### Упражнения 0.1.-0.6.

**0.1: HTML**

Ознакомьтесь с основами HTML, прочитав этот учебник от Mozilla: [Учебник по HTML](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics) .

*Это упражнение не выложено на GitHub, достаточно просто прочитать туториал*

**0.2: CSS**

Ознакомьтесь с основами CSS, прочитав этот учебник от Mozilla: [Учебник по CSS](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics) .

*Это упражнение не выложено на GitHub, достаточно просто прочитать туториал*

**0.3: HTML-формы**

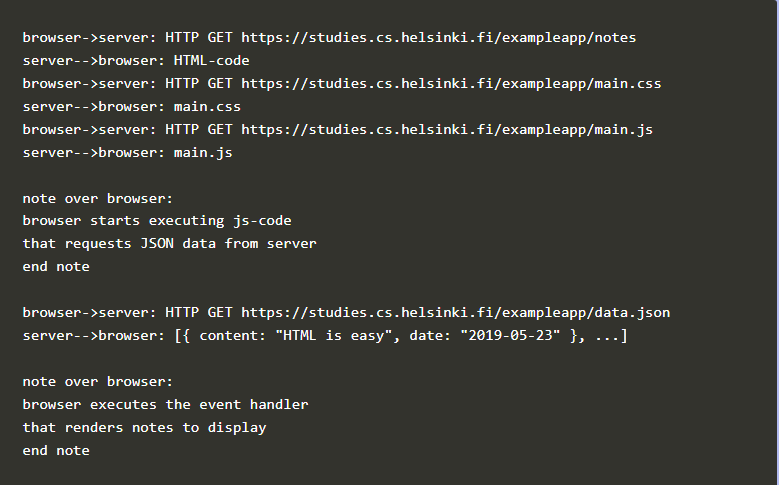
Узнайте об основах HTML-форм, прочитав учебник Mozilla [Ваша первая форма](https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Your_first_HTML_form) .

*Это упражнение не выложено на GitHub, достаточно просто прочитать туториал*

**0.4: Новая диаграмма заметок**

В главе [Загрузка страницы, содержащей JavaScript - просмотрите](https://fullstackopen.com/en/part0/fundamentals_of_web_apps#loading-a-page-containing-java-script-review) цепочку событий, вызванных открытием страницы <https://studies.cs.helsinki.fi/exampleapp/notes> , изображенную в виде [диаграммы последовательности .](https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/)

Диаграмма была сделана с помощью сервиса [websequencediagrams](https://www.websequencediagrams.com/" \t "_blank) следующим образом:



**Создайте аналогичную диаграмму** , изображающую ситуацию, когда пользователь создает новую заметку на странице <https://studies.cs.helsinki.fi/exampleapp/notes> , вводя что-то в текстовое поле и нажимая кнопку *отправки* .

При необходимости покажите операции в браузере или на сервере в виде комментариев к диаграмме.

Диаграмма не обязательно должна быть диаграммой последовательности. Любой разумный способ представления событий прекрасен.

Всю необходимую информацию для выполнения этого и следующих двух упражнений можно найти в тексте [этой части](https://fullstackopen.com/en/part0/fundamentals_of_web_apps#forms-and-http-post) . Идея этих упражнений состоит в том, чтобы еще раз прочитать текст и обдумать, что там происходит. [Читать код](https://github.com/mluukkai/example_app) приложения не обязательно, но можно конечно.

**Обратите внимание,** что, возможно, лучший способ создания диаграмм — это синтаксис [Mermaid](https://github.com/mermaid-js/mermaid#sequence-diagram-docs---live-editor) , который теперь реализован на страницах уценки [GitHub](https://github.blog/2022-02-14-include-diagrams-markdown-files-mermaid/) !

**0.5: Диаграмма одностраничного приложения**

Создайте схему, изображающую ситуацию, когда пользователь переходит к одностраничной [версии](https://fullstackopen.com/en/part0/fundamentals_of_web_apps#single-page-app) приложения для заметок по адресу <https://studies.cs.helsinki.fi/exampleapp/spa> .

**0.6: Новое примечание на диаграмме одностраничного приложения**

Создайте схему, изображающую ситуацию, когда пользователь создает новую заметку, используя одностраничную версию приложения.

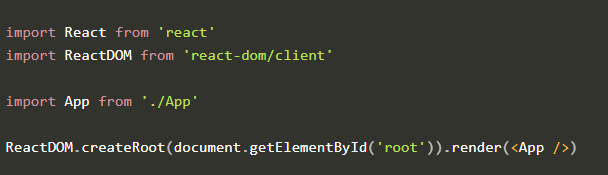
Это было последнее упражнение, и пришло время отправить ваши ответы на GitHub и отметить упражнения как выполненные в [системе отправки](https://studies.cs.helsinki.fi/stats/courses/fullstackopen) .

### Упражнения 1.1.-1.2.

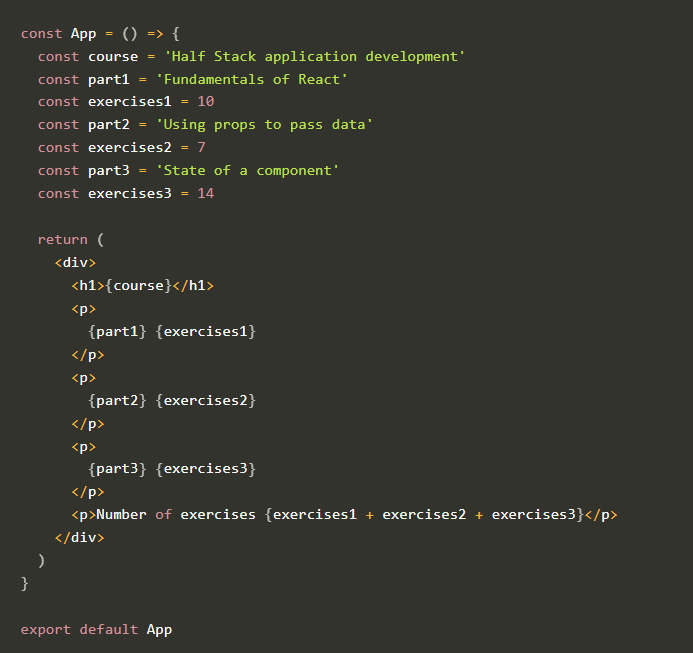
#### 1.1: информация о курсе, шаг 1

*Приложение, над которым мы начнем работать в этом упражнении, будет доработано в нескольких следующих упражнениях. В этом и других следующих наборах упражнений в этом курсе достаточно только передать финальное состояние приложения. При желании вы также можете создать коммит для каждого упражнения серии, но это совершенно необязательно.*

Используйте create-react-app для инициализации нового приложения. Измените *index.js* , чтобы он соответствовал следующему



и *App.js*, чтобы соответствовать следующему

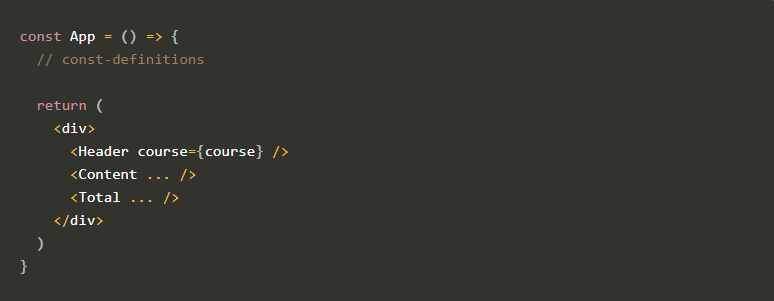


и удалите лишние файлы (App.css, App.test.js, index.css, logo.svg, setupTests.js, reportWebVitals.js.

К сожалению, все приложение находится в одном компоненте. измените код, чтобы он состоял из трех новых компонентов: *Header*, *Content* и *Total*. Все данные по-прежнему находятся в компоненте *App*, который передает необходимые данные каждому компоненту с помощью *props*. *Header* отвечает за отображение названия курса, *Content* отображает части и количество упражнений в них, а *Total* отображает общее количество упражнений.

Определите новые компоненты в файле *App.js.*

Тело компонента *App* будет примерно таким:



**ПРЕДУПРЕЖДЕНИЕ** create-react-app автоматически превращает проект в репозиторий git, если приложение не создано в уже существующем репозитории. Скорее всего вы **не хотите,** чтобы проект стал репозиторием, поэтому выполните команду rm -rf.git в корне проекта.

