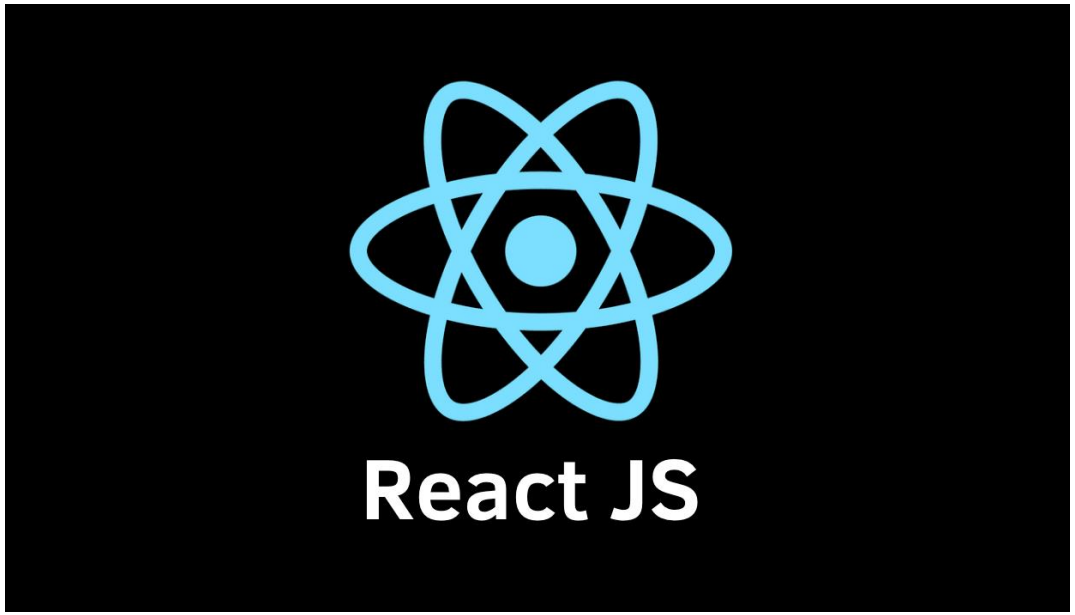
프론트 세미나 Week7
ReactJS (2)

Review



Last lecture: ReactJS (1)

- React의 개념, 특성과
- Component, Props, States 등에 대해서 알아보았습니다.

- 이번 시간에는 React의 Lifecycle과 form을 어떻게 제어하는지에 대해 배워볼 것입니다!

Forms

```
<div className="form">
    <form onSubmit={this.handleSubmit} method="get">
        <input type="text" name="first_name" onChange={this.handleChange}/>
        <input type="text" name="last_name" onChange={this.handleChange}/>
        <input type="submit" value="Submit"/>
        <ul>
          <li>{this.state.first_name}</li>
          <li>{this.state.last_name}</li>
        </ul>
    </form>
</div>
```

