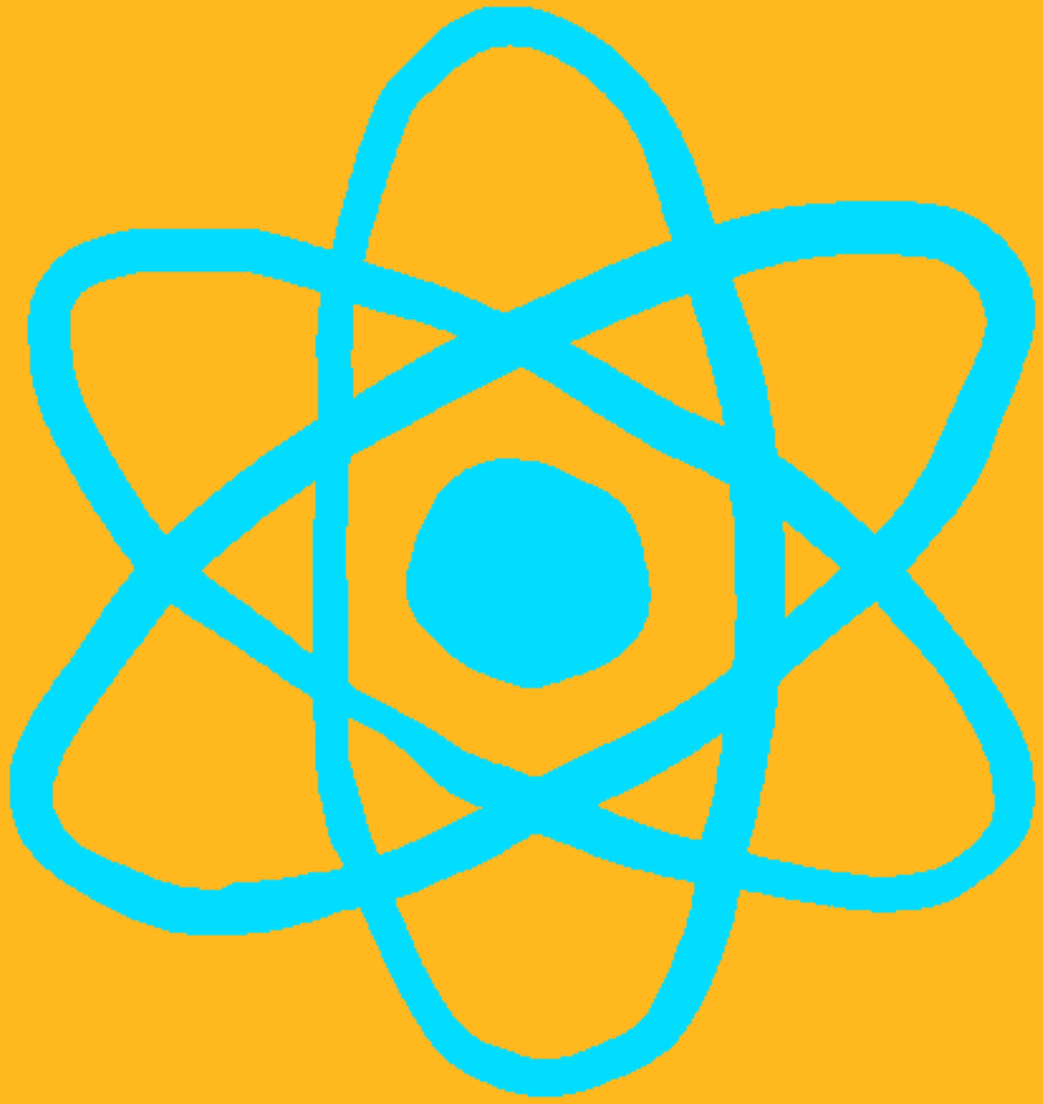
FRONTEND. REACT. ЛЕКЦІЯ 2

COMPONENTS. PROPS. STATE. EVENTS.



JSX

Розширення мови Javascript, що дозволяє використовувати HTML-подібний синтаксис.

Можна безпечно вставляти дані, введенні користувачем, в JSX:

```
const title = response.potentiallyMaliciousInput;
// Це безпечно:
const element = <h1>{title}</h1>;
```

За замовчуванням, React DOM екранує будь-які значення, що включені в JSX, перед їх рендерингом. Таким чином, це гарантує, що ви ніколи не включите в код те, що явно не написано у вашому додатку. Перед виводом все перетворюється на рядок. Це допомагає запобігти атакам XSS (cross-site-scripting).