#### 1.2: информация о курсе, шаг 2

измените компонент *Content*, чтобы он сам не отображал названия частей или их количество упражнений. Вместо этого он отображает только три компонента *«Part»*, каждый из которых отображает название и количество упражнений одной части.



Наше приложение на данный момент передает информацию довольно примитивным способом, так как оно основано на отдельных переменных. Эта ситуация скоро улучшится.

### **Упражнения 1.3.-1.5.**

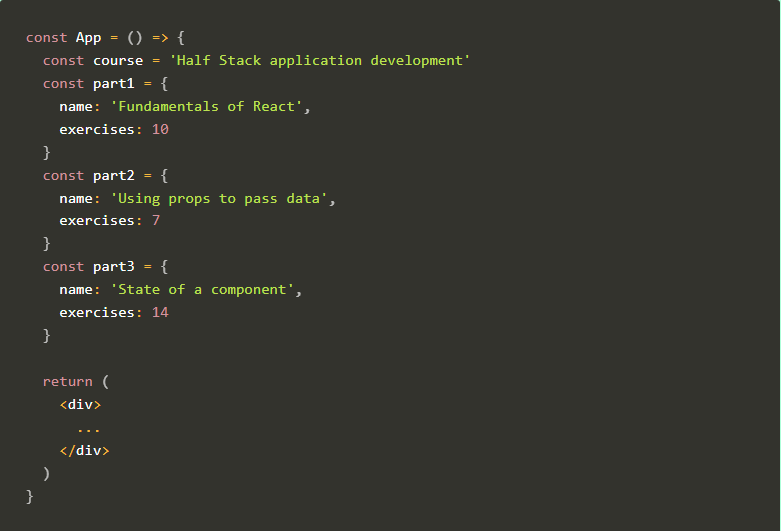
*Мы продолжаем создавать приложение, над которым начали работать в предыдущих упражнениях. Вы можете написать код в этот же проект, так как нас интересует только финальное состояние отправленного приложения.*

**Совет:** вы можете столкнуться с проблемами, когда дело доходит до структуры свойств*,* которые получают компоненты. Хороший способ прояснить ситуацию — вывести свойства в консоль, например, следующим образом:



#### 1.3: информация о курсе шаг 3

Давайте перейдем к использованию объектов в нашем приложении. Измените определения переменных компонента *App* следующим образом, а также реорганизуйте приложение, чтобы оно по-прежнему работало:



#### 1.4: информация о курсе шаг 4

Затем поместите объекты в массив. Измените определения переменных *App* в следующей форме и соответствующим образом измените другие части приложения:



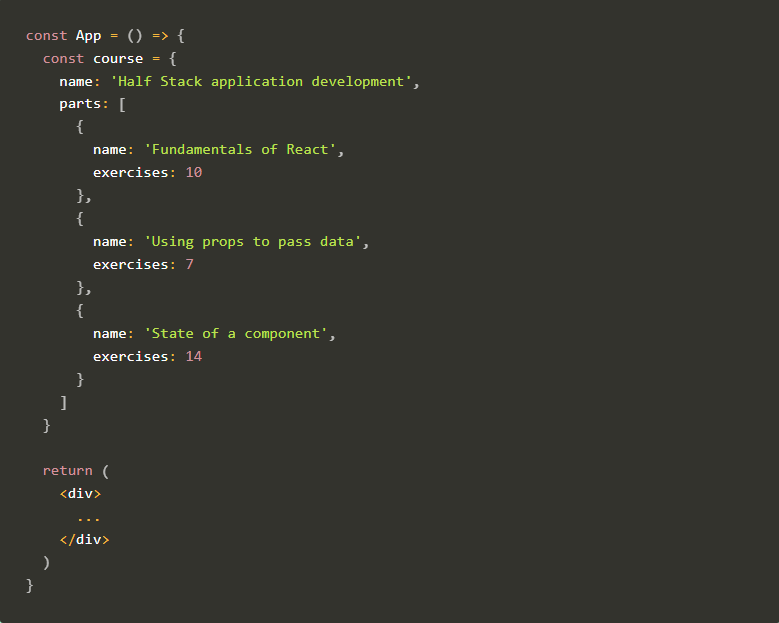
**NB** на этом этапе *вы можете предположить, что всегда есть три элемента* , поэтому нет необходимости проходить массивы с помощью циклов. Мы вернемся к теме рендеринга компонентов на основе элементов в массивах с более тщательным изучением в [следующей части курса](https://fullstackopen.com/en/part2).

Однако не передавайте разные объекты как отдельные props из компонента *App* компонентам *Content* и *Total*. Вместо этого передайте их напрямую в виде массива:



#### 1.5: информация о курсе шаг 5

Сделаем еще один шаг вперед. Превратите курс и его части в один объект JavaScript. а также реорганизуйте приложение, чтобы оно по-прежнему работало.



### Упражнения 1.6.-1.14.

Отправьте свои решения к упражнениям, сначала отправив свой код на GitHub, а затем отметив выполненные упражнения на вкладке «Мои отправки» приложения для [отправки](https://studies.cs.helsinki.fi/stats/courses/fullstackopen) .

Помните, что **все** упражнения одной части следует отправлять **в одном представлении** . После того как вы отправили свои решения для одной части, **вы больше не можете отправлять дополнительные упражнения в эту часть** .

*Некоторые упражнения работают в одном и том же приложении. В этих случаях достаточно представить только окончательный вариант заявки. При желании вы можете делать коммит после каждого выполненного упражнения, но это не обязательно.*

**ПРЕДУПРЕЖДЕНИЕ** create-react-app автоматически превратит ваш проект в git-репозиторий, если только вы не создадите свое приложение внутри существующего git-репозитория. **Скорее всего, вы не хотите, чтобы каждый из ваших проектов был отдельным репозиторием** , поэтому просто запустите команду rm -rf.git в корне вашего приложения.

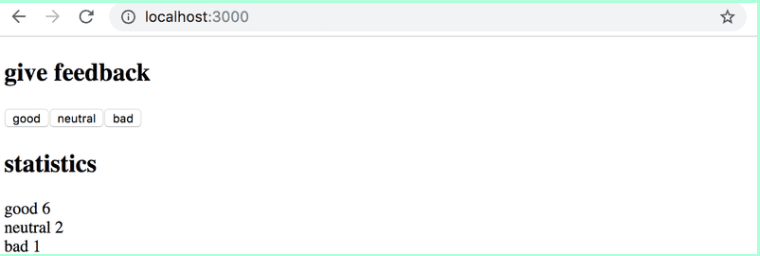
В некоторых ситуациях вам также может потребоваться выполнить приведенную ниже команду из корня проекта:



#### 1.6: уникафе шаг 1

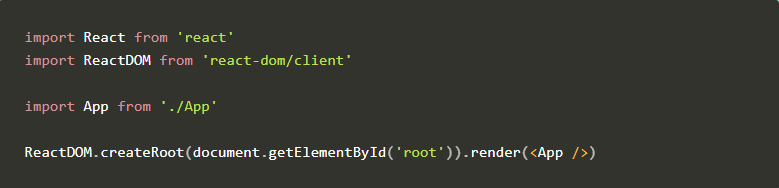
Как и большинство компаний, [Unicafe](https://www.unicafe.fi/" \l "/9/4" \t "_blank) собирает отзывы своих клиентов. Ваша задача — реализовать веб-приложение для сбора отзывов клиентов. Есть только три варианта отзыва: *хороший* , *нейтральный* и *плохой* .

Приложение должно отображать общее количество собранных отзывов по каждой категории. Ваше окончательное приложение может выглядеть так:

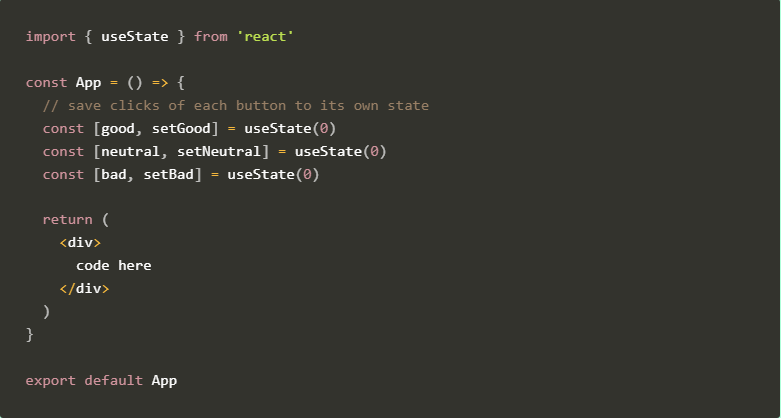


Обратите внимание, что ваше приложение должно работать только в течение одного сеанса браузера. Как только вы обновите страницу, собранные отзывы могут исчезнуть.