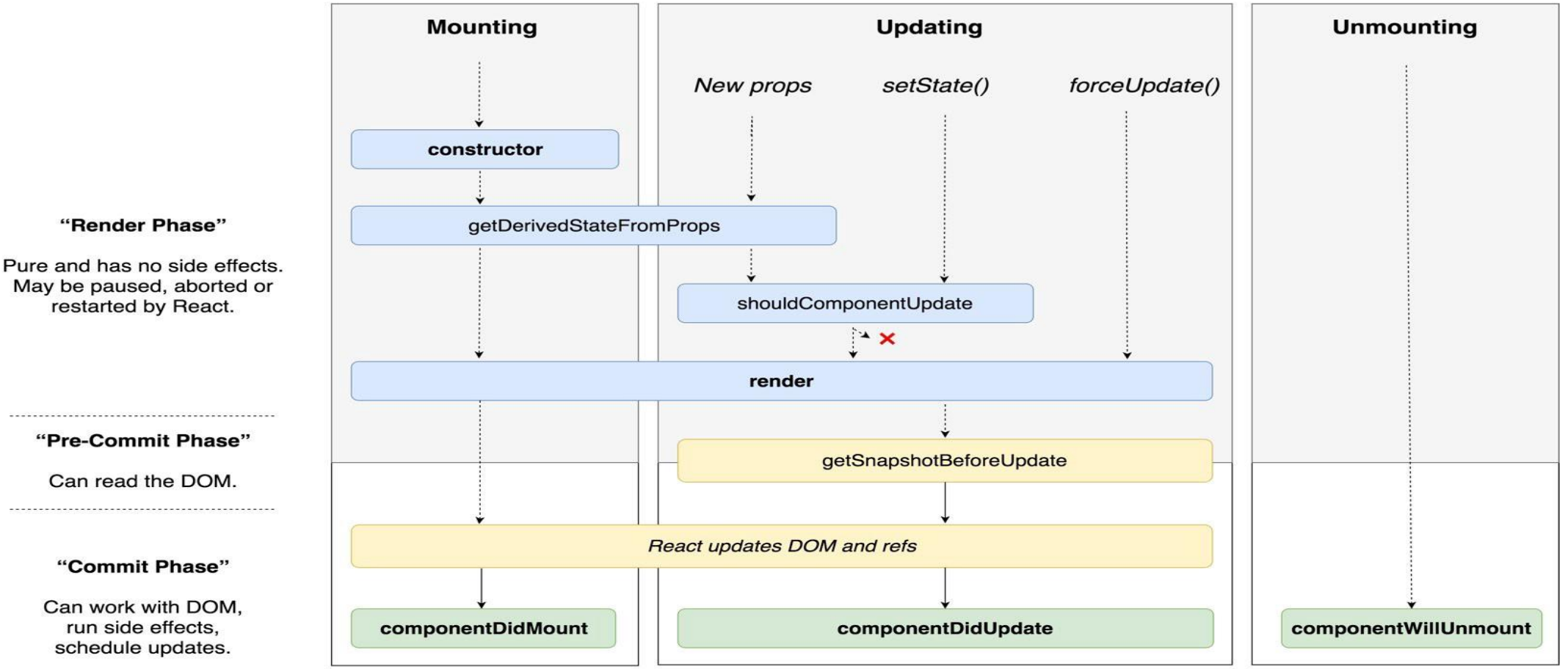
- onSubmit, onChange 속성 이용
- onChange: input 값이 바뀔 때마다 호출
- onSubmit: submit 버튼 클릭 시 호출

Forms

```
40    handleSubmit=(e)=>{
41      alert("submit first_name: "+this.state.first_name+", last_name: "+this.state.last_name);
42    }
43
44    handleChange=(e)=>{
45      var name=e.target.name;
46      var value=e.target.value;
47      console.log(name,value);
48      this.setState({[name]:value})
49    }
```

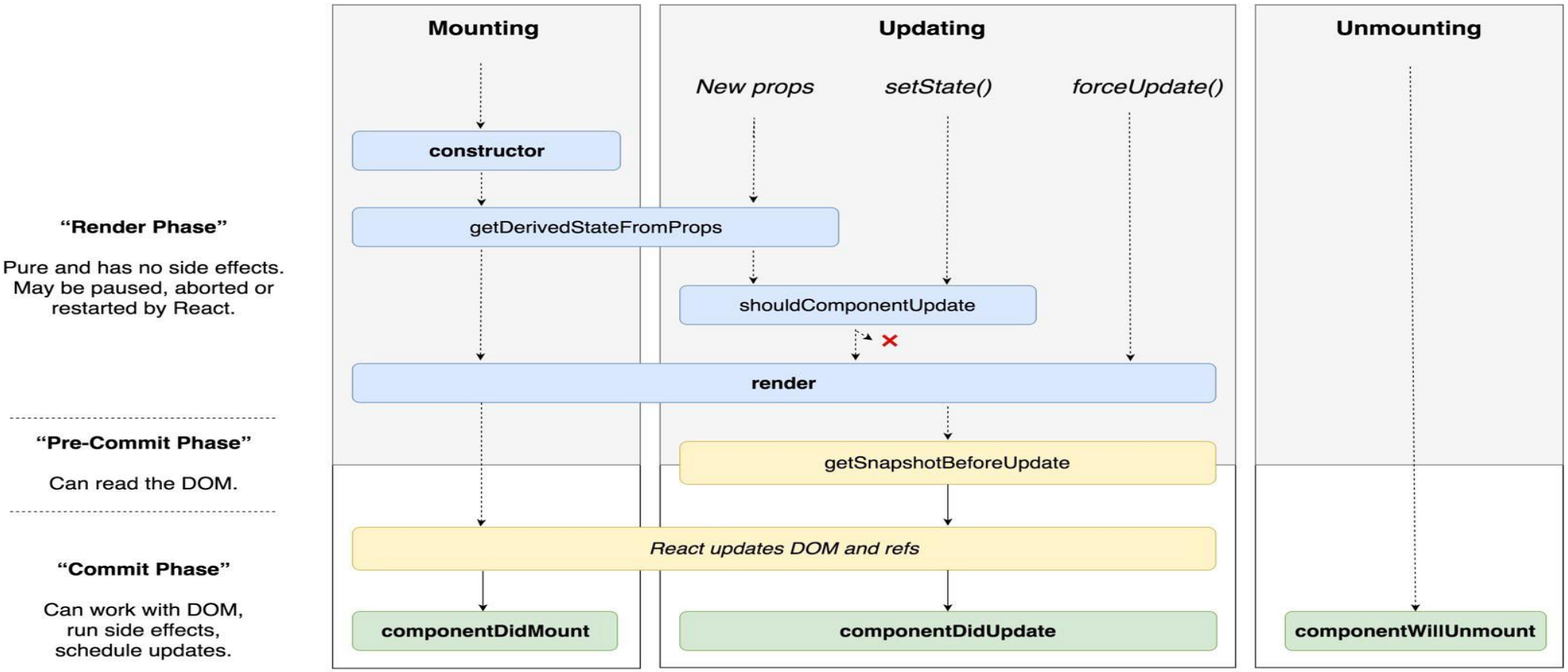
- 인자로 e(event)를 받아 속성 확인 가능
- e.target.name → input의 name, e.target.value → input에 들어온 실제 값
- input의 name과 value를 통해 state를 업데이트함

