

# CS385 Mobile Application Development (Lecture 3)



**Peter Mooney**

# Lecture 3 and Lecture 4 - menu

- Working with objects (map function again)
- Storing our object arrays in files.
- Using filtering with the map function when we have large arrays of objects.
- Popup Quiz – Topic 2
- Overview of the CS385 project
- Information about your first CS385 Lab

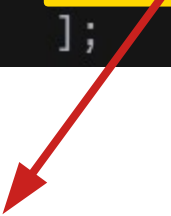
**Some revision from Lecture 1  
and Lecture 2 (Arrays of Objects)**

# A first look at Objects and arrays in Javascript (continued from lecture 2)

- Javascript is natively capable of working with objects (like those you have seen in Java)
- Indeed, Javascript allows us to very easily define objects and manipulate them.
- **One very important concept in Mobile Application Development is working with ARRAYS of Objects. This will become more obvious in a few weeks time.**
- So let's look at some objects and then move to arrays.

# Objects – understanding properties and values

```
6 let planets = [  
7   { id: 1, name: "Venus", gravity: 8.9, dayhrs: 2808, rings: "No" },  
8   { id: 2, name: "Earth", gravity: 9.8, dayhrs: 24, rings: "No" },  
9   { id: 3, name: "Mars", gravity: 3.7, dayhrs: 24.7, rings: "No" },  
10  { id: 4, name: "Jupiter", gravity: 23.1, dayhrs: 9.9, rings: "Yes" },  
11  { id: 5, name: "Saturn", gravity: 9, dayhrs: 10.7, rings: "Yes" },  
12  { id: 6, name: "Uranus", gravity: 8.7, dayhrs: 17.2, rings: "Yes" },  
13  { id: 7, name: "Neptune", gravity: 11, dayhrs: 16.1, rings: "Yes" },  
14  { id: 8, name: "Pluto", gravity: 0.7, dayhrs: 153, rings: "No" },  
15  { id: 9, name: "Mercury", gravity: 3.7, dayhrs: 4222.6, rings: "No" }  
16  ];
```



Usually, in an array of objects, there is one property (in this case `id`) which is UNIQUE – if you compare it to other objects.



This is a single object

It has 5 PROPERTIES or ATTRIBUTES with the names: `id`, `name`, `gravity`, `dayhrs`, `rings`

PROPERTY names can be enclosed in double quotes if you wish



Property



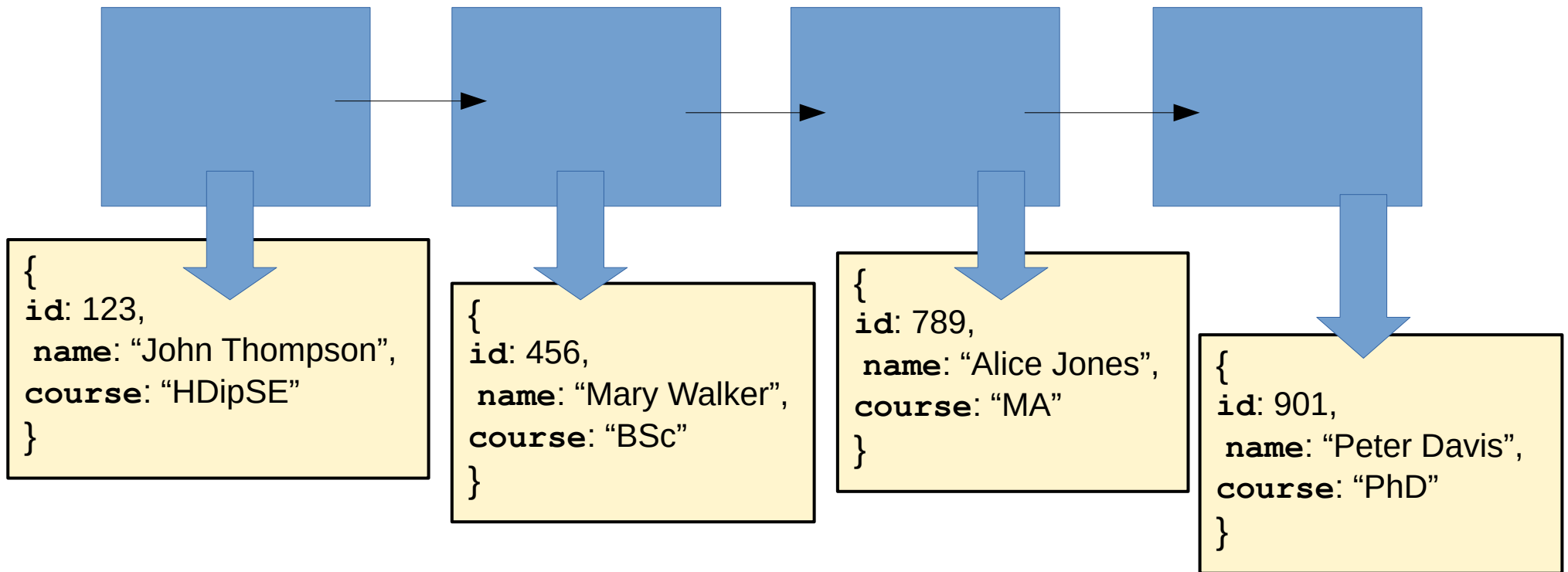
Value

# Array of objects (visual)

```
let students = [  
  { id: 123, name: "John Thompson", course: "HDipSE" },  
  { id: 456, name: "Mary Walker", course: "BSc" },  
  { id: 789, name: "Alice Jones", course: "MA" },  
  { id: 901, name: "Peter Davis", course: "PhD" }  
]
```

- The array of objects called students – an array is always declared using square brackets.
- Each object is an array element – we can add more if we need

Elements: `students[0]`   `students[1]`   `students[2]`   `students[3]`



# Understanding the map function applied to an array

- The map function is used to render the array.
- It is analogous to a for loop – but we NEVER specify the length of the array. The map function always starts at the first element and moves to the end.
- Notice how we use the curly bracket notation to print out each **property** value of each object.
- The variable **p** is the name we give each object as the map function moves from element to element in the array

```
3 function App() {
4   // let's declare an array of planet objects.
5   let planets = [
6     { id: 1, name: "Venus", gravity: 8.9, dayhrs: 2808},
7     { id: 2, name: "Earth", gravity: 9.8, dayhrs: 24 },
8     { id: 3, name: "Mars", gravity: 3.7, dayhrs: 24.7},
9     { id: 4, name: "Jupiter", gravity: 23.1, dayhrs: 9.9},
10    { id: 5, name: "Saturn", gravity: 9, dayhrs: 10.7},
11    { id: 6, name: "Uranus", gravity: 8.7, dayhrs: 17.2},
12    { id: 7, name: "Neptune", gravity: 11, dayhrs: 16.1 },
13    { id: 8, name: "Pluto", gravity: 0.7, dayhrs: 153 },
14    { id: 9, name: "Mercury", gravity: 3.7, dayhrs: 4222.6}
15  ];
16  return (
17    <>
18    <h1>CS385 Solar System</h1>
19    {planets.map((p,index) => (
20      <p key={index}>
21        <b>{p.name}</b>, Gravity = {p.gravity}m/s<sup>2</sup>
22      </p>
23    )})
24    </>
25  );
26 }
```





