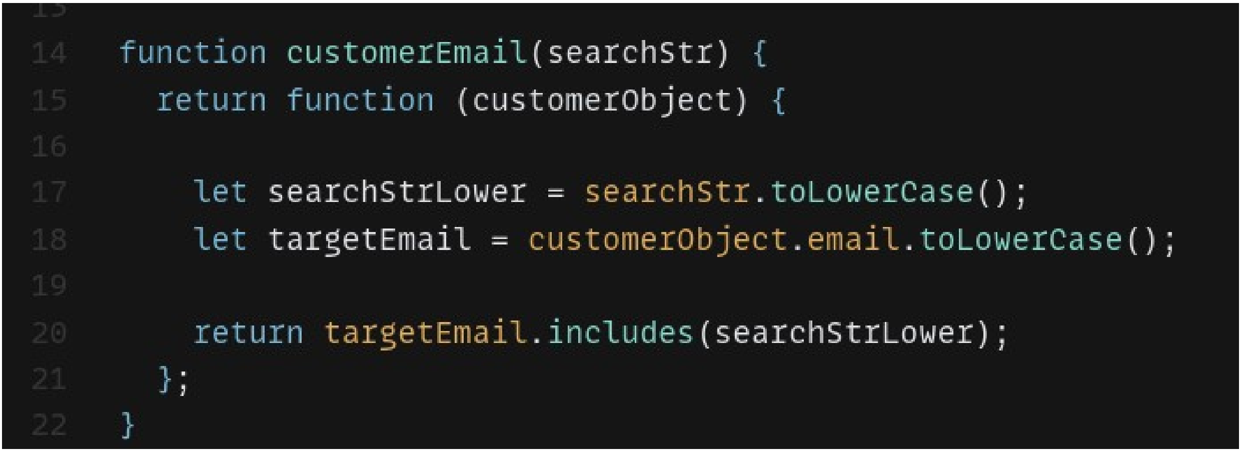
SOLUTION DESIGN: Suppose we want to search for customer emails?

- This task solution is approached the way most string matching or searching tasks are solved.
- Convert the **search** string (for ex `.org`) and the **target** email string to lower case.
- Then use **a string function** to check if the search string is in the target string

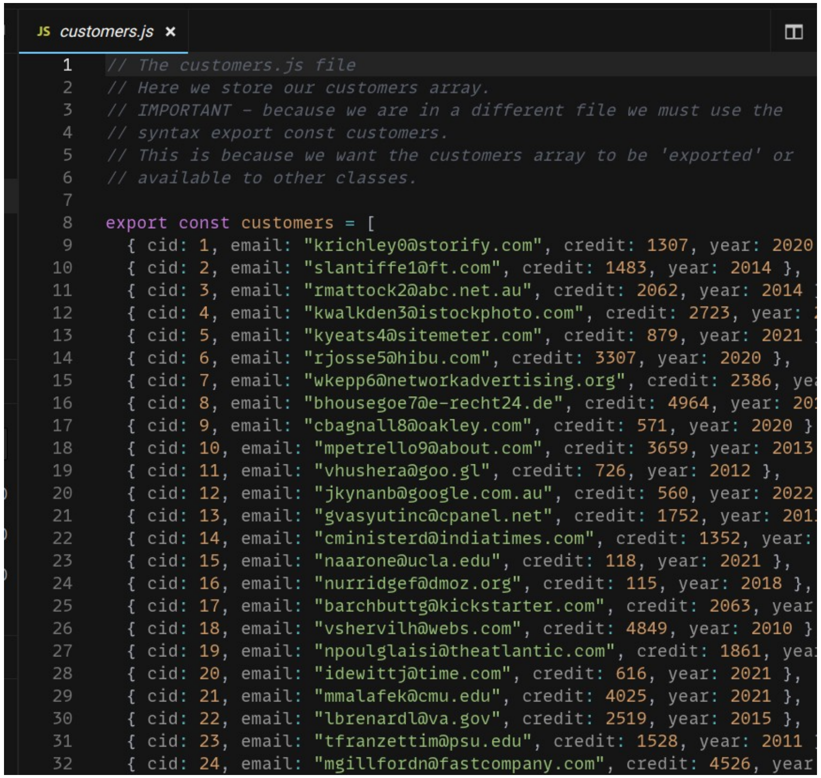


```
// The customers.js file
// Here we store our customers array.
// IMPORTANT - because we are in a different file we must use the
// syntax export const customers.
// This is because we want the customers array to be 'exported' or
// available to other classes.

export const customers = [
  { cid: 1, email: "krichley0@storify.com", credit: 1307, year: 2020
  { cid: 2, email: "slantiffe1@ft.com", credit: 1483, year: 2014 },
  { cid: 3, email: "rmattock2@abc.net.au", credit: 2062, year: 2014
  { cid: 4, email: "kwalkden3@istockphoto.com", credit: 2723, year:
  { cid: 5, email: "kyeats4@sitemeter.com", credit: 879, year: 2021
  { cid: 6, email: "rjosse5@hibu.com", credit: 3307, year: 2020 },
  { cid: 7, email: "wkepp6@networkadvertising.org", credit: 2386, ye
  { cid: 8, email: "bhousegoe7@e-recht24.de", credit: 4964, year: 20
  { cid: 9, email: "cbagnall8@oakley.com", credit: 571, year: 2020 }
  { cid: 10, email: "mpetrello9@about.com", credit: 3659, year: 2013
  { cid: 11, email: "vhushera@goo.gl", credit: 726, year: 2012 },
  { cid: 12, email: "jkynanb@google.com.au", credit: 560, year: 2022
  { cid: 13, email: "gvasyutinc@cpanel.net", credit: 1752, year: 201
  { cid: 14, email: "cministerd@indiatimes.com", credit: 1352, year:
  { cid: 15, email: "naarone@ucla.edu", credit: 118, year: 2021 },
  { cid: 16, email: "nurridgef@dmoz.org", credit: 115, year: 2018 },
  { cid: 17, email: "barchbuttg@kickstarter.com", credit: 2063, year
  { cid: 18, email: "vshervilh@webs.com", credit: 4849, year: 2010 }
  { cid: 19, email: "npoulglaisi@theatlantic.com", credit: 1861, yea
  { cid: 20, email: "idewittj@time.com", credit: 616, year: 2021 },
  { cid: 21, email: "mmalafek@cmu.edu", credit: 4025, year: 2021 },
  { cid: 22, email: "lbrenardl@va.gov", credit: 2519, year: 2015 },
  { cid: 23, email: "tfranzettim@psu.edu", credit: 1528, year: 2011
  { cid: 24, email: "mgillfordn@fastcompany.com", credit: 4526, year
```