React Lifecycle



- Three main phasese: Mounting, Updating, Unmounting
- Mounting: component를 DOM에 넣는 단계
- Updating: component 업데이트
- Unmounting: component가 DOM에서 삭제

Mounting



- Mounting: component를 DOM에 넣는 단계
- 4개의 method 존재: constructor, getDerivedStateFromProps, render, componentDidMount
- render는 항상 호출되어야 함, 나머지는 선택적

Constructor

```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  render() {
    return (
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
    );
  }
}

ReactDOM.render(<Header />, document.getElementById('root'));
```

- Constructor: component 초기화 시 호출, state 값이 초기화됨
- super(props) 항상 제일 먼저 호출!

getDerivedStateFromProps

```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  static getDerivedStateFromProps(props, state) {
    return {favoritecolor: props.favcol };
  }
  render() {
    return (
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
    );
  }
}

ReactDOM.render(<Header favcol="yellow"/>, document.getElementById('root'));
```

- element가 DOM에 redner되기 직전에 호출
- props 값에 따라 state를 변화시킬 때 사용
- 위 예제에서는 favoritecolor 값이 최종적으로 yellow가 됨

render

```
class Header extends React.Component {
  render() {
    return (
      <h1>This is the content of the Header component</h1>
    );
  }
}

ReactDOM.render(<Header />, document.getElementById('root'));
```

