Въведение в ReactJS

Какво е ReactJS?

Рамка за разработка или библиотека?

React е декларативна, ефективна и гъвкава JavaScript библиотека за изграждане на потребителски допирни точки

Основи

JSX (JavaScript XML)

JSX (JavaScript XML)

1. JSX е синтактично разширение

2. Позволява писането на псевдо-HTML

3. Логиката за изобразяване и структура живеят заедно в компонентите

```
<article>
  <h1>My First Component</h1>
  <ol>
    <li>Components: UI Building Blocks</li>
    <li>Defining a Component</li>
    <li>Using a Component</li>
  </ol>
</article>
```

```
<MyArticle>
  <MyTitle />
  <MyOrderedList />
</MyArticle>
```

React компоненти

React ни позволява да комбинираме HTML, CSS и JavaScript в зададени от потребителя преизползваеми модули наречени „компоненти“

Компоненти като класове или функции

Клас компоненти

1. Компоненти със състояние (Stateful)
2. Наследяват от базовия React Component и имплементират render() функция

```
class Counter extends Component {
  state = {
    age: 42,
  };

  render() {
    return (
      <p>You are {this.state.age}.</p>
    );
  }
}
```

Функционални компоненти

1. Компоненти без състояние (Stateless)
2. “Чисти” JavaScript функции

```
const Counter = () => {
  const [age, setAge] = useState(42);

  return (
    <p>You are {age}.</p>
  );
```

Реакт куки

Специални функции, които позволяват да се “закачим” за промяна в състоянието или жизнен цикъл на компонента

Вградени куки

useState

useState

```
const [age, setAge] = useState(42);
```

1. Позволява на компонента да „запомня“ въведена информация
2. Връща стойност на състоянието и функция за актуализиране
3. Задейства пререндериране на компонента при промяна на стойността

```
const Counter = () => {
  const [age, setAge] = useState(42);

  const onClick = () => {
    setAge(age + 1);
  }

  return (
    <p onClick={onClick}>You are {age}.</p>
  );
}
```

useEffect

useEffect в отговор на промените в състоянието

```
useEffect(() => console.log('Component mounted') , []);
```

1. Действа като наблюдател, който изпълнява код в отговор на промените в дадена стойност от списъка със зависимости
2. Изисква функция за изпълнение и списък от стойности за наблюдаване

```
const Counter = () => {
  const [age, setAge] = useState(42);

  useEffect(() => console.log('Age is updated!') , [age]);

  const onClick = () => {
    setAge(age + 1);
  };

  return (
    <p onClick={onClick}>You are {age}.</p>
  );
}
```

Когато е подаден празен списък,

useEffect се изпълнява само веднъж!

useEffect в отговор на жизнените цикли

1. Монтиране на компонента за DOM

```
useEffect(() => console.log('Component mounted') , []);
```

2. Демонтиране на компонента от DOM

```
useEffect(() => {
  console.log('Component mounted.');

  return () => console.log('Component unmounted.');
}, []);
```

Куки зададени от потребителя

Куки зададени от потребителя

1. Добро място за споделяне на логика между компонентите
2. Имат изолирано състояние от това на компонента
3. Както функционалните компоненти, трябва да са “чисти” функции

```
const useOnlineStatus = () => {
  const [isOnline, setIsOnline] = useState(true);

  useEffect(() => {
    const handleOnline = () => setIsOnline(true);
    const handleOffline = () => setIsOnline(false);

    window.addEventListener('online', handleOnline);
    window.addEventListener('offline', handleOffline);

    return () => {
      window.removeEventListener('online', handleOnline);
      window.removeEventListener('offline', handleOffline);
    };
  }, []);

  return isOnline;
}
```

Правила за писане

Правила за писане

1. Имената трябва да започват с use
2. Извикат се на най-високото ниво в тялото на функционален компонент или друга зададена от потребителя кука
3. Не се извикват в цикли, условия или вложени функции
4. Не могат да се извикват в клас компоненти (а как?)

Предаване на свойства между компоненти

```jsx
const ListItem = (props) => {
  return <li>{props.text}</li>;
};

const MyOrderedList = () => {
  const [items, ] = useState(['First item', 'Second item', 'Third item']);

  return (
    <ol>
      {items.map((item, index) => (
        <ListItem key={index} text={item} />
      ))}
    </ol>
  );
}
```

Четене на свойства

Подаване на свойства

Как можем да имаме достъп до данни от кука в клас компонент?