JSX - це об'єкти по суті

Babel компілює JSX до викликів `React.createElement()`.

Ці два приклади ідентичні між собою:

КОМПОНЕНТИ

КОМПОНЕНТИ

- Перевикористання

КОМПОНЕНТИ

- Інкапсуляція

```
import { useState } from 'react';

export function Price(props) {
  const { price } = props;
  const [color, setColor] = useState('yellow');
  const [stateSmth] = useState('stateSmth');

  const style = {
    backgroundColor: color
  };

  const changeColor = () => {
    const randomColor = `#${Math.floor(Math.random()*16777215).toString(16)}`;
    setColor(randomColor);
  }

  return (<div style="{{backgroundColor:" color}}="">
    {price}
```

КОМПОНЕНТИ

- Читабельність

```
export function Product(props) {
const { tag, text, price } = props.product;
const [unit, setUnit] = useState("$");
const id = Math.random().toString(16).slice(2);

return (
  <div id="{id}">
        <title tag="{tag}">{text}</title>
        <price price="{price}" unit="{unit}/">
  </price></div>
);
 }
```

КОМПОНЕНТИ

- Тестування

ТИПИ КОМПОНЕНТІВ

ТИПИ КОМПОНЕНТІВ

- Stateless

```
export function Count(props) {
        const { count } = props;

        return (<p>Наявна кількість: {count}</p>);
  }
 }
```

ТИПИ КОМПОНЕНТІВ

- Stateful

```
export function Product(props) {
const { tag, text, price } = props.product;
const [unit, setUnit] = useState("$");
const id = Math.random().toString(16).slice(2);

return (
  <div id="{id}">
        <title tag="{tag}">{text}</title>
        <price price="{price}" unit="{unit}/">
  </price></div>
);
 }
```

ВНУТРІШНІЙ СТАН

ВНУТРІШНІЙ СТАН

ВНУТРІШНІЙ СТАН

- Лише стейт робить ререндер

ВНУТРІШНІЙ СТАН

- Лише стейт робить рендер
- State повинен використовуватись при рендері сторінки

ВНУТРІШНІЙ СТАН

- Лише стейт робить ререндер
- State повинен використовуватись при рендері сторінки
- Props не змінюється, на відміну від стейт

Як створити компонент зі станом

```
1  import { useState } from 'react';
2
3  export function Price(props) {
4    const { price } = props;
5    const [color, setColor] = useState('yellow');
6    const [stateSmth] = useState('stateSmth');
7
8    const changeColor = () => {
9      const randomColor = `#${Math.floor(Math.random()*16777215).toString(16)}`;
10     setColor(randomColor);
11   }
12
13   return (<div style="{{backgroundColor:" color}}="">
14     {price}
15     <button onclick="{changeColor}">Change color {color} with {stateSmth}</but
16     </div>);
17 }
```

Як створити компонент зі станом

```
1  import { useState } from 'react';
2
3  export function Price(props) {
4    const { price } = props;
5    const [color, setColor] = useState('yellow');
6    const [stateSmth] = useState('stateSmth');
7
8    const changeColor = () => {
9      const randomColor = `#${Math.floor(Math.random()*16777215).toString(16)}`;
10     setColor(randomColor);
11   }
12
13   return (<div style="{{backgroundColor:" color}}="">
14     {price}
15     <button onclick="{changeColor}">Change color {color} with {stateSmth}</but
16     </div>);
17 }
```

ЯК СТВОРИТИ КОМПОНЕНТ ЗІ СТАНОМ

```
1  import { useState } from 'react';
2
3  export function Price(props) {
4    const { price } = props;
5    const [color, setColor] = useState('yellow');
6    const [stateSmth] = useState('stateSmth');
7
8    const changeColor = () => {
9      const randomColor = `#${Math.floor(Math.random()*16777215).toString(16)}`;
10     setColor(randomColor);
11   }
12
13   return (<div style="{{backgroundColor:" color}}="">
14     {price}
15     <button onclick="{changeColor}">Change color {color} with {stateSmth}</but
16     </div>);
17 }
```