# Array of objects (The planets)

## Source code: **Lecture3-Planets.js**

```
JS App.js • [icons] Browser Tests
https://fxkqlt.csb.app/

1 import React from "react";
2
3 function App() {
4   // let's declare an array of planet objects.
5   let planets = [
6     { id: 1, name: "Venus", gravity: 8.9, dayhrs: 2808 },
7     { id: 2, name: "Earth", gravity: 9.8, dayhrs: 24 },
8     { id: 3, name: "Mars", gravity: 3.7, dayhrs: 24.7 },
9     { id: 4, name: "Jupiter", gravity: 23.1, dayhrs: 9.9 },
10    { id: 5, name: "Saturn", gravity: 9, dayhrs: 10.7 },
11    { id: 6, name: "Uranus", gravity: 8.7, dayhrs: 17.2 },
12    { id: 7, name: "Neptune", gravity: 11, dayhrs: 16.1 },
13    { id: 8, name: "Pluto", gravity: 0.7, dayhrs: 153 },
14    { id: 9, name: "Mercury", gravity: 3.7, dayhrs: 4222.6 }
15  ];
16  return (
17    <>
18      <h1>CS385 Solar System</h1>
19      {planets.map((p, index) => (
20        <p key={index}>
21          <b>{p.name}</b>, Gravity = {p.gravity}m/s<sup>2</sup>
22        </p>
23      )]}
24    </>
25  );
26 }
27 export default App;
```

### CS385 Solar System

Venus, Gravity = 8.9m/s<sup>2</sup>

Earth, Gravity = 9.8m/s<sup>2</sup>

Mars, Gravity = 3.7m/s<sup>2</sup>

Jupiter, Gravity = 23.1m/s<sup>2</sup>

Saturn, Gravity = 9m/s<sup>2</sup>

Uranus, Gravity = 8.7m/s<sup>2</sup>

Neptune, Gravity = 11m/s<sup>2</sup>

Pluto, Gravity = 0.7m/s<sup>2</sup>

Mercury, Gravity = 3.7m/s<sup>2</sup>

#### KEY UNDERSTANDING:

The map function Line 19 – 23

The variable 'p'

The rendering of the property values with **0**



# Object property names are always case sensitive.

- Property names are ALWAYS case sensitive. Look at the example of “Gravity” below

```
3 function App() {
4   // let's declare an array of planet objects.
5   let planets = [
6     { id: 1, name: "Venus", gravity: 8.9, dayhrs: 2808},
7     { id: 2, name: "Earth", gravity: 9.8, dayhrs: 24 },
8     { id: 3, name: "Mars", gravity: 3.7, dayhrs: 24.7},
9     { id: 4, name: "Jupiter", gravity: 23.1, dayhrs: 9.9},
10    { id: 5, name: "Saturn", gravity: 9, dayhrs: 10.7},
11    { id: 6, name: "Uranus", gravity: 8.7, dayhrs: 17.2},
12    { id: 7, name: "Neptune", gravity: 11, dayhrs: 16.1 },
13    { id: 8, name: "Pluto", Gravity: 0.7, dayhrs: 153 },
14    { id: 9, name: "Mercury", Gravity: 3.7, dayhrs: 4222.6}
15  ];
16  return (
17    <>
18    <h1>CS385 Solar System</h1>
19    {planets.map((p,index) => (
20      <p key={index}>
21        <b>{p.name}</b>, Gravity = {p.gravity}m/s<sup>2</sup>
22      </p>
23    )})
24    </>
25  );
26 }
```



## CS385 Solar System

Venus, Gravity = 8.9m/s<sup>2</sup>

Earth, Gravity = 9.8m/s<sup>2</sup>

Mars, Gravity = 3.7m/s<sup>2</sup>

Jupiter, Gravity = 23.1m/s<sup>2</sup>

Saturn, Gravity = 9m/s<sup>2</sup>

Uranus, Gravity = 8.7m/s<sup>2</sup>

Neptune, Gravity = 11m/s<sup>2</sup>

Pluto, Gravity = m/s<sup>2</sup>

Mercury, Gravity = m/s<sup>2</sup>

React/Javascript just ignores the Gravity (with capital G) property as it does not exist. Therefore it cannot be printed or rendered

# The map function and arrays

- If you can work to understand the previous example then you're well on your way to grasping one of the most useful concepts in Javascript/React. The importance of the map function and arrays cannot be over emphasised
- The map function will work regardless of the number of elements in the array.
- We, as the programmer, decides how we render the values of each property (in our example we put the name property value in bold text)



# Understanding the structure of a **React** class and **codesandbox**



**The 'src' folder:** This is where ALL of your source code, data, and media files are stored for your 'App'

The screenshot shows a web-based development environment. On the left is a file manager sidebar with icons for adding files and folders. The 'src' folder is highlighted. In the center is a code editor showing the contents of 'App.js'. On the right is a browser window displaying the application output.

**File Manager Icons:**

- File icon: Add a new file.
- Folder icon: Add a new folder.
- Up arrow: Move the selected file/folder up.
- Down arrow: Move the selected file/folder down.
- Trash icon: Delete the selected file/folder.

**Code Editor Content (App.js):**

```
1 import React from "react";
2
3 function App() {
4   // let's declare an array of planet objects.
5   let planets = [
6     { id: 1, name: "Venus", gravity: 8.9, dayhrs: 2808 },
7     { id: 2, name: "Earth", gravity: 9.8, dayhrs: 24 },
8     { id: 3, name: "Mars", gravity: 3.7, dayhrs: 24.7 },
9     { id: 4, name: "Jupiter", gravity: 23.1, dayhrs: 9.9 },
10    { id: 5, name: "Saturn", gravity: 9, dayhrs: 10.7 },
11    { id: 6, name: "Uranus", gravity: 8.7, dayhrs: 17.2 },
12    { id: 7, name: "Neptune", gravity: 11, dayhrs: 16.1 },
13    { id: 8, name: "Pluto", gravity: 0.7, dayhrs: 153 },
14    { id: 9, name: "Mercury", gravity: 3.7, dayhrs: 4222.6 }
15  ];
16
17  return (
18    <div>
19      <p>Gravity</p>
20      <table>
21        <tr>
22          <th>Planet</th>
23          <th>Gravity</th>
24          <th>Day</th>
25        </tr>
26        <tr>
27          <td>Venus</td>
```