Функционален родител компонент взима данните от куката и подава свойства към клас компонента

Изобразяване на компоненти

Изобразяване на компоненти

1. Когато се изпълни функцията на компонент, ние казваме, че той се “рендерира” или “изобразява”
2. Предполагаме, че компонент може да се пререндерира/преобрази по всяко време

Имаме две възможности да задействаме преобразяване на компонент

1. Сменя се стойността на подадените на компонента свойства

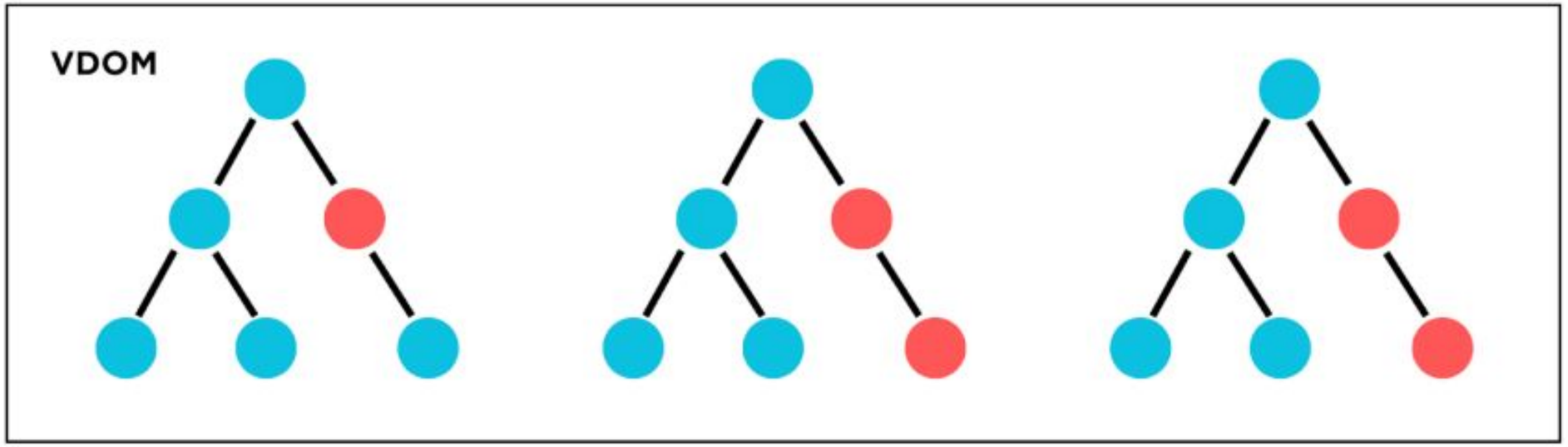
2. useState функцията за обновяване на състоянието е извикана

Виртуален DOM

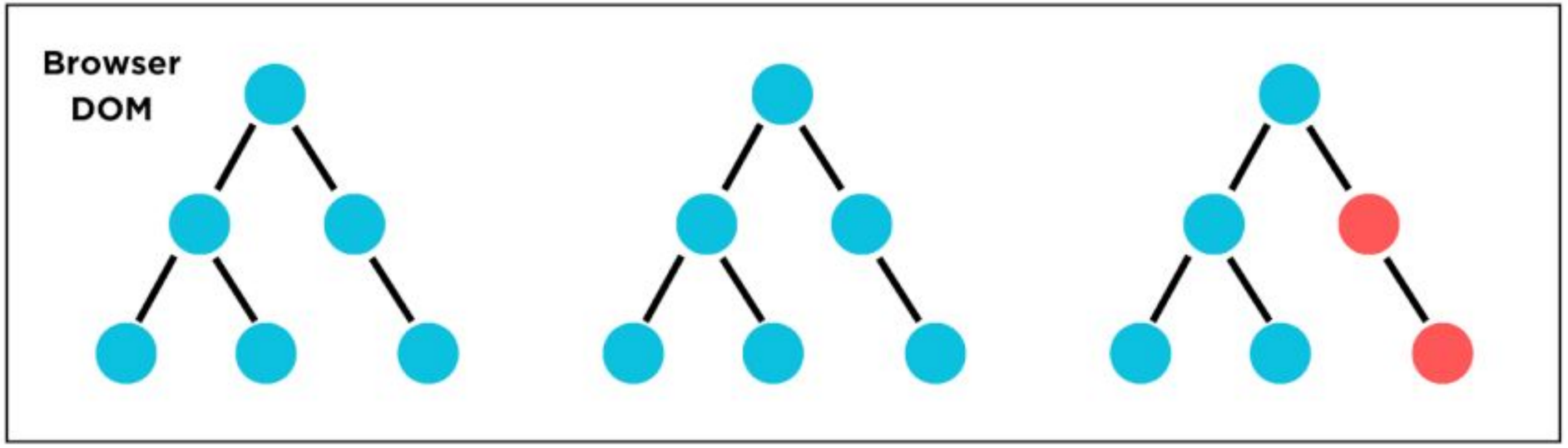
Виртуалният DOM (VDOM) е модел за идеално или „виртуално“ представяне на DOM елементите в паметта, който се синхронизира с „реалния“ DOM

Как работи?