Желательно использовать ту же структуру, которая используется в материале и предыдущем упражнении. Файл *index.js* выглядит следующим образом:

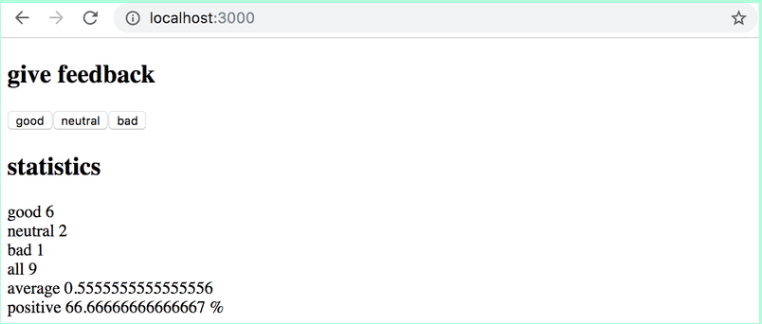


Вы можете использовать приведенный ниже код в качестве отправной точки для файла *App.js* :



#### 1.7: уникафе шаг 2

Разверните свое приложение, чтобы оно отображало больше статистики о собранных отзывах: общее количество собранных отзывов, средний балл (хороший: 1, нейтральный: 0, плохой: -1) и процент положительных отзывов.



#### 1.8: уникафе, шаг 3

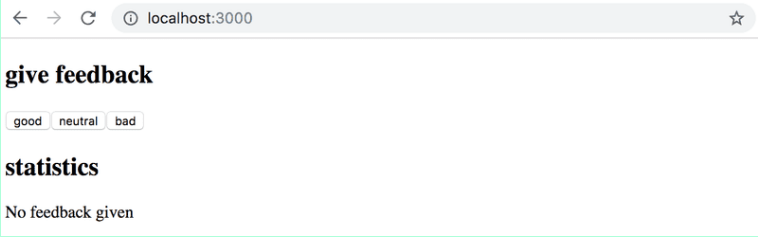
Рефакторинг вашего приложения, чтобы отображение статистики было извлечено в отдельный компонент *статистики* . Состояние приложения должно оставаться в корневом компоненте *приложения* .

Помните, что компоненты не должны определяться внутри других компонентов:



#### 1.9: уникафе, шаг 4

Измените свое приложение, чтобы отображать статистику только после сбора отзывов.

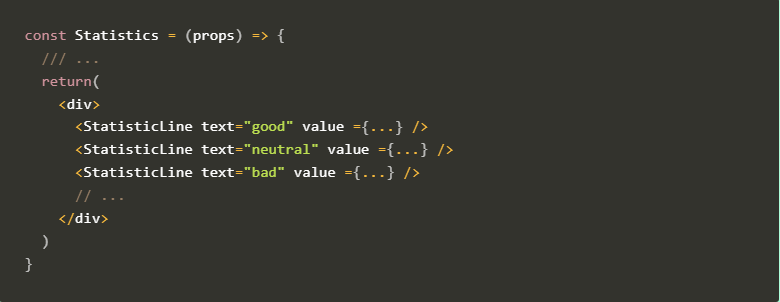


#### 1.10: уникафе шаг 5

Продолжим рефакторинг приложения. Извлеките следующие два компонента:

* *Кнопка* для определения кнопок, используемых для отправки отзыва
* *StatisticLine* для отображения отдельной статистики, например среднего балла.

Для ясности: компонент *StatisticLine* всегда отображает одну статистику, а это означает, что приложение использует несколько компонентов для отображения всей статистики:



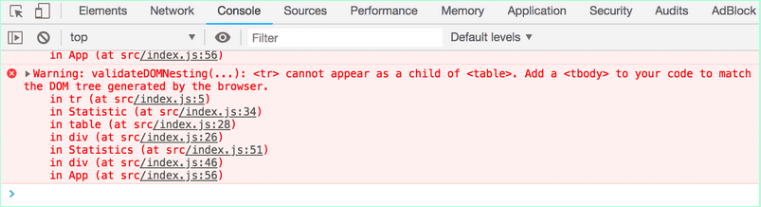
Состояние приложения должно по-прежнему храниться в корневом компоненте *приложения* .

#### 1.11\*: уникафе шаг 6

Отобразите статистику в HTML- [таблице](https://developer.mozilla.org/en-US/docs/Learn/HTML/Tables/Basics) , чтобы ваше приложение выглядело примерно так:



Не забывайте постоянно держать консоль открытой. Если вы видите это предупреждение в консоли:



Затем выполните необходимые действия, чтобы предупреждение исчезло. Попробуйте вставить сообщение об ошибке в поисковую систему, если вы застряли.

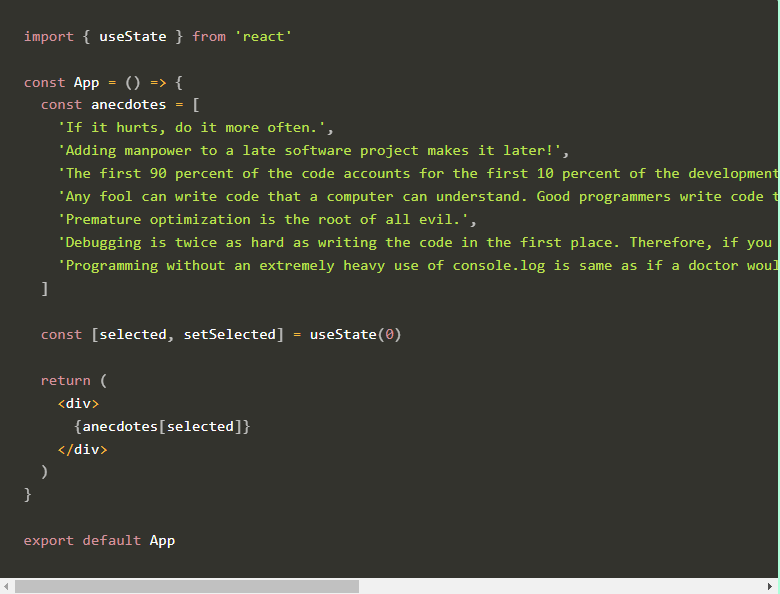
*Типичным источником ошибки Unchecked runtime.lastError: Could not establish connection. Receiving end does not exist.является расширение Chrome. Попробуйте перейти chrome://extensions/и попробовать отключить их один за другим и обновить страницу приложения React; ошибка должна со временем исчезнуть.*

**Убедитесь, что с этого момента вы не видите никаких предупреждений в своей консоли!**

#### 1.12\*: анекдоты шаг 1

Мир программной инженерии полон [анекдотов](http://www.comp.nus.edu.sg/~damithch/pages/SE-quotes.htm) , которые превращают вечные истины из нашей области в короткие остроты.

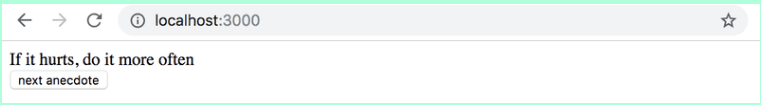
Расширьте следующее приложение, добавив кнопку, по которой можно щелкнуть, чтобы отобразить *случайный* анекдот из области разработки программного обеспечения:



Содержимое файла *index.js* такое же, как и в предыдущих упражнениях.

Узнайте, как генерировать случайные числа в JavaScript, например. через поисковую систему или в [Mozilla Developer Network](https://developer.mozilla.org/) . Помните, что вы можете протестировать генерацию случайных чисел, например, прямо в консоли вашего браузера.

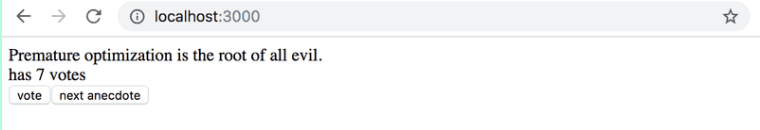
Готовое приложение может выглядеть примерно так:



**ПРЕДУПРЕЖДЕНИЕ** create-react-app автоматически превратит ваш проект в git-репозиторий, если только вы не создадите свое приложение внутри существующего git-репозитория. **Скорее всего, вы не хотите, чтобы каждый из ваших проектов был отдельным репозиторием** , поэтому просто запустите команду rm -rf.git в корне вашего приложения.

