Simply - Learn React JS in 2 hours

Free Reference

Unit 1 - The Basics
Unit 5 - Deploying The Application Online

www.ramy.pro

Get the full course: https://ramy.pro/react-js-course/

1.1 Start a New Project

- Download nodejs from the official site https://nodejs.org/
- Install node js
- With the terminal on the "macOS" or the PowerShell on "windows" go into the location you want to build the "React JS" application files

CD my_location

To build the "React JS" application, enter the following command.
npx create-react-app my_project_name

now go inside the project folder by entering this command

cd my_project_name

to run the server just enter command

npm start

If the page doesn't open automatically, use this link address "http://localhost:3000" to open the application home page on your browser.

1.2 The Project Folders and Files

After creating the "React JS" you will find a three folders (node_modules, public, and src).- public folder

this folder contains the public files including the main html file “index.html” that is associated with the domain “localhost:3000”.

React will render the results to this file.

- src folder

src short of (source), this folder contains the basic js files that react needs.

Index.js

The main file that associates with index.html, so if you going to do any changes to this file the changes will be applied to index.html.

1.3 The Default “React JS” App

After you create your new “React JS” project you will see the home page of the “React JS” default application. Let's dive into this application to figure out how is "ReactJS" works.

Let's go inside the folder “src”, with any of the code editors you like. I’m going to check the "index.js" and the "App.js" files.

index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import reportWebVitals from './reportWebVitals';
import './index.css';
import App from './App';

const root=ReactDOM.createRoot(document.getElementById('root'));

root.render(
  <React.StrictMode>
    <App />

  </React.StrictMode>
);
reportWebVitals( “report.log” );

import App from './App';

const root=ReactDOM.createRoot(document.getElementById('root'));

root.render(
  <React.StrictMode>
    <App />

  </React.StrictMode>
);
reportWebVitals( “report.log” );
```

As we see here, index.js imported some of the modules from files located on the server.

The first two Modules related to React and must be imported in the index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
```

The third one

```
import reportWebVitals from './reportWebVitals';
```

This Module is optional, you can use it to create a report about your application, you can choose the file that you want to save the report in, through the function “reportWebVitals();”

```
reportWebVitals( “report.log” );
```

Now let's focus on this module.

```
import App from './App';
```

if you have a basics knowledge of “ECMAScript” (the new version of Javascript) you will be aware that the name of the module comes after the keyword “import” (App), and the path that comes after the "from" keyword is the path of the JavaScript file that contains the module, so let's check the “./App.js”, located in the “src” folder.

App.js

Fortunately, this file contains just a few lines, but telling you:

- how to import external files such as images or style sheets (logo, CSS).
- the module “App” that is imported in the “index.js” file, is just a regular javascript function.
- you must export this function as you see in the last line.

But there is something weird in the return area! It’s HTML tags without single or double quotations, what’s going on? "ReactJS" use “JSX” or (JavaScript XML), JSX allows you to write HTML tags in Javascript.

Render the content into index.js.

Let's go back to the “index.js” file. If you going to draw your first sketch you have to get the empty white page first! So, to render the content you need to create the "root" element as a white page of your “React JS” application. Let's focus on this line.

```
const root=ReactDOM.createRoot(document.getElementById('root'));
```

ReactDOM is the built-in object that includes the functions that can help you to do changes to the HTML elements. "createRoot()" function is responsible to create the root div that will display the final result of the rendering.

So in this line we just save the object into the "root" constant, to start rendering the content we going to use the “render” function.

```
root.render(
  <React.StrictMode>
    <App />

  </React.StrictMode>
);
```

"Render()" function accepts the content as an argument, in this example, we rendered the return that comes from “App” function located in the “App.js” file, so if you want to do any changes, it must be inside the “return” area of the “App” function.

Render your custom content.

If you desire to render custom text directly through the render argument, you can do that by adding the text into an empty HTML tag.

```
 root.render(
    <> Hi my name is ramy </>
);
```

1.4 The Components

The question is, what the “React JS” Component is?

The simple answer is the functions that we have used in the previous lessons.

So when you hear “Component” in the "ReactJS" world, it means the functions you going to create. In this lesson, I am going to create some of the functions in the “index.js” file, and render these functions as components, so, yes you can put the functions or the components in the same file without any problems.

First, make sure the “Root” was created.

```
const root =
ReactDOM.createRoot(document.getElementById('root'));
```

Let's create the first simple component that will display the welcome message to the visitor.

```
function Welcome(){

          return(
          <h4>Welcome to my first React App</h4>
          );
}
```

Now, in the render argument let's pass this component.

```
root.render(
          <Welcome />
);
```

TADA! You can see the welcome message on the home page! So, you can use the HTML tags in lowercase, don’t try to write the HTML tags like this:

<H4>Welcome</H4> or <DIV></DIV>, this will cause an error.

Next: let's create a new component that will display the name depending on the argument you going to pass!

```
function MyName(arg_obj){

          return (
          <div>My name is {arg_obj.name}.</div>
```

```
        );
}
```

Don’t be confused! The secret is, if you want to pass any of the arguments into the component, you must include those arguments in an object as properties. So, in this example, I have created an object called “arg_obj” (You can name it whatever you like), and then, used the “name” argument as a reference of the name that I will pass later.

