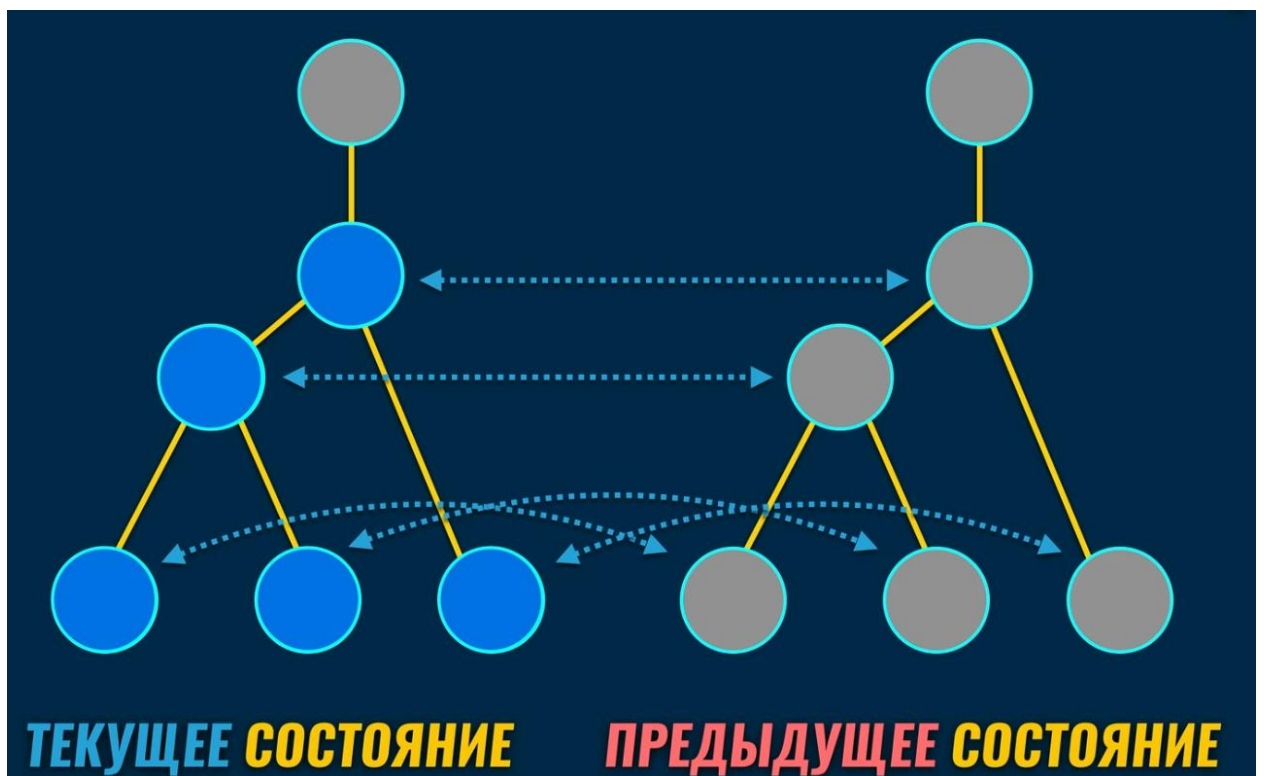
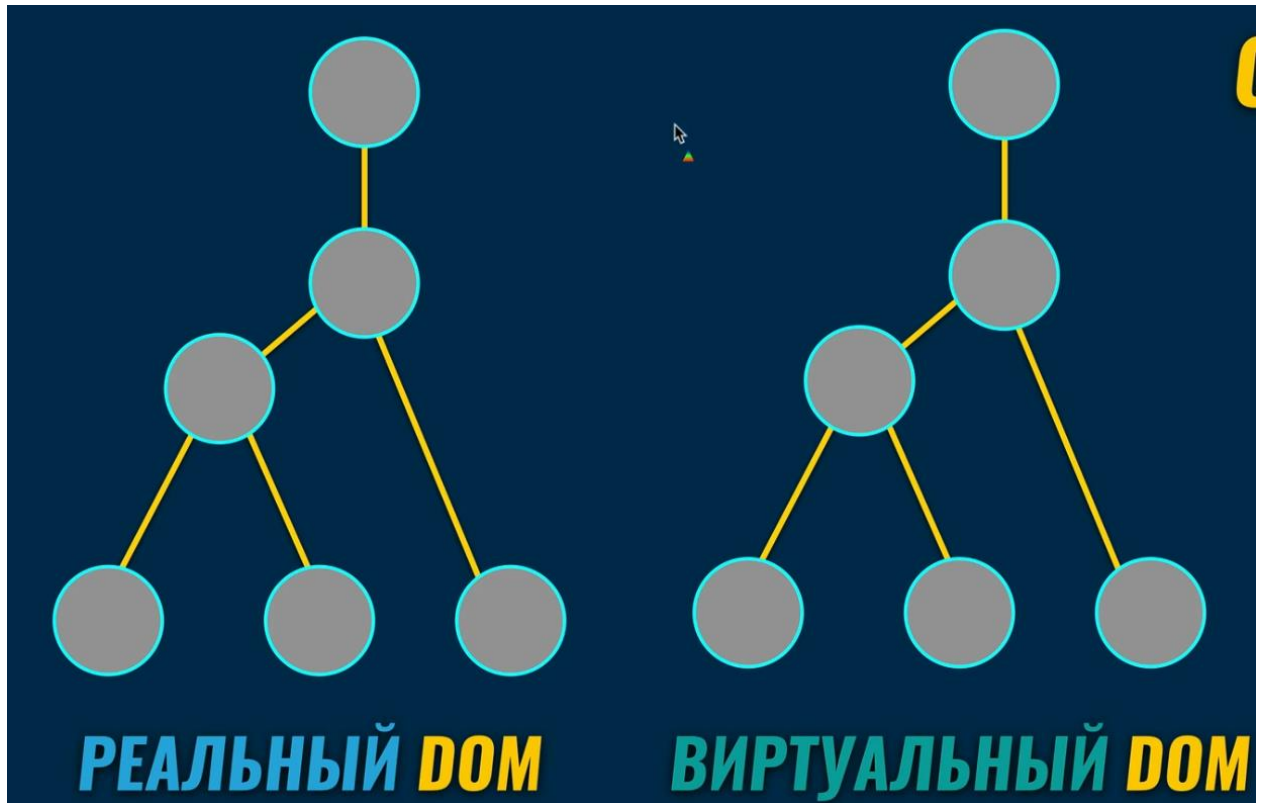