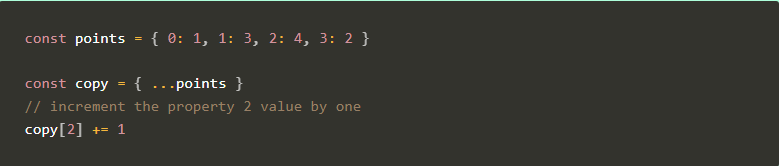
#### 1.13\*: анекдоты шаг 2

Разверните свое приложение, чтобы вы могли проголосовать за отображаемый анекдот.

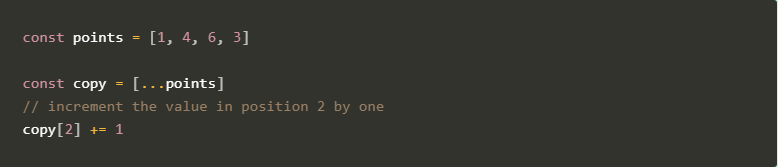


**NB** хранит голоса каждого анекдота в массиве или объекте в состоянии компонента. Помните, что правильный способ обновления состояния, хранящегося в сложных структурах данных, таких как объекты и массивы, заключается в создании копии состояния.

Вы можете создать копию объекта следующим образом:



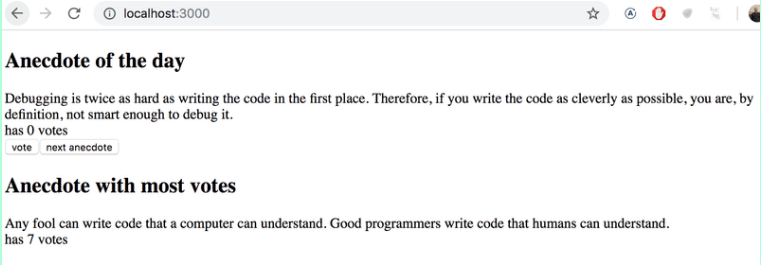
ИЛИ копию такого массива:



В этом случае использование массива может быть более простым выбором. Поиск в Интернете даст вам множество советов о том, как [создать заполненный нулями массив желаемой длины](https://stackoverflow.com/questions/20222501/how-to-create-a-zero-filled-javascript-array-of-arbitrary-length/22209781) .

#### 1.14\*: анекдоты шаг 3

Теперь реализуйте финальную версию приложения, отображающую анекдот, набравший наибольшее количество голосов:



Если несколько анекдотов делят первое место, достаточно просто показать один из них.

Это было последнее упражнение в этой части курса, и пришло время отправить ваш код на GitHub и отметить все выполненные вами упражнения на вкладке «мои материалы» приложения для [отправки](https://studies.cs.helsinki.fi/stats/courses/fullstackopen) .

### Упражнения 2.1.-2.5.

Упражнения отправляются через GitHub и помечаются как выполненные в [системе отправки](https://studies.cs.helsinki.fi/stats/courses/fullstackopen) .

Вы можете отправить все упражнения в один и тот же репозиторий или использовать несколько разных репозиториев. Если вы отправляете упражнения из разных частей в один и тот же репозиторий, правильно назовите свои каталоги.

Упражнения представлены **по одной части за один раз** . Когда вы отправили упражнения для части, вы больше не можете отправлять пропущенные упражнения для этой части.

Обратите внимание, что в этой части больше упражнений, чем в предыдущих, поэтому *не отправляйте* , пока не выполните все упражнения из этой части, которые хотите отправить.

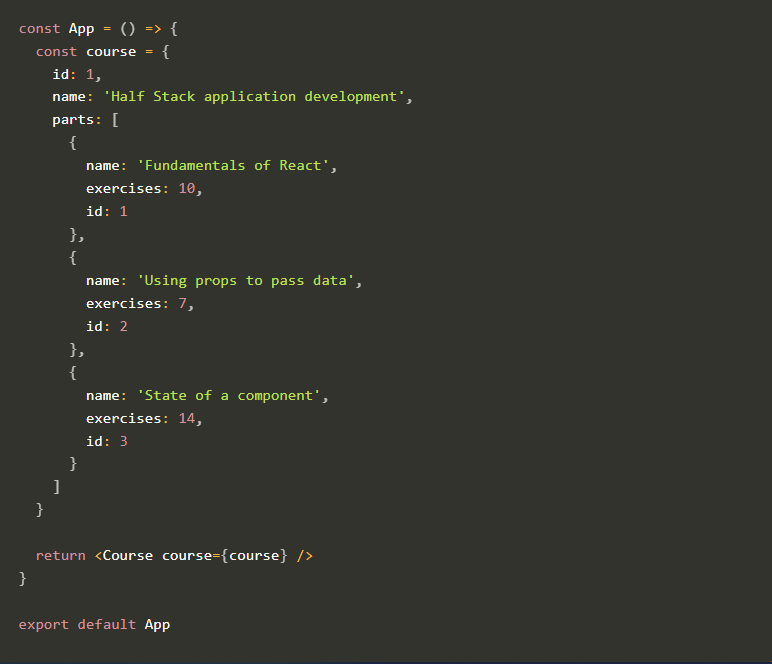
**ВНИМАНИЕ** create-реагировать-приложение автоматически превращает проект в git-репозиторий, если проект не создан внутри уже существующего репозитория. Вероятно, вы **не** хотите, чтобы проект стал репозиторием, поэтому выполните команду rm -rf .git из его корня.

#### 2.1: Информация о курсе, шаг 6

Закончим код для рендеринга содержимого курса из упражнений 1.1 — 1.5. Вы можете начать с кода из ответов модели. Типовые ответы для части 1 можно найти, перейдя в [систему подачи](https://studies.cs.helsinki.fi/stats/courses/fullstackopen) , щелкнув *мои представления* вверху и в строке, соответствующей части 1, под столбцом *решений* щелкните *показать* . Чтобы увидеть решение упражнения *с информацией о курсе* , щелкните index.js в разделе *kurssitiedot* («kurssitiedot» означает «информация о курсе»).

**Обратите внимание, что если вы копируете проект из одного места в другое, вам может потребоваться удалить каталог *node\_modules* и снова установить зависимости с помощью команды npm install , прежде чем вы сможете запустить приложение.**Как правило, не рекомендуется копировать все содержимое проекта и/или добавлять каталог *node\_modules* в систему контроля версий.

Давайте изменим компонент *приложения* следующим образом:



Определите компонент, отвечающий за форматирование одного курса с именем *Course* .

Компонентная структура приложения может быть, например, следующей:



Следовательно, компонент *«Курс»* содержит компоненты, определенные в предыдущей части, которые отвечают за отображение названия курса и его частей.

Визуализированная страница может, например, выглядеть следующим образом:



Вам пока не нужна сумма упражнений.

Приложение должно работать *независимо от количества частей курса* , поэтому убедитесь, что приложение работает, если вы добавляете или удаляете части курса.

Убедитесь, что консоль не показывает ошибок!

#### 2.2: Информация о курсе, шаг 7

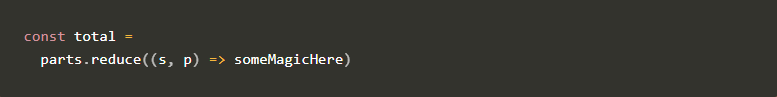
Покажите также сумму упражнений курса.



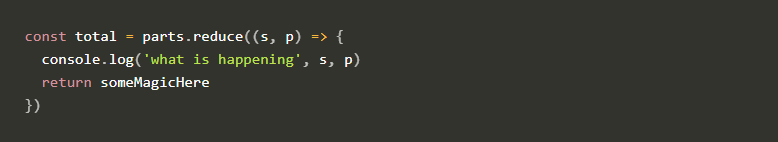
#### 2.3\*: Информация о курсе, шаг 8

Если вы еще этого не сделали, подсчитайте сумму упражнений с помощью метода массива [reduce](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce) .

**Совет для профессионалов:** когда ваш код выглядит следующим образом:

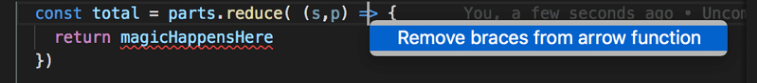


и не работает, стоит использовать *console.log* , который требует, чтобы стрелочная функция была написана в более длинной форме:



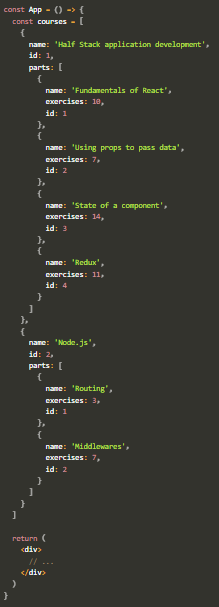
**Не работает? :** Воспользуйтесь поисковой системой, чтобы узнать, как используется сокращение в **массиве объектов** .