SOLUTION CODE: The `customerEmail` function is our filter function



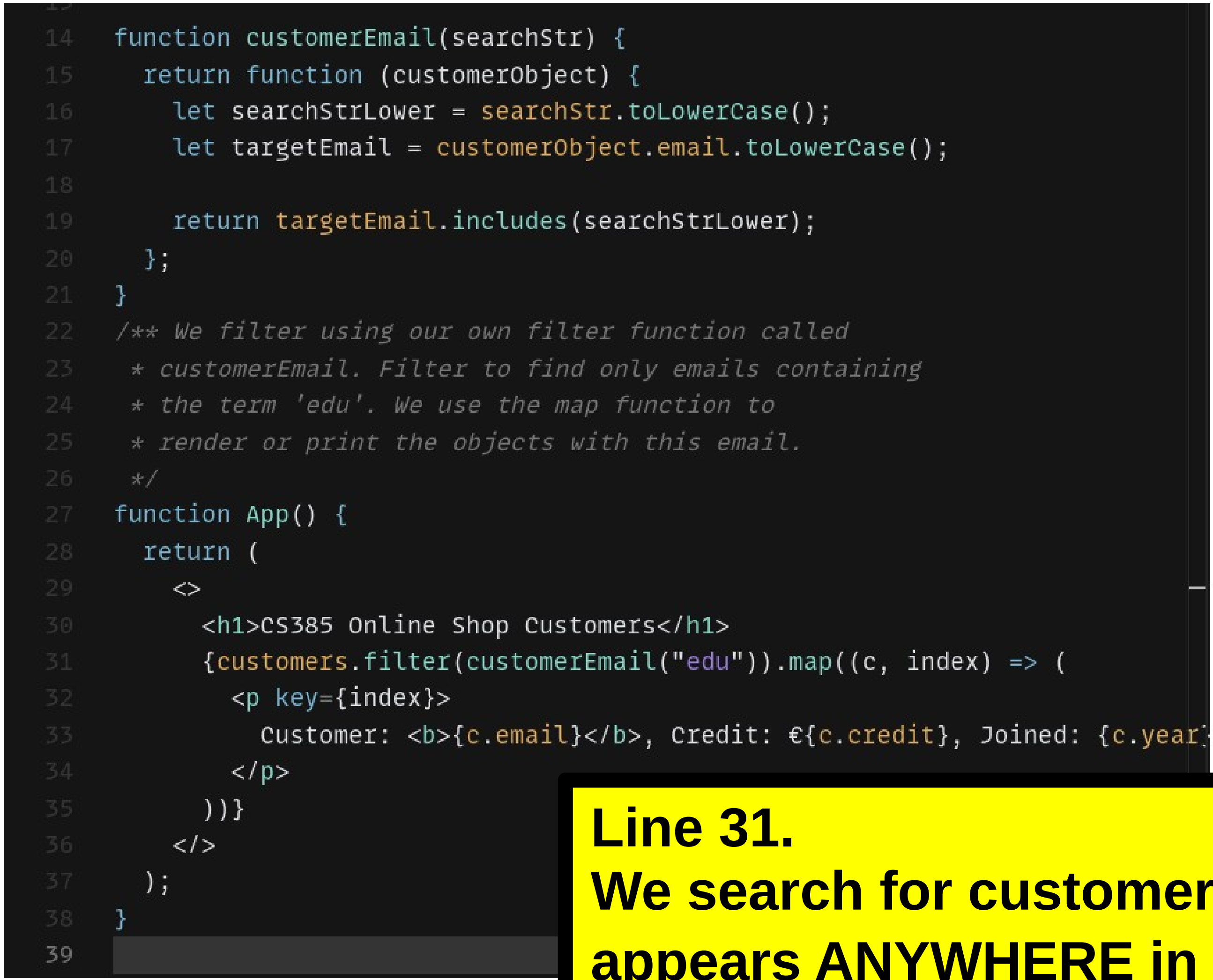
```javascript
function customerEmail(searchStr) {
  return function (customerObject) {

    let searchStrLower = searchStr.toLowerCase();
    let targetEmail = customerObject.email.toLowerCase();

    return targetEmail.includes(searchStrLower);
  };
}
```



