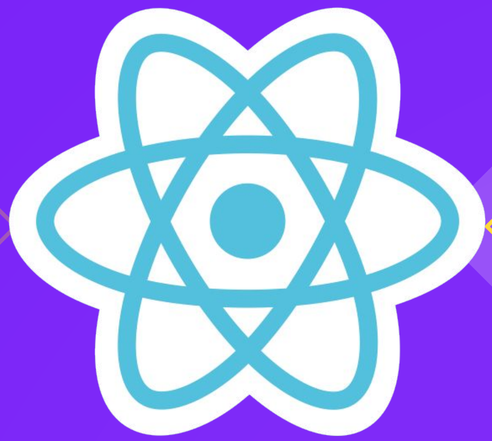
STEP ACADEMY

УРОК №1

Компоненты в React



Создание компонента

```
import React from ‘react’

class MyComp extends React.Component {
    render(){
        return <p>MyComp</p>
    }
}
```

Внимание! Важно!

Компоненты нужно называть с большой буквы, иначе в процессе рендеринга React может ошибиться и принять название компонента за атрибут.

defaultProps

Свойство defaultProps класса - это значение props по умолчанию, т.е. те значения, которые будут переданы в props, если мы ничего не пропишем в атрибутах компонента

MyComp.defaultProps = {name: "Username"}

Состояние компонента

Работа с компонентами в React завязана на состояниях.

Состояние - набор свойств компонента в конкретный момент времени.

props - данные извне, state - данные, необходимые и доступные внутри компонента.

Каждый раз, когда происходит изменение состояния, происходит ререндеринг компонента.

Установка начального состояния

```
constructor(){
        super();
        this.state = {color: "red"}
    }
```

state представляет собой объект, в котором записаны данные

Изменение состояния

События.

События в React записываются похожим образом, как если бы это происходило в HTML

onClick={this.changeColor}//со скобками это был бы вызов метода

Важно! Проверьте, чем является внутри колбэка.

Привязка this

```
this.changeColor = this.changeColor.bind(this);
```

Используйте такую запись в конструкторе для того, чтобы привязать значение this к компоненту.

Изменение состояния через событие

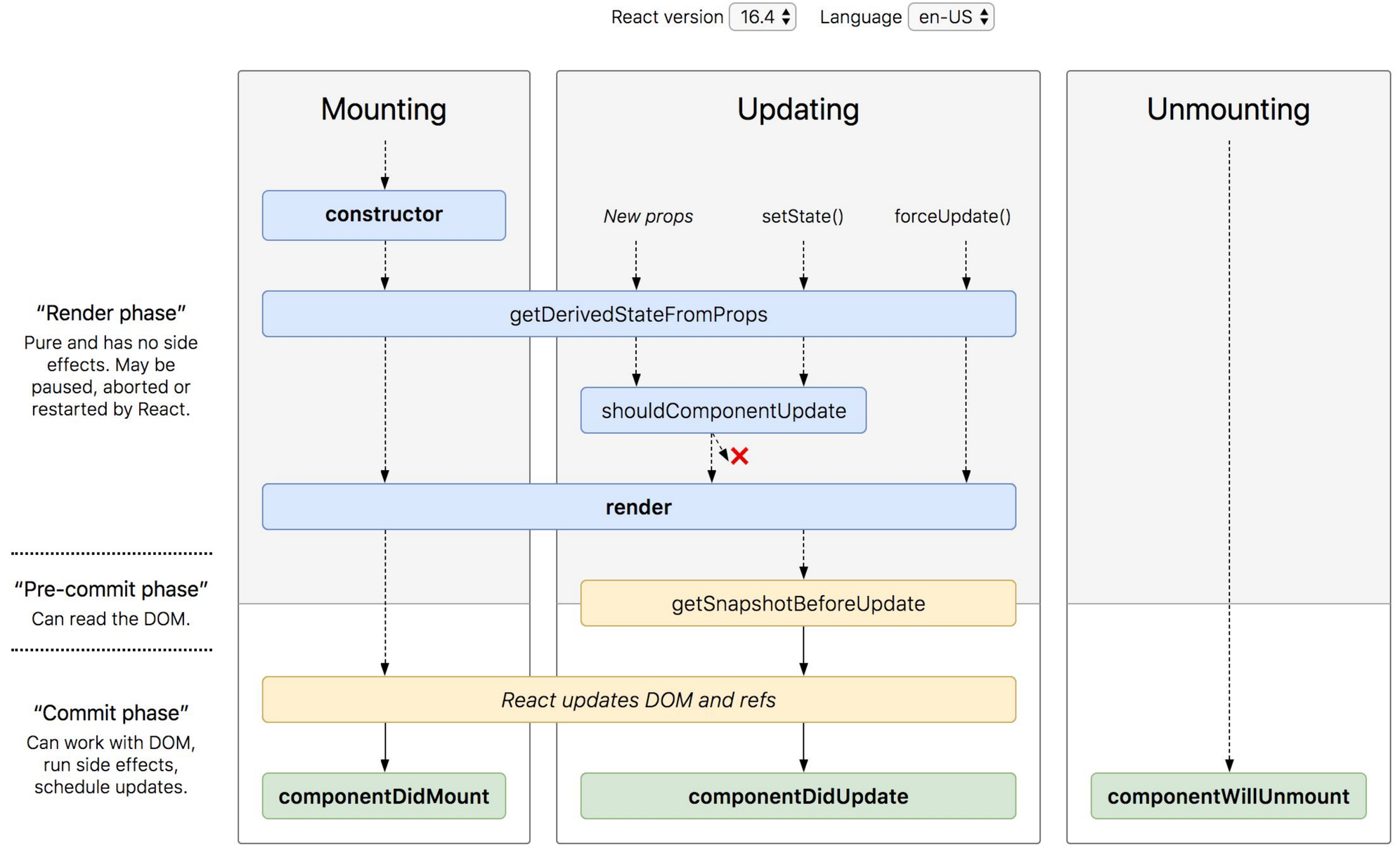
```
changeColor(){
        this.setState({color: "green"})
    }
```

Метод setState устанавливает состояние (state) компонента.