Ok, ok, don’t be mad, let's render this component, and everything will be easy.

```
root.render(
        <MyName name="ramy" />
);
```

In the "root" element, you will see the result of the render "My name is ramy".

At this point, we learned that in the "React JS" world, we can use the Javascript functions as HTML tags, and we can pass the arguments as attributes.

Let's create another component to display my age dynamically depending on the current year.

```
function MyAge(arg_obj){
        let get_date = new Date();

        return (
          <>I am {get_date.getFullYear() - arg_obj.birth_year }
years old.</>
        );
}
```

Did you notice this line?

```
I am {get_date.getFullYear() - arg_obj.birth_year } old.
```

If you want to merge any of (Variables, constants, or math operations) with the output text, you do that by putting them in between curly brackets {}, this is called "Expressions" in JSX.

So, by using the "get_date.getFullYear()" function we will get the current year, but what about arg_obj.birth_year? Can you answer this question?

Yes, it's just the object holding the argument "birth_year" that we going to use as an attribute to pass the year of the birth.

```
root.render(
          <MyAge birth_year = "1987" />
);
```

The current year is 2022, and I have passed 1987 through the "birth_year" argument.

The result will be 35, let's see the final output.

I am 35 years old.

Next, I want to create a component to display my contacts, but I want to pass them as a Javascript object. Let's say, I have this js object.

```
{phone:"1234567",email:"ramy@anysite.com"}
```

So, how I can pass it through the "ReactJS" component?

First, let's create “MyContacts” component.

```
function MyContacts(arg_obj){

          return(
             <>
                <p>Phone: {arg_obj.info.phone}</p>
                <p>email: {arg_obj.info.email}</p>
             </>
          );
}
```

Inside the return, I have used the "info" argument as a js object, so let's see how we can pass the object's properties into the component "MyContacts".

```
<MyContacts info={{phone:"1234567",email:"ramy@anysite.com"}} />
```

Instead of just putting one value, I have put the JS object with the properties.

So after the render, here is the result:

Phone: 1234567

email: ramy@anysite.com

Finally, here is the final code of this lesson, keep it for the next lesson.

```
function Welcome(){

          return(
          <h4>Welcome to my first React App</h4>
          );
}

function MyName(arg_obj){

          return (
          <div>My name is {arg_obj.name}.</div>

          );
}

function MyAge(arg_obj){
          let get_date = new Date();

          return (
            <>I am {get_date.getFullYear() - arg_obj.birth_year }
old.</>
          );
}

function MyContacts(arg_obj){

          return(
             <>
                    <p>Phone: {arg_obj.info.phone}</p>
                    <p>email: {arg_obj.info.email}</p>
                    </>
          );
}
const root =
ReactDOM.createRoot(document.getElementById('root'));
root.render(
 <React.StrictMode>
          <Welcome />
          <MyName name="ramy" />
          <MyAge birth_year = "1987" />
          <MyContacts
info={{phone:"1234567",email:"ramy@anysite.com"}} />

  </React.StrictMode>
);
```

1.5 Components in The Files

In the last lesson, we created some of the components in "index.js", of course, this is not the best way to build our App. What if I going to need these components later for a different page or file? Importing "index.js" can cause a lot of issues.

So, in this lesson, we just will move the functions of the previous lesson from the "index.js" file to a new file I'll name this file "about.js", so we can import this file to any part of our App without problems.

First, I'll create the "about.js" file in "src" folder. Next, I'll move the functions from "index.js" to "about.js", with adding the "export" keyword before every function.

```
export function Welcome(){

     return(
     <h4>Welcome to my first React App</h4>
     );
}

export function MyName(arg_obj){

     return (
     <div>My name is {arg_obj.name}.</div>

     );
}

export function MyAge(arg_obj){
     let get_date = new Date();

     return (
       <>I am {get_date.getFullYear() - arg_obj.birth_year }
old.</>
     );
}

export function MyContacts(arg_obj){

     return(
        <>
          <p>Phone: {arg_obj.info.phone}</p>
          <p>email: {arg_obj.info.email}</p>
          </>
```

```
      );
}
```

Now, back to index.js, It must import "about.js" into "index.js" but also we have to choose the components we need to import, in this example, I'm going to import the whole components I created.

```
import {Welcome,MyName,MyAge,MyContacts} from './about';
```

Did notice any difference?!

Yes, I have put the components in between curly brackets and separated them with a comma.

So, you can export multiple components from one file, and import them using this way, but if you want to export the component as default, you must export just one default component.

So, let's create a new component in “about.js” I'll name it "Head".

```
export default function Head(){

     return(
  <div>
   <Welcome />
   <a href="https://ramy.pro" target="_blank" >www.ramy.pro</a>
   <hr />
  </div>
          );
}
```

In the "Head" component I have rendered the "Welcome" component, which is, you can render a component inside another component. Also, I added a link to my website and an "hr" tag to display a separator line.

The export of this component is a default, so I can import this component into the "index.js" file by inserting the component name out of the curly brackets group. So in the "index.js" file, I'll edit the import line.