**Совет 2:**[для VS Code](https://marketplace.visualstudio.com/items?itemName=cmstead.js-codeformer) существует плагин , который автоматически изменяет короткие функции стрелок на более длинные и наоборот.

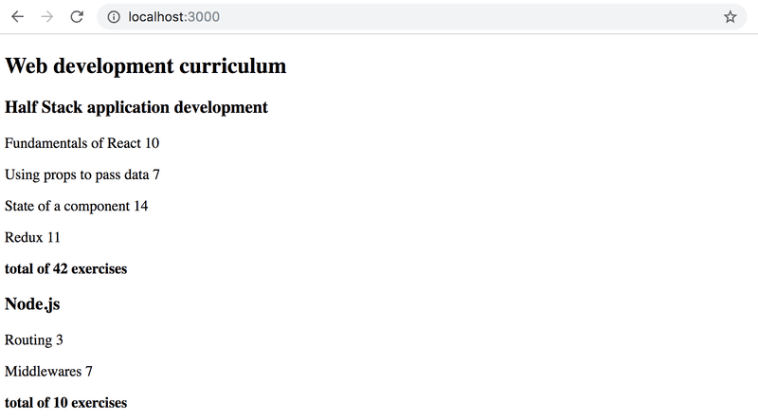


#### 2.4: Информация о курсе шаг 9

Давайте расширим наше приложение, чтобы разрешить *произвольное количество* курсов:



Приложение может выглядеть, например, так:



#### 2.5: отдельный модуль

Объявите компонент *«Курс»* как отдельный модуль, который импортируется компонентом *«Приложение»* . Вы можете включить все подкомпоненты курса в один и тот же модуль.

### Упражнения 2.6.-2.10.

В первом упражнении мы начнем работать над приложением, которое будет доработано в последующих упражнениях. В связанных наборах упражнений достаточно вернуть окончательный вариант вашего приложения. Вы также можете сделать отдельную фиксацию после завершения каждой части набора упражнений, но это не обязательно.

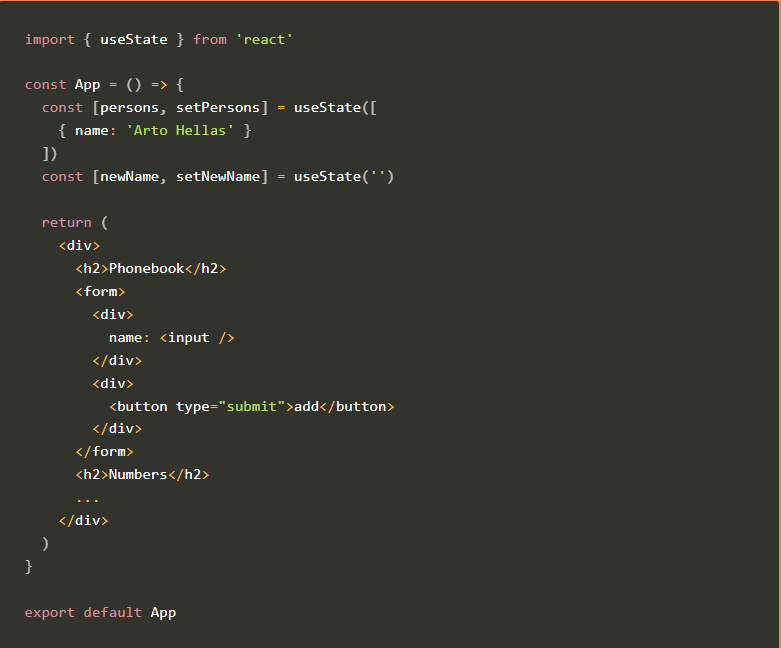
**ПРЕДУПРЕЖДЕНИЕ** create-react-app автоматически превратит ваш проект в git-репозиторий, если только вы не создадите свое приложение внутри существующего git-репозитория. Скорее всего, вы **не хотите,** чтобы ваш проект был репозиторием, поэтому просто запустите команду rm -rf .git в корне вашего приложения.

#### 2.6: Телефонная книга Шаг 1

Давайте создадим простую телефонную книгу. ***В этой части мы будем только добавлять имена в телефонную книгу.***

Начнем с реализации добавления человека в телефонную книгу.

Вы можете использовать приведенный ниже код в качестве отправной точки для компонента *приложения* вашего приложения:



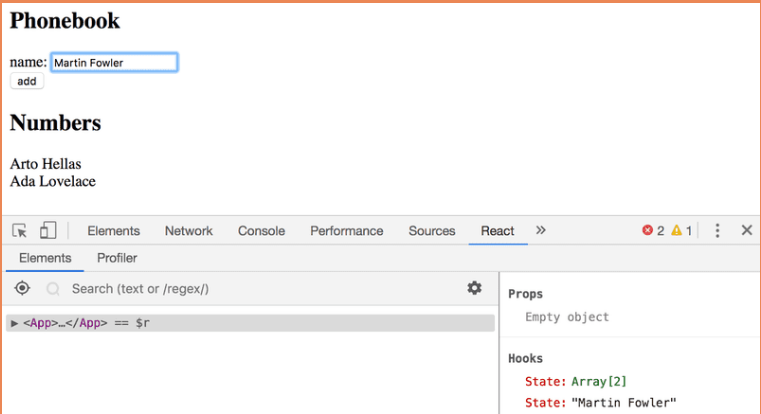
Состояние newName предназначено для управления элементом ввода формы.

Иногда может быть полезно отображать состояние и другие переменные в виде текста для целей отладки. Вы можете временно добавить следующий элемент к отображаемому компоненту:



Также важно правильно использовать то, что мы узнали в главе [об отладке приложений React](https://fullstackopen.com/en/part1/a_more_complex_state_debugging_react_apps) первой части. Расширение [инструментов разработчика React](https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi) особенно полезно для отслеживания изменений, происходящих в состоянии приложения.

После выполнения этого упражнения ваше приложение должно выглядеть примерно так:



Обратите внимание на использование расширения инструментов разработчика React на картинке выше!

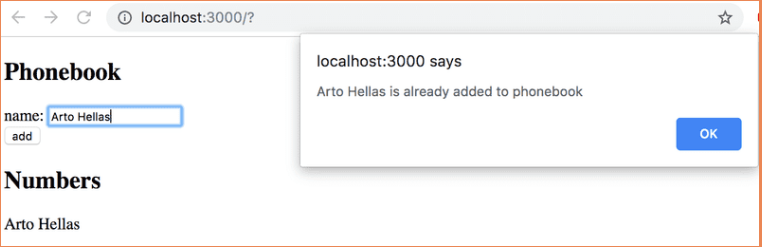
**Примечание:**

* вы можете использовать имя человека как значение *ключевого* свойства
* не забудьте предотвратить действие по умолчанию отправки HTML-форм!

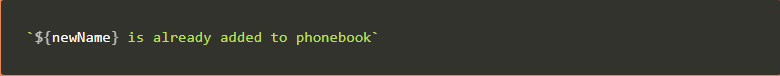
#### 2.7: Телефонная книга Шаг 2

Запретите пользователю добавлять имена, которые уже существуют в телефонной книге. Массивы JavaScript имеют множество подходящих [методов](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array) для выполнения этой задачи. Помните [, как работает равенство объектов](https://www.joshbritz.co/posts/why-its-so-hard-to-check-object-equality/) в Javascript.