1. Целият виртуален DOM се актуализира
2. Виртуалният DOM се сравнява с това как е изглеждал преди да се актуализира
3. React открива кои възли са се променили
4. Само променените възли се актуализират в реалния DOM



State change ⟶ Compute diff ⟶ Re-render



Как това е бърз процес?

Алгоритъм за сравняване

Най-съвременните алгоритми имат сложност от порядъка на O(n^3), където n е броят на елементите в дървото.

1000 елемента = 1 000 000 000 сравнения

React прилага евристичен O(n) алгоритъм,

базиран на две предположения

Реакт разчита на 2 принципа

1. Два елемента от различни видове ще произведат различни дървета

2. Разработчикът може да подскаже кои дъщерни елементи могат да бъдат стабилни в различните рендери чрез ключова опора (key prop)

```jsx
const ListItem = (props) => {
  return <li>{props.text}</li>;
};

const MyOrderedList = () => {
  const [items, ] = useState(['First item', 'Second item', 'Third item']);

  return (
    <ol>
      {items.map((item, index) => (
        <ListItem key={index} text={item} />
      ))}
    </ol>
  );
}
```

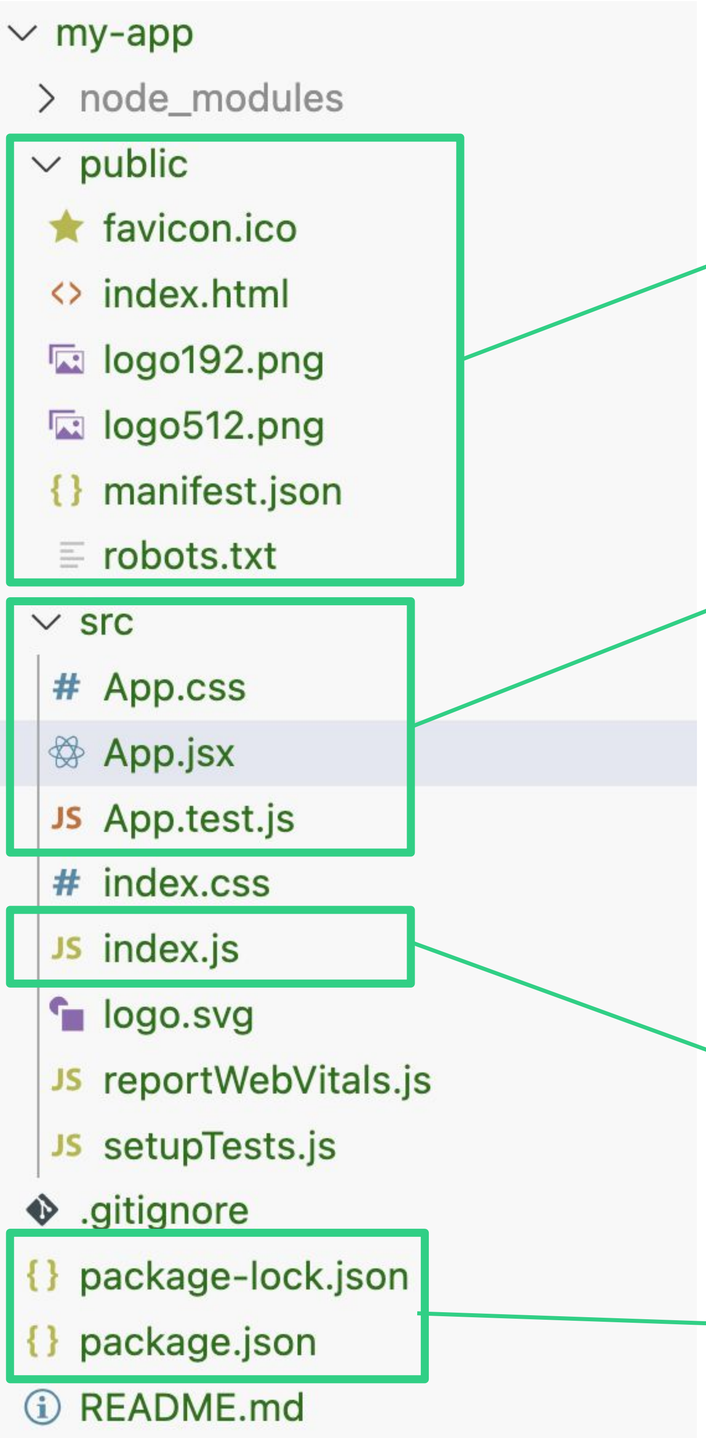
Първото ни React приложение

Първото ни React приложение

1. npm install create-react-app

2. create-react-app <име_на_директория>

3. npm start



/public - Главният HTML документ и други статични ресурси

/src - Генериран компонент за стартиране на разработка

index.js - React се свързва с DOM

jsons - конфигурация на npm зависимости

Благодаря!