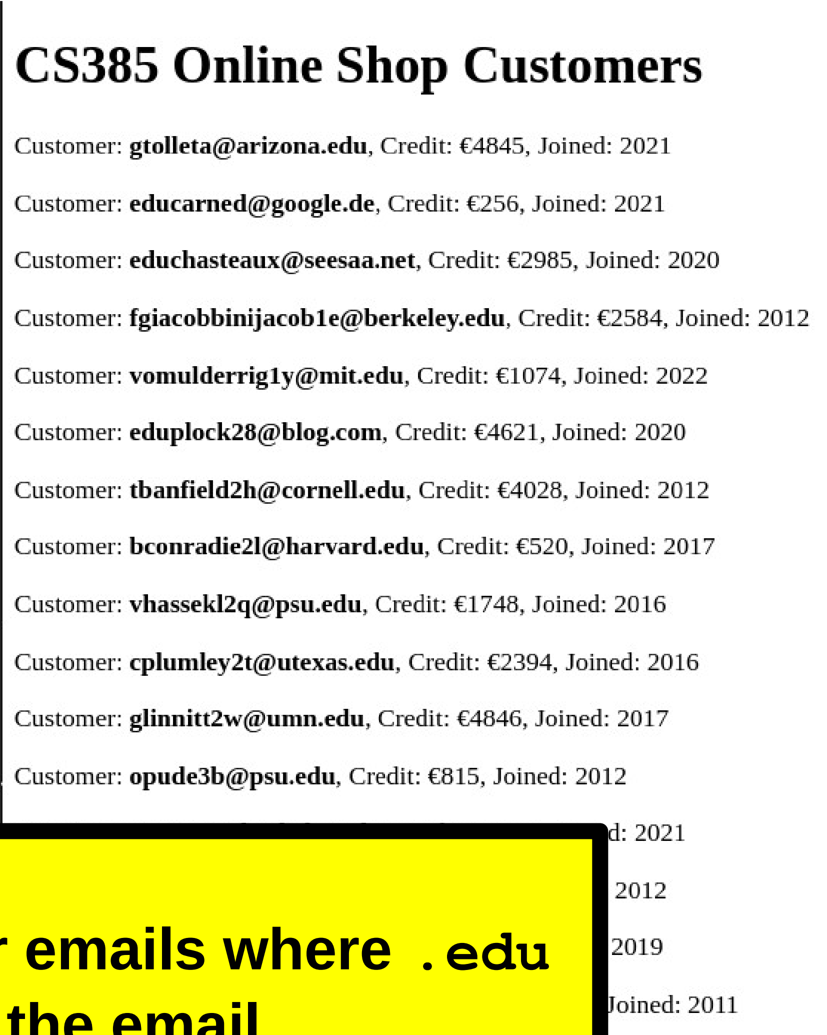
```javascript
// The customers.js file
// Here we store our customers array.
// IMPORTANT - because we are in a different file we must use the
// syntax export const customers.
// This is because we want the customers array to be 'exported' or
// available to other classes.

export const customers = [
  { cid: 1, email: "krichley0@storify.com", credit: 1307, year: 2020
  { cid: 2, email: "slantiffe1@ft.com", credit: 1483, year: 2014 },
  { cid: 3, email: "rmattock2@abc.net.au", credit: 2062, year: 2014
  { cid: 4, email: "kwalkden3@istockphoto.com", credit: 2723, year:
  { cid: 5, email: "kyeats4@sitemeter.com", credit: 879, year: 2021
  { cid: 6, email: "rjosse5@hibu.com", credit: 3307, year: 2020 },
  { cid: 7, email: "wkepp6@networkadvertising.org", credit: 2386, ye
  { cid: 8, email: "bhousegoe7@e-recht24.de", credit: 4964, year: 20
  { cid: 9, email: "cbagnall8@oakley.com", credit: 571, year: 2020 }
  { cid: 10, email: "mpetrello9@about.com", credit: 3659, year: 2013
  { cid: 11, email: "vhushera@goo.gl", credit: 726, year: 2012 },
  { cid: 12, email: "jkynanb@google.com.au", credit: 560, year: 2022
  { cid: 13, email: "gvasyutinc@cpanel.net", credit: 1752, year: 201
  { cid: 14, email: "cministerd@indiatimes.com", credit: 1352, year:
  { cid: 15, email: "naarone@ucla.edu", credit: 118, year: 2021 },
  { cid: 16, email: "nurridgef@dmoz.org", credit: 115, year: 2018 },
  { cid: 17, email: "barchbuttg@kickstarter.com", credit: 2063, year
  { cid: 18, email: "vshervilh@webs.com", credit: 4849, year: 2010 }
  { cid: 19, email: "npoulglaisi@theatlantic.com", credit: 1861, yea
  { cid: 20, email: "idewittj@time.com", credit: 616, year: 2021 },
  { cid: 21, email: "mmalafek@cmu.edu", credit: 4025, year: 2021 },
  { cid: 22, email: "lbrenardl@va.gov", credit: 2519, year: 2015 },
  { cid: 23, email: "tfranzettim@psu.edu", credit: 1528, year: 2011
  { cid: 24, email: "mgillfordn@fastcompany.com", credit: 4526, year
```

Read the comments carefully.
We convert both strings to lowercase to avoid confusion around upper and lower case.
We use the Javascript `includes` function (returns `true` or `false`) to check if the target email contains the search string

We use our `customerEmail` function in our `filter` for the map function



```
14  function customerEmail(searchStr) {
15    return function (customerObject) {
16      let searchStrLower = searchStr.toLowerCase();
17      let targetEmail = customerObject.email.toLowerCase();
18
19      return targetEmail.includes(searchStrLower);
20    };
21  }
22  /** We filter using our own filter function called
23   * customerEmail. Filter to find only emails containing
24   * the term 'edu'. We use the map function to
25   * render or print the objects with this email.
26   */
27  function App() {
28    return (
29      <>
30        <h1>CS385 Online Shop Customers</h1>
31        {customers.filter(customerEmail("edu")).map((c, index) => (
32          <p key={index}>
33            Customer: <b>{c.email}</b>, Credit: €{c.credit}, Joined: {c.year
34          </p>
35        ))}
36      </>
37    );
38  }
39
```