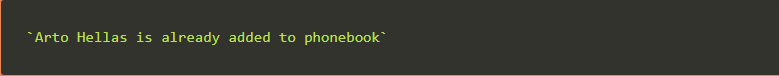
Выдайте предупреждение с помощью команды [alert](https://developer.mozilla.org/en-US/docs/Web/API/Window/alert) при попытке такого действия:



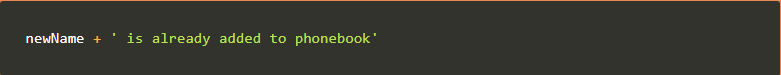
**Подсказка:** при формировании строк, содержащих значения переменных, рекомендуется использовать строку- [шаблон](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals) :



Если переменная newName содержит значение *Arto Hellas* , строковое выражение шаблона возвращает строку



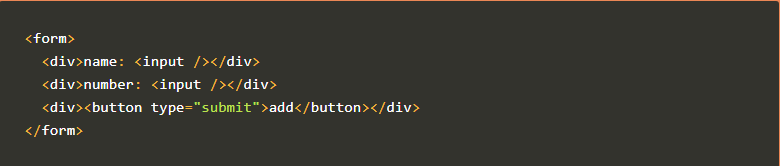
То же самое можно сделать более похожим на Java способом, используя оператор плюс:



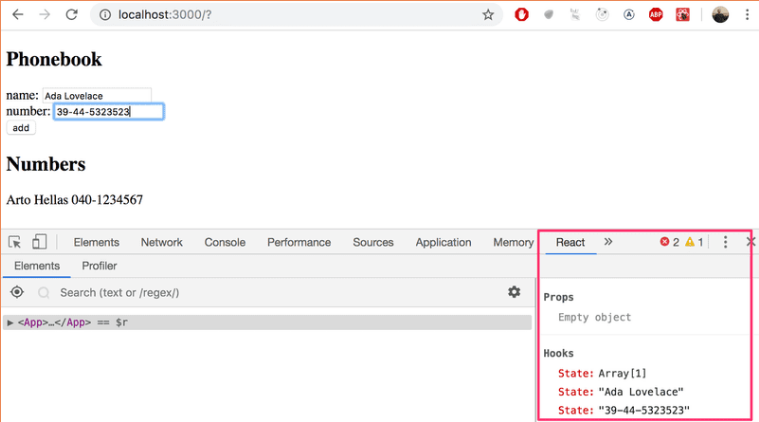
Использование шаблонных строк — более идиоматический вариант и признак настоящего профессионала JavaScript.

#### 2.8: Телефонная книга Шаг 3

Расширьте свое приложение, разрешив пользователям добавлять телефонные номера в телефонную книгу. Вам нужно будет добавить в форму второй элемент *ввода* (вместе с собственным обработчиком событий):

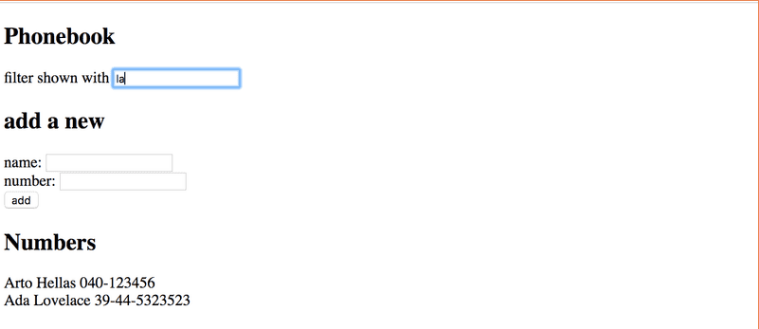


На этом этапе приложение может выглядеть примерно так. На изображении также отображается состояние приложения с помощью [инструментов разработчика React](https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi" \t "_blank) :



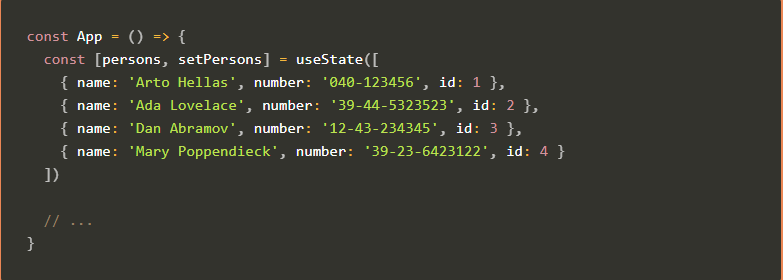
#### 2.9\*: Телефонная книга Шаг 4

Реализуйте поле поиска, которое можно использовать для фильтрации списка людей по имени:



Поле поиска можно реализовать как элемент *ввода* , размещенный за пределами HTML-формы. Логика фильтрации, показанная на изображении, *нечувствительна к регистру* , а это означает, что поисковый запрос *arto* также возвращает результаты, содержащие Arto с прописной буквой A.

**NB:** когда вы работаете над новой функциональностью, часто полезно «жестко закодировать» некоторые фиктивные данные в ваше приложение, например



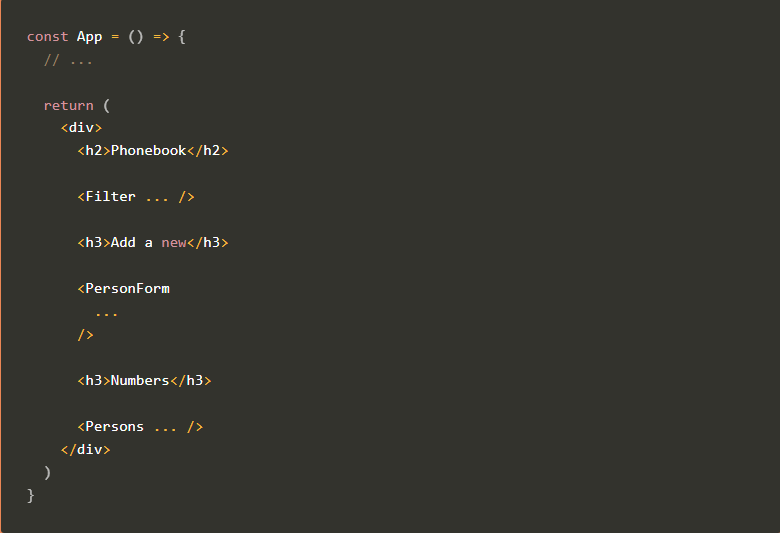
Это избавляет вас от необходимости вручную вводить данные в приложение для тестирования новой функциональности.

#### 2.10: Телефонная книга Шаг 5

Если вы реализовали свое приложение в одном компоненте, проведите его рефакторинг, извлекая подходящие части в новые компоненты. Поддерживайте состояние приложения и всех обработчиков событий в корневом компоненте *приложения* .

Достаточно извлечь из приложения ***три компонента.***Хорошими кандидатами для отдельных компонентов являются, например, поисковый фильтр, форма добавления новых людей в телефонную книгу, компонент, отображающий всех людей из телефонной книги, и компонент, отображающий сведения об одном человеке.

Корневой компонент приложения может выглядеть примерно так после рефакторинга. Рефакторинговый корневой компонент ниже отображает только заголовки и позволяет извлеченным компонентам позаботиться обо всем остальном.

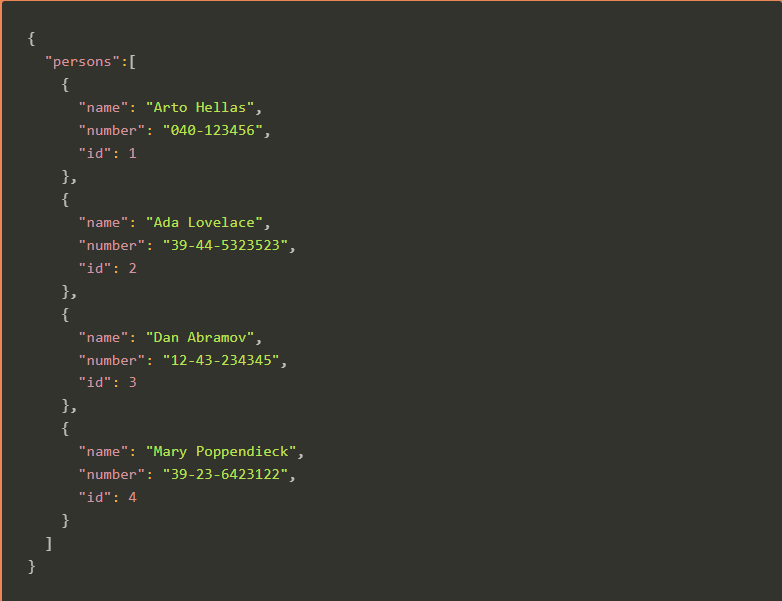


**NB** : Вы можете столкнуться с проблемами в этом упражнении, если вы определите свои компоненты "не в том месте". Сейчас самое время порепетировать главу, [не определяющую компонент в другом компоненте](https://fullstackopen.com/en/part1/a_more_complex_state_debugging_react_apps#do-not-define-components-within-components) из предыдущей части.

### Упражнения 2.11.-2.14.

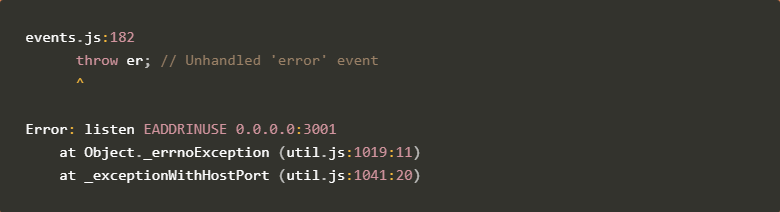
#### 2.11: Телефонная книга Шаг 6

Мы продолжаем развивать телефонную книгу. Хранить начальное состояние приложения в файле *db.json* , который нужно поместить в корень проекта.