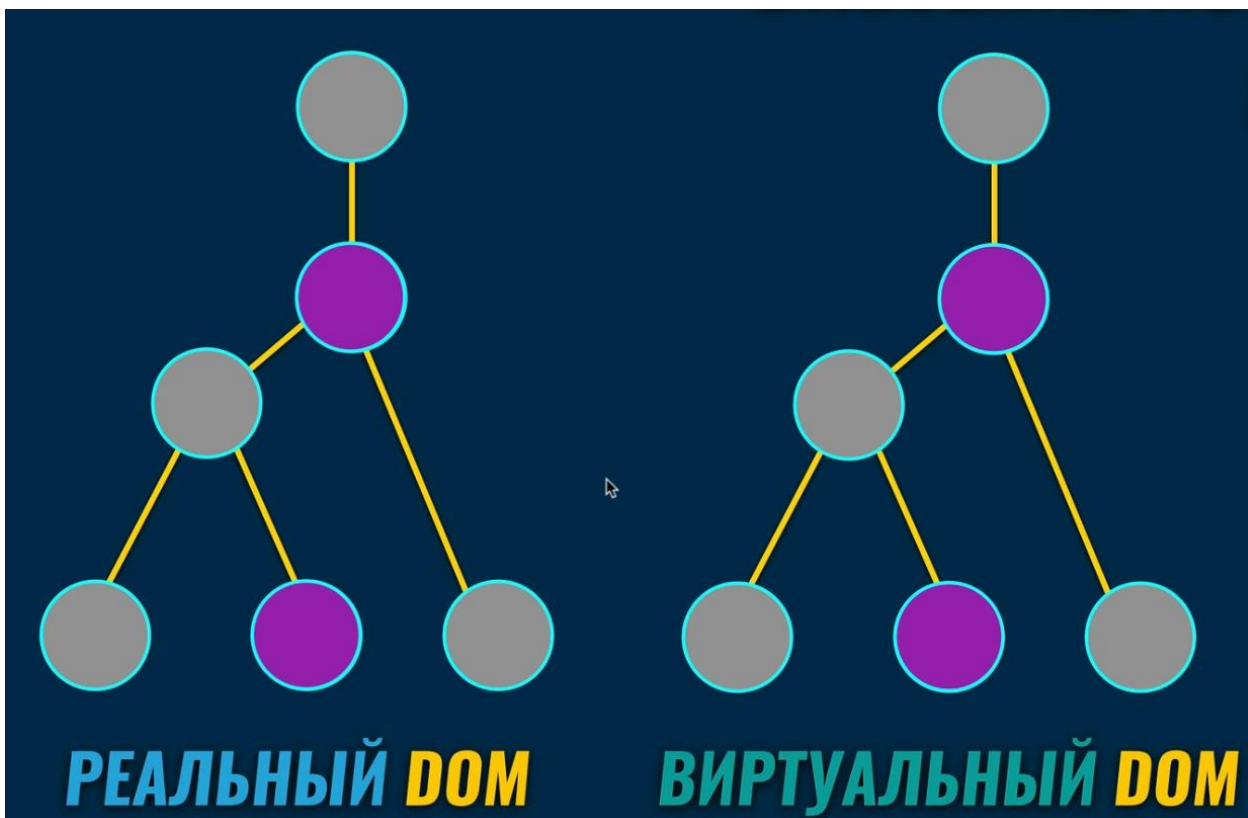
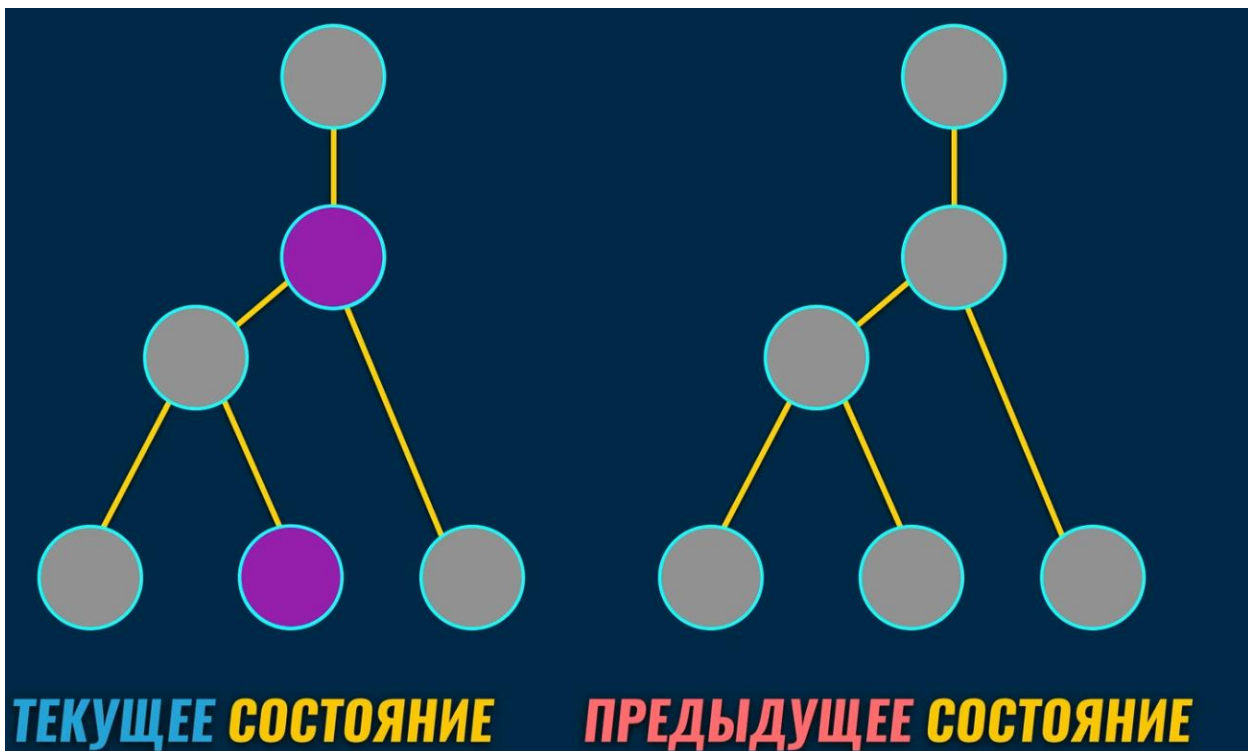