CS385 Online Shop Customers

Customer: **gtolleta@arizona.edu**, Credit: €4845, Joined: 2021

Customer: **educarned@google.de**, Credit: €256, Joined: 2021

Customer: **educhasteaux@seesaa.net**, Credit: €2985, Joined: 2020

Customer: **fgiacobbinijacob1e@berkeley.edu**, Credit: €2584, Joined: 2012

Customer: **vomulderrig1y@mit.edu**, Credit: €1074, Joined: 2022

Customer: **eduplock28@blog.com**, Credit: €4621, Joined: 2020

Customer: **tbanfield2h@cornell.edu**, Credit: €4028, Joined: 2012

Customer: **bconradie2l@harvard.edu**, Credit: €520, Joined: 2017

Customer: **vhassekl2q@psu.edu**, Credit: €1748, Joined: 2016

Customer: **cplumley2t@utexas.edu**, Credit: €2394, Joined: 2016

Customer: **glinnitt2w@umn.edu**, Credit: €4846, Joined: 2017

Customer: **opude3b@psu.edu**, Credit: €815, Joined: 2012

d: 2021

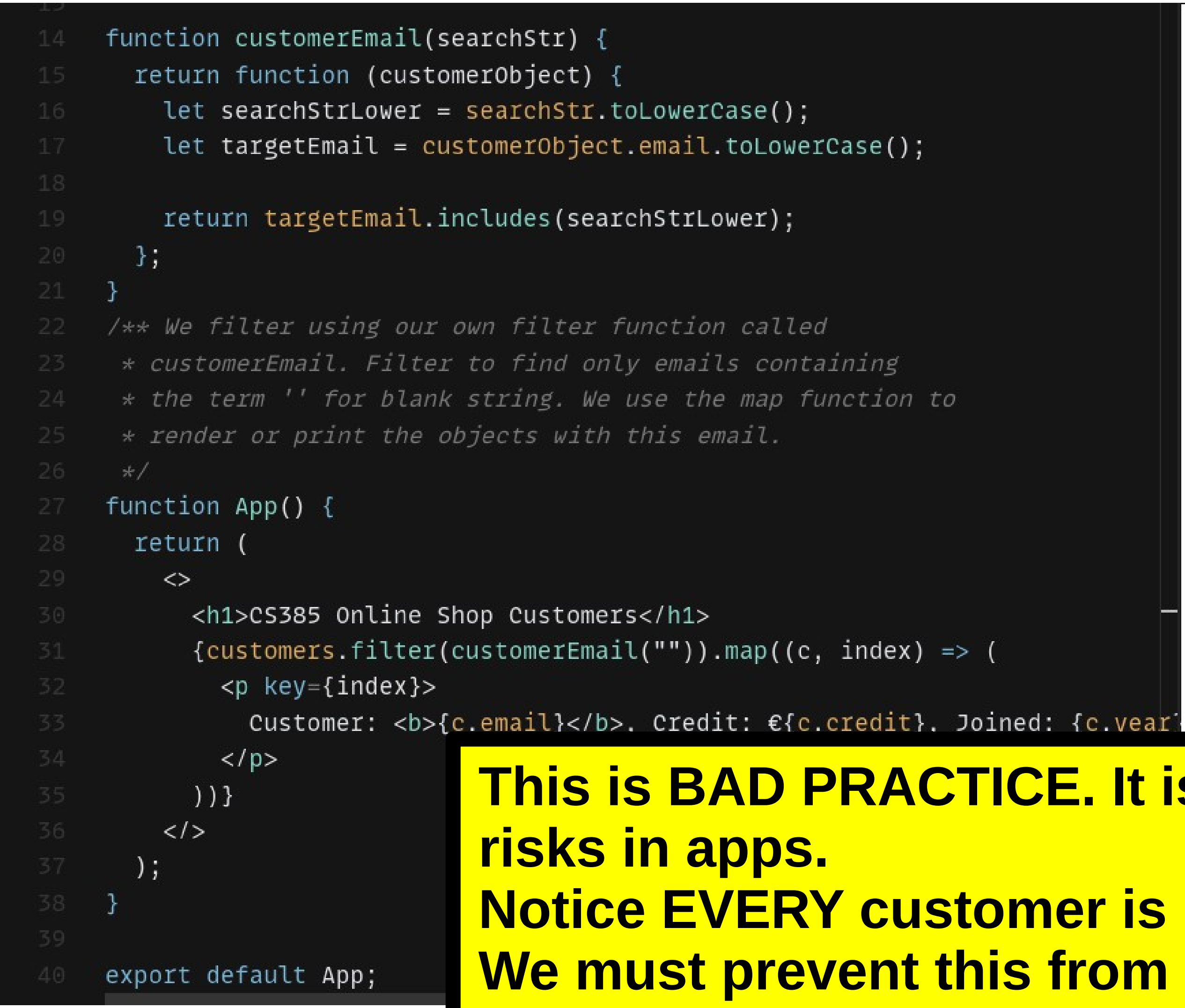
2012

2019

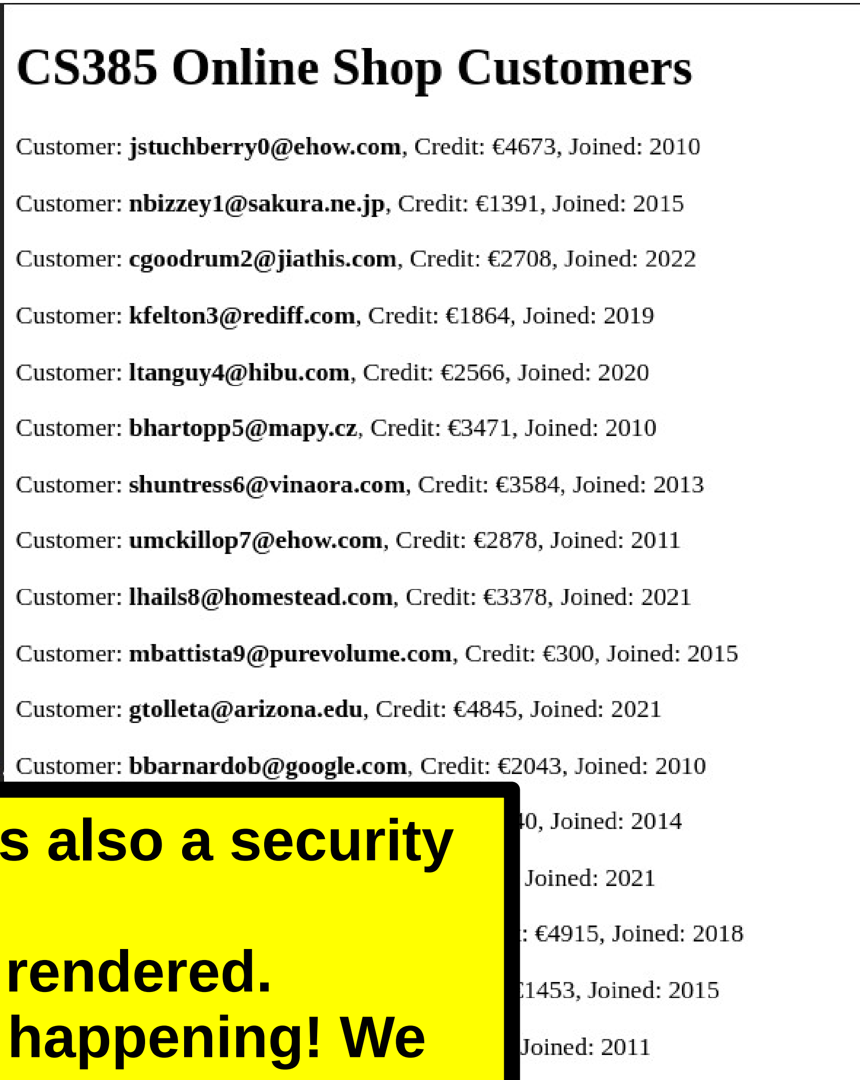
Joined: 2011

Line 31.
We search for customer emails where `.edu` appears ANYWHERE in the email.
The results are rendered and we verify that our function is working correctly

Finally, what if we do not enter a search string (empty string)?



```
function customerEmail(searchStr) {
  return function (customerObject) {
    let searchStrLower = searchStr.toLowerCase();
    let targetEmail = customerObject.email.toLowerCase();

    return targetEmail.includes(searchStrLower);
  };
}
/** We filter using our own filter function called
 * customerEmail. Filter to find only emails containing