Запустите json-сервер на порту 3001 и убедитесь, что сервер возвращает список людей, перейдя по адресу <http://localhost:3001/persons> в браузере.

Если вы получаете следующее сообщение об ошибке:



это означает, что порт 3001 уже используется другим приложением, например, уже запущенным json-сервером. Закройте другое приложение или измените порт, если это не работает.

Измените приложение таким образом, чтобы начальное состояние данных извлекалось с сервера с помощью библиотеки *axios* . Завершите получение с помощью [хука Effect](https://reactjs.org/docs/hooks-effect.html) .

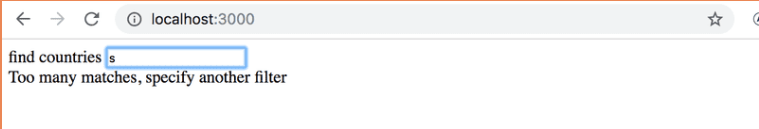
#### 2.12\* Данные по странам, шаг 1

API [https://restcountries.com](https://restcountries.com/) предоставляет данные по разным странам в машиночитаемом формате, так называемый REST API.

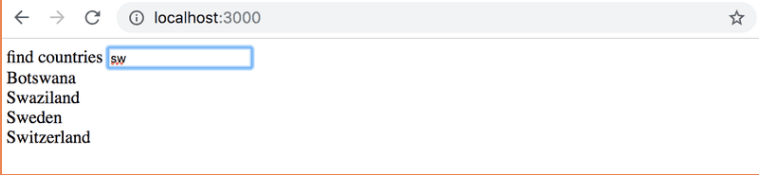
Создайте приложение, в котором можно посмотреть данные разных стран. Приложение, вероятно, должно получать данные из конечной точки [все](https://restcountries.com/v3.1/all) файлы .

Пользовательский интерфейс очень прост. Страна для отображения определяется путем ввода поискового запроса в поле поиска.

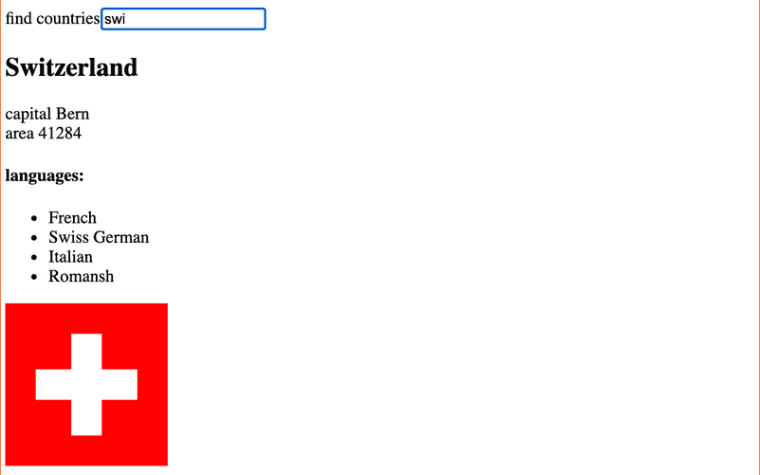
Если слишком много (более 10) стран, соответствующих запросу, пользователю будет предложено сделать свой запрос более конкретным:



Если стран десять или меньше, но больше одной, то отображаются все страны, соответствующие запросу:



Когда запросу соответствует только одна страна, отображаются основные данные страны (например, столица и территория), ее флаг и языки, на которых говорят в ней:



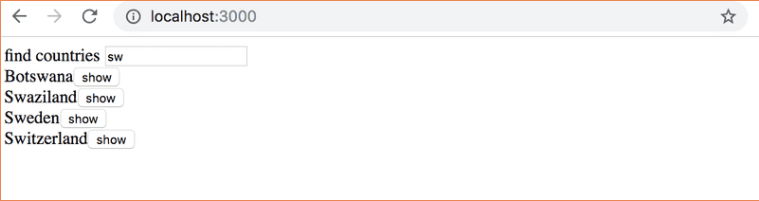
**NB** : Достаточно, чтобы ваше приложение работало в большинстве стран. Некоторые страны, такие как *Судан* , может быть трудно поддерживать, поскольку название страны является частью названия другой страны, *Южного Судана* . Вам не нужно беспокоиться об этих крайних случаях.

**ПРЕДУПРЕЖДЕНИЕ** create-react-app автоматически превратит ваш проект в git-репозиторий, если только вы не создадите свое приложение внутри существующего git-репозитория. **Скорее всего, вы не хотите, чтобы каждый из ваших проектов был отдельным репозиторием** , поэтому просто запустите команду rm -rf.git в корне вашего приложения.

#### 2.13\*: Данные по странам, шаг 2

**В этой части еще многое предстоит сделать, так что не зацикливайтесь на этом упражнении!**

Улучшите приложение из предыдущего упражнения, чтобы при отображении на странице названий нескольких стран рядом с названием страны была кнопка, при нажатии которой отображается представление для этой страны:

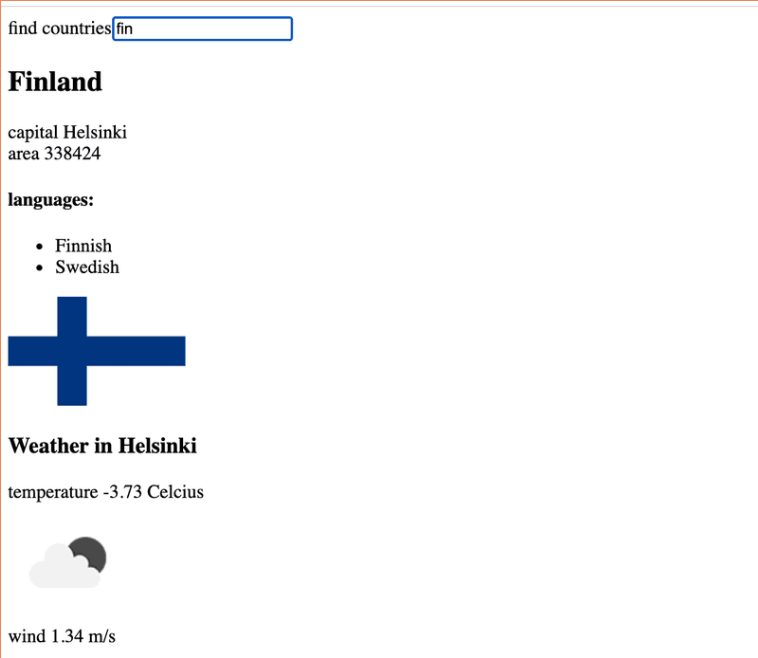


В этом упражнении также достаточно, чтобы ваше приложение работало в большинстве стран. Страны, название которых встречается в названии другой страны, например, *Судан* , можно игнорировать.

#### 2.14\*: Данные по странам, шаг 3

**В этой части еще многое предстоит сделать, так что не зацикливайтесь на этом упражнении!**

Добавьте к представлению, показывающему данные по одной стране, отчет о погоде для столицы этой страны. Существуют десятки поставщиков данных о погоде. Один из предлагаемых API — [https://openweathermap.org](https://openweathermap.org/) . Обратите внимание, что может пройти несколько минут, прежде чем сгенерированный ключ API станет действительным.

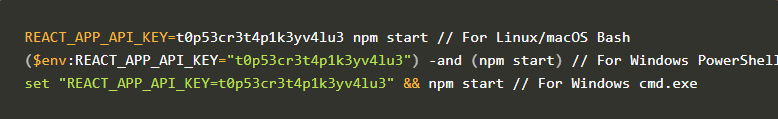


Если вы используете открытую карту погоды, [вот](https://openweathermap.org/weather-conditions#Icon-list) описание того, как получить значки погоды.

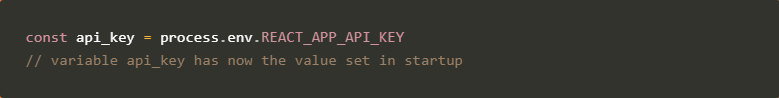
**NB:** В некоторых браузерах (например, Firefox) выбранный API может отправить ответ об ошибке, который указывает, что шифрование HTTPS не поддерживается, хотя URL-адрес запроса начинается с http:// . Эту проблему можно решить, выполнив упражнение в Chrome.