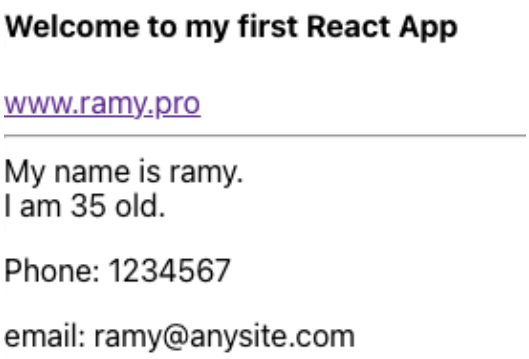
```
import Head,{Welcome,MyName,MyAge,MyContacts} from './about';
```

Perfect! Now, let's put the components into the render function as we want.

```
root.render(

     <>
     <Head />
     <MyName name="ramy" />
```

```
      <MyAge birth_year = "1987" />
      <MyContacts
info={{phone:"1234567",email:"ramy@anysite.com"}} />
      </>

);
```

So, the outcome will be like this.



1.6 React and CSS

It's time to learn how to use CSS style with "React JS". In "React JS" still, you have three ways to use CSS, Inline styling, CSS Modules, and CSS stylesheets.

Let's add some CSS to the "about.js" components.

do you remember this file?! If you don't, just review the last lesson before continuing this one.

Let's add "Inline style" to the main "div" in the "Head" component in the "about.js" file.

```
export default function Head(){

     return(
     <div
style={{backgroundColor:"#ddd",width:"600px",margin:"auto",paddi
ng:"10px" }}>
    <Welcome  />
    <a href="https://ramy.pro" target="_blank" >www.ramy.pro</a>
          <hr />
     </div>
          );
}
```

Focus on this line.

```
<div
style={{backgroundColor:"#ddd",width:"600px",margin:"auto",paddi
ng:"10px" }}>
```

To add CSS style inline, you must pass the properties as a Javascript object.

```
{backgroundColor:"#ddd",width:"600px",margin:"auto",padding:"10p
x" }
```

Also, make sure you write the CSS properties as JS object's properties with camel case syntax. For example, in CSS files we write "background-color" with the dash, but in Javascript "backgroundColor", here is the outcome.

Next, the second way, instead of writing the "CSS" Properties as a value of the "style" attribute directly, we can create an object and use it instead.

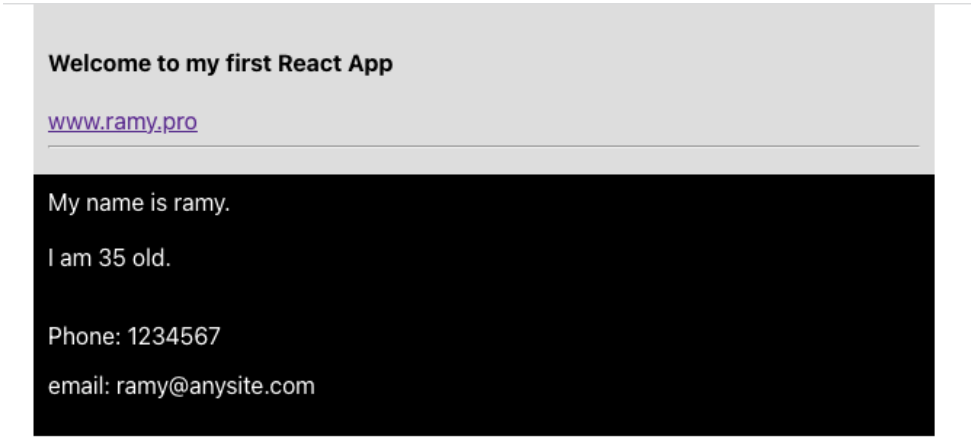
In the file "about.js" I'll create a new Javascript object "Content_style", this object will contain the "CSS" Properties.

```
const Content_style={backgroundColor:"#000",color:"#fff",
width:"600px",margin:"auto",padding:"10px"}
```

Now, I can use this object as a "CSS" style multiple times with the components' HTML elements.

```
export function MyName(arg_obj){

     return (
     <div style={Content_style}>My name is {arg_obj.name}.</div>

     );
}

export function MyAge(arg_obj){
     let get_date = new Date();

     return (
       <div style={Content_style}>I am {get_date.getFullYear() -
arg_obj.birth_year } old.</div>
     );
}

export function MyContacts(arg_obj){

     return(
        <div style={Content_style}>
          <p>Phone: {arg_obj.info.phone}</p>
          <p>email: {arg_obj.info.email}</p>
        </div>
     );
}
```

Here is the outcome!


Welcome to my first React App
www.ramy.pro
My name is ramy.
I am 35 old.
Phone: 1234567
email: ramy@anysite.com

1.7 CSS in External Stylesheets

Still, you can write the "CSS" into external files and import them. You have 2 ways to do that. The first is importing "The CSS" file as a regular style sheet, and the second is importing "The CSS" file as a module. Let's focus first on the first way.

Let's create a new CSS file called "my_style.css" in the "src" folder.

