🌌 Полный разбор компонента **ApodRandom.tsx**

Вот компонент, разбитый на логические части, с детальными объяснениями:

1️⃣ Импорты

```
import { useEffect, useState } from "react";
import { useDispatch, useSelector } from "react-redux";
import { fetchRandomApod } from "../features/apod/apodSlice";
import { RootState, AppDispatch } from "../app/store";
import ImageModal from "./ImageModal";
```

Здесь происходит:

- **useEffect** — чтобы выполнить загрузку данных при рендере.
- **useState** — чтобы хранить состояние модального окна.
- **useDispatch** — чтобы отправлять экшены Redux.
- **useSelector** — чтобы получать данные из Redux store.

- **fetchRandomApod** — Redux thunk, который загружает из API 3 случайных изображения.
- **RootState, AppDispatch** — типы Redux, чтобы TypeScript корректно понимал типизацию.
- **ImageModal** — компонент модального окна для просмотра фото в увеличенном виде.

2️⃣ Состояние компонента (React useState)

```
const [modalImage, setModalImage] = useState<string | null>(null);
const [modalTitle, setModalTitle] = useState<string>("");
```

Что и зачем:

- `modalImage` хранит URL изображения, которое пользователь хочет посмотреть в большом размере.
- `modalTitle` — название выбранной картинки.
- Пока `modalImage === null`, модальное окно **не показывается**.

3️⃣ Загрузка данных при рендере (useEffect)

```
useEffect(() => {
  dispatch(fetchRandomApod());
}, [dispatch]);
```

Как это работает:

- `useEffect` запускается один раз — при первом рендере компонента.
- Внутри мы вызываем `dispatch(fetchRandomApod())` — это запускает Redux thunk.
- Thunk делает запрос к NASA API и сохраняет результат в Redux `state.apod.items`.

💡 Почему `[dispatch]` в зависимостях?

Типичная практика: это избавляет TypeScript от предупреждений. По факту это аналог `[]`.

4️⃣ Получение данных из Redux (useSelector)

```
const { items, loading, error } = useSelector(
  (state: RootState) ⇒ state.apod
);
```

Разбор:

- `items` — массив картинок APOD.
- `loading` — true пока идет загрузка.
- `error` — текст ошибки, если что-то пошло не так.

Это всё хранится внутри Redux slice (`apodSlice`).

5️⃣ Обработчики модального окна

```
const openModal = (url: string, title: string) ⇒ {
  setModalImage(url);
  setModalTitle(title);
};
```

```
const closeModal = () => setModalImage(null);
```

Как работает:

- `openModal(url)` — сохраняет URL выбранного изображения ⇒ модальное окно открывается.
- `closeModal()` — сбрасывает состояние обратно ⇒ модалка закрывается.

6️⃣ Отображение модального окна

```
{modalImage && (
  <ImageModal url={modalImage} title={modalTitle} onClose={closeModal} />
)}
```

Логика:

- Если `modalImage !== null`, компонент рендерит `<ImageModal />`.
- Если `modalImage === null`, ничего не отображает.

7️⃣ Карточки изображений

```
<div
  style={{
    display: "flex",
    gap: "20px",
    flexWrap: "wrap",
    justifyContent: "center",
    padding: "20px"
  }}
```

```
>
```

Это контейнер всех карточек:

- `flex` — выстраивает карточки в ряд.
- `flexWrap: "wrap"` — позволяет переносить карточки на следующую строку.
- `gap: "20px"` — расстояние между карточками.
- `justifyContent: "center"` — красиво центрирует.

8️⃣ Генерация карточек (map)

```
{items.map((item) ⇒ {
  const imageUrl = item.hdurl ?? item.url;
  return (
    <divkey={item.date}
      style={{
        width: "300px",
        background: "#fff",
        borderRadius: "12px",
        padding: "12px",
        boxShadow: "0 4px 12px rgba(0,0,0,0.15)",
        transition: "transform .2s ease",
        cursor: "pointer"
      }}
      onClick={() ⇒ openModal(imageUrl, item.title)}
      onMouseEnter={(e) ⇒
        (e.currentTarget.style.transform = "scale(1.03)")
      }
      onMouseLeave={(e) ⇒
        (e.currentTarget.style.transform = "scale(1)")
      }
```

```
    >
```

Пояснение:

- **key={item.date}** — уникальный ключ (APOD публикации уникальны по датам).
- **imageUrl = item.hdurl ?? item.url** — если есть HD-картинка, берём её.
- Карточка имеет:
  - белый фон
  - скругленные углы
  - box-shadow
  - hover-анимацию увеличения
  - красивую плавность

Обработчики:

- **onClick** — открытие модалки.
- **onMouseEnter / onMouseLeave** — эффект увеличения карточки.

9️⃣ Изображение в карточке

```
<img
  src={imageUrl}
  alt={item.title}
  style={{
    width: "100%",
    borderRadius: "10px",
    height: "200px",
    objectFit: "cover",
  }}
```

```
/>
```

Что здесь важно:

- `objectFit: "cover"` — изображение аккуратно обрезается, без искажения.
- `height: "200px"` — все карточки одинакового размера.
- Используем **hdurl** для лучшего качества.

🔟 Название и дата

```
<h3 style={{ marginTop: "10px", fontSize: "18px" }}>
  {item.title}
</h3>

<p style={{ fontSize: "14px", opacity: 0.7 }}>
  {item.date}
</p>
```

Что происходит:

- Заголовок + дата оформлены максимально минималистично.
- `opacity: 0.7` делает дату менее заметной, акцент — на изображении.

🎯 Итог: что делает весь компонент

1. Загружает 3 случайных изображения NASA APOD через Redux Toolkit.
2. Сохраняет их в Redux store.
3. Показывает **красивые карточки**.
4. При наведении карточка увеличивается.

5. При клике — открывается **модальное окно с HD версией фото**.
6. Модалка закрывается по клику на фон.