05. Условный рендеринг

" ИЛИ - || " запинается на правде , "

И - && " запинается на лжи...

[ИЛИ](https://learn.javascript.ru/logical-operators" \l "ili-nahodit-pervoe-istinnoeznachenie) **[«||»](https://learn.javascript.ru/logical-operators" \l "ili-nahodit-pervoe-istinnoeznachenie)** [находит](https://learn.javascript.ru/logical-operators" \l "ili-nahodit-pervoe-istinnoeznachenie) **[первое истинное](https://learn.javascript.ru/logical-operators" \l "ili-nahodit-pervoe-istinnoeznachenie)**[или](https://learn.javascript.ru/logical-operators" \l "ili-nahodit-pervoe-istinnoeznachenie) **[последнее ложное (если все ложные)](https://learn.javascript.ru/logical-operators" \l "ili-nahodit-pervoe-istinnoeznachenie)** [значение](https://learn.javascript.ru/logical-operators" \l "ili-nahodit-pervoe-istinnoeznachenie) и возвращает его

[И](https://learn.javascript.ru/logical-operators" \l "i-nahodit-pervoe-lozhnoeznachenie) **[«&&»](https://learn.javascript.ru/logical-operators" \l "i-nahodit-pervoe-lozhnoeznachenie)** [находит](https://learn.javascript.ru/logical-operators" \l "i-nahodit-pervoe-lozhnoeznachenie) **[первое ложное](https://learn.javascript.ru/logical-operators" \l "i-nahodit-pervoe-lozhnoeznachenie)** [или](https://learn.javascript.ru/logical-operators" \l "i-nahodit-pervoe-lozhnoeznachenie) **[последнее истинное (если все истинные)](https://learn.javascript.ru/logical-operators" \l "i-nahodit-pervoe-lozhnoeznachenie)**[значение](https://learn.javascript.ru/logical-operators" \l "i-nahodit-pervoe-lozhnoeznachenie) и возвращает его

**В JS есть 5 псевдоложных значений:**

* undefined
* null
* 0
* пустая строка
* NaN

**Псевдоистинные значения, кроме исключающих псевдоложные:**

-Любой массив, даже пустой

-любая строка не пустая

Так не пишем, только для примера. Например, есть компонент, и в зависимости от true/false нам нужно отобразить компонент либо нет. Если мы запишем таким образом, используя логические операторы:

export function Accordion(props: AccordionPropsType) {  
 return (  
 <div>  
 <AccordionTitle titleValue={props.title}/>  
 {true && <AccordionBody/>}  
 </div>  
 )  
}

То сработает последний true, компонент <AccordionBody/> отобразится.

Если напишем так:

export function Accordion(props: AccordionPropsType) {  
 return (  
 <div>  
 <AccordionTitle titleValue={props.title}/>  
 {false && <AccordionBody/>}  
 </div>  
 )  
}

То сработает первый false и компонент <AccordionBody/> **не** отобразится. Используя это, запишем условную конструкцию так (первое условие **даст true**):

export function Accordion(props: AccordionPropsType) {  
 return (  
 <div>  
 <AccordionTitle titleValue={props.title}/>  
 {props.collapsed === false && <AccordionBody/>}  
 </div>  
 )  
}

или сделаем simplify

export function Accordion(props: AccordionPropsType) {  
 return (  
 <div>  
 <AccordionTitle titleValue={props.title}/>  
 {!props.collapsed && <AccordionBody/>}  
 </div>  
 )  
}

Таким же образом можно отрефакторить **<Star/>** отображение пяти звездочек:

function App() {  
 return (  
 <div className="App">  
 <PageTitle title={"This is app component"}/>  
 <PageTitle title={"My friends"}/>  
 Article 1  
 <Rating value={2}/>  
 <Accordion title={"Menu"} collapsed={true}/>  
 <Accordion title={"Users"} collapsed={false}/>  
 Article 2  
 <Rating value={1}/>  
 <Rating value={2}/>  
 <Rating value={3}/>  
 <Rating value={4}/>  
 <Rating value={5}/>  
  