Now I'm going to add the "CSS" Properties to this file in a normal way to apply changes to (the body, paragraph elements, links elements, and create a class called footer_area).

my_style.css

```
body{ background-color: #FDFFDD; }

p{ font-size: 17px; font-weight: bold; }

a{text-decoration: none; color: #2B7CFF; font-weight: bold;}
a:hover{text-decoration: underline;}

.footer_area{background-color:#fff; width:600px; margin:auto;
padding:10px; }
```

Next, let's import this file into "index.js", you can also import it into any of the other files, but I chose the "index.js" file, to target those tags in general.

```
import './my_style.css';
```

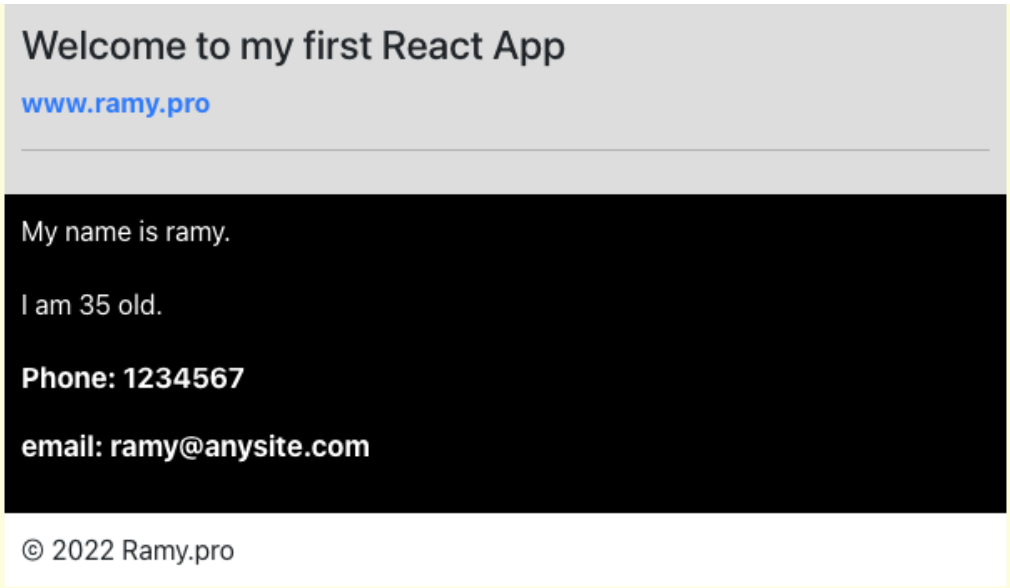
Now I'm going to create a new component called "Footer" in the "about.js" file.

```
export  function Footer(){
     let get_date = new Date();

     return(
       <div className="footer_area">
          &copy; {get_date.getFullYear()} Ramy.pro
       </div>
     );
}
```

Notice the "className" value is "footer_area", the name of the class I have created in the "my_style.css" file. Now, let's add this new component to the "index.js" file in the render function.

```
<Head />
<MyName name="ramy" />
<MyAge birth_year = "1987" />
<MyContacts info={{phone:"1234567",email:"ramy@anysite.com"}} />
<Footer />
```

Let's see the result.



Now we can see the changes in the body's background color, the paragraphs are thicker, the link has no underline anymore, and the "Footer" component has added with the class "footer_area" style.

What about the second way?! You can import the "CSS" file as a module, which is, the classes will be treated as the object's properties. I have to create another file and name it with the extension ".module.css". Be Careful, if you just name it without ".module.css" the "React JS" will not recognize this file as a "CSS Module" and won't work!

So, I'll create "my_style.module.css", and write into this file the next line.

my_style.module.css

```
.highlight{color: red; background-color: yellow; padding: 5px;
border-radius: 7px;}
```

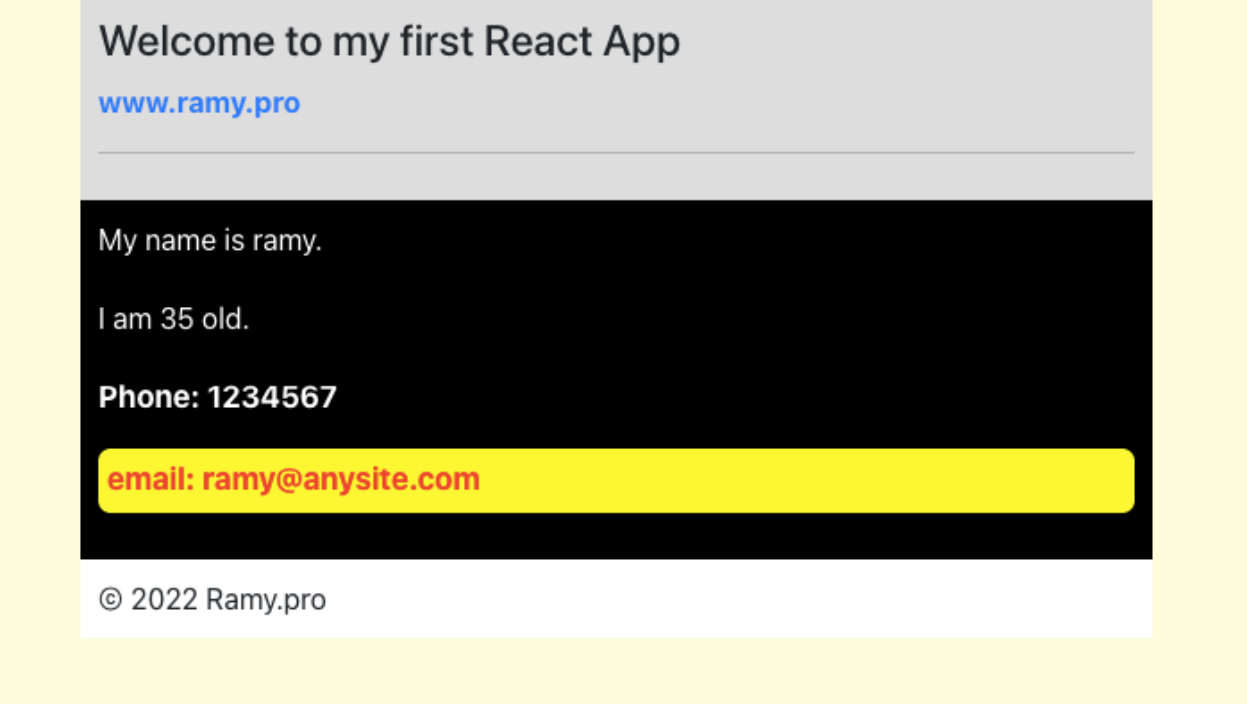
Now, I have a class called "highlight", to apply this class to any of the elements I must import "my_style.module.css" first, I am going to import into "about.js".

```
import styles from "./my_style.module.css";
```