 * the term '' for blank string. We use the map function to
 * render or print the objects with this email.
 */
function App() {
  return (
    <>
      <h1>CS385 Online Shop Customers</h1>
      {customers.filter(customerEmail("")).map((c, index) => (
        <p key={index}>
          Customer: <b>{c.email}</b>, Credit: €{c.credit}, Joined: {c.year
        </p>
      ))}
    </>
  );
}

export default App;
```



CS385 Online Shop Customers

Customer: **jstuchberry0@ehow.com**, Credit: €4673, Joined: 2010

Customer: **nbizzey1@sakura.ne.jp**, Credit: €1391, Joined: 2015

Customer: **cgoodrum2@jiathis.com**, Credit: €2708, Joined: 2022

Customer: **kfelton3@rediff.com**, Credit: €1864, Joined: 2019

Customer: **ltanguy4@hibu.com**, Credit: €2566, Joined: 2020

Customer: **bhartopp5@mapy.cz**, Credit: €3471, Joined: 2010

Customer: **shuntress6@vinaora.com**, Credit: €3584, Joined: 2013

Customer: **umckillop7@ehow.com**, Credit: €2878, Joined: 2011

Customer: **lhails8@homestead.com**, Credit: €3378, Joined: 2021

Customer: **mbattista9@purevolume.com**, Credit: €300, Joined: 2015

Customer: **gtolleta@arizona.edu**, Credit: €4845, Joined: 2021

Customer: **bbarnardob@google.com**, Credit: €2043, Joined: 2010

0, Joined: 2014

Joined: 2021

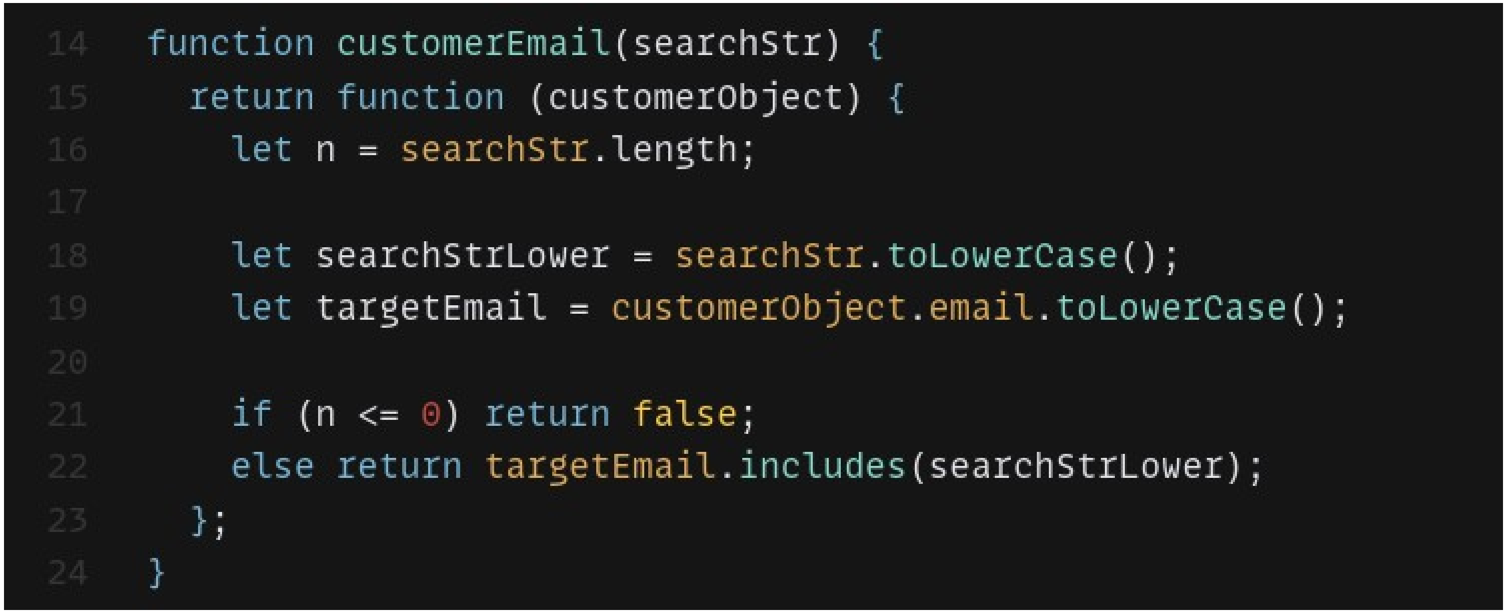
: €4915, Joined: 2018

1453, Joined: 2015

Joined: 2011

This is BAD PRACTICE. It is also a security risks in apps.
Notice EVERY customer is rendered.
We must prevent this from happening! We have to add some code to `customerEmail`