Основы React и взаимодействие с DOM





КЛЮЧЕВЫЕ ПОНЯТИЯ В REACT

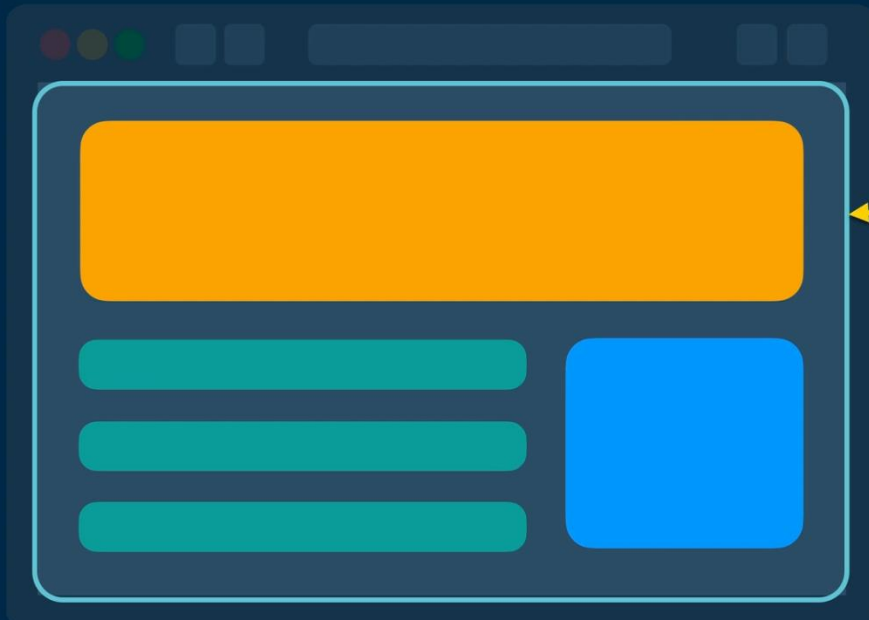
КОМПОНЕНТЫ
components

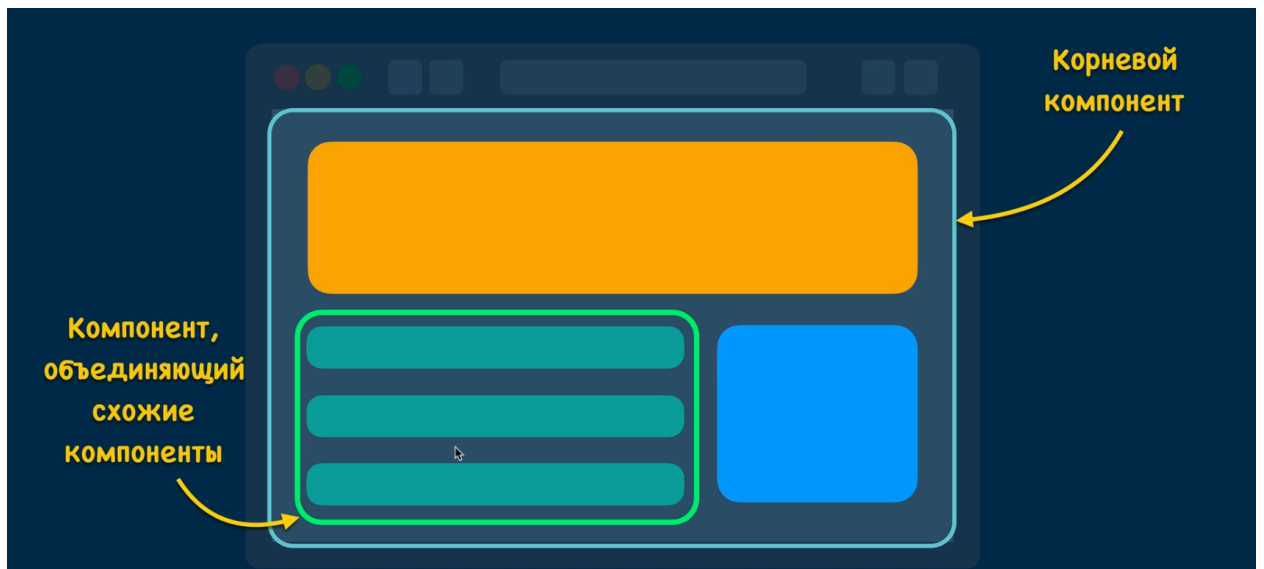
JSX
JavaScript Syntax
Extension

СВОЙСТВА
props

СОСТОЯНИЕ
state

ИЕРАРХИЯ КОМПОНЕНТОВ

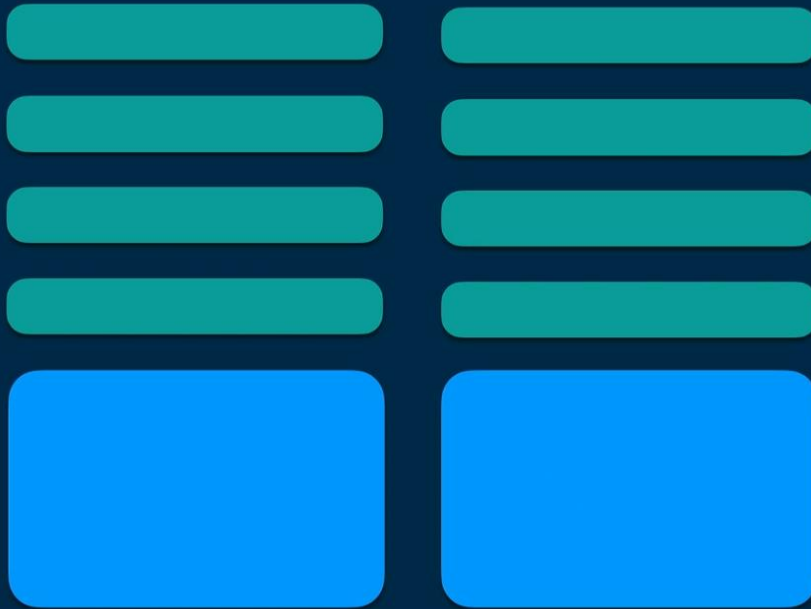




ВИДИМЫЕ КОМПОНЕНТЫ



ПЕРЕИСПОЛЬЗОВАНИЕ КОМПОНЕНТОВ



АНАТОМИЯ КОМПОНЕНТА REACT

КОМПОНЕНТ REACT

HTML

CSS

JavaScript

ФУНКЦИОНАЛЬНЫЕ КОМПОНЕНТЫ

```
function HelloWorld() {  
  return <h1>Hello World</h1>  
}
```

ФУНКЦИОНАЛЬНЫЕ КОМПОНЕНТЫ

```
const HelloWorld = () => {  
  return <h1>Hello World</h1>  
}
```

КЛАССОВЫЕ КОМПОНЕНТЫ

```
class HelloWorld extends Component {  
  render() {  
    return <h1>Hello World</h1>  
  }  
}
```


JAVASCRIPT SYNTAX EXTENSION

(JSX)

```
<Card
  style={{
    backgroundColor: `rgb(${initialColor}, ${opacity})`,
  }}
  className="m-2"
>
  <Card.Img
    variant="top"
    style={
      imageLoaded
        ? { opacity: 1, transition: 'opacity 2s ease-in-out' }
        : { opacity: 0 }
    }
    src={image}
    alt={title}
    onLoad={() => setImageLoaded(true)}
  />
</Card>
```

Похоже на CSS

Похоже на HTML

JavaScript



JSX - СИНТАКСИЧЕСКАЯ НАДСТРОЙКА НАД JS

```
React.createElement(Card, {
  style: {
    backgroundColor: `rgb(${initialColor}, ${opacity})`,
  },
  className: "m-2"
}, React.createElement(Card.Img, {
  variant: "top",
  style: imageLoaded ? {
    opacity: 1,
    transition: 'opacity 2s ease-in-out'
  } : {
    opacity: 0
  },
  src: image,
  alt: title,
  onLoad: () => setImageLoaded(true)
}));
```

ПОЛЬЗОВАТЕЛЬСКИЕ И ВСТРОЕННЫЕ КОМПОНЕНТЫ В JSX

Встроенный
компонент `h1`

```
return (  
  <div>  
    <h1>JSX в React</h1>  
    <p>JSX похож на HTML</p>  
    <Footer text={text} />  
  </div>  
)
```