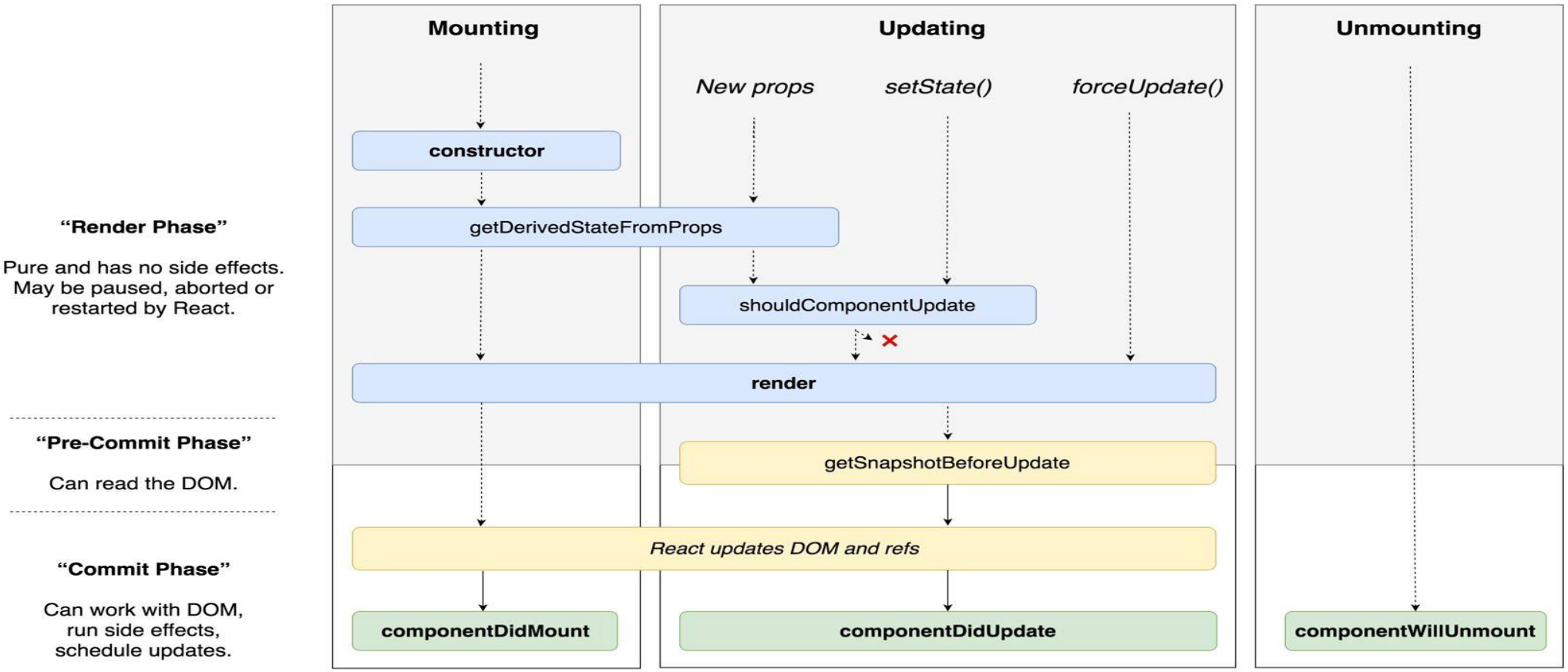
- 실제로 웹페이지를 화면에 출력
- 항상 필요!!!

componentDidMount

```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  componentDidMount() {
    setTimeout(() => {
      this.setState({favoritecolor: "yellow"})
    }, 1000)
  }
```

- component가 render된 후 호출
- 이미 DOM에 출력된 값을 바꿀 때 사용
- 위 예제에서는 화면에 favoritecolor 값이 red로 출력되고나서 1초 후 yellow로 값이 바뀜

Updating



- Updating: component 업데이트
- component의 state나 props 값이 바뀔때
- 다섯 개 method 존재: getDerivedStateFromProps, shouldComponentUpdate, render, getSnapshotBeforeUpdate, componentDidUpdate
- render는 항상 호출, 나머지는 선택적

shouldComponentUpdate

```
shouldComponentUpdate() {
  return false;
}
```

- state/props 값이 바꼈을 때 render를 새로 할지를 결정
- true/false 값을 반환 → true일 경우 render 진행, false일 경우 render x
- default value: true (명시하지 않으면 항상 새로 render)
- 현재 state, props 값에 따라 업데이트한 값을 화면에 바로 출력하고 싶지 않을 경우 사용

getSnapshotBeforeUpdate

```
getSnapshotBeforeUpdate(prevProps, prevState) {
  document.getElementById("div1").innerHTML =
  "Before the update, the favorite was " + prevState.favoritecolor;
}
```