Жизненный цикл

Жизненным циклом компонента называют этапы, через которые он проходит в момент его существования.

Используя следующие методы, можно подвязать действия к определенному моменту в жизненном цикле.


React version 16.4
Language en-US
Mounting
Updating
Unmounting
constructor
New props
setState()
forceUpdate()
getDerivedStateFromProps
shouldComponentUpdate
render
getSnapshotBeforeUpdate
React updates DOM and refs
componentDidMount
componentDidUpdate
componentWillUnmount
"Render phase"
Pure and has no side effects. May be paused, aborted or restarted by React.
"Pre-commit phase"
Can read the DOM.
"Commit phase"
Can work with DOM, run side effects, schedule updates.

constructor()

Конструктор компонента React вызывается до того, как компонент будет примонтирован. В начале конструктора необходимо вызывать super(props).

Конструктор – единственное место, где можно напрямую изменять this.state. В остальных методах необходимо использовать this.setState().

getDerivedStateFromProps(props, state)

Редко используемый хук.

getDerivedStateFromProps вызывается непосредственно перед вызовом метода render, как при начальном монтировании, так и при последующих обновлениях. Он должен вернуть объект для обновления состояния или null, чтобы ничего не обновлять.

componentDidMount()

componentDidMount() вызывается сразу после монтирования (то есть, вставки компонента в DOM). В этом методе должны происходить действия, которые требуют наличия DOM-узлов.

shouldComponentUpdate(nextProps, nextState)

Используйте shouldComponentUpdate(), чтобы указать необходимость следующего рендера на основе изменений состояния и пропсов. По умолчанию происходит повторный рендер при любом изменении состояния. Возвращает true или false

getSnapshotBeforeUpdate(prevProps, prevState)

getSnapshotBeforeUpdate() вызывается прямо перед этапом «фиксирования» (например, перед добавлением в DOM). Он позволяет вашему компоненту брать некоторую информацию из DOM (например, положение прокрутки) перед её возможным изменением. Любое значение, возвращаемое этим методом жизненного цикла, будет передано как параметр componentDidUpdate().

componentDidUpdate(prevProps, prevState, snapshot)

componentDidUpdate() вызывается сразу после обновления. Не вызывается при первом рендере.

Метод позволяет работать с DOM при обновлении компонента.

componentWillUnmount()

- componentWillUnmount() вызывается непосредственно перед размонтированием и удалением компонента. В этом методе выполняется необходимый сброс

Пример использования жизненного цикла

```
componentDidMount(){
        this.setState({
            timer: setInterval(()=>{this.setState({date:
new Date()})}, 1000)
        })
    }
```

```
componentWillUnmount(){
      clearInterval(this.state.timer);
  }
```

Дочерние компоненты, передача данных в дочерний компонент

Здесь всё просто.

Дочерние компоненты - это другие компоненты, используемые в родительском

В дочерние компоненты данные можно передавать через пропсы.

Любимая задача: ToDoList

Файл App

```
<Todolist list={list}></Todolist>
```

Файл Todolist

```
            <ul>
                {this.props.list.map(val=>{
                    return <li>{val.do} в {val.time}</li>
                })}
            </ul>
```

Передача данных из дочернего компонента родительскому (App.js)

```
 constructor(props){
    super(props);
    this.state = {list};
    this.deleteTodo = this.deleteTodo.bind(this);
  }
  render(){
    return (<Todolist list={this.state.list}
deleteTodo={this.deleteTodo}></Todolist> )
  }
  deleteTodo(index) {
    let tempList = this.state.list;
    tempList.splice(index, 1);
    this.setState({list: tempList})
  }
```

Todolist

```
class Todolist extends React.Component{
    render(){
        return (
            <ul>
                {this.props.list.map( (val, index)=>{
                    return <li
onClick={()=>{this.props.deleteTodo(index)}}>{val.do} в {val.time}</li>
                })}
            </ul>
        )
    }
}
```

Поднятие состояния

Чтобы перерисовать компонент, нужно изменить состояние. Но когда в компоненте находятся два дочерних, состояния которых нужно связать (один изменяется в зависимости от другого), нужно использовать поднятие состояния.

Под данным термином подразумевается процесс, в котором изменение состояния производится в родительском компоненте, чтобы изменить второй компонент.

Задание

Сделать страницу со списком магазинов, обслуживаемых поставщиком.

Свойства: название, время открытия, время закрытия, удалённость от центра доставки, является ли “особенным”.

Компоненты: основной, таблица с представлением, фильтр-поиск-сортировка, добавление нового магазина.

Конец

ПОСЛЕСЛОВИЕ

Давайте подведем итоги урока!
Чему мы научились? Что мы использовали?
К чему мы пришли?