


Список тренингов (/teach/control/stream/index) /

Профессия Джуниор Frontend-разработчик /

(/teach/control/stream/view/id/695281059)

Модуль 2. Frontend (/teach/control/stream/view/id/701720009) /

3. Композиция компонентов (/teach/control/stream/view/id/751162787)

 Содержание

# Задание #1

Предыдущий урок (/pl/t...

4. Формы

## Реализовать игру «Крестики-Нолики»:

- игровое поле — 3x3 клетки;
- над полем информация — чей текущий ход (крестика или нолика) / информация о победе одной из сторон или ничья;
- при клике на клетку в ней должен отрисовываться символ стороны, у которой был текущий ход (крестик или нолик);
- если 3 одинаковых символа размещается в одну линию (горизонтально, вертикально или по диагонали), то остановить игру и сообщить о победе крестика или нолика;
- реализовать кнопку «Начать заново», при клике на которую поле будет очищаться и игра начнётся сначала;
- дизайн на усмотрение разработчика.

## План реализации:

**Подготовьте стартовый React-проект.** В качестве стартового проекта используйте шаблон приложения, который мы подготовили в первом уроке (с настроенными ESLint, Prettier и EditorConfig).

**Строгий режим.** Убедитесь, что корневой компонент обернут в StrictMode.

**Подготовьте компоненты.** В проекте создайте следующие компоненты (пока пустые):

- `Field` — будет игровым полем с клетками, где каждая клетка представляет из себя кнопку. При нажатии на определенную клетку

будет происходить ход одной из сторон;

- `Information` — будет выводить, чей на данный момент ход (крестика или нолика), либо информацию о победе одной из сторон;
- `Game` (или `App`) — корневой компонент. Будет выводить компоненты `Information`, `Field`, а также кнопку «Начать заново».

Уведомления

Магазин

Каждый из этих компонентов является `stateful`-контейнером, который внутри содержит `stateless`-шаблон. То есть компонент `Field` (или их можно называть по типу `FieldContainer`) является компонентом-контейнером, который внутри выводит компонент-шаблон `FieldLayout`.

Обучение

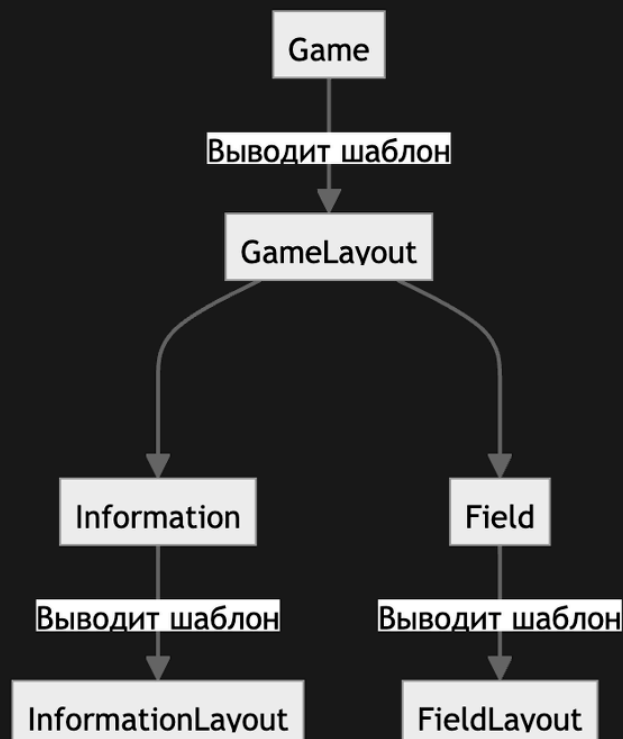
Аналогично `Information` выводит `InformationLayout`, `Game` ВЫВОДИТ `GameLayout`.

Сообщения

Покупки

Создайте компоненты-шаблоны, соедините все компоненты друг с другом и выведите их в соответствии с иерархией:

```
└─ Game
  └─ GameLayout
    ├── Information
    │   └─ InformationLayout
    └─ Field
        └─ FieldLayout
```



Внутри компонентов-шаблонов ( `GameLayout` , `InformationLayout` , `FieldLayout` ) подготовьте всю верстку игры (в статичном виде, без функционала). При необходимости для компонентов создавайте CSS-модули, они будут отдельными для каждого компонента.