Пользовательский
компонент `Footer`

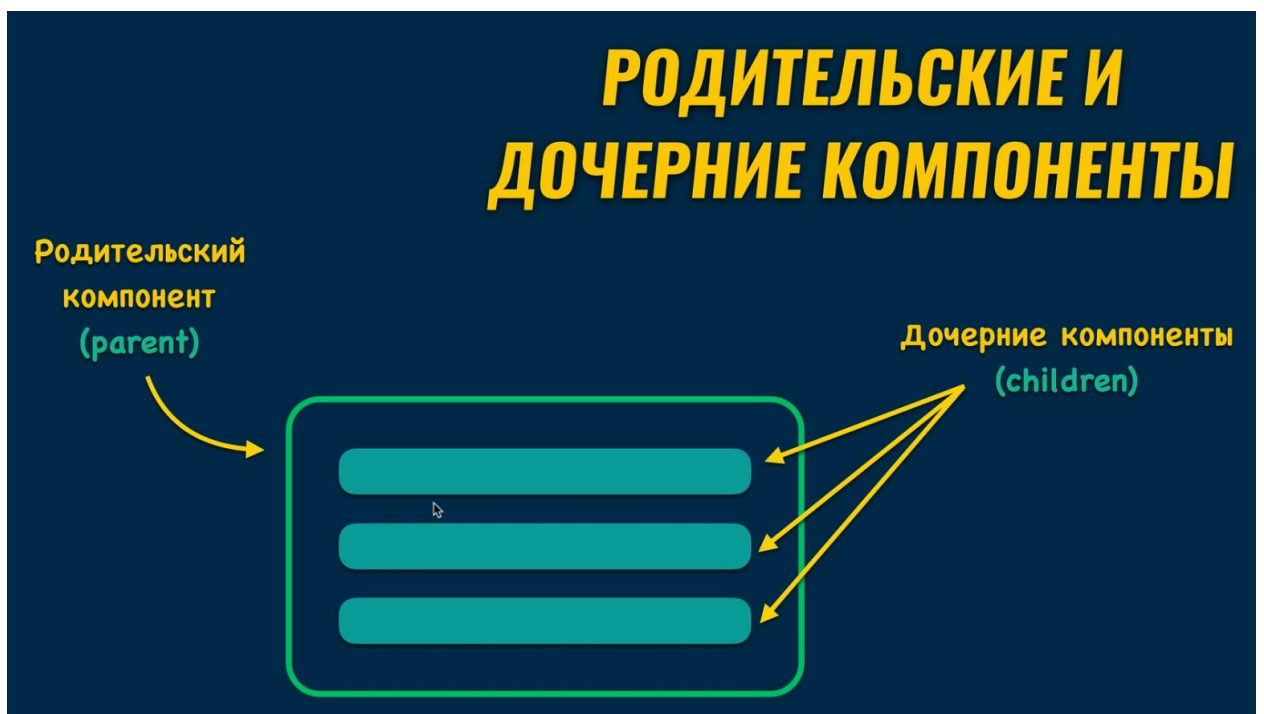
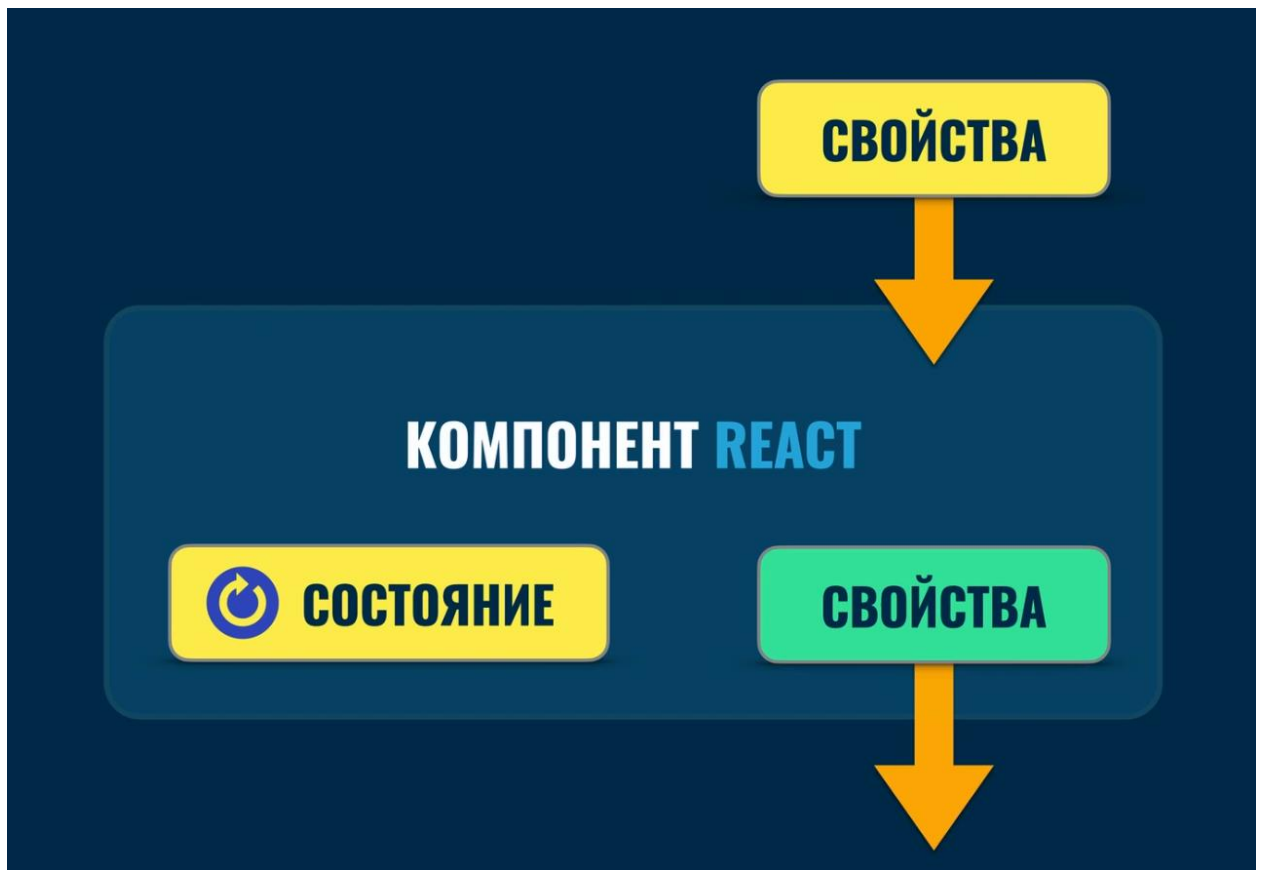
JSX ДОЛЖЕН ИМЕТЬ ОДИН КОРНЕВОЙ ЭЛЕМЕНТ

```
return (  
  <div>  
    <h1>JSX в React</h1>  
    <p>JSX похож на HTML</p>  
    <Footer text={text} />  
  </div>  
)
```

