**Project note**

Q: How to use ‘for Each’?

A: mySimpleArray = ['a', 1, 'etc'];

mySimpleArray.forEach(function(val) {

console.log(val);

});

* Used over arrays

Q: for-in:

A: Used over objects

his can be error prone because for-in does not loop from 0 to length - 1 but over all the present keys in the object and its prototype chain.

* So:

myObj = {'firstName':'Ada','lastName':'Lovelace'};

for (var key in myObj) {

if (myObj.hasOwnProperty(key)) {

console.log(myObj[key]);

}

}

\_For new repo on github, use git remote add origin

Q: Sometimes you create a remote repo and a local repo. Then when you try to merge the 2 together (pull), you receive this message “fatal: refusing to merge unrelated histories

“. What do you do?

A: git pull origin branchname --allow-unrelated-histories

Q: componentDidMount(){

BooksAPI.getAll().then((books) =>{

this.state({books})

})

}

A: getAll goes into the BooksAPI where it’s defined, and then it goes through all the promises and saves into this.state. books: books is simplified to just “books”

Q: So when do we use =() =>{} on the method and when do we just use method()?

A: They’re basically the same thing if we consider what’s written on the render() method:

1. Properly initialized syntax:
2. class LoggingButton extends React.Component {
3. // This syntax ensures `this` is bound within handleClick.
4. // Warning: this is \*experimental\* syntax.
5. handleClick = () => {
6. console.log('this is:', this);
7. }
8. render() {
9. return (
10. <button onClick={this.handleClick}>
11. Click me
12. </button>
13. );
14. }
15. }

2) Non-properly initialized syntax:

class LoggingButton extends React.Component {

handleClick() {

console.log('this is:', this);

}

render() {

// This syntax ensures `this` is bound within handleClick

return (

<button onClick={(e) => this.handleClick(e)}>

Click me

</button>

);

}

}

* They are the same, but preferably use ()=>{}