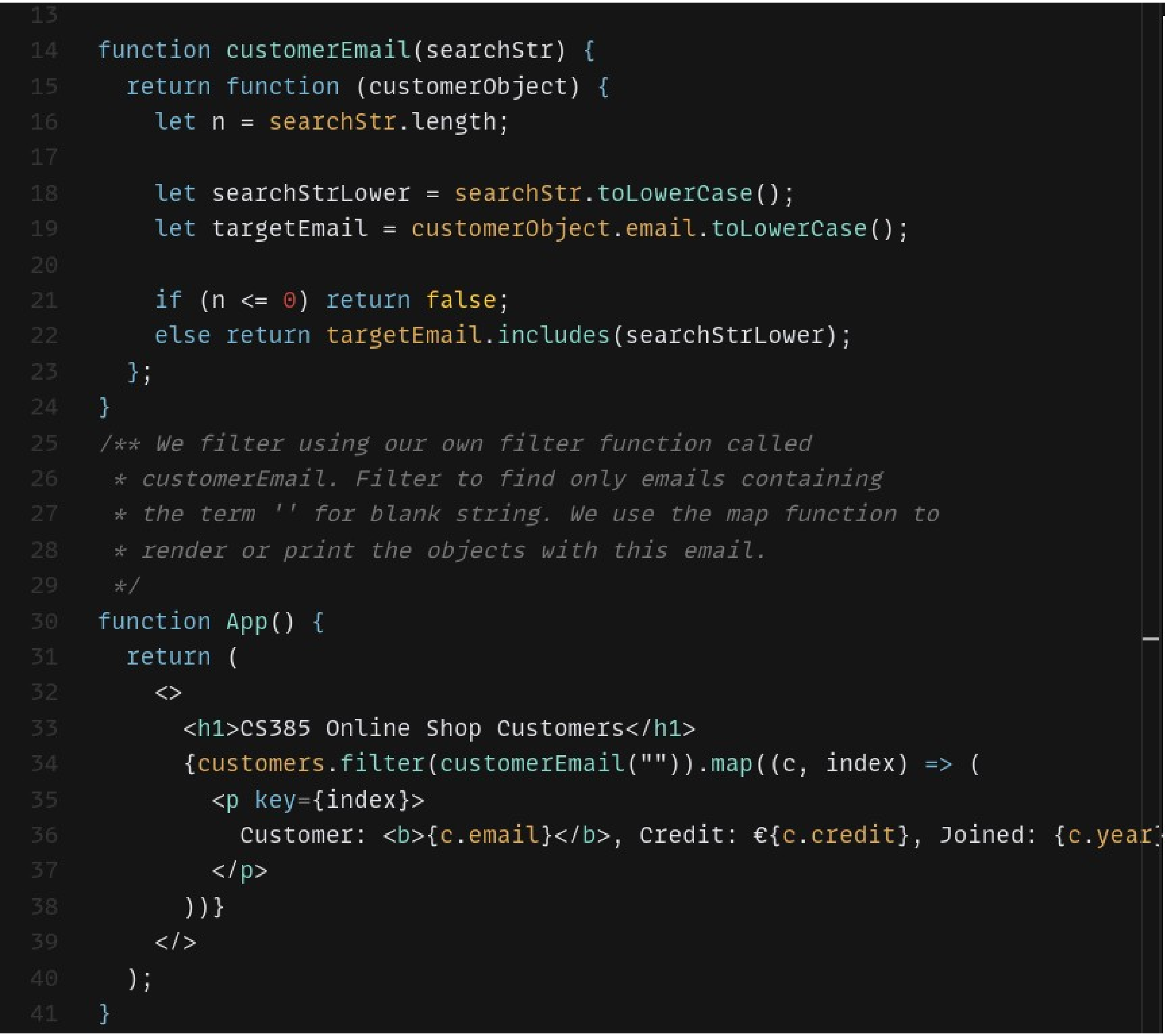
Safety check – for an empty `searchStr`



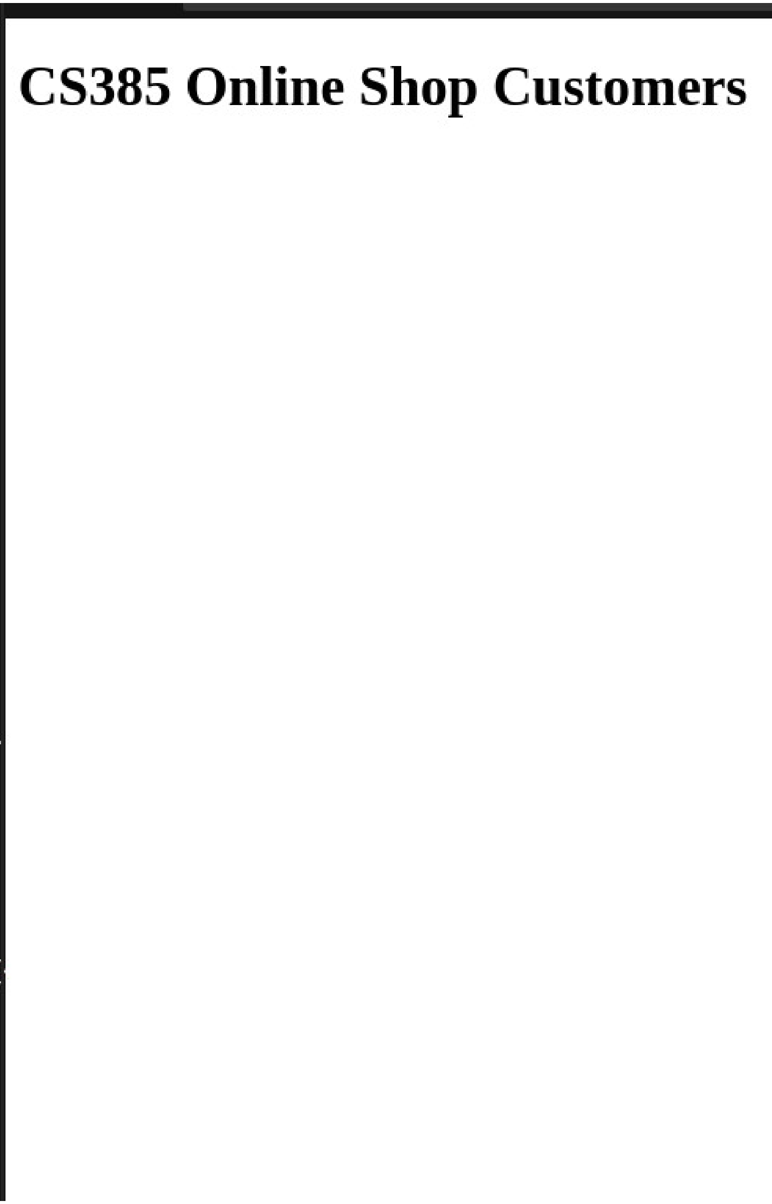
```
14  function customerEmail(searchStr) {
15    return function (customerObject) {
16      let n = searchStr.length;
17
18      let searchStrLower = searchStr.toLowerCase();
19      let targetEmail = customerObject.email.toLowerCase();
20
21      if (n <= 0) return false;
22      else return targetEmail.includes(searchStrLower);
23    };
24  }
```

Line 21. We use the `length` property of a string in Javascript to find the length (number of characters in a string). Empty strings have 0 characters

Working! Line 34 shows the search for an empty string



```
13
14  function customerEmail(searchStr) {
15    return function (customerObject) {
16      let n = searchStr.length;
17
18      let searchStrLower = searchStr.toLowerCase();
19      let targetEmail = customerObject.email.toLowerCase();
20