**NB:** Вам нужен API-ключ, чтобы использовать почти каждый погодный сервис. Не сохраняйте API-ключ в систему управления версиями! Не закодируйте API-ключ в исходный код. Вместо этого используйте [переменную среды](https://create-react-app.dev/docs/adding-custom-environment-variables/) для сохранения ключа.

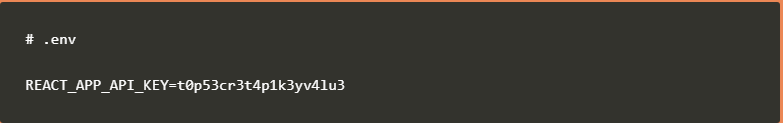
Предполагая, что API-ключ *t0p53cr3t4p1k3yv4lu3* , когда приложение запускается следующим образом:



вы можете получить доступ к значению ключа из объекта process.env :



Обратите внимание: если вы создали приложение с помощью npx create-react-app ...и хотите использовать другое имя для своей переменной среды, то имя переменной среды все равно должно начинаться с REACT\_APP\_. Вы также можете использовать .envфайл вместо того, чтобы каждый раз определять его в командной строке, создавая файл с именем «.env» в корне проекта и добавляя следующее.



Обратите внимание, что вам потребуется перезапустить сервер, чтобы изменения вступили в силу.

### Упражнения 2.15.-2.18.

#### 2.15: Телефонная книга шаг 7

Вернемся к нашему приложению телефонной книги.

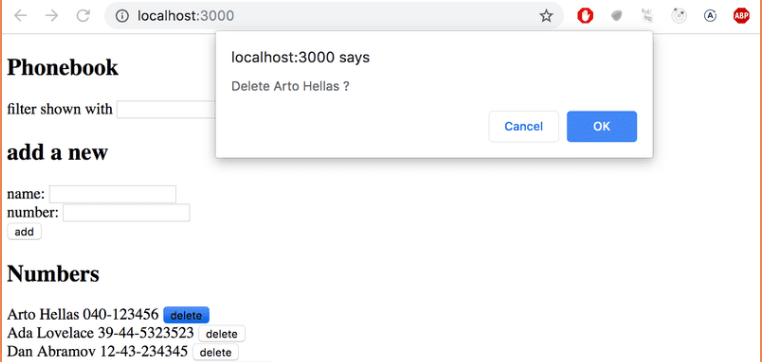
В настоящее время номера, добавленные в телефонную книгу, не сохраняются на внутреннем сервере. Исправьте эту ситуацию.

#### 2.16: Телефонная книга шаг 8

Извлеките код, который обрабатывает связь с серверной частью, в отдельный модуль, следуя примеру, показанному ранее в этой части материала курса.

#### 2.17: Телефонная книга шаг 9

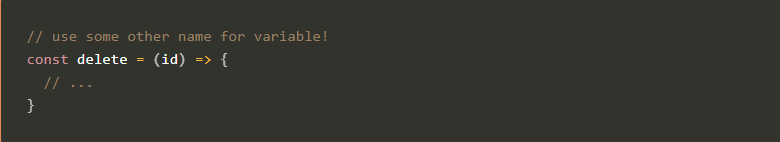
Предоставьте пользователям возможность удалять записи из телефонной книги. Удаление можно выполнить с помощью специальной кнопки для каждого человека в списке телефонной книги. Вы можете подтвердить действие пользователя с помощью метода [window.confirm](https://developer.mozilla.org/en-US/docs/Web/API/Window/confirm) :



Связанный ресурс для человека в бэкэнде можно удалить, отправив HTTP-запрос DELETE на URL-адрес ресурса. Если мы удаляем, например, человека с идентификатором *2* , нам нужно будет сделать HTTP-запрос DELETE на URL-адрес *localhost:3001/persons/2* . Никакие данные не отправляются вместе с запросом.

Вы можете сделать HTTP-запрос DELETE с библиотекой [axios](https://github.com/axios/axios" \t "_blank) так же, как мы делаем все остальные запросы.

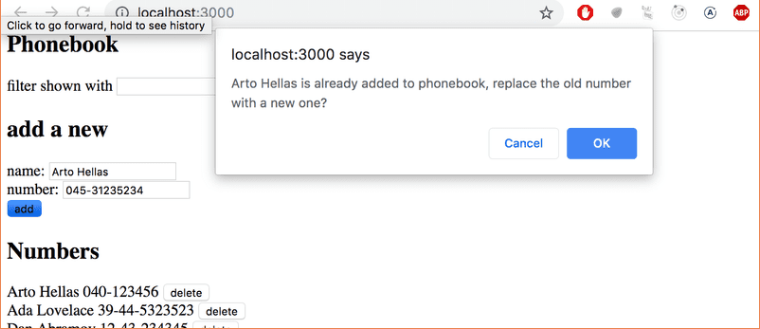
**NB:** вы не можете использовать имя удаления для переменной, потому что это зарезервированное слово в JavaScript. Например, следующее невозможно:



#### 2.18\*: Телефонная книга, шаг 10

Измените функциональность, чтобы при добавлении номера к уже существующему пользователю новый номер заменял старый номер. Для обновления номера телефона рекомендуется использовать метод HTTP PUT.

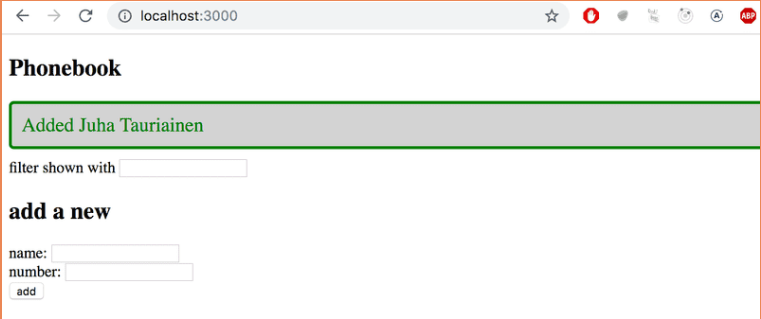
Если информация о человеке уже есть в телефонной книге, приложение может подтвердить действие от пользователя:



### Упражнения 2.19.-2.20.

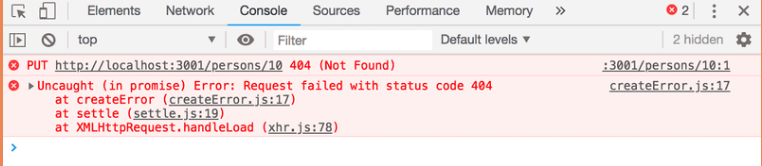
#### 2.19: Телефонная книга шаг 11

Используйте [улучшенный](https://fullstackopen.com/en/part2/adding_styles_to_react_app#improved-error-message) пример сообщения об ошибке из части 2 в качестве руководства, чтобы показать уведомление, которое длится несколько секунд после успешного выполнения операции (добавления человека или изменения номера):

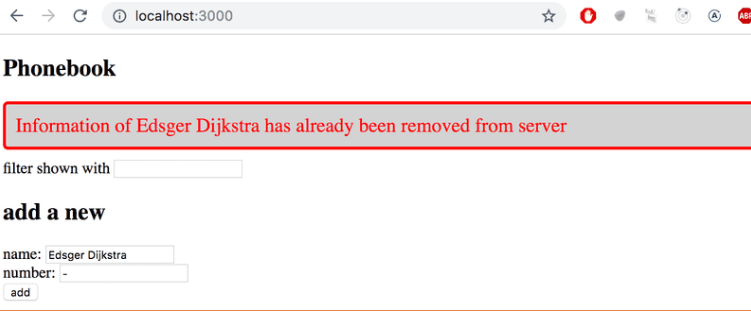


#### 2.20\*: Телефонная книга шаг 12

Откройте ваше приложение в двух браузерах. **Если вы удалите человека в браузере 1** незадолго до того, как попытаетесь *изменить номер телефона человека* в браузере 2, вы получите следующее сообщение об ошибке:



Исправьте проблему в соответствии с примером, показанным в [обещании и ошибках](https://fullstackopen.com/en/part2/altering_data_in_server#promises-and-errors) в части 2. Измените пример, чтобы пользователю показывалось сообщение, когда операция не удалась. Сообщения, отображаемые для успешных и неуспешных событий, должны выглядеть по-разному:



**Обратите внимание** , что даже если вы обрабатываете исключение, сообщение об ошибке выводится на консоль.

Это было последнее упражнение в этой части курса. Пришло время отправить ваш код на GitHub и отметить все выполненные вами упражнения в [системе отправки упражнений](https://studies.cs.helsinki.fi/stats/courses/fullstackopen) .