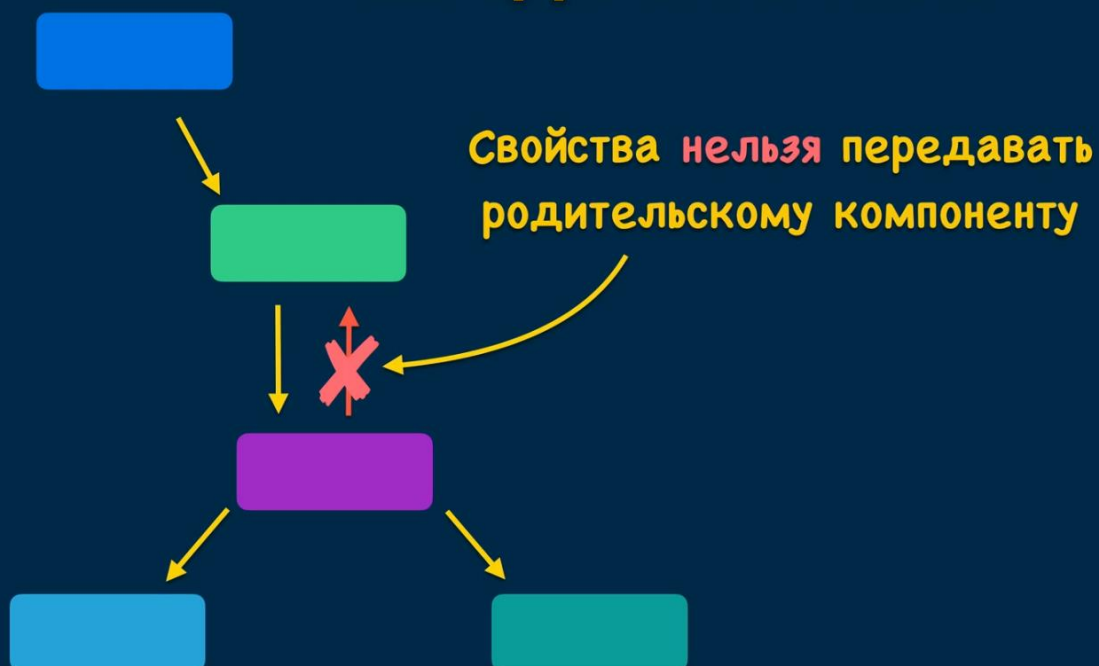
Валидный JSX

```
return (  
  <h1>JSX в React</h1>  
  <p>JSX похож на HTML</p>  
)
```

Невалидный
JSX



ПЕРЕДАЧА СВОЙСТВ

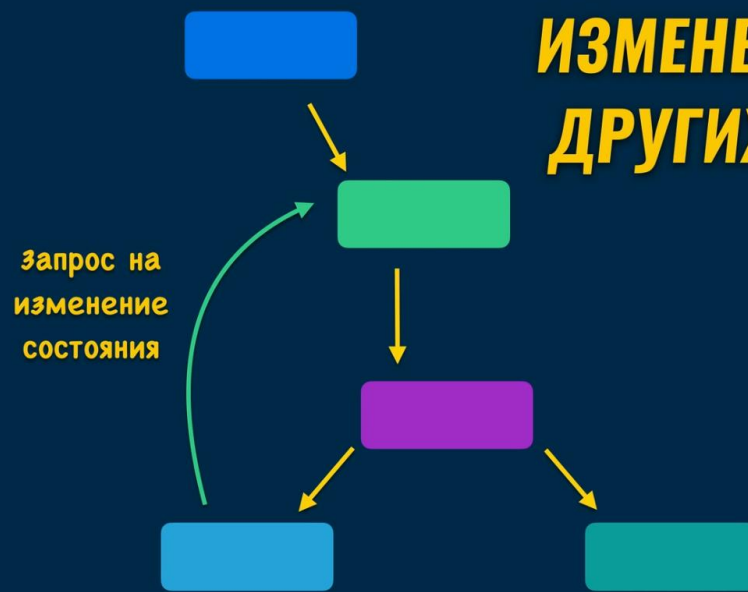


**КОМПОНЕНТ НЕ ДОЛЖЕН
ИЗМЕНЯТЬ
СОБСТВЕННЫЕ
СВОЙСТВА**

**КОМПОНЕНТ МОЖЕТ
ИЗМЕНЯТЬ
СОБСТВЕННОЕ
СОСТОЯНИЕ**

**КОМПОНЕНТ НЕ МОЖЕТ
ИЗМЕНЯТЬ СОСТОЯНИЕ
ДРУГИХ
КОМПОНЕНТОВ**

**НО МОЖНО ВЛИЯТЬ НА
ИЗМЕНЕНИЕ СОСТОЯНИЯ
ДРУГИХ КОМПОНЕНТОВ**



**МОЖНО ПЕРЕДАВАТЬ ЧАСТЬ СВОИХ
СВОЙСТВ И СОСТОЯНИЯ ДОЧЕРНИМ
КОМПОНЕНТАМ
В ВИДЕ СВОЙСТВ**

КОМПОНЕНТ ПОДЛЕЖИТ РЕРЕНДЕРИНГУ ПРИ ИЗМЕНЕНИИ СВОЙСТВ ИЛИ СОСТОЯНИЯ

Сам React этого не делает. Это выполняет другая библиотека:
ReactDOM – для Веб-приложений и React Native (RN) – для
мобильных приложений