Did you notice "styles" after the "import" keyword? the "styles" now holding the "classes", let's see how I can apply this class to the paragraph element in the "MyContacts" component.

```
<p className={styles.highlight}> email: {arg_obj.info.email}</p>
```

In HTML we usually use the "class" attribute, but in Javascript, we must use "className". So if you want to call any class in the "my_style.module.css" just write "styles." and the class name. Now, you can see this!



1.8 Bootstrap with React JS

You can install "Bootstrap" in your "React JS" application in many ways, but I am going to use the easier one. Simply, you can install "Bootstrap" by the "Node Package Manager" NPM.
We going to use a commands window for the next steps, so remember, if "Windows" is your operating system, you need to open "PowerShell" or "Command Prompt" for the older versions, if "macOS" or "Linux" is your operating system, you can use "Terminal".

First, you must be in the "project folder" location, you put the full path of the folder after "cd" command, for example:

```
cd Desktop/my_project
```

Now, enter the installation command.
For "Windows":

```
npm install bootstrap
```

For "macOS" or "Linux":

```
sudo npm install bootstrap
```

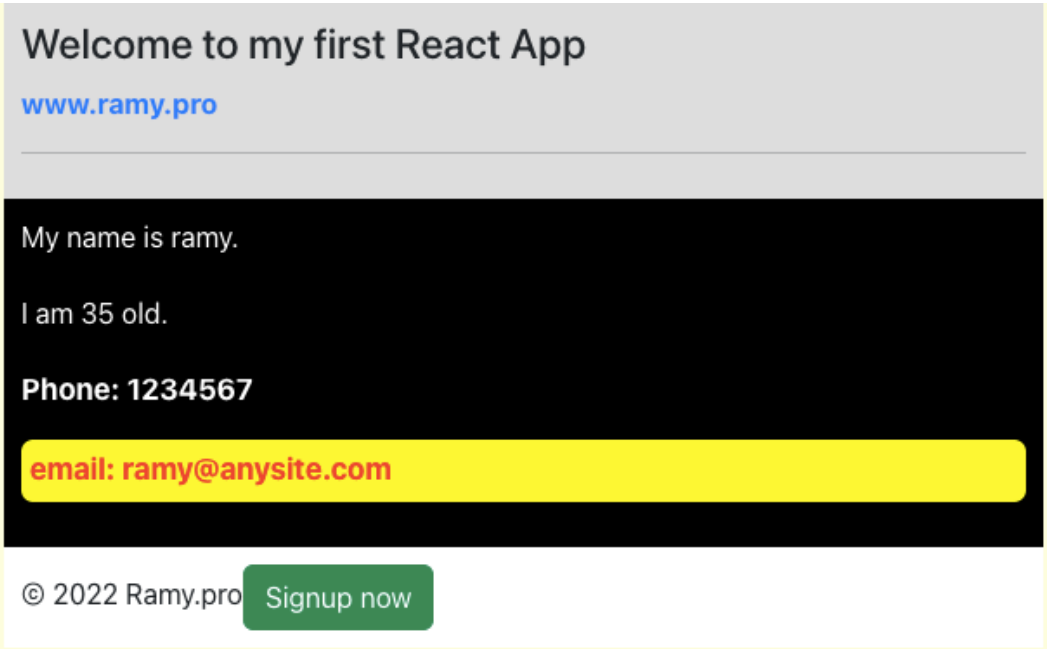
The terminal will ask you to put in the "admin" password.
Next, you need to import "bootstrap" in the "index.js" file, so, I'm going to add this line.

```
import 'bootstrap/dist/css/bootstrap.css';
```

Test time! I'm gonna add a "bootstrap" button into the "div" of the "Footer" component in the "about.js" file.

```
export  function Footer(){
     let get_date = new Date();

     return(
       <div className="footer_area">
        &copy; {get_date.getFullYear()} Ramy.pro
        <button type="button" class="btn btn-success">
           Signup now
        </button>
       </div>
     );
}
```

Here is the outcome.


Welcome to my first React App
www.ramy.pro
My name is ramy.
I am 35 old.
Phone: 1234567
email: ramy@anysite.com
© 2022 Ramy.pro
Signup now

5.0 Intro

How can we deploy our "React JS" project online?

Since we started our course and we use our devices as a server to host the project, in this case, those devices are called localhost.

This means we can access the website that we have made with "React JS" directly from the same device or from any other device connected to the same network.

But still, no one can get access to our "React JS" App through the Internet.

