

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 42

КУРСОВАЯ РАБОТА
ЗАЩИЩЕНА С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ

доцент
должность, уч. степень, звание

подпись, дата

О. И. Красильникова
инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

по дисциплине: Web-технологии

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

4128

подпись, дата

Воробьев. В.А.
инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1	ВВЕДЕНИЕ	3
2	Модель сайта	4
2.1	Применение CSS	4
2.2	Графическое оформление web-страницы	10
3	Web-сценарии сайта на языке TypeScript	14
3.1	Назначение TypeScript	14
3.2	Разработка TS скриптов	14
3.3	State-management	15
4	Валидация кода и продвижение сайта	16
4.1	Использование приемов продвижения	16
5	Заключение	18
	ПРИЛОЖЕНИЕ	19

1 ВВЕДЕНИЕ

В современном мире веб-технологии играют важную роль в развитии информационных систем и коммуникаций. Разработка сайтов стала неотъемлемой частью бизнеса, образования и повседневной жизни людей. В связи с этим возникает необходимость изучения основ веб-технологий и овладения навыками создания качественных и функциональных веб-ресурсов.

Целью данной курсовой работы является изучение основных принципов и методов разработки веб-сайтов, а также практическое применение полученных знаний для создания собственного сайта. В ходе выполнения работы будут рассмотрены различные аспекты веб-разработки, включая проектирование структуры сайта, создание страниц с использованием HTML и CSS, а также разработку интерактивных элементов с применением TypeScript и React.

Согласно уточненным требованиям, нами осуществлена разработка сайта, предназначенного исключительно для игры в блекджек без необходимости регистрации и онлайн-режима.

Сайт представляет собой игровую платформу, на которой пользователи могут свободно играть в блекджек без использования виртуальной валюты или участия профессионального дилера. Игровой процесс реализован в соответствии с международными правилами и стандартами, что обеспечивает пользователям честную и прозрачную игру.

Интерфейс сайта выполнен в минималистичном стиле и не содержит ненужных элементов, что позволяет пользователям сосредоточиться на игровом процессе. Сайт не предусматривает сбора персональных данных пользователей и не требует регистрации, что гарантирует сохранение конфиденциальности и анонимности игроков.

Таким образом, данный сайт представляет собой простую и удобную платформу для игры в блекджек без необходимости регистрации, обеспечивающую честной и прозрачной игрой в соответствии с международными стандартами.

2 Модель сайта

Макет сайта был реализован в Figma и изображен на рисунке 2.1.



Рисунок 2.1 - Дизайн в Figma

Макет состоит из:

- 1) Статуса. Отображает текущее состояния игрока во время партии.
- 2) Денег. Виртуальная валюта, необходимая для игры.
- 3) Кнопок действия. Кнопки, с помощью которых игрок может предпринимать действия.
- 4) Руки дилера и игрока. Отображает текущий счет и карты.

Дизайн является адаптивным и с помощью медиазапросов изменяется размер элементов сайта.

2.1 Применение CSS

Для реализации красивого дизайна похожего на дизайн-систему Cupertino было решено применить CSS.

Выделю главные преимущества, которое нам дало CSS.

Определение цветов темы:

```
1  :root {
2    --card-bg: #f2f2f2;
3    --card-shadow: rgba(0, 0, 0, 0.25);
4    --hidden-card-bg: rgb(230, 230, 230);
5    --text-black: black;
6    --text-red: red;
7  }
8
9
10 * {
11   margin: 0;
12   padding: 0;
13 }
14
15 body {
16   background: var(--background-color);
17   margin: 0;
18   font-family: 'Lexend Exa', sans-serif;
19   -webkit-font-smoothing: antialiased;
20 }
```

Определение цветов кнопок.

```
1  :root {
2    --primary-color: #fff;
3    --secondary-color: #000;
4    --shadow-color: rgba(0, 0, 0, 0.2);
5    --background-color: #f2f2f2;
6  }
7
8  .controlsContainer {
9    display: flex;
10   justify-content: center;
11   margin: 0.5em 1em 1em 1em;
12   background-color: var(--background-color);
13 }
14
15 .betContainer {
16   display: flex;
17   align-items: center;
18   margin: 0 0.5em;
19   padding: 0 1em;
```

```

20     width: 40%;
21     border-radius: 10px;
22     background-color: var(--primary-color);
23     box-shadow: 0px 3px 6px var(--shadow-color);
24 }
25
26 .input {
27     width: 1px;
28     flex-grow: 1;
29     margin: 0 0 0 0.5em;
30     font-size: 200%;
31     text-align: right;
32     margin: 5px;
33     padding: 0;
34     border: 0;
35     outline: 0;
36     background-color: var(--primary-color);
37     color: var(--secondary-color);
38 }
39
40 .inputError {
41     width: 1px;
42     flex-grow: 1;
43     margin: 0 0 0 0.5em;
44     font-size: 200%;
45     text-align: right;
46     margin: 5px;
47     padding: 0;
48     border: 0;
49     outline: 0;
50     background-color: var(--primary-color);
51     color: red;
52 }
53
54 .input::-webkit-inner-spin-button,
55 .inputError::-webkit-inner-spin-button {
56     margin: 0;
57     -webkit-appearance: none;
58 }
59
60 .button {