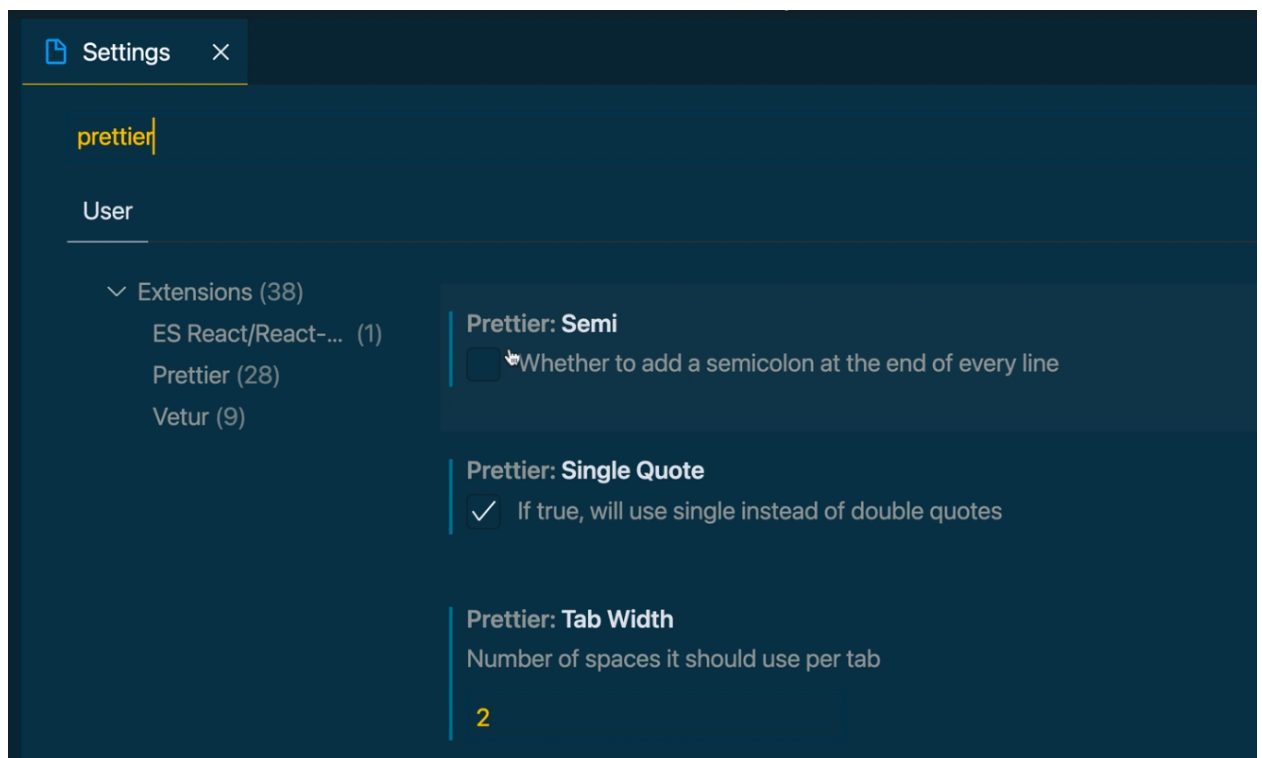
РЕАКТ HOOKS ПОЗВОЛЯЮТ УПРАВЛЯТЬ СОСТОЯНИЕМ В ФУНКЦИОНАЛЬНЫХ КОМПОНЕНТАХ

ОСНОВНЫЕ ХУКИ REACT

useState

useEffect

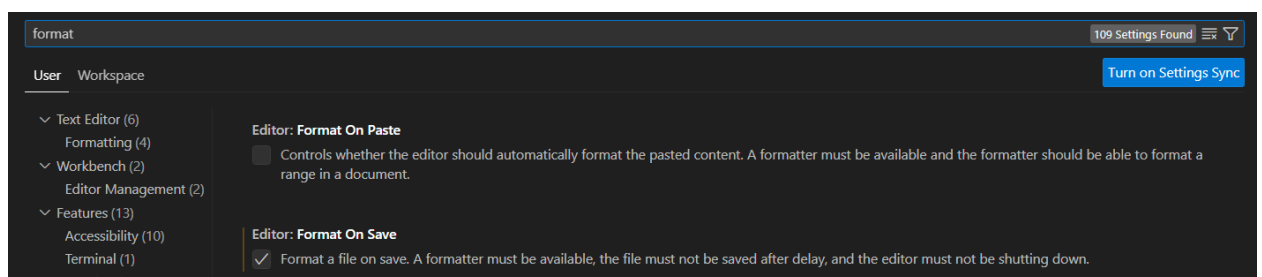
Настройка Prettier



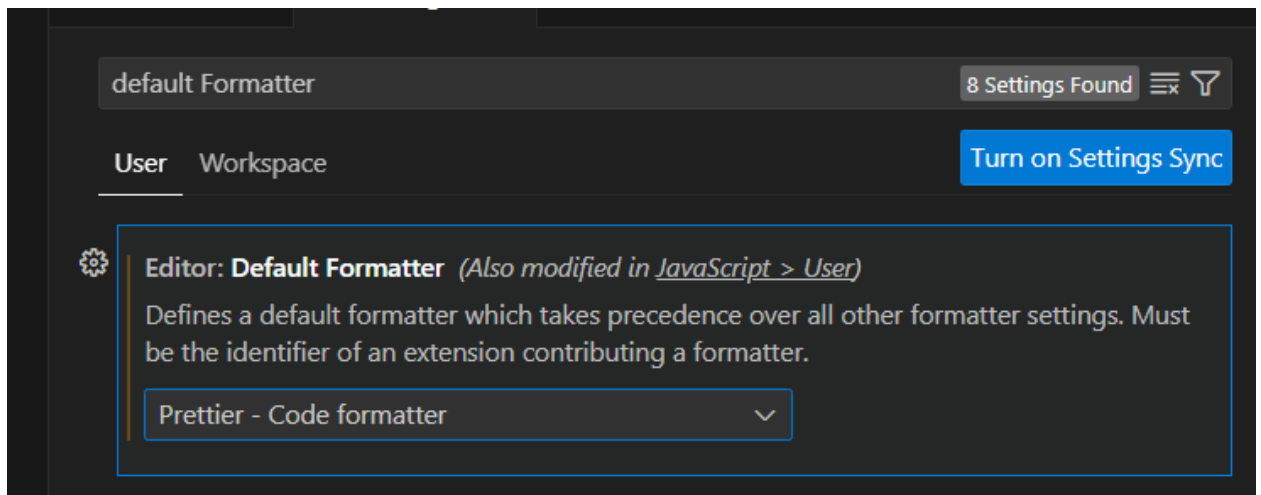
Semi – автоматическое выставление «;» в конце строки кода