In this unit, we will cover how to deploy our project online so anyone can visit and use our application.

Let's take a quick view of the plan.

1 - Will know about the 2 options to deploy the app.
2 - Get the VPS server.
3 - Overview of the SSH & FTP.
4 - Set the server and install the required packages.
5 - Convert the "React JS" project to the production edition.
6 - Upload the production version to the server.
7 - Run the server and deploy the "React JS" App.

5.1 The two options to deploy the project

We have 2 options to deploy the “React JS” project.

1st make your device a permanent server, you can do that if you have admin access to the router that this device is connected to, and you have some knowledge about the Network settings.

The negativity of this option is your device must be running 24 / 7 days a week.
Of course, this will cost you money for the electricity, the security devices like a firewall, and need also another server to back up the main server.
So it's a lot of requirements for this option.

2nd option is to rent a server run by someone else for a couple of bucks, and you can control this server online.

Those servers are called VPS.

VPS means Virtual Private Server.
It's called virtual because it's using limited resources of the actual server.

This server will keep on until you shut it down or do whatever you want because you will get full access to the server configurations.

So, you can use it to host your website or your application data.
Or to host the database, or even to host your files.

In other words, this server will be your device and you can do whatever you want.

In the following lessons, I'll focus on how to buy an affordable VPS server, and how to get access to this server to host our React JS project.

5.2 Purchase the VPS Server

There are a lot of companies providing VPS services. Different offers and prices, but I found out that a company called "Applexe" offers affordable prices and excellent service starting from $6 a month.

The best thing is, you can specify the features you want for the server and based on that will give you the price.

Also, there are different services such as web development, windows app development, and more.

Let's start to get my VPS for around $6 a month.
Go to www.applexe.com
In the top menu click “servers and host”.
In the top menu click on "VPS Servers".
If you would like to know more about the service and the feature just scroll down.

We have here 2 types for the VPS server, KVM and OVZ.
The OVZ is cheaper, but KVM has more improvements, you can read about it.

Anyway, whatever you going to choose it will be the same path, and you will follow the same steps in this lesson to publish your "React JS" project online.

Enter the user name you would like, but write it down on paper or on your phone.

Choose the offer you want from the dropping list, and make sure to read the features for every server.
I'll choose the cheapest one to test it together.

With OVZ Server you can choose the datacenter location for your server.

I'll keep it in Chicago.

Enter the hostname you would like, it can be your current domain or your future domain, anyway, the server name must be in the format of the domain, like mywebsite.com, or server.mywebsite.com.

I'm going to put my website domain.

OS: the operating system, if you have experience with any one of them you can choose it, but in this lesson, I'm going to choose the last version of ubuntu.

If you want to add SSL to your domain you can do that but will charge you extra money for 1 year.

But I don't need that.

In the next section, you can purchase a new domain but it's optional, if you don't need it just skip it.

The next section is very important "Account owner details", enter your details correctly, and don't use any fake information.

If you want to review the agreements, check out those links then check on this box, and create a new account.

Next, will ask you to create a new password to the control panel for your account.

Choose the payment method and put in the information and complete the purchase.

You going to receive an email from Applexe includes the access information to the server and a link to the server control panel.

If you cannot see it in the inbox, check the spam or the junk box.

They will send you the main IP number that is linked to your server.
Also, you will find the Root pass.
Root pass means, the password of the root account.
"root account" has full privileges, so, when you access your server using this account, it means you can do whatever you want, so you have to be careful, and make sure you know what you do.

Also, you will find the link to your account control panel.
you can use it to purchase new services such as domain, ssl, or even a new server, or upgrade your current server.

You can sign in to the control panel using the user name that will be sent, but you have to set the password just before the first-time access through this link.

Scroll down a little bit to find the link to the full control panel on your server with the access information like user name and password.

let's take a quick view.
On the home page, you can see 2 sections, the first one is just a summary that includes general information about the server.

While second section "Controls" is divided into 6 sections.
Through the "General," you can restart, shut down, or turn on the server.

If you would like to reset the server, just click on "Reinstall", choose the operating system, and click Reinstall again, but keep in your mind you going to lose any files that on the server, so you can make a backup first in the "central backup" section.

Just click on 'create backup' and you can restore from this section too.

In the "Hostname" section you can change the server’s name.

While in the "Root/Admin Password" section you can change the root account password, I recommend doing that.

in the "Network" section you can set the DNS.

You can take a quick view of the other sections, if you have experience in the network field, and the API, you can use them any time.

Ok, let's jump to the next lesson, to know how we can get access to the server using SSH protocol.

5.3 SSH and FTP

There are 2 things you must know first.
SSH and FTP

SSH means "secure shell" It's a network communication protocol allowing us to connect to other devices remotely.

While FTP means "File transfer protocol".
FTP gives us the ability to transfer files from and to the server remotely.

To use ssh you need to know the server’s name or IP and the user account to access this server, you can get the access information from the company that going to provide the service.

In the next lesson will use command-line software such as PowerShell in Windows or "Terminal" in MAC OS to start an SSH session.

Also, you need to get an FTP client, it's just software to use to connect to the server through the FTP protocol.

I'll use FileZilla, it's free, easy, and available for most operating systems.
So, you need to download and install it from the official site.
https://filezilla-project.org/

Remember: download the client version, not the server version.

If you prefer to use another FTP client, it's fine as long as you know what you do.