```

```

61     color: var(--secondary-color);
62     font-weight: bold;
63     margin: 0 0.5em;
64     padding: 1em;
65     width: 30%;
66     outline: none;
67     background-color: var(--primary-color);
68     border-radius: 10px;
69     box-shadow: 0px 3px 6px var(--shadow-color);
70     text-align: center;
71     cursor: pointer;
72     border: none;
73     transition: all 0.3s ease;
74 }
75
76 @media (hover: hover) {
77     .button:hover {
78         background: rgba(0, 0, 0, 0.05);
79     }
80 }
81
82 .button:disabled {
83     color: gray;
84     background: rgb(245, 245, 245);
85 }
86
87 @media screen and (max-width: 992px) {
88     .betContainer {
89         width: 50%;
90     }
91 }
92
93 @media screen and (max-width: 600px) {
94     .betContainer {
95         width: 70%;
96     }
97
98     .betContainer h4 {
99         font-size: 75%;
100    }
101

```

```

102 .betContainer input {
103     font-size: 125%;
104 }
105 }

```

Определение дизайна карт.

```

1  :root {
2      --primary-color: #fff;
3      --secondary-color: #000;
4      --shadow-color: rgba(0, 0, 0, 0.2);
5      --background-color: #f2f2f2;
6  }
7
8  .controlsContainer {
9      display: flex;
10     justify-content: center;
11     margin: 0.5em 1em 1em 1em;
12     background-color: var(--background-color);
13 }
14
15 .betContainer {
16     display: flex;
17     align-items: center;
18     margin: 0 0.5em;
19     padding: 0 1em;
20     width: 40%;
21     border-radius: 10px;
22     background-color: var(--primary-color);
23     box-shadow: 0px 3px 6px var(--shadow-color);
24 }
25
26 .input {
27     width: 1px;
28     flex-grow: 1;
29     margin: 0 0 0 0.5em;
30     font-size: 200%;
31     text-align: right;
32     margin: 5px;
33     padding: 0;
34     border: 0;
35     outline: 0;
36     background-color: var(--primary-color);

```



```

37     color: var(--secondary-color);
38 }
39
40 .inputError {
41     width: 1px;
42     flex-grow: 1;
43     margin: 0 0 0 0.5em;
44     font-size: 200%;
45     text-align: right;
46     margin: 5px;
47     padding: 0;
48     border: 0;
49     outline: 0;
50     background-color: var(--primary-color);
51     color: red;
52 }
53
54 .input::-webkit-inner-spin-button ,
55 .inputError::-webkit-inner-spin-button {
56     margin: 0;
57     -webkit-appearance: none;
58 }
59
60 .button {
61     color: var(--secondary-color);
62     font-weight: bold;
63     margin: 0 0.5em;
64     padding: 1em;
65     width: 30%;
66     outline: none;
67     background-color: var(--primary-color);
68     border-radius: 10px;
69     box-shadow: 0px 3px 6px var(--shadow-color);
70     text-align: center;
71     cursor: pointer;
72     border: none;
73     transition: all 0.3s ease;
74 }
75
76 @media (hover: hover) {
77     .button:hover {

```

```

78     background: rgba(0, 0, 0, 0.05);
79 }
80 }
81
82 .button:disabled {
83     color: gray;
84     background: rgb(245, 245, 245);
85 }
86
87 @media screen and (max-width: 992px) {
88     .betContainer {
89         width: 50%;
90     }
91 }
92
93 @media screen and (max-width: 600px) {
94     .betContainer {
95         width: 70%;
96     }
97
98     .betContainer h4 {
99         font-size: 75%;
100    }
101
102     .betContainer input {
103         font-size: 125%;
104     }
105 }

```

2.2 Графическое оформление web-страницы

Ниже представлен визуал страниц:

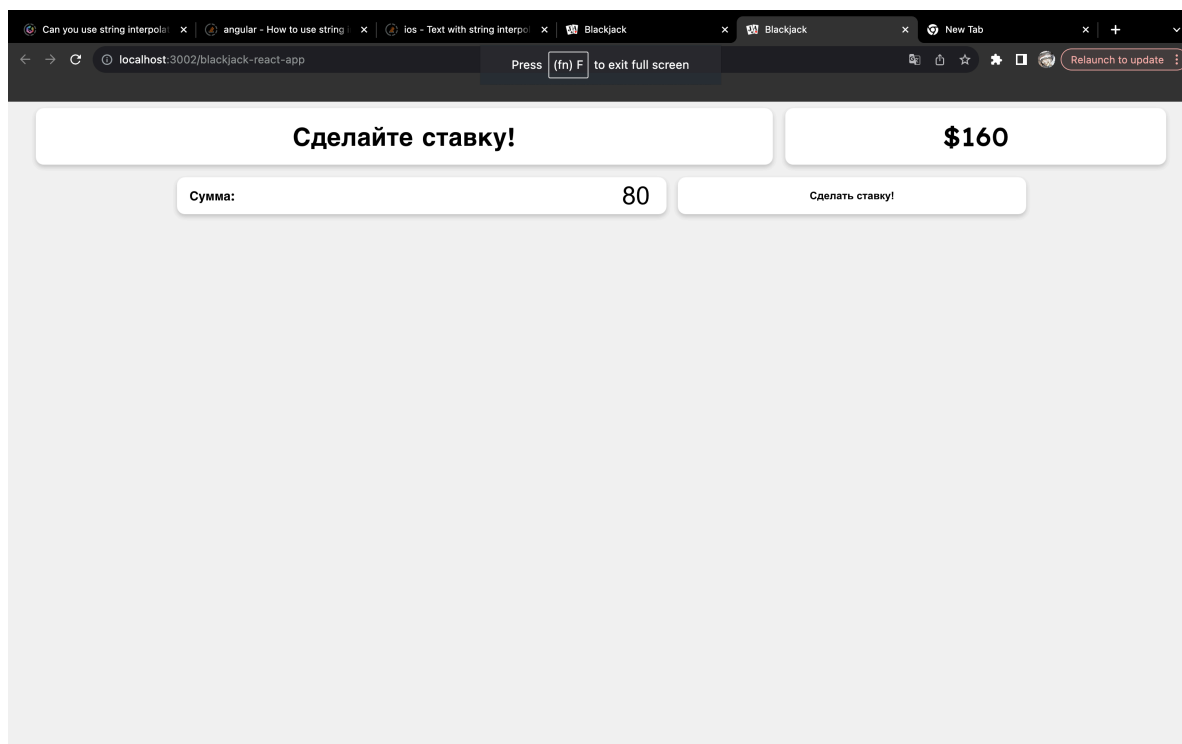


Рисунок 2.2 - Начальная страница

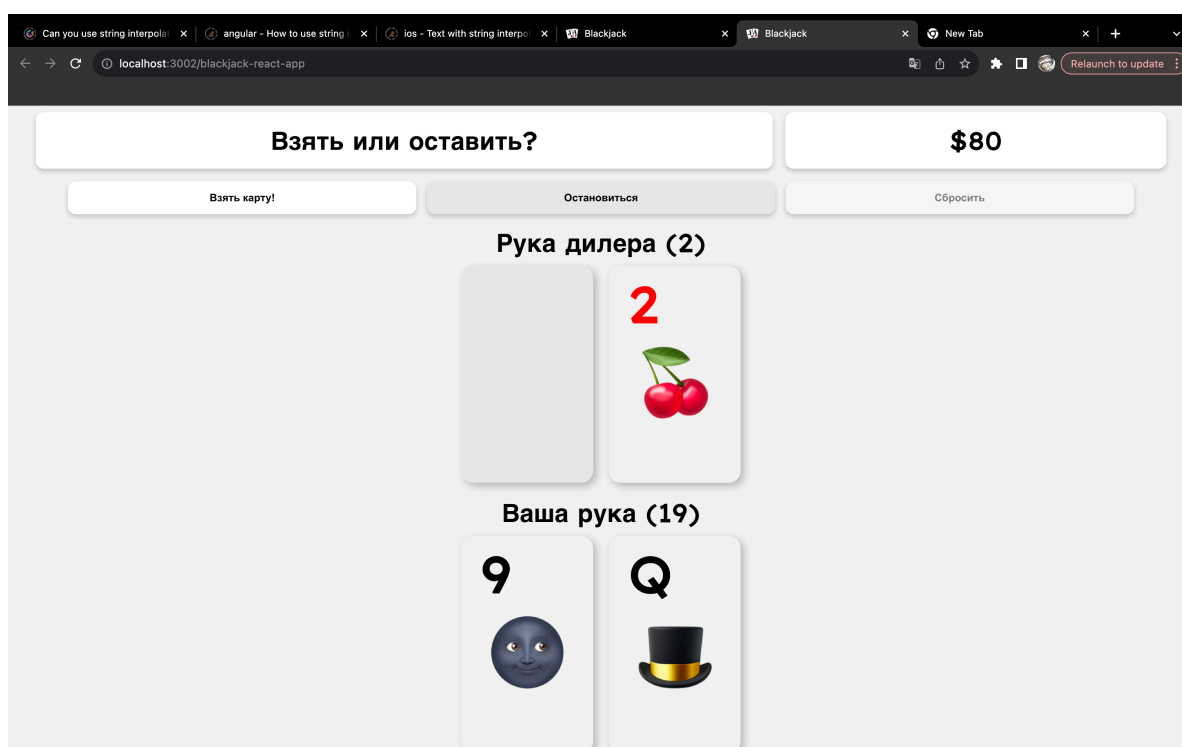


Рисунок 2.3 - Начало игры

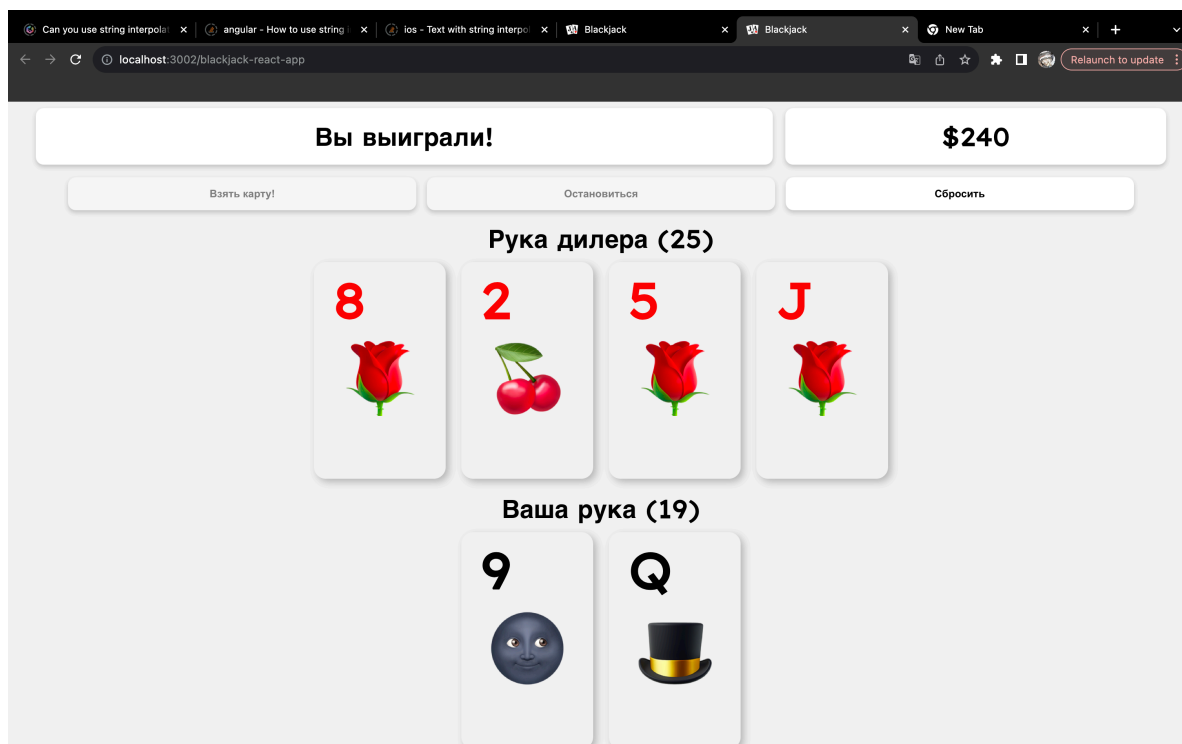


Рисунок 2.4 - Выигрыш игрока

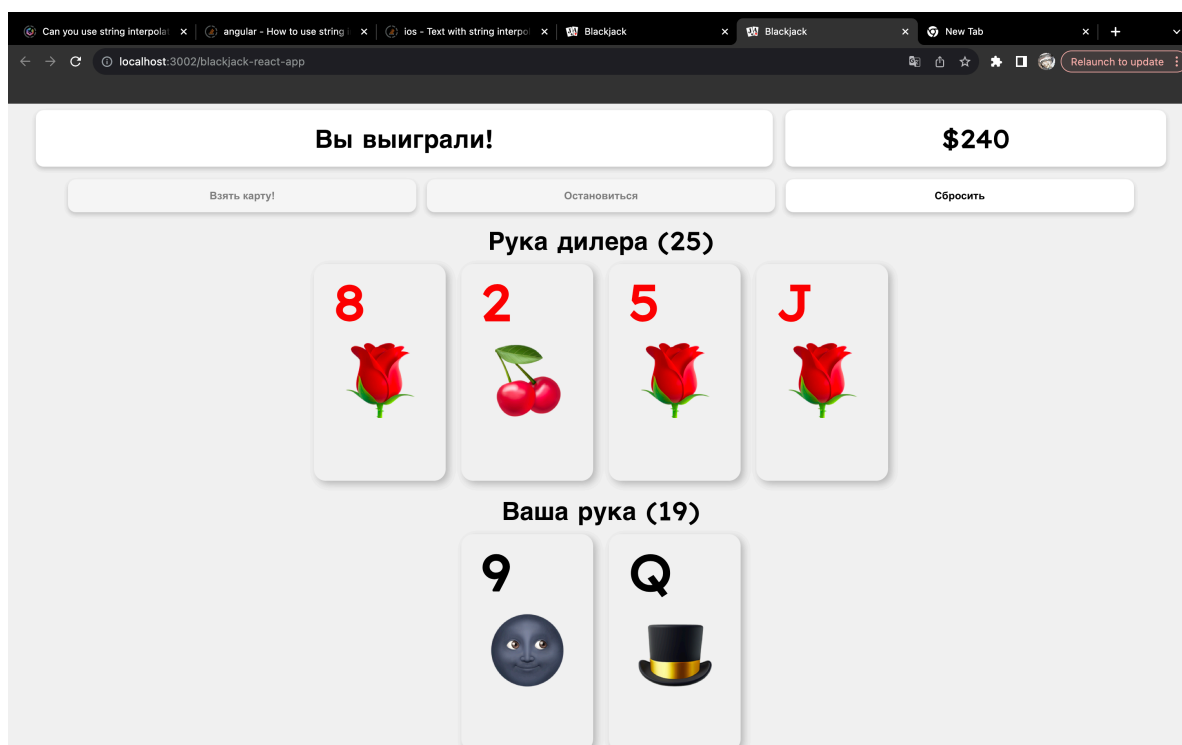


Рисунок 2.5 - Выигрыш игрока

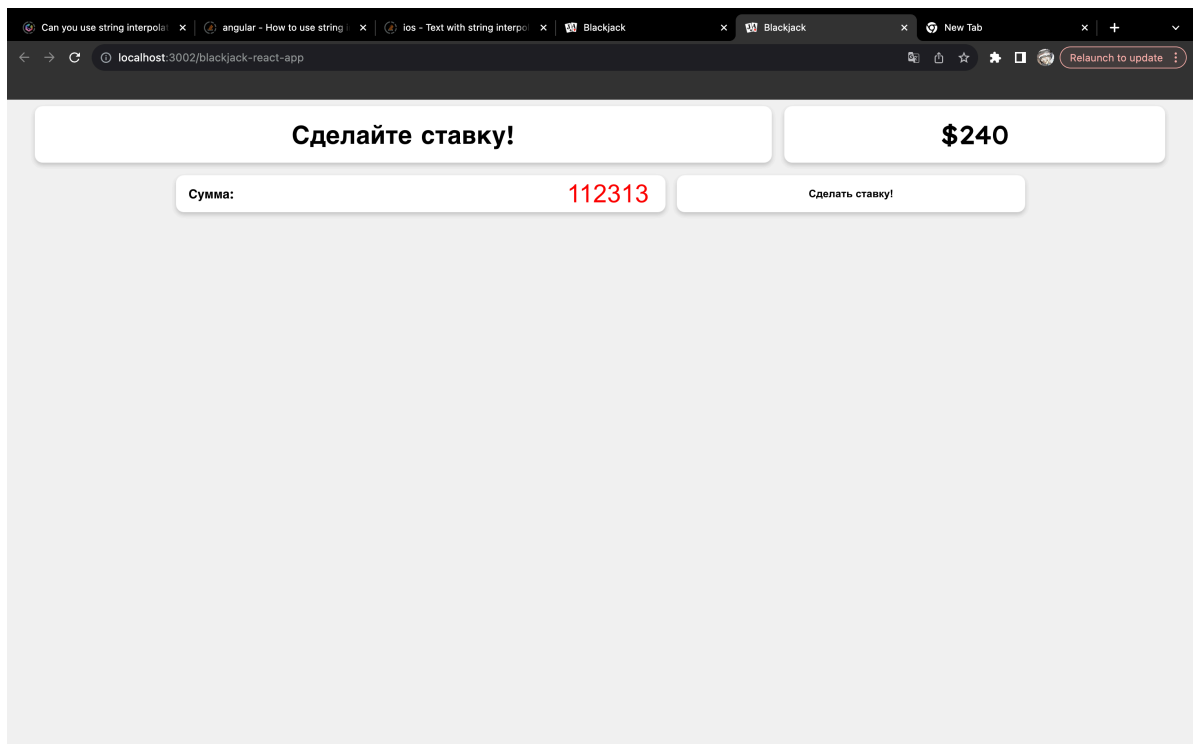


Рисунок 2.6 - Валидация текста

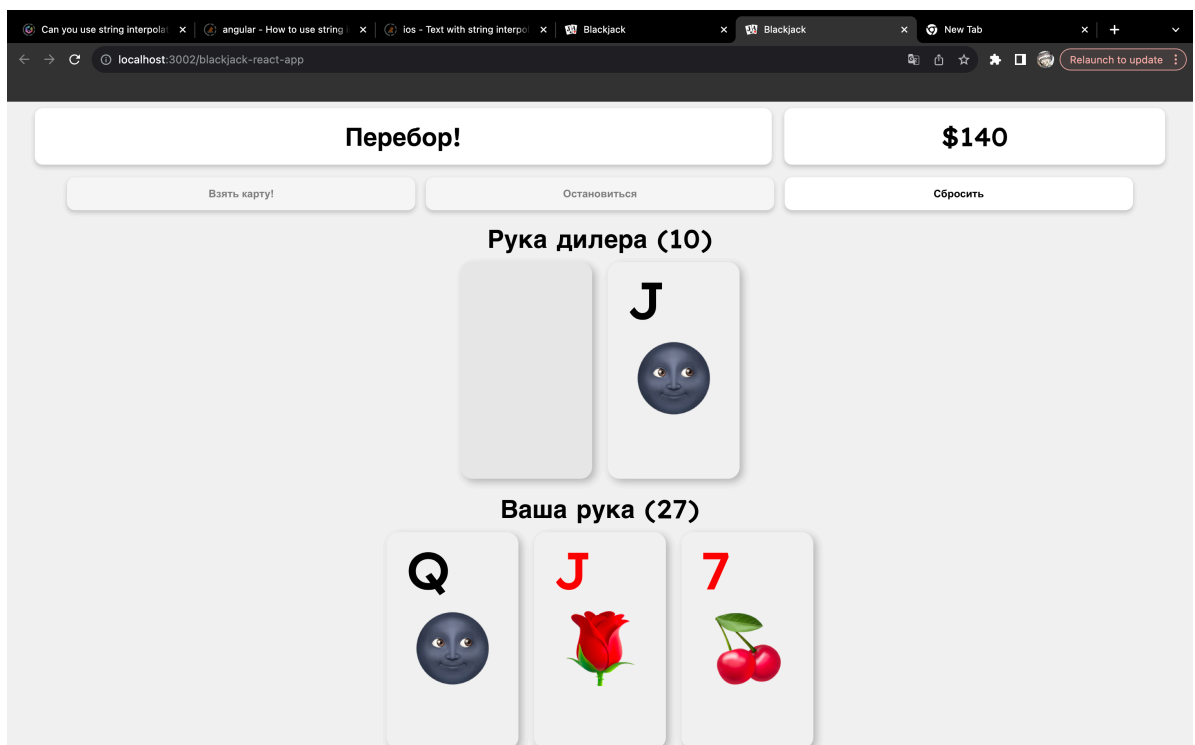


Рисунок 2.7 - Перебор карты

3 Web-сценарии сайта на языке TypeScript

Для реализации динамического поведения, которое необходимо для проекта такой направленной было решено выбрать TypeScript.

Выбор обусловлен:

- 1) Наличием строгой типизации.
- 2) Приятным и удобным синтаксисом.