Уведомления

**Подготовьте состояния.** В корневом компоненте `Game` создаем следующие состояния:

Магазин

- `currentPlayer` — кто ходит в данный момент, значениями может быть `'x'` или `'o'` . По умолчанию — `'x'` ;
- `isGameEnded` — была ли завершена игра. По умолчанию `false` ;
- `isDraw` — была ли ничья. По умолчанию `false` ;
- `field` — массив клеток игрового поля, состоящий из 9 пустых строк (3x3). Будет храниться массив вида:

Обучение

Сообщения

Покупки

```
// Для наглядности перенесем на 3 строки
[
  '', '', '',
  '', '', '',
  '', '', ''
]
```

**Выведите данные.** Передаем состояния в нужные компоненты (через промежуточные компоненты) и выводим в шаблон:

- `FieldLayout` выводит массив `field` (с помощью метода `map()` );
- `InformationLayout` выводит статус:
  - Если `isDraw` равен `true` — `'Ничья'` ;
  - Если `isDraw` равен `false` , но `isGameEnded` равен `true` — ``Победа: ${currentPlayer}`` ;
  - Если `isDraw` равен `false` и `isGameEnded` равен `false` — ``Ходит: ${currentPlayer}`` .

Попробуйте вручную поменять значения состояний и убедитесь, что вывод компонентов меняется.

**Напишите логику игры.**

При клике на клетку устанавливайте символ ходившего игрока ( `currentPlayer` ) в выбранную ячейку массива `field` (при условии, что выбранная ячейка пуста и игра не завершена).

В процессе игры состояние `field` может выглядеть следующим образом:

```
// Для наглядности перенесем на 3 строки
[
  'X', '0', '',
  'X', 'X', '',
  '', '', '0',
]
```

В данном случае клетки с индексами 0, 1, 3, 4, 8 недоступны для выбора.

Кроме того, при клике на ячейку необходимо определять, выиграл ли кто-то после выполненного хода. Могут быть следующие ситуации:

- Если кто-то выиграл — устанавливаем в `isGameEnded` значение `true` ;
- Если победителей нет и нет пустых ячеек — устанавливаем в `isDraw` значение `true` (ничья);
- Если победителей нет и есть пустые ячейки — устанавливаем в `currentPlayer` игрока, который будет следующим совершать ход ( `currentPlayer === 'X' ? '0' : 'X'`  ).

Для того чтобы определить, кто является победителем, мы можем сравнить `field` с заранее определенными вариантами побед:

```
const WIN_PATTERNS = [
  [0, 1, 2], [3, 4, 5], [6, 7, 8], // Варианты побед по горизонтали
  [0, 3, 6], [1, 4, 7], [2, 5, 8], // Варианты побед по вертикали
  [0, 4, 8], [2, 4, 6] // Варианты побед по диагонали
];
```

Если массив `field` содержит один из вариантов побед (когда одинаковый символ игрока стоит на победных индексах), то игра завершается и данный игрок считается победителем. Например, если `field[1]` , `field[4]` , `field[7]` равны `'X'` , то это будет считаться победой `'X'` .

Для проверки массивов можно использовать цикл `for` или встроенные методы массивов (`some()` , `every()` и т. д.).

При клике на кнопку «Начать заново» для состояний `currentPlayer` , `isGameEnded` , `isDraw` , `field` устанавливаем начальные значения.

**Типизируйте props.** В компонентах, которые принимают props, используйте `PropTypes` для указания конкретных типов передаваемых значений.

**Примечание:** выполните задание и прикрепите ссылку на ответ в форме отправки ответа ниже. После того как наставник примет ваш ответ, здесь будет открыто решение.

**Дедлайн** для получения **бонусных монет** по заданию — до 15.09 включительно.

**Дедлайн** для получения **обратной связи от наставника** по заданию — до 22.09 включительно.

**Каждый вопрос, который у вас возникает при изучении материала - это звено цепочки изучения и понимания JavaScript.** Если пропустить/отложить какой-то вопрос, то в дальнейшем это увеличит количество вопросов, создаст непонимание и, со временем, учиться станет все сложнее и сложнее. **НЕ ДОПУСКАЙТЕ** этого.

**Задавайте все вопросы, которые у вас возникают СЕЙЧАС в процессе изучения.**

**ВСЕГДА** пишите наставнику и/или в чат группы, чтобы его поскорее прояснить. Это сэкономит ваше время в будущем и укрепит цепочку понимания в настоящем.

# Задание



Эрдэни Чимитов

Уведомления

Ваш ответ

Магазин

Обучение

Добавить файлы

Сообщения

максимальный размер файла - 100мб

Покупки

Отправить ответ

Сохранить черновик

50 монет за выполнение этого урока