Single Quote – использование одинарных кавычек

Tab Width - табуляция



Format on Save – применить автоматическое форматирование кода при сохранении



```
269 // =====
270 // Promise
271
272 // Пример 1
273 let a = 7;
274 console.log(a);
275
276 let b = new Promise(function (resolve, reject) {
277   setTimeout(() => {
278     resolve((a = 99));
279   }, 2000);
280 });
281
282 b.then(function () {
283   console.log(a);
284 }).catch(console.error);
```

```

286 // Пример 2
287 console.log('Request data...');
288 const p = new Promise(function (resolve, reject) {
289     setTimeout(() => {
290         console.log('Preparing data...');
291         const backendData = {
292             server: 'aws',
293             port: 2000,
294             status: 'working',
295         };
296         resolve(backendData);
297     }, 2000);
298 });
299
300 p.then((data) => {
301     return new Promise((resolve, reject) => {
302         setTimeout(() => {
303             data.modified = true;
304             resolve(data); //reject(data);
305         }, 2000);
306     });
307 })
308 .then((clientData) => {
309     clientData.fromPromise = true;
310     return clientData;
311 })
312 .then((data) => {
313     console.log('Modified', data);
314 })
315 .catch((err) => console.log('Error: ', err))
316 .finally(() => {
317     console.log('Finally');
318 });

```

Модифицируем код, что бы ветка catch заработала: строку `resolve(data);` во втором промисе заменим на `reject(data);`

```
320 // Promise.all и Promise.race
321
322 const sleep = (ms) => {
323   return new Promise((resolve) => {
324     setTimeout(() => resolve(), ms);
325   });
326 };
327
328 // sleep(2000).then(() => console.log('After 2 sec'));
329 // sleep(3000).then(() => console.log('After 3 sec'));
330
331 Promise.all([sleep(2000), sleep(5000)]).then(() => {
332   console.log('All promises');
333 });
334
335 Promise.race([sleep(2000), sleep(5000)]).then(() => {
336   console.log('Race promises');
337 });
```



```

339 // Пример. Определение победителя в автогонках
340
341 class Car {
342   constructor(mark, model, speed) {
343     this.mark = mark;
344     this.model = model;
345     this.speed = speed;
346     this.time = Math.floor(Math.random() * 10000) - this.speed;
347   }
348 }
349
350 const result = [];
351
352 const competitors = (car) => {
353   return new Promise((resolve) => {
354     setTimeout(() => {
355       console.log(
356         'Car',
357         car.mark,
358         car.model,
359         'finished in',
360         car.time,
361         'ms'
362       );
363       result.length == 0 ? (car.isWinner = true) : (car.isWinner = false);
364       result.push(car);
365       resolve();
366     }, car.time);
367   });
368 };
369
370 let race = [
371   competitors(new Car('ford', 'mustang', 400)),
372   competitors(new Car('mazda', 'rx-7', 350)),
373   competitors(new Car('BMW', 'x5', 300)),
374 ];
375
376 Promise.all(race).then(() => {
377   console.table(result);
378 });

```

```
380 // async/await
381
382 const delay = (ms) => {
383   return new Promise((r) => setTimeout(() => r(), ms));
384 };
385
386 // delay(2000).then(() => console.log('2 sec'));
387
388 const url = 'https://jsonplaceholder.typicode.com/todos';
389
390 function fetchTodos() {
391   console.log('Fetch todo started ...');
392   return delay(2000)
393     .then(() => fetch(url))
394     .then((response) => response.json());
395 }
396
397 fetchTodos()
398   .then((data) => {
399     console.log('Data: ', data);
400   })
401   .catch((e) => console.error(e));
402
403 async function fetchAsyncTodos() {
404   console.log('Fetch todo started ...');
405   try {
406     await delay(2000);
407     const response = await fetch(url);
408     const data = await response.json();
409     console.log('Data ', data);
410   } catch (e) {
411     console.error(e);
412   } finally {
413     console.log('Finally');
414   }
415 }
416
417 fetchAsyncTodos();
418
419 //https://jsonplaceholder.typicode.com/
```

```
421 /***** Замыкание - Closure *****/
422
423 function outer() {
424     let x = 10;
425     function inner() {
426         console.log(x);
427     }
428     return inner;
429 }
430
431 let closureFn = outer();
432 closureFn();
433
434 function createCalcFn(n) {
435     return function () {
436         console.log(100 * n);
437     };
438 }
439
440 const calc = createCalcFn(42);
441 calc();
442
443 function createIncrementor(n) {
444     return function (m) {
445         return n + m;
446     };
447 }
448
449 const addOne = createIncrementor(12);
450 console.log(addOne(3));
451
452 const addTen = createIncrementor(10);
453 console.log(addTen(100));
454
455 function urlGeneration(domain) {
456     return function (url) {
457         return `https://${url}.${domain}`;
458     };
459 }
460
461 const comUrl = urlGeneration('com');
462 console.log(comUrl('google'));
463 console.log(comUrl('VK'));
464
```

```
465 // самостоятельно: написать функцию ruUrl, которая замыкает функцию urlGeneration() и выводит в консоль сообщения типа
466 // https://VK.ru
467 // https://netflix.ru
468 // https://google.ru
469 //
470
```

ИМЕНА ПЕРЕМЕННЫХ

PascalCase

Компоненты React

DB_PASSWORD

значения известны до
запуска приложения и
не меняются

camelCase

Все остальные переменные

СЕМАНТИЧЕСКОЕ УПРАВЛЕНИЕ ВЕРСИЯМИ

13.5.0

Основная
major

Минорная
minor

патч
patch

ДОПУСТИМЫЕ ОБНОВЛЕНИЯ

Допускаются **минорные**
и **патч** версии

^13.5.0

Допускаются **патч**
версии

~13.5.0

Обновления
не допускаются

13.5.0

Файл package.json

СКРИПТЫ NPM

```
"scripts": {  
  "start": "node scripts/start.js",  
  "build": "node scripts/build.js",  
  "test": "node scripts/test.js"  
}
```

npm start

npm run build

npm test