- 3) Поддержка современными библиотеками.

3.1 Назначение TypeScript

В данном коде TypeScript используется для указания типов переменных и аргументов функций. Например, для переменной `deck` указан тип `any[]`, что означает, что она может содержать массив любых значений. Для переменных `userCards` и `dealerCards` также указан тип `any[]`.

Для аргументов функций также указаны типы. Например, для функции `placeBet` указан тип аргумента `amount` как `number`, что означает, что в эту функцию можно передать только числовое значение.

Кроме того, в коде использованы перечисления (`enum`) `GameState` и `Deal`, которые позволяют явно указать допустимые значения для соответствующих переменных и улучшить читаемость кода.

Наконец, в коде используются конструкции `useState` и `useEffect` из библиотеки `React`, для которых также указаны типы аргументов и возвращаемых значений с помощью TypeScript.

3.2 Разработка TS скриптов

Этот код представляет собой компонент `React`, который реализует игру в блэкджек. Он включает в себя несколько подкомпонентов, таких как `Status`, `Controls` и `Hand`. Состояние игры и ее логика управляются с помощью хуков `React` `useState` и `useEffect`.

Игра начинается с того, что игрок делает ставку, затем выдаются две карты как игроку, так и дилеру. Ход игрока идет первым, и он может выбрать взять еще одну карту (`hit`) или закончить свой ход (`stand`). Если сумма очков игрока превышает 21, он проигрывает (`bust`). Если игрок остановился, наступает ход дилера. Дилер обязан брать карты, пока сумма его очков не достигнет 17 или более. Победитель определяется на основе того, кто имеет большую сумму очков, не превышая 21.

Компонент использует массив объектов карт для представления колоды, и включает функции для выдачи карт, подсчета очков и определения победителя. Также отслеживается и обновляется текущий баланс и сумма ставки в зависимости от результата каждой игры.

3.3 State-management

В этом коде используется управление состоянием React с помощью хуков `useState` и `useEffect` для управления игрой в блэкджек.