5.4 Prepare the server

In this lesson, you will learn how to access the VPS server over SSH protocol using the root account, and do some stuff to prepare our server to be ready to host our "React JS" App.

Let's start.
To access any server, you need the host name or the IP number.
The user account and the password.
As I mentioned before, the VPS service provider will send you an email including this information.
Applexe already sent me this email with the root account.

The root account has full privileges, and by using this account you have full permission to do whatever you want on the server.

"Root account" has been created by the operating system as a default account.
So it's recommended to use this account just in some cases to avoid harming your server accidentally.

So for security purposes, I recommend creating a new user account to use it to access the server instead of the root account.

Let's start.
In this lesson, we need a command-line application such as Terminal on Mac or PowerShell on Windows.

To start the ssh session, will enter the command
ssh account name @ server name IP.

So for just the first time, I'll use the root account.
ssh root@ the server IP number.

Ok, if you using Terminal on Mac and you get this warning message just enter the following command, to tell the security system of the operating system that you trust this connection and register the key that will be coming from this server.

ssh-keygen -R and the server IP number or the server name

```
ssh-keygen -R 198.23.52.243
```

Now let's start the session again
For just the first-time access, will ask about something called the fingerprint, just enter "yes".

Now it's asking me to put in the "root account" password, keep it in your mind, while you enter the password it will be hidden, so you will not see anything printing out on the screen.

Perfect, now I can see the operating system name and version.
In the last line, you can see the account name which is "root" and @ the server name I have named "applexe".

I'll just enter the"clear" command to clear the screen.

Quick overview of the "apt" command.

the "apt" is short for "Advanced Packaging Tool ".
The "apt command" takes the responsibility to install, uninstall, and upgrades the software packages on the server.

The "apt" command uses an index for the available packages on the system.
So, we need to update this index using the "update" argument.

So, enter the command:

```
apt update
```

Now we have 10 packages that can be upgraded, we can do that using the "upgrade" argument.

```
apt upgrade
```

This is confirmation about the disk space, just enter y.

Next step.
I need to create a new user account and add it to the sudo group.
"sudo" is short for "Super User Do", in other words, I can use the "root privileges" through this account.

To create a new user enter command "adduser" and put the user name.

```
adduser ramy
```

Next, enter the new password for this account.
Now will ask me to put some of the information about this user such as the full name, room number, and work phone, but I'll put just the name and skip the rest by pressing Enter.

Enter y as an answer to the confirmation question.

Now, I'll add this account to sudo group.
adduser ramy sudo

Perfect, let's log out from the root account, and start a new ssh session to access the server by using the ramy's account.

By entering the "logout" command, the SSH session will be ended and the connection to the server will be closed.

Now, let's start a new SSH session using ramy's account.

ssh ramy@ip number
enter the password of this account.

Perfect, now we can see in the last line the account name which is ramy@applexe.

The next step will be disabling the root account access, so no one can access to the server using the root account.

In this case, I need the sudo privileges to do that.
If you need the "sudo" privileges anytime just enter the "sudo" command first like the following.

sudo passwd -d root
This command will disable the password of the root account.

If asking for the password, I'll put ramy's account password, as long this account is in the sudo group.

The following command locks the root account.

```
sudo passwd -l root
```

Now, I'll log out from my account, and try to access using the root account to make sure it's disabled.

Permission denied.
Perfect, to terminate the ssh session in Terminal, press Command and Q.

5.5 – Install Node JS & NPM and create Test React JS App

In the previous lesson, I prepared the server, and it's ready now to install the "node js" . There are many ways to install "node js" through the command line, I'll use NVM package to install the node js.

NVM the short for "Node Version Manager", through the NVM we can install a specific node js version, upgrade, or downgrade to the previous version.

The "NVM" package is not installed by default on the server, so we can install it by using the "curl" package.

The "curl" package is used to download files to the server.

Let's install the "curl" package first.

To proceed to the next steps, I need to switch to the root account.
To switch to the "root account" I'll use sudo command.

```
sudo -s
```

I'll enter my account password, as long my account part of the sudo group.

Next, I'll use the NVM installer

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
```

Now, let's follow the instructions, I'll copy the three commands and run them.
Now let's check if the nvm is installed correctly, the version number will be printed out.
So I'll enter the nvm command space dash v

```
nvm -v
```

Now, I can print out the available node js versions so I can choose one from them using the ls-remote argument.

```
nvm ls-remote
```

Perfect, so the latest version right now is v20.3.1, to install this version I can use the "install" argument with nvm.

```
nvm install v20.3.1
```

Now, let's check if the Node js and NPM are installed correctly.

If we can see the versions means everything going well.

Now, let's build a new "React JS" app as a test, and run it on the server.

I'll create the "React JS" App in the var directory.
var directory located in the main directory.
using the cd command and space .. I'll go to the main directory or I can just enter

```
cd /
```

To display the current directory content, enter ls command.

```
ls
```

so, now I can see the var in the current directory.
to go inside the var directory enter

```
cd var
```

perfect, now I am inside the var directory, and I'll create the "reacte js app" here.

```
npx create-react-app test
```

Perfect, I just notice this message informing me that there is a new npm version, so let's update the npm quickly.

The "React JS" app has created in the var directory a new directory called test that includes the "React JS App" files.
So we need to be inside the "test" directory.