**Browser Content:**

### CS385 Solar System

Venus	Gravity = 8.9m/s <sup>2</sup>
Earth	Gravity = 9.8m/s <sup>2</sup>
Mars	Gravity = 3.7m/s <sup>2</sup>
Jupiter	Gravity = 23.1m/s <sup>2</sup>
Saturn	Gravity = 9m/s <sup>2</sup>
Uranus	Gravity = 8.7m/s <sup>2</sup>
Neptune	Gravity = 11m/s <sup>2</sup>
Pluto	Gravity = m/s <sup>2</sup>
Mercury	Gravity = m/s <sup>2</sup>



**Dependencies** – this is where the libraries or external source code your app needs are linked to your ‘app’

The screenshot shows a code editor interface with a dark theme. On the left, a sidebar displays the project structure under 'Files' and a list of 'Dependencies'. The 'Dependencies' list includes 'loader-utils' (3.2.1), 'react' (18.2.0), 'react-dom' (18.2.0), and 'react-scripts' (5.0.1). A red arrow points from the 'react' dependency to the code editor. The code editor shows a file named 'App.js' with the following content:

```
1 import React from "react";
2
3 function App() {
4   // let's declare an array of planet objects.
5   let planets = [
6     { id: 1, name: "Venus", gravity: 8.9, dayhrs: 2808 },
7     { id: 2, name: "Earth", gravity: 9.8, dayhrs: 24 },
8     { id: 3, name: "Mars", gravity: 3.7, dayhrs: 24.7 },
9     { id: 4, name: "Jupiter", gravity: 23.1, dayhrs: 9.9 },
10    { id: 5, name: "Saturn", gravity: 9, dayhrs: 10.7 },
11    { id: 6, name: "Uranus", gravity: 8.7, dayhrs: 17.2 },
12    { id: 7, name: "Neptune", gravity: 11, dayhrs: 16.1 },
13    { id: 8, name: "Pluto", gravity: 0.7, dayhrs: 153.3 },
14    { id: 9, name: "Mercury", gravity: 3.7, dayhrs: 88 },
15  ];
16  return (
17    <div>
18      <h1>CS385 Solar System</h1>
19      {planets.map((p, index) => (
20        <p key={index}>
21          <b>{p.name}</b>, Gravity = {p.gravity}m/s2, Day = {p.dayhrs}hrs
22        </p>
23      ))}
24    </div>
25  );
26 }
27 export default App;
```

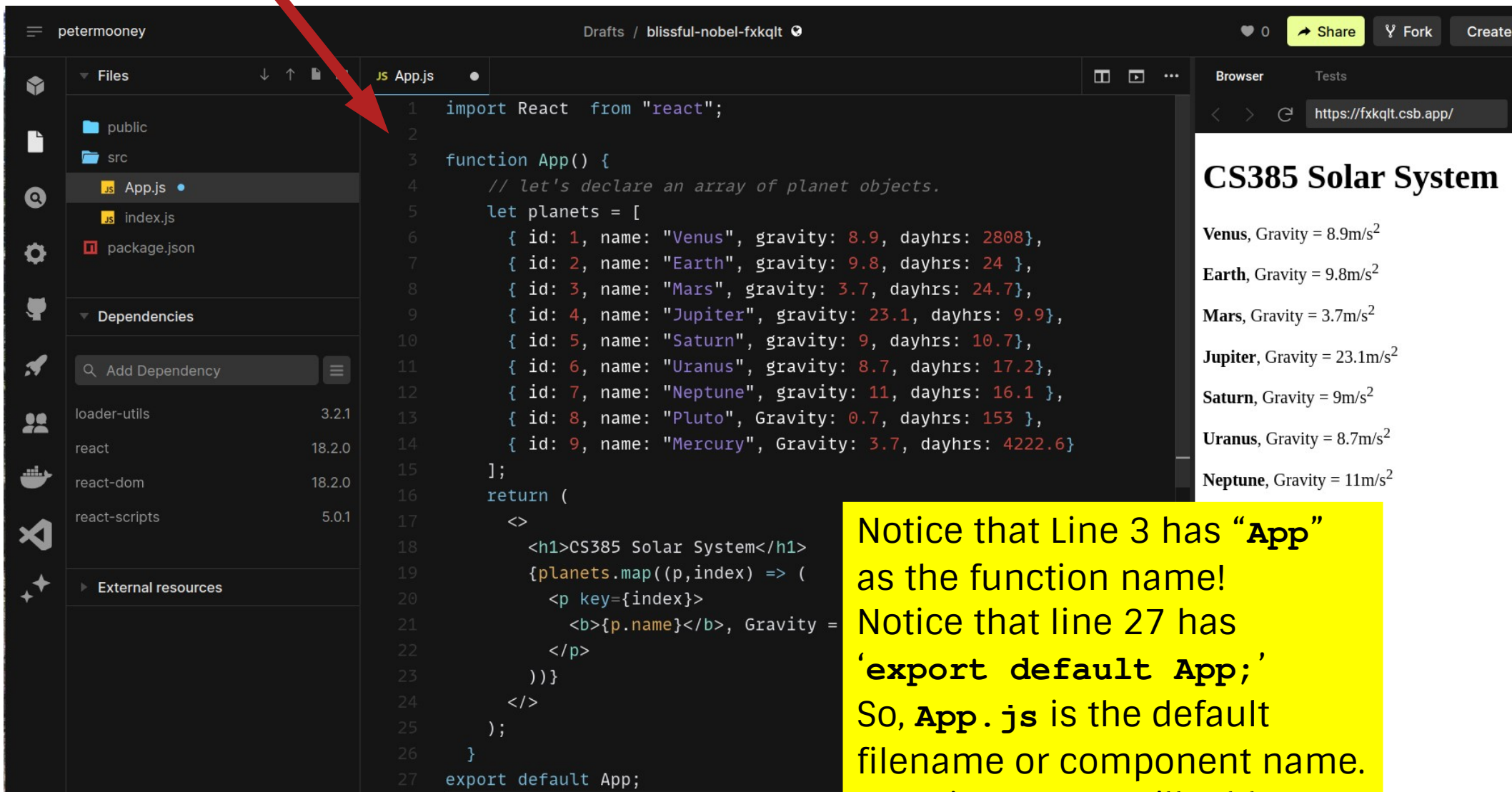
On the right, a browser preview shows the rendered application at <https://fxkqlt.csb.app/>. The browser title is 'CS385 Solar System'. The content lists the planets and their gravity values:

- Venus, Gravity = 8.9m/s<sup>2</sup>
- Earth, Gravity = 9.8m/s<sup>2</sup>
- Mars, Gravity = 3.7m/s<sup>2</sup>
- Jupiter, Gravity = 23.1m/s<sup>2</sup>
- Saturn, Gravity = 9m/s<sup>2</sup>

A yellow text box with a red arrow pointing to the 'react' dependency in the sidebar contains the following text:

Later in CS385 we will add our own chosen 'dependencies'. React (version 18.2.0) is listed here – this is saying that we want to import React.js and use it to write our app

**“App.js”** – this is the filename of the JS (JavaScript) file containing your app code



The screenshot shows a code editor interface with a dark theme. On the left, a sidebar displays the file structure with folders 'public' and 'src', and files 'App.js', 'index.js', and 'package.json'. Below this, the 'Dependencies' section lists 'loader-utils' (3.2.1), 'react' (18.2.0), 'react-dom' (18.2.0), and 'react-scripts' (5.0.1). The main editor area shows the content of 'App.js', which starts with 'import React from "react";' and defines a function 'App()' that returns a JSX element. The JSX element includes a heading 'CS385 Solar System' and a list of planets with their names and gravity values. The file ends with 'export default App;'. A red arrow points from the text '“App.js”' in the top heading to the 'App.js' file in the sidebar. On the right, a browser window shows the rendered output of the app, titled 'CS385 Solar System', which lists the planets and their gravity values.

```
1 import React from "react";
2
3 function App() {
4   // let's declare an array of planet objects.
5   let planets = [
6     { id: 1, name: "Venus", gravity: 8.9, dayhrs: 2808 },
7     { id: 2, name: "Earth", gravity: 9.8, dayhrs: 24 },
8     { id: 3, name: "Mars", gravity: 3.7, dayhrs: 24.7 },
9     { id: 4, name: "Jupiter", gravity: 23.1, dayhrs: 9.9 },
10    { id: 5, name: "Saturn", gravity: 9, dayhrs: 10.7 },
11    { id: 6, name: "Uranus", gravity: 8.7, dayhrs: 17.2 },
12    { id: 7, name: "Neptune", gravity: 11, dayhrs: 16.1 },
13    { id: 8, name: "Pluto", gravity: 0.7, dayhrs: 153 },
14    { id: 9, name: "Mercury", gravity: 3.7, dayhrs: 4222.6 }
15  ];
16  return (
17    <>
18      <h1>CS385 Solar System</h1>
19      {planets.map((p, index) => (
20        <p key={index}>
21          <b>{p.name}</b>, Gravity =
22        </p>
23      ))}
24    </>
25  );
26 }
27 export default App;
```

loader-utils 3.2.1  
react 18.2.0  
react-dom 18.2.0  
react-scripts 5.0.1

CS385 Solar System

Venus, Gravity = 8.9m/s<sup>2</sup>  
Earth, Gravity = 9.8m/s<sup>2</sup>  
Mars, Gravity = 3.7m/s<sup>2</sup>  
Jupiter, Gravity = 23.1m/s<sup>2</sup>  
Saturn, Gravity = 9m/s<sup>2</sup>  
Uranus, Gravity = 8.7m/s<sup>2</sup>  
Neptune, Gravity = 11m/s<sup>2</sup>

Notice that Line 3 has “**App**” as the function name!  
Notice that line 27 has ‘**export default App;**’  
So, **App.js** is the default filename or component name.  
Later in CS385 we’ll add other **.js** files here.



**function App:** This is a function or component called 'App'.

The screenshot shows a code editor with a file explorer on the left, a code editor in the center, and a browser window on the right. The file explorer shows a project named 'blissful-nobel-fxkqlt' with files 'App.js', 'index.js', and 'package.json'. The code editor shows the following code:

```
1 import React from "react";
2
3 function App() {
4   // let's declare an array of planet objects.
5   let planets = [
6     { id: 1, name: "Venus", gravity: 8.9, dayhrs: 2808},
7     { id: 2, name: "Earth", gravity: 9.8, dayhrs: 24 },
8     { id: 3, name: "Mars", gravity: 3.7, dayhrs: 24.7},
9     { id: 4, name: "Jupiter", gravity: 23.1, dayhrs: 9.9},
10    { id: 5, name: "Saturn", gravity: 9, dayhrs: 10.7},
11    { id: 6, name: "Uranus", gravity: 8.9, dayhrs: 9.9},
12    { id: 7, name: "Neptune", gravity: 14.7, dayhrs: 16.1},
13    { id: 8, name: "Pluto", gravity: 0.6, dayhrs: 2.3},
14    { id: 9, name: "Mercury", gravity: 3.7, dayhrs: 1.6},
15  ];
16  return (
17    <>
18      <h1>CS385 Solar System</h1>
19      {planets.map((p, index) => (
20        <p key={index}>
21          <b>{p.name}</b>, Gravity = {p.gravity}m/s2
22        </p>
23      )}
24    </>
25  );
26 }
27 export default App;
```

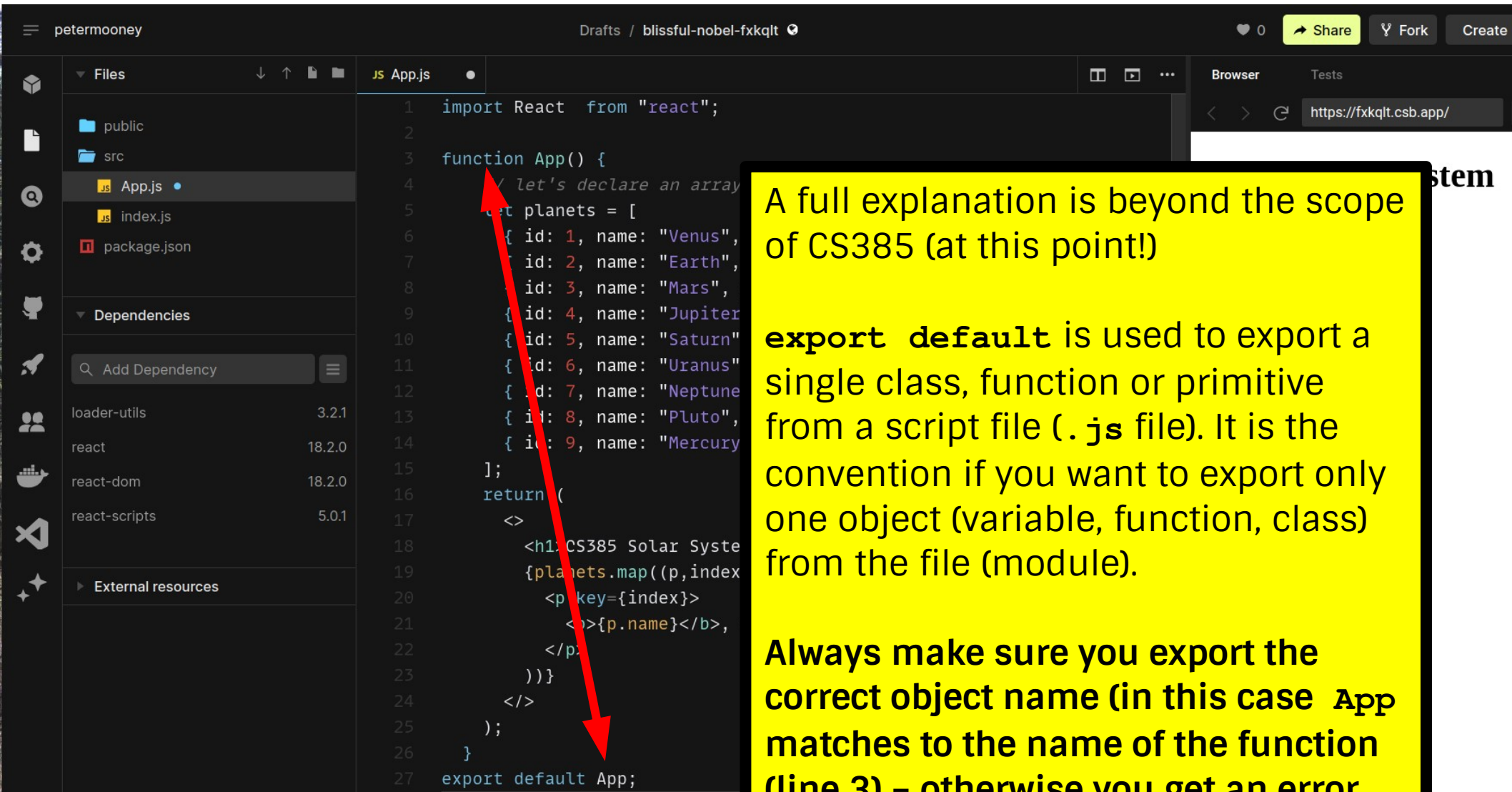
The browser window shows the rendered output of the App component, displaying the title "CS385 Solar System" and a list of planets with their gravity values:

- Venus, Gravity = 8.9m/s<sup>2</sup>
- Earth, Gravity = 9.8m/s<sup>2</sup>
- Mars, Gravity = 3.7m/s<sup>2</sup>
- Jupiter, Gravity = 23.1m/s<sup>2</sup>

The App function takes no parameters (line 3).

It has a **return** statement (line 16) where the method returns JSX – essentially a mixture of JavaScript and HTML. This is RENDERED or displayed on the RHS as HTML output

**export default App;** Line 27 –  
essentially “exports” the App function  
for usage elsewhere



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'public' folder, a 'src' folder, and files 'App.js', 'index.js', and 'package.json'. The 'App.js' file is selected. The code editor shows the following code:

```
1 import React from "react";
2
3 function App() {
4   // let's declare an array
5   let planets = [
6     { id: 1, name: "Venus",
7       id: 2, name: "Earth",
8       id: 3, name: "Mars",
9       id: 4, name: "Jupiter",
10      id: 5, name: "Saturn",
11      id: 6, name: "Uranus",
12      id: 7, name: "Neptune",
13      id: 8, name: "Pluto",
14      id: 9, name: "Mercury"
15   ];
16   return (
17     <>
18       <h1>CS385 Solar System
19       {planets.map((p, index) => (
20         <p key={index}>
21           <b>{p.name}</b>,
22         </p>
23       ))}
24     </>
25   );
26 }
27 export default App;
```

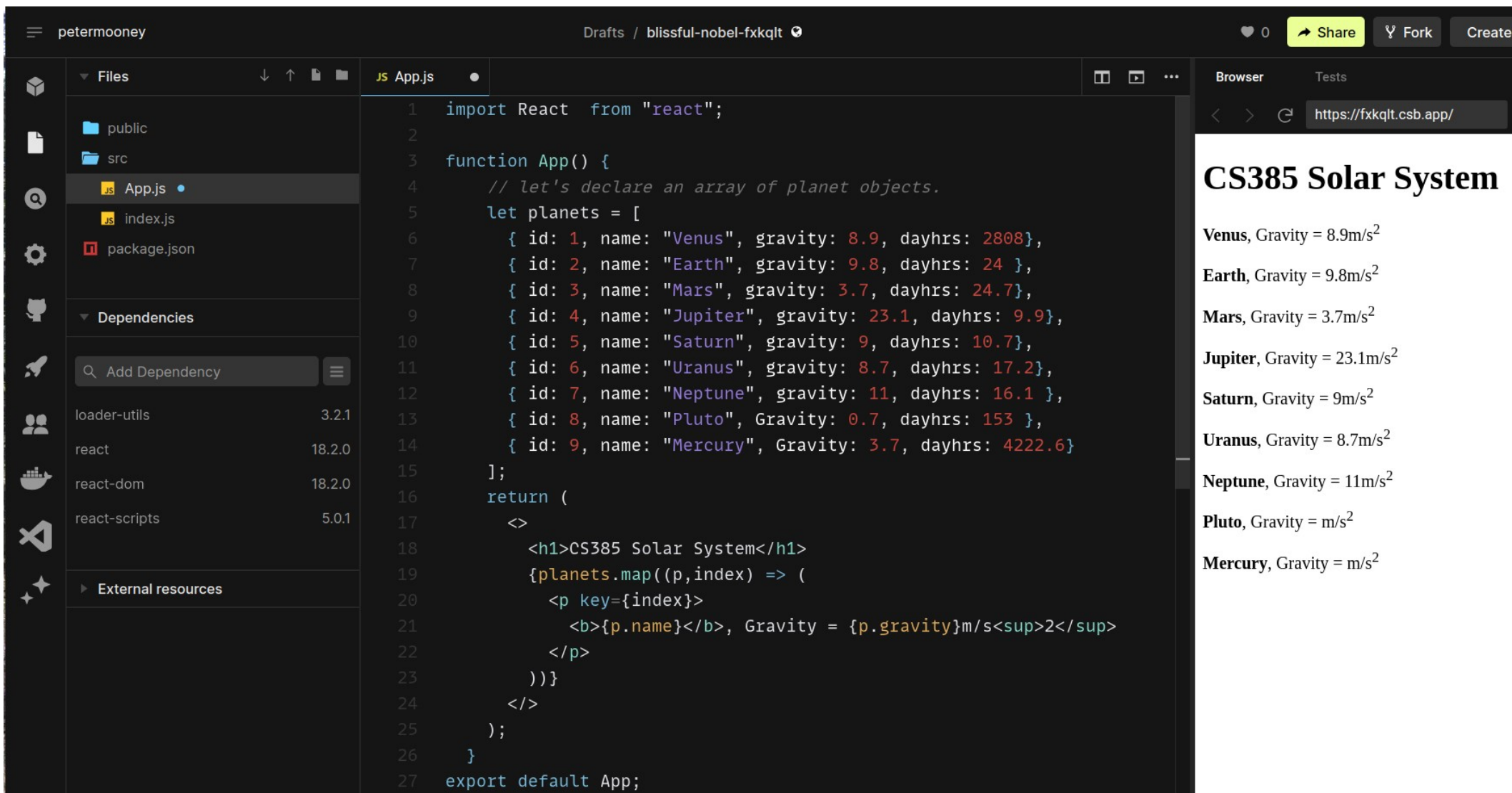
A red arrow points from the 'App()' function definition on line 3 to the 'export default App;' statement on line 27.