Хук `useState` используется для создания нескольких переменных состояния, таких как `deck`, `userCards`, `userScore`, `dealerCards`, `dealerScore`, `balance`, `bet`, `gameState`, `message` и `buttonState`. Эти переменные состояния используются для хранения текущего состояния игры, включая колоду карт, карты игрока и дилера, текущий счет, баланс, ставку, сообщение и состояние кнопок.

Хук `useEffect` используется для выполнения побочных эффектов в зависимости от изменений в состоянии. Например, когда изменяется состояние `gameState`, вызывается функция `drawCard` для выдачи карт. Когда изменяются `userCards` или `dealerCards`, вызывается функция `calculate` для подсчета очков.

Кроме того, в этом коде используются другие функции для управления состоянием игры, такие как `resetGame`, `placeBet`, веб-страниц на соответствие стандартам разметки, определенным Консорциумом Всемирной паутины (W3C). Валидация кода важна для обеспечения правильной работы сайта во всех браузерах и устройствах, а также для улучшения SEO-оптимизации. Ошибки в коде могут приводить к плохому отображению сайта, медленной загрузке и низкому рейтингу в поисковых системах. `drawCard`, `dealCard`, `revealCard`, `hit`, `stand`, `bust` и `checkWin`. Все эти функции обновляют состояние игры, вызывая соответствующие функции обновления состояния, переданные из хука `useState`.

Общий подход к управлению состоянием в этом коде состоит в том, чтобы хранить все необходимые данные в состоянии React и обновлять их с помощью функций обновления состояния при необходимости. Это позволяет React отслеживать изменения состояния и автоматически обновлять компонент, когда это необходимо.

4 Валидация кода и продвижение сайта

Форматтер

Форматтер - это инструмент, который автоматически форматирует код в соответствии с определенными стандартами и правилами. Использование форматтера помогает сохранять код чистым, упорядоченным и легко читаемым, что облегчает его поддержку и сопровождение. Кроме того, форматтер может помочь избежать ошибок, связанных с неправильным форматированием кода.

В качестве форматера используется Prettier.

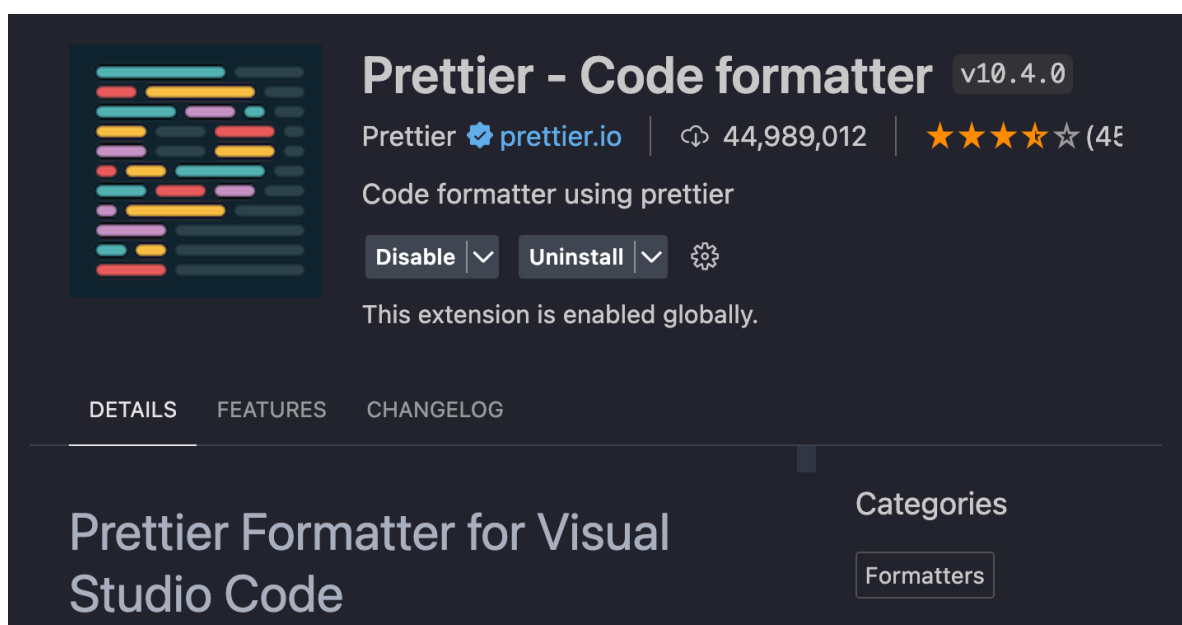


Рисунок 4.1 - Страница на VS Code

Метаданные сайта

Метаданные - это информация о сайте, которая не отображается на странице, но используется поисковыми системами и другими веб-сервисами для определения содержания и цели сайта. Метаданные могут включать в себя заголовок страницы, описание, ключевые слова, язык, авторство и другие сведения. Правильное использование метаданных может улучшить SEO-оптимизацию сайта и повысить его рейтинг в результатах поиска.

