# React Components

First to know is that react is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called “Components”

React has a few different kinds of components, but we’ll start with React.Component subclasses:

class ShoppingList extends React.Component{

    render(){

        return(

            <div className="shopping-list">

                <h1>Shopping List for {this.props.name}</h1>

                <ul>

                    <li>Cat Bed</li>

                    <li>Cat food</li>

                    <li>dog Blanquet</li>

                </ul>

            </div>

        );

    }

}

There is a practical tic tac toe program to get involve with, I am going to pause on it to go back and learn a little bit more on props, this is what I got down (tutorial link: Dan Abramov (username gaearon), Codepen.io, <https://reactjs.org/tutorial/tutorial.html>).

https://reactjs.org/docs/components-and-props.html

https://reactjs.org/tutorial/tutorial.html

https://reactjs.org/docs/react-api.html#createelement

https://create-react-app.dev/docs/available-scripts

# Components

React lets you create component as classes or functions. Components defined as classes currently provide more features. To define a React component class, you need to extend React.Component:

The only method you must define in a React.Component subclass is called [render()](https://reactjs.org/docs/react-component.html#render). All the other methods described on this page are optional.

# How to render a function

Great, so render is a must subclass in a class, this to show something in screen. There is also React.DOM which has a render method as well, for the syntax there is a couple of things going on in just a several lines of code. With 16 lines I made a component:

Defied a component

class Welcome extends React.Component {

  render(){

    return <h1>Hello, {this.props.name}</h1>;

  }

}

, defined a name:

const element = <Welcome name = "Ashley"/>

, ReactDOM.render() was called with the name and was passed as a child in the component.

ReactDOM.render(

  element, document.getElementById('root')

);

This is what happened (reactjs.org, 19)

1. We call ReactDOM.render() with the <Welcome name="Sara" /> element.
2. React calls the Welcome component with {name: 'Sara'} as the props.
3. Our Welcome component returns a <h1>Hello, Sara</h1> element as the result.
4. React DOM efficiently updates the DOM to match <h1>Hello, Sara</h1>.

Note: To always start the components with a capital letter for errors with JSX

**We strongly recommend against creating your own base component classes.** In React components, [code reuse is primarily achieved through composition rather than inheritance](https://reactjs.org/docs/composition-vs-inheritance.html).

It is getting a little easier…

Example 2:

This time I will render a title and a paragraph

function FancyBorder(props){

  return(

    <div className = {'FancyBorder FancyBorder-' + props.color}>

      {props.children}

    </div>

  );

}

FancyBorder is something defined with some CSS code I had:

.FancyBorder {

  padding: 10px 10px;

  border: 10px solid;

}

.FancyBorder-blue {

  border-color: blue;

}

.Dialog-title {

  margin: 0;

  font-family: sans-serif;

}

.Dialog-message {

  font-size: larger;

}

The big chunk of code comes from WelcomeDialog():

function WelcomeDialog(){

  return(

    <FancyBorder color = "Brown">

      <h1 className="Dialog-title">

        Welcome

      </h1>

      <p className="Dialog-Message">

        Thank you for visiting the healthy zone!

      </p>

    </FancyBorder>

  );

}

And at last comes ReactDOM.render which is going to render the full function called WelcomeDialog companioned with a document.getElementById(‘root’):

ReactDOM.render(

  <WelcomeDialog/>, document.getElementById('root')

);

This second part (document.getElementById(‘root’)); is the border represented as a child within the function fancyborder.

The render function should be pure, meaning that it does not modify component state, it returns the same result each time it’s invoked

If you need to interact with the browser, perform your work in componentDidMount() or the other lifecycle methods instead. Keeping render() pure makes components easier to think about.

render() will not be invoked if [shouldComponentUpdate()](https://reactjs.org/docs/react-component.html#shouldcomponentupdate) returns false.

How does react now which one is root?

I did find where the index.html had hid and it is the public folder of my project, there I can define my html root tags.

# Bibliography

<https://reactjs.org/docs/react-component.html#shouldcomponentupdate>

https://reactjs.org/docs/react-component.html#render

<https://reactjs.org/docs/components-and-props.html>

# Constructor

So to start a constructor in react you need to initialize state or if you have bind methods, this is not required for a component.