A full explanation is beyond the scope  
of CS385 (at this point!)

**export default** is used to export a  
single class, function or primitive  
from a script file (.js file). It is the  
convention if you want to export only  
one object (variable, function, class)  
from the file (module).

Always make sure you export the  
correct object name (in this case **App**  
matches to the name of the function  
(line 3) – otherwise you get an error

# What does the `<p key={index}>` actually mean in the map function?



The screenshot shows a code editor with a React application. The main code file is `App.js`, which imports `React` from `"react"` and defines a function `App()`. Inside `App()`, an array of planet objects is declared: `let planets = [ { id: 1, name: "Venus", gravity: 8.9, dayhrs: 2808 }, { id: 2, name: "Earth", gravity: 9.8, dayhrs: 24 }, { id: 3, name: "Mars", gravity: 3.7, dayhrs: 24.7 }, { id: 4, name: "Jupiter", gravity: 23.1, dayhrs: 9.9 }, { id: 5, name: "Saturn", gravity: 9, dayhrs: 10.7 }, { id: 6, name: "Uranus", gravity: 8.7, dayhrs: 17.2 }, { id: 7, name: "Neptune", gravity: 11, dayhrs: 16.1 }, { id: 8, name: "Pluto", Gravity: 0.7, dayhrs: 153 }, { id: 9, name: "Mercury", Gravity: 3.7, dayhrs: 4222.6 } ]`. The `return` statement uses `ReactDOM.render` to render the `planets` array using `planets.map((p, index) => ( <p key={index}> <b>{p.name}</b>, Gravity = {p.gravity}m/s<sup>2</sup> </p> ))`. The `key` prop is used to uniquely identify each element in the list. The right sidebar shows the rendered output, titled "CS385 Solar System", which lists the planets and their gravity values: Venus, Gravity = 8.9m/s<sup>2</sup>; Earth, Gravity = 9.8m/s<sup>2</sup>; Mars, Gravity = 3.7m/s<sup>2</sup>; Jupiter, Gravity = 23.1m/s<sup>2</sup>; Saturn, Gravity = 9m/s<sup>2</sup>; Uranus, Gravity = 8.7m/s<sup>2</sup>; Neptune, Gravity = 11m/s<sup>2</sup>; Pluto, Gravity = m/s<sup>2</sup>; Mercury, Gravity = m/s<sup>2</sup>.

```
1 import React from "react";
2
3 function App() {
4   // let's declare an array of planet objects.
5   let planets = [
6     { id: 1, name: "Venus", gravity: 8.9, dayhrs: 2808 },
7     { id: 2, name: "Earth", gravity: 9.8, dayhrs: 24 },
8     { id: 3, name: "Mars", gravity: 3.7, dayhrs: 24.7 },
9     { id: 4, name: "Jupiter", gravity: 23.1, dayhrs: 9.9 },
10    { id: 5, name: "Saturn", gravity: 9, dayhrs: 10.7 },
11    { id: 6, name: "Uranus", gravity: 8.7, dayhrs: 17.2 },
12    { id: 7, name: "Neptune", gravity: 11, dayhrs: 16.1 },
13    { id: 8, name: "Pluto", Gravity: 0.7, dayhrs: 153 },
14    { id: 9, name: "Mercury", Gravity: 3.7, dayhrs: 4222.6 }
15  ];
16  return (
17    <>
18      <h1>CS385 Solar System</h1>
19      {planets.map((p, index) => (
20        <p key={index}>
21          <b>{p.name}</b>, Gravity = {p.gravity}m/s<sup>2</sup>
22        </p>
23      ))}
24    </>
25  );
26 }
27 export default App;
```

**CS385 Solar System**

Venus, Gravity = 8.9m/s<sup>2</sup>

Earth, Gravity = 9.8m/s<sup>2</sup>

Mars, Gravity = 3.7m/s<sup>2</sup>

Jupiter, Gravity = 23.1m/s<sup>2</sup>

Saturn, Gravity = 9m/s<sup>2</sup>

Uranus, Gravity = 8.7m/s<sup>2</sup>

Neptune, Gravity = 11m/s<sup>2</sup>

Pluto, Gravity = m/s<sup>2</sup>

Mercury, Gravity = m/s<sup>2</sup>

# What does the `<p key={index}>` actually mean in the map function?

The screenshot shows a code editor with a file explorer on the left, a code editor in the center, and a browser preview on the right. The code editor shows a file named `App.js` with the following code:

```
1 import React from "react";
2
3 function App() {
4   // let's declare an array of planet objects.
5   let planets = [
6     { id: 1, name: "Venus", gravity: 8.9, dayhrs: 2808 },
7     { id: 2, name: "Earth", gravity: 9.8, dayhrs: 24 },
8     { id: 3, name: "Mars", gravity: 3.7, dayhrs: 24.7 },
9     { id: 4, name: "Jupiter", gravity: 23.1, dayhrs: 9.9 },
10    { id: 5, name: "Saturn", gravity: 9, dayhrs: 10.7 },
11    { id: 6, name: "Uranus", gravity: 8.7, dayhrs: 17.2 },
12    { id: 7, name: "Neptune", gravity: 11, dayhrs: 16.1 },
13    { id: 8, name: "Pluto", gravity: 0.7, dayhrs: 153 },
14    { id: 9, name: "Mercury", gravity: 3.7, dayhrs: 4222.6 }
15  ];
16  return (
17    <>
18      <h1>CS385 Solar System</h1>
19      {planets.map((p, index) => (
20        <p key={index}>
21          <b>{p.name}</b>, Gravity = {p.gravity}m/s<sup>2</sup>
22        </p>
23      ))}
24    </>
25  );
26 }
27 export default App;
```

The browser preview shows the rendered output of the application, titled "CS385 Solar System". It displays a list of planets with their names in bold and their gravity values. Two yellow arrows point from the `<p key={index}>` code in the code editor to the rendered output, highlighting the mapping of the `index` prop to the `key` prop in the rendered HTML.