21      if (n <= 0) return false;
22      else return targetEmail.includes(searchStrLower);
23    };
24  }
25  /** We filter using our own filter function called
26   * customerEmail. Filter to find only emails containing
27   * the term '' for blank string. We use the map function to
28   * render or print the objects with this email.
29   */
30  function App() {
31    return (
32      <>
33        <h1>CS385 Online Shop Customers</h1>
34        {customers.filter(customerEmail("")).map((c, index) => (
35          <p key={index}>
36            Customer: <b>{c.email}</b>, Credit: €{c.credit}, Joined: {c.year
37          </p>
38        ))}
39      </>
40    );
41  }
```

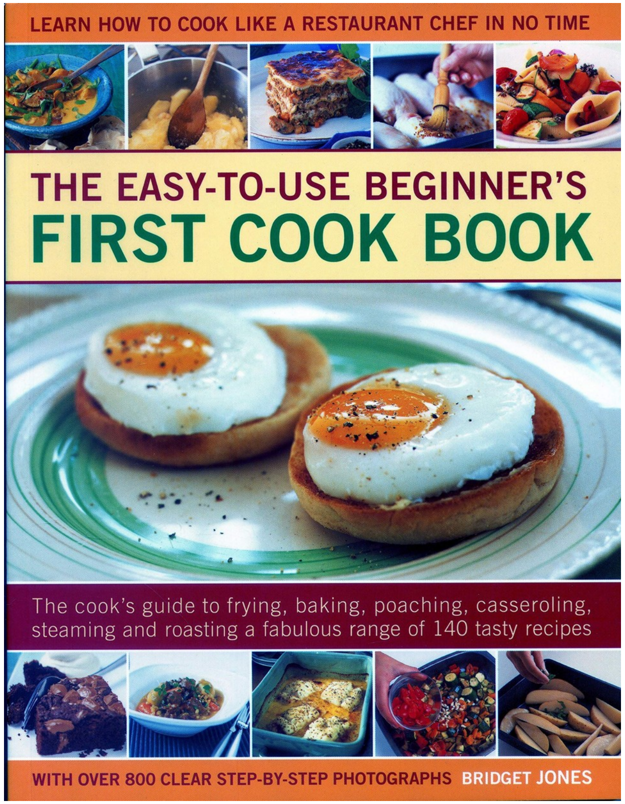


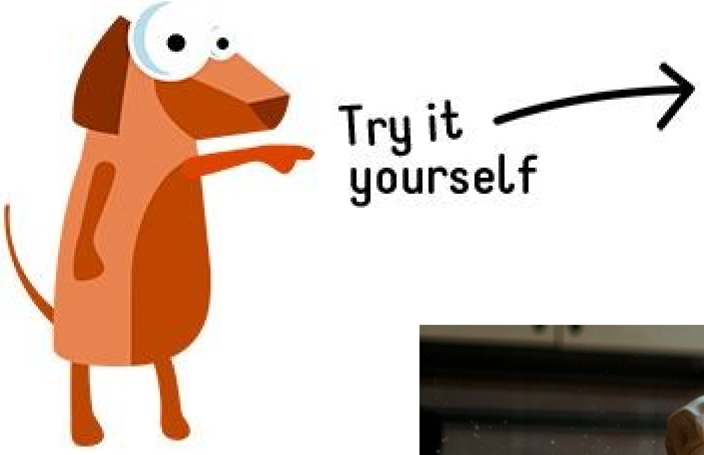
CS385 Online Shop Customers

In-lecture DEMO

Best way to learn? Try out the examples in codesandbox.io by youself

- No one every learned how to cook without actually making a mess in the kitchen!