Typically, in React constructors are only used for two purposes:

* Initializing [local state](https://reactjs.org/docs/state-and-lifecycle.html) by assigning an object to this.state.
* Binding [event handler](https://reactjs.org/docs/handling-events.html) methods to an instance.

In a constructor you should not call setState(), instead if your component needs to use local state, assign the initial state to this.state directly in the constructor.

# Component did mount

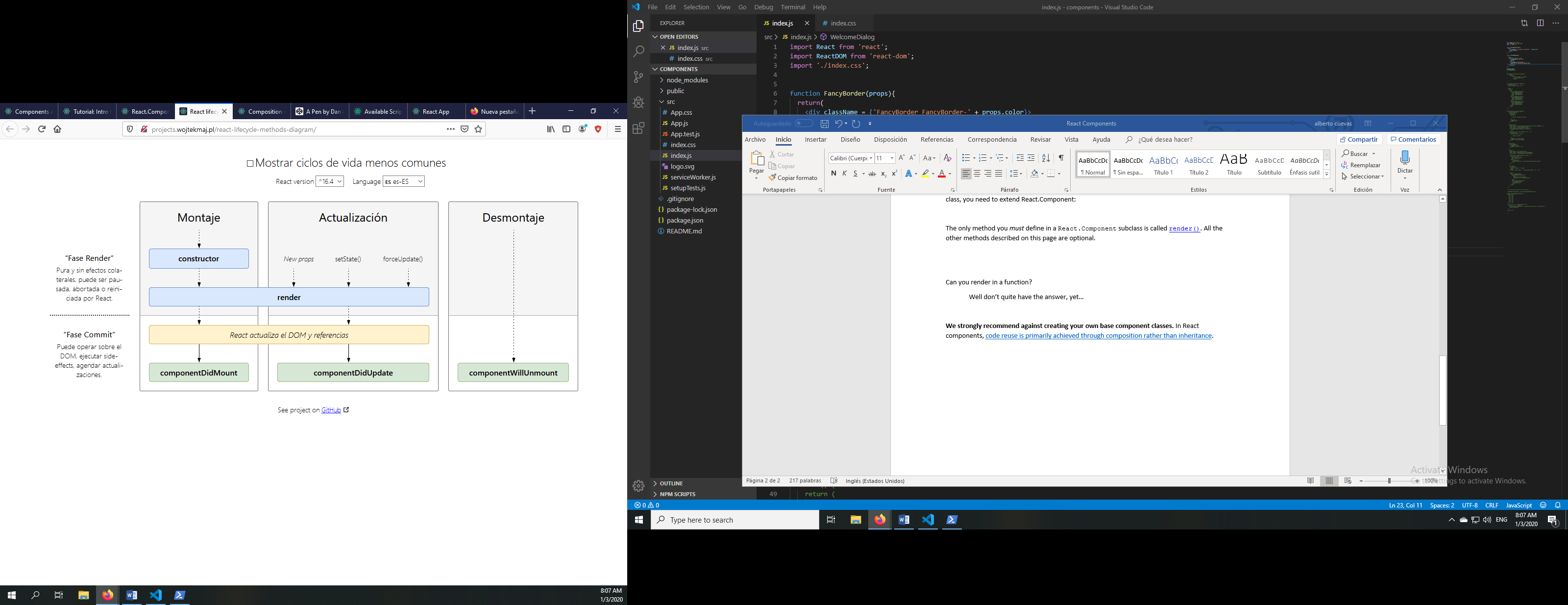
componentDidMount is invoked immediately after a component is mounted (inserted into the tree). If you need to load data from a remote endpoint, this is a good place to instantiate the network request.

Is to continue…

# Next Lesson Extracting components

https://reactjs.org/docs/components-and-props.html

La vida y el ciclo de un programa React, funciona de la siguiente manera:



Se construye, se renderiza todo, después de ello React tiene una fase de renderización en DOM.

Tienes ahora el rectángulo verde que te confirma que se monto algo en react. Esta es la fase de montaje, esta misma comparte una columna llamada actualización, esta va a la par de montaje, sin embargo esta columna no monta nada. Solo se encarga de actualizar lo que se ocupe.