Planet	Gravity
Venus	8.9m/s <sup>2</sup>
Earth	9.8m/s <sup>2</sup>
Mars	3.7m/s <sup>2</sup>
Jupiter	23.1m/s <sup>2</sup>
Saturn	9m/s <sup>2</sup>
Uranus	8.7m/s <sup>2</sup>
Neptune	11m/s <sup>2</sup>
Pluto	m/s <sup>2</sup>
Mercury	m/s <sup>2</sup>

# In the rendered app – think of each <p> (paragraph) as a single block or element

CS385 Solar System	
	Index =
Venus, Gravity = $8.9\text{m/s}^2$	0
Earth, Gravity = $9.8\text{m/s}^2$	1
Mars, Gravity = $3.7\text{m/s}^2$	2
Jupiter, Gravity = $23.1\text{m/s}^2$	3
Saturn, Gravity = $9\text{m/s}^2$	4
Uranus, Gravity = $8.7\text{m/s}^2$	5
Neptune, Gravity = $11\text{m/s}^2$	6
Pluto, Gravity = $0.7\text{m/s}^2$	7
Mercury, Gravity = $3.7\text{m/s}^2$	8

- For the browser to be able to control the movement of blocks or element (such as a user clicking to delete one) – **the DOM (Document Object Model)** must keep track of the blocks or elements.
- Hence **the need for a UNIQUE IDENTIFIER for each block.**



# The MAP FUNCTION provides use with an index variable – and this just provides a count from 0

```
3 function App() {
4   // let's declare an array of planet objects.
5   let planets = [
6     { id: 1, name: "Venus", gravity: 8.9, dayhrs: 2808 },
7     { id: 2, name: "Earth", gravity: 9.8, dayhrs: 24 },
8     { id: 3, name: "Mars", gravity: 3.7, dayhrs: 24.7 },
9     { id: 4, name: "Jupiter", gravity: 23.1, dayhrs: 9.9 },
10    { id: 5, name: "Saturn", gravity: 9, dayhrs: 10.7 },
11    { id: 6, name: "Uranus", gravity: 8.7, dayhrs: 17.2 },
12    { id: 7, name: "Neptune", gravity: 11, dayhrs: 16.1 },
13    { id: 8, name: "Pluto", gravity: 0.7, dayhrs: 153 },
14    { id: 9, name: "Mercury", gravity: 3.7, dayhrs: 4222.6 }
15  ];
16  return (
17    <>
18    <h1>CS385 Solar System</h1>
19    {planets.map((p,index) => (
20      <p key={index}>
21        <b>{p.name}</b>, Gravity = {p.gravity}m/s<sup>2</sup>
22        [Block Index = {index}]
23      </p>
24    )]}
25  </>
```

## CS385 Solar System

Venus, Gravity = 8.9m/s<sup>2</sup>[Block Index = 0]

Earth, Gravity = 9.8m/s<sup>2</sup>[Block Index = 1]

Mars, Gravity = 3.7m/s<sup>2</sup>[Block Index = 2]

Jupiter, Gravity = 23.1m/s<sup>2</sup>[Block Index = 3]

Saturn, Gravity = 9m/s<sup>2</sup>[Block Index = 4]

Uranus, Gravity = 8.7m/s<sup>2</sup>[Block Index = 5]

Neptune, Gravity = 11m/s<sup>2</sup>[Block Index = 6]

Pluto, Gravity = 0.7m/s<sup>2</sup>[Block Index = 7]

Mercury, Gravity = 3.7m/s<sup>2</sup>[Block Index = 8]

Line 22 – actually renders the value of index from the map function. Think of it as just a counter variable provided to you by REACT