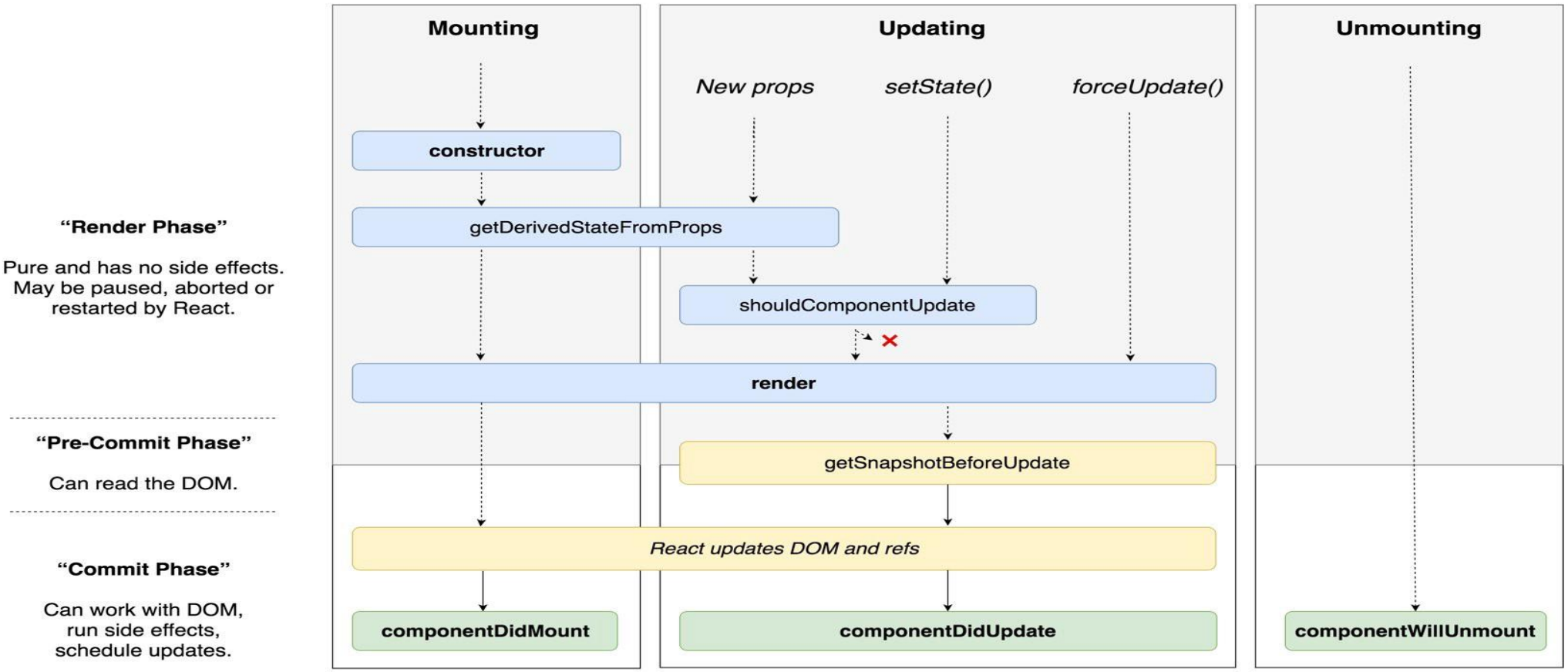
- 업데이트 이전의 props/state 값 access 가능
- 인자로 prevProps, prevState를 받음
- getSnapshotBeforeUpdate 함수 존재할 경우 항상 componentDidUpdate 함수 필요함

componentDidUpdate

```
componentDidUpdate() {
  document.getElementById("mydiv").innerHTML =
  "The updated favorite is " + this.state.favoritecolor;
}
```

- 업데이트 후 호출

Unmounting



- Unmounting: component가 DOM에서 삭제
- 한 개의 method만 존재: componentWillUnmount

componentWillUnmount

```
componentWillUnmount() {
  alert("The component named Header is about to be unmounted.");
}
```

- component가 DOM에서 삭제되기 직전에 호출

Reference

- https://www.w3schools.com/react/
- https://opentutorials.org/module/4058
- https://ko.reactjs.org/tutorial/tutorial.html

과제

- 처음에 화면에 숫자 5가 표시됨, 이후 1초마다 숫자 1씩 감소
- 증가 버튼 누르면 숫자 1씩 증가
- 숫자 10에서 버튼 눌러서 11이 되기 직전에 update 멈춤 → 숫자 변화 x
- 숫자가 음수가 되면 화면에 숫자 대신 “You Lost” 출력
- 기한: 다음주 일요일 오후 10시까지!

- hint) setInterval 함수 사용
- hint) constructor, componentDidMount, shouldComponentUpdate, componentDidUpdate, render 등 DOM Lifecycle Method들을 적절히 이용
- Final Project는 종강 쯤 다시 공지할 예정이며 기한은 2주 정도로 생각하고 있습니다!

한 학기 동안 수고 많으셨습니다~~

열심히 들어주신 분들 모두 감사드립니다!