ЯК СТВОРИТИ КОМПОНЕНТ ЗІ СТАНОМ

```
1  import { useState } from 'react';
2
3  export function Price(props) {
4    const { price } = props;
5    const [color, setColor] = useState('yellow');
6    const [stateSmth] = useState('stateSmth');
7
8    const changeColor = () => {
9      const randomColor = `#${Math.floor(Math.random()*16777215).toString(16)}`;
10     setColor(randomColor);
11   }
12
13   return (<div style="{{backgroundColor:" color}}="">
14     {price}
15     <button onclick="{changeColor}">Change color {color} with {stateSmth}</but
16     </div>);
17 }
18                                                    }
```

ЯК СТВОРИТИ КОМПОНЕНТ ЗІ СТАНОМ

```
1  import { useState } from 'react';
2
3  export function Price(props) {
4    const { price } = props;
5    const [color, setColor] = useState('yellow');
6    const [stateSmth] = useState('stateSmth');
7
8    const changeColor = () => {
9      const randomColor = `#${Math.floor(Math.random()*16777215).toString(16)}`;
10     setColor(randomColor);
11   }
12
13   return (<div style="{{backgroundColor:" color}}="">
14     {price}
15     <button onclick="{changeColor}">Change color {color} with {stateSmth}</but
16     </div>);
17 }
18                                                  }
```

ОСОБЛИВОСТІ STATE

ОСОБЛИВОСТІ STATE

- Непотрібно присвоювати значення напряму

ОСОБЛИВОСТІ STATE

- Непотрібно присвоювати значення напряму
- Не потрібно змінювати стейт в асинхронних функціях.