```
cd test
```

Ok, let's start the test app, by using the npm start command.

```
npm start
```

Perfect, the "React App" is running now using port 3000, to open in the browser, will replace the localhost with the server IP number in the URL address.

This very good sign that everything going perfectly, except for one thing.

To keep the server running I must keep the SSH session running too, and If I'll terminate it, the server will be stopped.

It doesn't make sense at all!
So, in the following lessons will learn something new about how to create the production version for our "React JS" project, and how to upload and host it on the VPS server.

5.6 – Build the production version and upload it to the VPS.

The production version is the outcome of your project.
The difference between the production version and the development version is the code of the production version is encrypted and not easy to understand.

Also, the size of the development version is very huge compared to the production.
Imagine that our project size is over 400 MB while the production version is just 4 MB.

Ok, let's start the first step, building the production version.

First, I need to get the full path for the main directory of our project on my computer.
Next, using the Command-Line software, go inside this directory by entering the "cd" command and the path.

Using the "npm" command will build the production version.

```
npm run build
```

Notice that a new folder called "build" has been created.
So, we need just to upload this folder on the server instead the whole project.

I'll rename the folder from "build" to the project name.

The second step will be to place the production folder on the server.

But before that, I need to create a new directory on the server, I'll name it "apps" so I can use it as a main directory for my applications.

So, I need to access the server using SSH protocol.
Next, I'll create the new directory using the "root" account privileges, so I'll switch to the "root" account using the "sudo" command.

```
sudo -s
```

Perfect, I'll create my new directory in the "var" directory, so I need to change the current directory to the "var" using the "cd" command.

If you don't know where is the "var" located? just go to the main directory by entering cd /

```
cd /
```

And enter the "ls" command to list the files and folders that the current directory contains.

Here is the "var" directory located on the main.

Let's go inside "var", and using "mkdir" command will create the "apps" directory.

I'll use "ls" in "var" to list the contents.
Perfect, here is the "apps" directory that has been created.

Now, it's time to use the FTP client to upload the app.
If this is the first time using the FTP, it's similar to the SSH but we use it to send our local files to the server.

As I mentioned before, I'll use FileZilla, you can download the free version from the official site.

Let's start a new connection, by clicking on the server's icon, and "New site".

From the protocol list, choose "SFTP" or Secure FTP instead of FTP, because it secures your data from leaking to the hackers.

In host, but the hostname or the server IP.
I'll use my account access information here, the username, and the password.

And click Connect.

May at the first time connection you will see the "Unkown host key" window, just check the "Always trust" box and click "OK".

Perfect, now you have 2 sub-windows in the FTP client, the one on the left side is listed contents of the current directory on your local device, while the right one is on the server.

So, in the right window, I'll change the directory to "apps" that I have created.

But, there is a small problem here I have to fix first so I can upload my files.

The owner of this directory is just the "root" account, which means my account has no permission to change on this directory.

So, if I'll try to upload even one file, I'll receive in the console the "permission denied" message.

To solve this issue, I must add my account "ramy" to the owner group for this directory using the "root" account.

Back to the SSH session.
To change the permission code to this directory, I'll enter:

```
chmod 775 /var/apps
```

So I can allow the owners to read, write, and execute.
To know more, just search "The Permission Indicators".

Next, I'll add my account as an owner of this directory.
Using the "chown" command.

I'll enter "chown", my account name, and the full directory path.

```
chown ramy /var/apps
```

Perfect, now I'll refresh the "var" directory in the FTP client.

Now, I can see my account name in the owner's group.

Now simply I can drag the production folder and drop it into the server.

And I'll enter the "ls" command to make sure that the folder has been uploaded successfully.

And here is the content.

Now, it's time to install the server to keep our application running online.

5.7 – Install PM2 and deploy the apps online

Welcome to the last step to deploying our "React JS" application online.

There are many options to keep your application running on the server.
But I have chosen the simplest one.

MP2 is software to manage the deployed applications on the server.
So you can run multi applications on the same server using different ports.

First, I'll install MP2 on our VPS server using the NPM or "Node Packages Manager".

Let's access the server through SSH.

And switch to the "root" account.

```
sudo -s
```

If I'll try to get the PM2 version.

```
pm2 -v
```

I'll receive this message, which means it's not installed yet.

So, I'll enter the installation command.

```
npm install pm2@latest -g
```

Now, let's check the MP2 version again.

```
pm2 -v
```

Now, I'll change the current directory to go inside "apps".
I'll display the directory content using the "ls" command.

and here is our application's directory.

To run the application using the PM2 we have to put the arguments in the right way.
The first argument must be "serve".
The second will be the directory name if you are inside the parent directory, or better to put the full path.

The third argument will be the port number.
at the end put "--spa" which it means will run a single application.

So, I'll put here just the directory name as long as the parent directory is the current one.

I'll use the "7070" port to access our application.
Amazing!

Now we can see my "React JS" application in the deployed list, and the status is online, so anyone who has internet can access my application using the server IP number and the 7070 port.

Let's try it or try by yourself by following this link.
http://198.23.52.243:7070/
Now, I'll log out and end the SSH session to see if the application will keep running or not.

Perfect!

To manage the deployed applications, you must use the "root" account.
So, to list the applications, enter.
PM2 list

You can restart, stop, and run the application again by using the ID number that in the first column.

Or you can determine all to apply the action on the applications that are on the list.