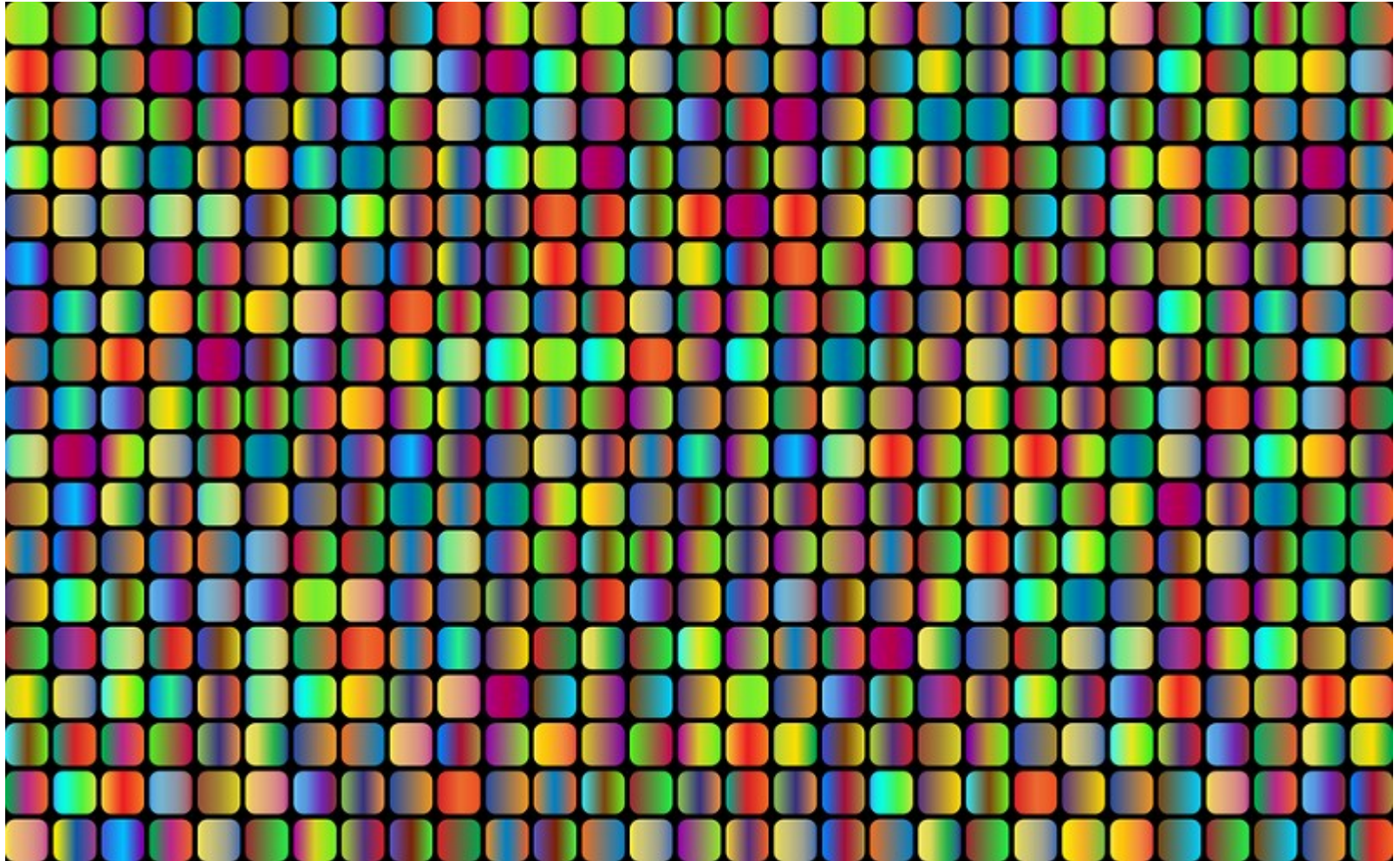


# Recap



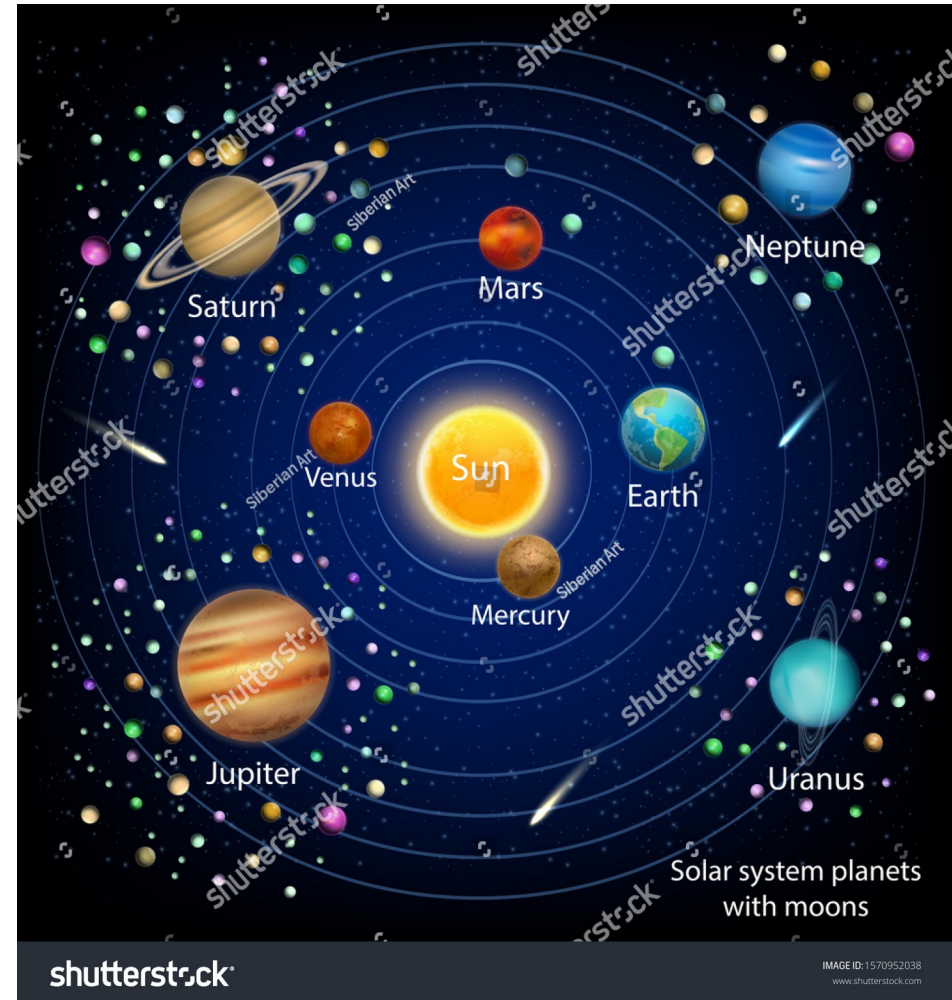
- **Property names MUST stay the same** (it just makes your coding life easier)
- **The map function provides you with its own internal “index” or counter value for BLOCKs**
- **The map function can render or print any valid array of JSON objects.**
- **Try out the Planets code** (available on Moodle)

# Working with large arrays of Objects in a mobile application



# Large arrays of objects in Javascript – how do we deal with these situations?

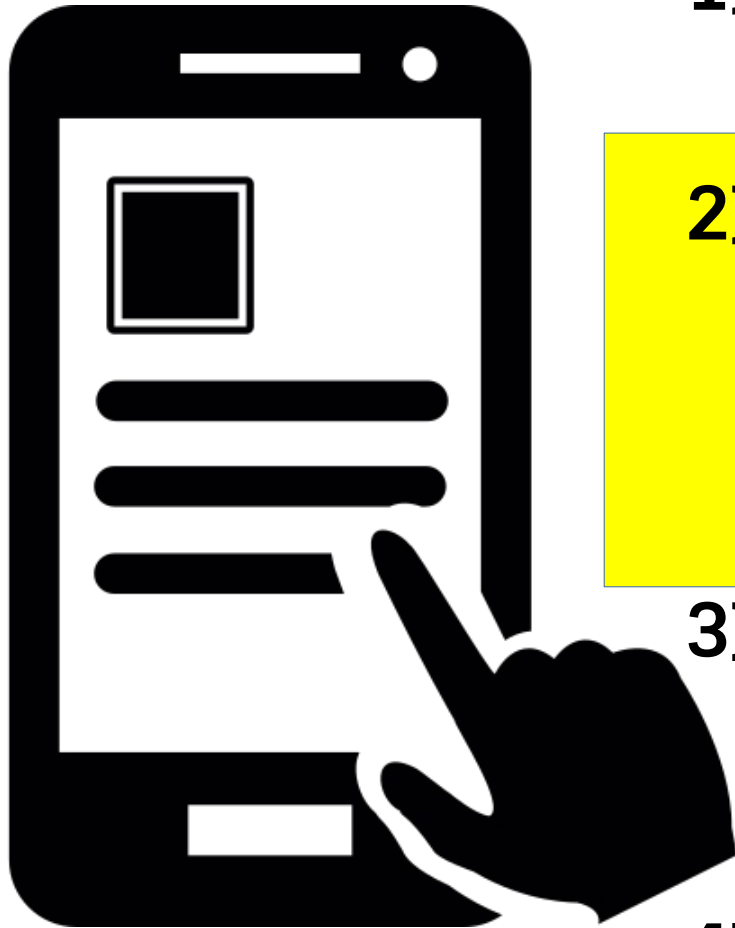
- In the planets example we had 9 objects. This isn't a big array but it takes up a lot of lines of code in our **App.js** file.
- What if we wanted to create an array of objects representing all of the moons or satellites of planets in our solar system?
- This would be over 200 objects – that is a big array and would take 200 lines of code!



# **We need to be able to process large arrays in our mobile applications**

- **In our usage of applications on our mobile devices we interact with arrays of objects ALL THE TIME. For example:**
  - **Scrolling through your contacts**
  - **Scrolling message threads in social apps** such as WhatsApp (each thread is an array of object where each message is an object)
  - **News aggregator feeds** – a constantly updating array of news objects which you can click on ...
  - **List of items in an online shop....**

# There are **FOUR** ways (with variations) on how you import data into your mobile app



1) Use an array of objects within your `App.js` class (or similar) ✓

2) Use a local file (within your app source) to hold or store an array of objects which is accessible to `App.js`

3) Use an external API (Application Programming Interface) – later in CS385

4) Use an external database – later in CS385

# End – Lecture 3