 </div>  
 );  
}

export function Rating(props: RatingPropsType) {  
  
 return (  
 <div>  
 <Star selected={props.value > 0}/>  
 <Star selected={props.value > 1}/>  
 <Star selected={props.value > 2}/>  
 <Star selected={props.value > 3}/>  
 <Star selected={props.value > 4}/>  
 </div>)  
}  
  
  
function Star(props: StarPropsType) {  
 if (props.selected === true) {  
 return (<span><b>star </b></span>)  
 } else {  
 return (  
 <span>star </span>  
 )  
 }  
}

06. Условный рендеринг

в Ract не является плохим тоном использовать inline-стили, этим можем воспользоваться.

- Создается объект в самой компоненте, стили с ДЕФИСОМ прописываются camelCase.

- Свойства прописываются через ‘,’ как в объекте,

- Значения свойств берутся в “кавычки”

import classes from './OnOf.module.css';  
  
type OnOffPropsType = {  
 on: boolean  
}  
  
export function OnOff(props: OnOffPropsType) {  
  
 const onStyle = {  
 width: '30px',  
 height: '30px',  
 border: '1px solid black'  
 }  
 const offStyle = {  
 width: '30px',  
 height: '30px',  
 border: '1px solid black'  
 }  
 const indicatorStyle = {  
 width: '30px',  
 height: '30px',  
 borderRadius: '30px',  
 border: '1px solid black'  
 }  
  
 return (  
 <div className={classes.wrapper}>  
 <div style={onStyle}>On</div>  
 <div style={offStyle}>Off</div>  
 <div style={indicatorStyle}>+</div>  
 </div>  
 )  
  
}

В объекте стилей мы можем прописать условия, в данном случае тернарное выражение:

backgroundColor: props.on === true? 'green' : 'red'

Итого получим:

import classes from './OnOf.module.css';  
  
type OnOffPropsType = {  
 on: boolean  
}  
  
export function OnOff(props: OnOffPropsType) {  
  
 const onStyle = {  
 width: '30px',  
 height: '30px',  
 border: '1px solid black',  
 **backgroundColor: props.on === true? 'green' : 'white'**  
 }  
 const offStyle = {  
 width: '30px',  
 height: '30px',  
 border: '1px solid black',  
 **backgroundColor: props.on === false? 'red' : 'white'**  
 }  
 const indicatorStyle = {  
 width: '30px',  
 height: '30px',  
 borderRadius: '30px',  
 border: '1px solid black',  
 **backgroundColor: props.on === false? 'red' : 'green'**  
 }  
  
 return (  
 <div className={classes.wrapper}>  
 <div style={onStyle}>On</div>  
 <div style={offStyle}>Off</div>  
 <div style={indicatorStyle}>+</div>  
 </div>  
 )  
  
}

В строках

backgroundColor: props.on === true? 'green' : 'white'

можем сократить запись

backgroundColor: props.on? 'green' : 'white'

backgroundColor: !props.on? 'red' : 'white'

Можно вынести часть стилей в CSS, часть оставить в inline:

import classes from './OnOf.module.css';  
  
type OnOffPropsType = {  
 on: boolean  
}  
  
export function OnOff(props: OnOffPropsType) {  
  
 const onStyle = {  
 backgroundColor: props.on ? 'green' : 'white'  
 }  
 const offStyle = {  
 backgroundColor: !props.on ? 'red' : 'white'  
 }  
 const indicatorStyle = {  
 backgroundColor: !props.on ? 'red' : 'green'  
 }  
  
 return (  
 <div className={classes.wrapper}>  
 <div style={onStyle} className={classes.button}>On  
 </div>  
 <div style={offStyle} className={classes.button}>Off  
 </div>  
 <div style={indicatorStyle} className={classes.circle}>+</div>  
 </div>  
 )  
}