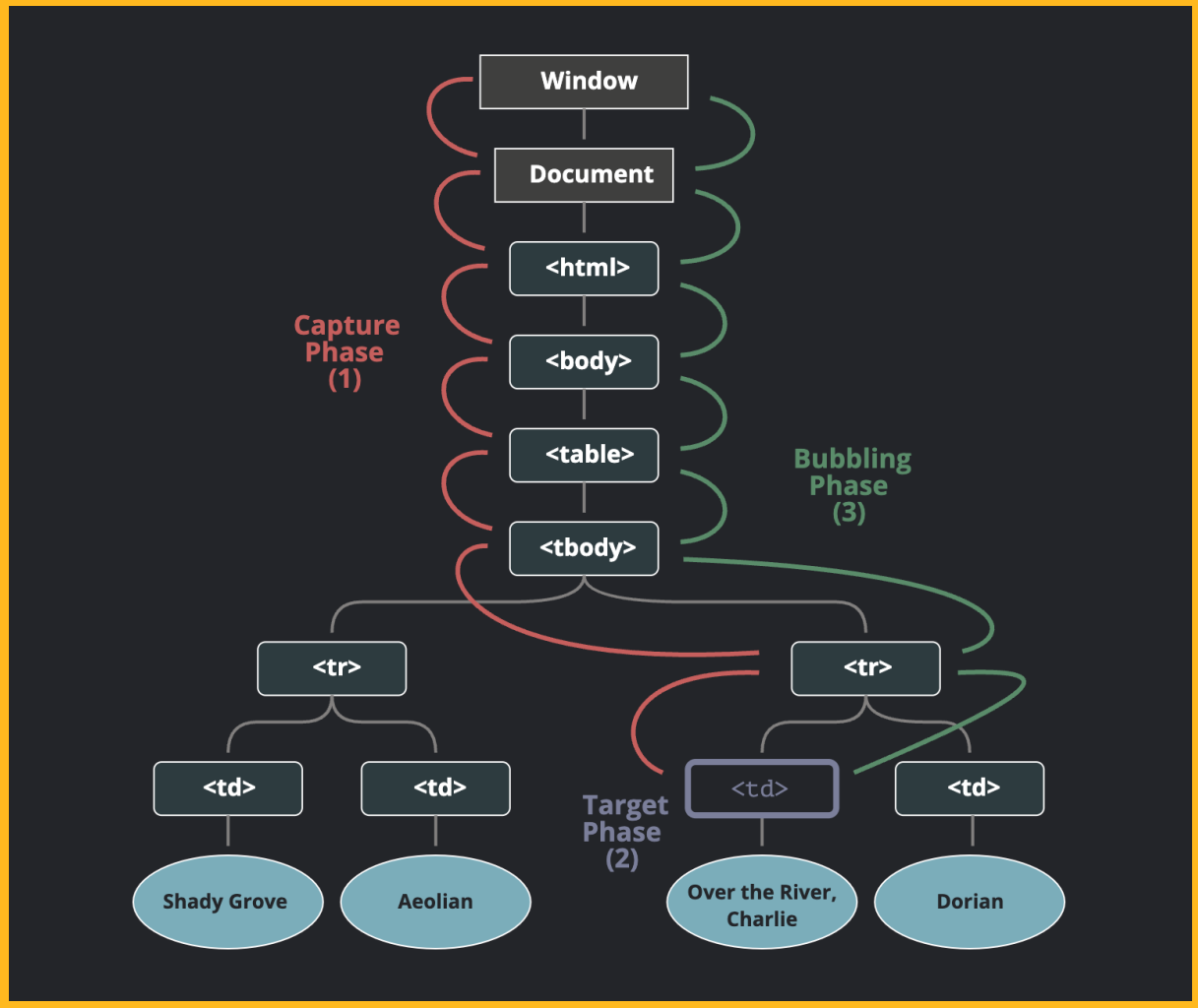
ОСОБЛИВОСТІ STATE

- Непотрібно присвоювати значення напряму
- Не потрібно змінювати стейт в асинхронних функціях.
- Реакт групує виклики setState

```
1  { useState } from 'react';
2
3    export function Price(props) {
4   const { price } = props;
5   const [color, setColor] = useState('yellow');
6   const [stateSmth] = useState('stateSmth');
7
8   const changeColor = () => {
9     const randomColor = `#${Math.floor(Math.random()*16777215).toString(16)}`;
10    setColor(randomColor);
11    setColor(randomColor);
12  }
13
14  return (<div style="{{backgroundColor:" color}}="">
15    {price}
16    <button onclick="{changeColor}">Change color {color} with {stateSmth}</but
17    </div>);
```

ЯК ПЕРЕДАТИ ДАННІ ДО КОМПОНЕНТА?

PROPS / EVENTS



ОСОБЛИВОСТІ PROPS

ОСОБЛИВОСТІ PROPS

- Лише для читання

ОСОБЛИВОСТІ PROPS

- Лише для читання
- Працюють, як чисті функції

Як передати props

ЯК ПЕРЕДАТИ PROPS

- За допомогою атрибутів

Як передати props

- За допомогою атрибутів
- Як вкладені компоненти

Як передати props

- За допомогою атрибутів
- Як вкладені компоненти
- Строка, число, булеве, масив, об'єкт, функція, інший jsx елемент

PROPS.CHILDREN

```
import { Children, useState } from 'react';

export function Title(props) {
const { tag } = props;

return (
        <>
        {(tag === 'h1')
                ? <h1>{props.children}</h1>
                : <p>{props.children}</p>
        }

);
}
```

```
export function Product(props) {
        const { tag, text, price, count } = props.product;
        const [unit, setUnit] = useState("$");
        const id = Math.random().toString(16).slice(2);

        return (
          <div id="{id}">
                <title tag="{tag}">{text}</title>
                <price price="{price}" unit="{unit}/">
                <count count="{count}"></count>
          </price></div>
```

```
        );
    }
```

ПОДІЇ

ОСОБЛИВОСТІ ПОДІЙ

ОСОБЛИВОСТІ ПОДІЙ

- Прив’язка у атрибутах компонентів

ОСОБЛИВОСТІ ПОДІЙ

- Прив'язка у атрибутах компонентів
- CamelCase нотація

ОСОБЛИВОСТІ ПОДІЙ

- Прив'язка у атрибутах компонентів
- CamelCase нотація
- Передача функції, а не рядок

ОСОБЛИВОСТІ ПОДІЙ

- Прив'язка у атрибутах компонентів
- CamelCase нотація
- Передача функції, а не рядок
- PreventDefault

SYNTHETICEVENT

SYNTHETICEVENT

- Кросбраузерна обгортка над нативною подією

SYNTHETICEVENT

- Кросбраузерна обгортка над нативною подією
- nativeEvent

SYNTHETICEVENT

- Кросбраузерна обгортка над нативною подією
- nativeEvent
- Пул подій, persist

SYNTHETICEVENT

- Кросбраузерна обгортка над нативною подією
- nativeEvent
- Пул подій, persist
- Документація https://react.dev/reference/react-dom/components/common

WO? WO?