Пользователь должен видеть, где он находится в приложении в любой момент времени. А видит он свое текущее местоположение в адресной строке браузера. Следовательно приложение должно уметь сопоставлять определённый URL с соответствующей ему страницей. То есть, если мы введём в адресную строку например <https://health-imperium/appointmens>, то приложение должно направить нас на страницу списка приёмов, но не на какую-либо другую.

**Router** определяет набор маршрутов и, когда к приложению, приходит запрос, то Router выполняет сопоставление запроса с маршрутами. И если какой-то маршрут совпадает с URL запроса, то этот маршрут выбирается для обработки запроса.

И также для выбора маршрута определен объект **Switch**. Он позволяет выбрать первый попавшийся маршрут и его использовать для обработки. Без этого объекта Router может использовать для обработки одного запроса теоретически несколько маршрутов, если они соответствуют строке запроса.

Каждый маршрут представляет объект **Route**. Он имеет ряд атрибутов. В частности, здесь для маршрута устанавливаются два атрибута:

* **path**: шаблон адреса, с которым будет сопоставляться запрошенный адрес URL
* **component** - тот компонент, который отвечает за обработку запроса по этому маршруту

<Route exact path="/" component={Main} /> main page

С помощью атрибута **children** объекта Route можно определить содержимое компонента, который будет обрабатывать маршрут:

<Router>

    <Switch>

        <Route exact path="/" component={Main} />

        <Route path="/about" children={()=><h2>About</h2>} />

        <Route path="/contact" children={()=><h2>Contact</h2>} />

        <Route component={NotFound} />

    </Switch>

</Router>

С помощью лямбда-выражения для маршрута "/about" фактически определен неявно компонент, который выполняет те же самые действия, что и компонент About. Аналогично будет обрабатываться маршурт "/contact".

**Квалификаторы exact и strict**

Возьмем один ранее определенных маршрутов и чуть изменим его, добавив конечный слеш:

|  |  |
| --- | --- |
| 1 | <Route path="/about/" children={()=><h2>About</h2>} /> |

Данный маршрут соответствует любой строке запроса, которая начинается с "/about". Примеры соответствий:

* http://localhost:3000/about
* http://localhost:3000/about/
* http://localhost:3000/about/25

Используем квалификатор **exact**:

|  |  |
| --- | --- |
| 1 | <Route exact path="/about/" children={()=><h2>About</h2>} /> |

Exact указывает, что строка запроса должна в точности соответствовать шаблону маршрута, и теперь данный маршрут будет соответствовать только двум следующим запросам:

* http://localhost:3000/about
* http://localhost:3000/about/

То есть условия соответствия маршруту стали строже.

Теперь используем квалификатор **strict**:

|  |  |
| --- | --- |
| 1 | <Route strict path="/about/" children={()=><h2>About</h2>} /> |

* Теперь этот маршрут будет соответствовать только одному запросу: <http://localhost:3000/about/>

**Сопоставление пути**

npm пакет path-to-regexp компилирует prop path в регулярное выражение и сопоставляет его против location.pathname. Строки path имеют более сложные опции форматирования чем объясняются здесь. Вы можете почитать [документацию](https://github.com/pillarjs/path-to-regexp).  
  
Когда пути сопоставляются создается объект match который содержит свойства:

* **url** — сопоставляемая часть текущего location.pathname
* **path** — путь в компоненте Route
* **isExact** — path в Route === location.pathname
* **params** — объект содержит значения из path которые возвращает модуль path-to-regexp

**Заметка:** Можете поиграться с [тестером роутов](https://pshrmn.github.io/route-tester/#/) и посмотреть как создается объект match.  
  
**Заметка:** path в Route должен быть абсолютным[4].

### Что делает рендер компонента Route?

У Route есть 3 props'a которые описывают каким образом выполнить рендер сопоставляя prop path с location.pathname и только один из prop должен быть представлен в Route:

* **component** — React компонент. Когда роут удовлетворяется сопоставление в path, то он возвращает переданный component (используя функцию React.createElement).
* **render** — функция которая должна вернуть элемент React. Будет вызвана когда удовлетворится сопоставление в path. Render довольно похож на component, но используется для inline рендеринга и подстановки необходимых для элемента props[5].
* **children** — в отличие от предыдущих двух props children будет всегда отображаться независимо от того сопоставляется ли path или нет.

**NavLink** | **Link**

Кроме объекта Link из модуля react-router-dom для создания ссылок мы можем использовать объект **NavLink**. Этот объект во многом аналогичен Link за тем исключением, что позволяет использовать состояние ссылки. В частности, с помощью атрибутов **activeClassName** и **activeStyle** можно установить стиль активной ссылки. Атрибут activeClassName указывает на класс активной ссылки. Для этой цели в стилях веб-станицы определен класс active.

Для маршрута "/about" определено два параметра id и name:

|  |  |
| --- | --- |
| 1 | <Route path="**about/:id/:name**" component={About} /> |

Чтобы получить эти параметры в компоненте, надо обратиться к объекту **this.props.match.params**:

### Необязательные параметры

В предыдущем определении маршрута "about/:id/:name" при обращении к этому маршруту ему обязательно должны быть переданы значения для параметров id и name. Но мы можем отметить параметры как необязательные, указав в их определении знак вопроса:

|  |  |  |  |
| --- | --- | --- | --- |
|  | <Route path="about/:id?" component={About} />  Или (если несколько параметров):   |  |  | | --- | --- | | 1 | <Route path="about/:id?/:name?" component={About} /> | |

link https://metanit.com/web/react/4.4.php