LEARN HOW TO COOK LIKE A RESTAURANT CHEF IN NO TIME
THE EASY-TO-USE BEGINNER'S
FIRST COOK BOOK
The cook's guide to frying, baking, poaching, casseroling, steaming and roasting a fabulous range of 140 tasty recipes
WITH OVER 800 CLEAR STEP-BY-STEP PHOTOGRAPHS
BRIDGET JONES



Popup Quiz time

- Go to Moodle – during the lecture – click on the Popup Quiz for Topic 2

The CS385 Project

- Full details about the project – discussed within the lecture.
- **Link to project website will be provided in the lectures**

CS385 Project – Week 2

- **First steps [if you haven't done this already]**
  - Step 1 – **decide if you will do an individual project or be part of a group** (2, 3, or 4 students) [I *advise group projects*]
  - Step 1a – if you want to be part of a group – start attempting to form that group NOW.
  - Step 2 – **start thinking of ideas** – brainstorming – you'll soon start to see how you can code up or implement those ideas.

Review the project documentation website carefully

See you on Tuesday 10th October for Lecture 5 + 6 (16:00 – 18:00)

See you for your first CS385 Lab (Friday 6th October 2023
10:00 – 12:00)
IN THE CALLAN Building
S/W Lab 4 (Ground floor)

All content available on Moodle

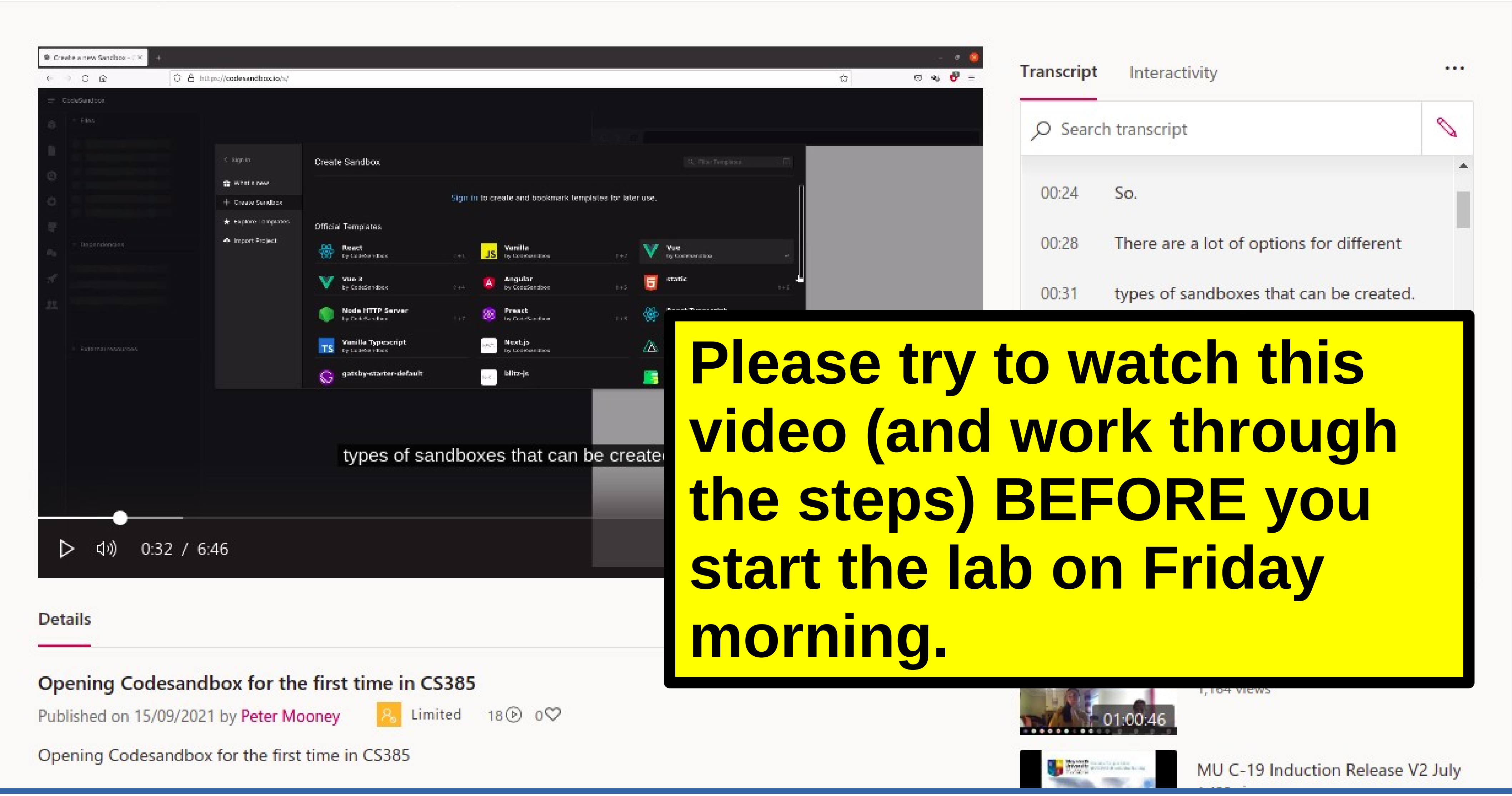
CS385 Lab 1

- **Lab 1 Assignment sheet will be available on Thursday evening (5th October 2023 after 6pm)**
- **Assignment: 2 parts**
  - Working with rendering of variables in React
  - Working with Filter functions in React with an external Javascript file.
  - You will be given some code to get your started!
- You must upload your Javascript Code (using Moodle) by 15:59 Tuesday 10th October 2023

CS385 Lab 1… some advice

- Watch the “Opening Codesandbox” screencast (Link is on Moodle ….) before the lab
- Read the Assignment Sheet CAREFULLY!
- You can work together in the lab!

Link to “Opening Code Sandbox” video available on Moodle



00:24 So.

00:28 There are a lot of options for different

00:31 types of sandboxes that can be created.

Opening Codesandbox for the first time in CS385

Published on 15/09/2021 by Peter Mooney

18 0

Opening Codesandbox for the first time in CS385

MU C-19 Induction Release V2 July

Please try to watch this video (and work through the steps) BEFORE you start the lab on Friday morning.

Lecture 3 & 4