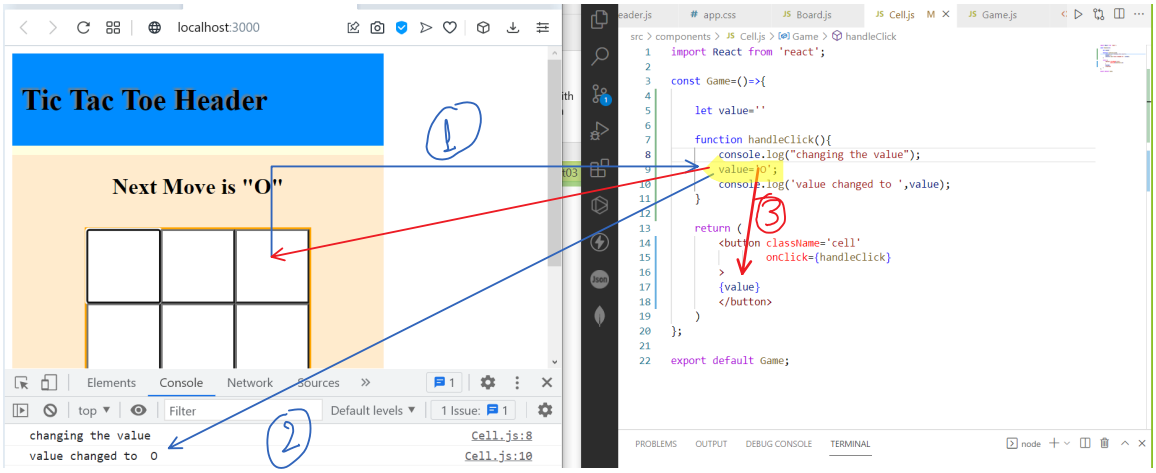
Updating the UI

Tuesday, March 22, 2022 3:40 PM

- When our code changes the value, it may not automatically update the UI



1. When button is clicked, it calls handleClick
2. handleClick modifies the value successfully
   - We can see value change in console.log
3. UI is trying to display the same value
   - But change is not reflected in the UI

Why is UI not updated?

- React re-renders the screen when it identifies values have changed
  - When component is re-rendered UI gets refreshed.
- But when wen change a normal variable, React doesn't know that the value has changed.
  - We know we changed the value.
  - React doesn’t know we changed the value.

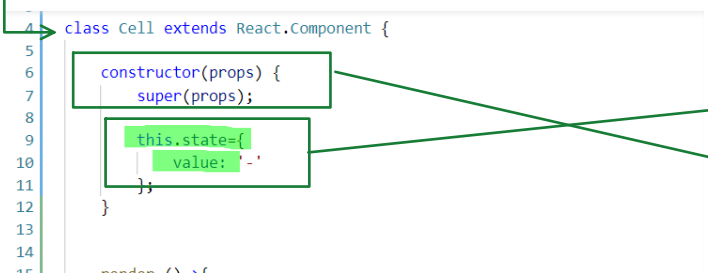
- A component can't modify the 'prop' passed to it. They are read-only.

How to mange data that may change?

- React defines a special type of variable for components.
- They are called **state**
- A state is traditionally available only in class based components
- It is a class property

Stateful class design

1. Create a class based component
2. Define constructor to set initial value of state
   - If you define the constructor it must
     - Take 'prop' object as parameter
     - Pass 'prop' to super constructor
3. State must be an object.
   - It should not normal vlaue like
     - int
     - String



1. Create initial state
   - It should be any normal object
   - Can have multiple value
2. If you are creating a constructor, you must take 'prop' as parameter

```
15      render=()=>{
16
17          return <button className="cell">{this.state.value}</button>;
18      }
19
20  }
```

and pass it to super class

3. Now you can access and use the value using this.state.value syntax.

2. Changing the state value —> this.setState

- Never change the state value directly

  ~~this.state.value="new value";~~

- React will not know you have changed value.

- To change the value you should use this.setState()

  ```
  this.setState( {  value : newValue } );
  ```

State management summary

```
class Cell extends React.Component {

    constructor(props) {
        super(props);

        this.state={
            value: '-'
        };
    }

    handleClick=()=>{
        //this change, will not reflect in react
        const newValue= this.state.value==='O'?'X':'O';
        console.log(`value for cell # ${this.props.id} changed to ${new

        //this change will reflect in React and UI will updated

        this.setState({value:newValue});
    }


    render=()=>{

        return <button
            onClick={this.handleClick}
            className="cell">{this.state.value}</button>;
    }
}
```

1. Initial setup (one time)

- Done by using normal object creation syntax.
- Nothing special.
- Never change state this way after initial setup

2. Changing the State

- Done by calling this.setState and passing new values
- Never change the state by directly assigning the value
- UI will not be updated.

3. Reading state value

- Done normally using this.state.value
- **This is not a special syntax**
- When state changes, it updates the UI.