4.1 Использование приемов продвижения

Были использованы следующие метаданные. Также React из коробки поддерживает некоторые SEO оптимизации для поддержки поиска по сайту.

```
1 <meta name="description"
```



```
2      content="Learn , play and master Blackjack with us . We  
      offer tips , strategies and the best Blackjack games  
      online ." />  
3 <meta name="keywords" content="Blackjack , online Blackjack ,  
      Blackjack strategies , Blackjack tips , Blackjack games" />  
4 <meta name="author" content="Your Name or Your Company Name"  
    />
```

5 Заключение

В ходе этой работы были изучены основы создания веб-сайтов с использованием React и TypeScript. Было пройдено несколько этапов, включая создание компонентов, работу с состоянием и пропсами, управление жизненным циклом компонентов.

Одним из ключевых моментов в этой работе было изучение методов продвижения сайта, включая оптимизацию метаданных для поисковых систем, использование ключевых слов и SEO-стратегий.

Благодаря этой работе я приобрел навыки разработки веб-приложений с использованием современных инструментов и библиотек, а также навыки продвижения сайтов в поисковых системах. Эти навыки являются необходимыми для разработки и продвижения конкурентоспособных веб-приложений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Красильникова О.И., Красильников Н.Н. Web-технологии для разработки клиентской части web-страниц: учеб. пособие в 2 ч. Ч. 2 – СПб.: ГУАП. 2018 - 44 с.
- 2 Красильникова, О. И. JavaScript в разработке клиентской части веб-страниц: учеб. пособие / О. И. Красильникова. – СПб.: ГУАП, 2022 – 87 с.
- 3 Диков А. В. Клиентские технологии веб-программирования: JavaScript и DOM: учебное пособие / А. В. Диков. СПб.: Лань, 2020 124 с.
- 4 Красильникова О.И. Основы frontend-разработки. HTML и CSS: учеб.-метод. пособие: / О. И. Красильникова. – СПб.: ГУАП, 2023 – 72 с.

ПРИЛОЖЕНИЕ

```
1  .card {
2    width: 120px;
3    height: 260px;
4    margin: 10px;
5    padding: 0.5em 1.5em;
6    background: var(--card-bg);
7    border-radius: 15px;
8    box-shadow: 4px 4px 10px var(--card-shadow);
9    cursor: default;
10   transition: transform 0.3s ease;
11 }
12
13 .card:hover {
14   transform: scale(1.1);
15 }
16
17 .hiddenCard {
18   width: 120px;
19   height: 260px;
20   margin: 10px;
21   padding: 0.5em 1.5em;
22   background: var(--hidden-card-bg);
23   background-size: 60.00px 4.20px;
24   border: 5px solid var(--hidden-card-border);
25   border-radius: 15px;
26   box-shadow: 4px 4px 10px var(--card-shadow);
27   cursor: default;
28   transition: transform 0.3s ease;
29 }
30
31 .hiddenCard:hover {
32   transform: scale(1.1);
33 }
34
35 .black {
36   color: var(--text-black);
37 }
38
39 .red {
40   color: var(--text-red);
```

```

41 }
42
43 .value {
44     font-size: 400%;
45     margin: 0;
46 }
47
48 .suit {
49     font-size: 600%;
50     margin: 0;
51     text-align: center;
52 }
53
54 @media screen and (max-width: 992px) {
55     .card {
56         width: 70px;
57         height: 180px;
58     }
59
60     .hiddenCard {
61         width: 60px;
62         height: 170px;
63     }
64
65     .value {
66         font-size: 300%;
67     }
68
69     .suit {
70         font-size: 500%;
71     }
72 }
73
74 @media screen and (max-width: 600px) {
75     .card {
76         width: 45px;
77         height: 100px;
78         padding: 5px 10px;
79     }
80
81     .hiddenCard {

```

```

82     width: 41px;
83     height: 96px;
84     padding: 5px 10px;
85     border: 2px solid white;
86 }
87
88 .value {
89     font-size: 150%;
90 }
91
92 .suit {
93     font-size: 250%;
94 }
95 }

```

```

1  :root {
2    --primary-color: #fff;
3    --secondary-color: #000;
4    --shadow-color: rgba(0, 0, 0, 0.2);
5    --background-color: #f2f2f2;
6  }
7
8  .controlsContainer {
9    display: flex;
10   justify-content: center;
11   margin: 0.5em 1em 1em 1em;
12   background-color: var(--background-color);
13 }
14
15 .betContainer {
16   display: flex;
17   align-items: center;
18   margin: 0 0.5em;
19   padding: 0 1em;
20   width: 40%;
21   border-radius: 10px;
22   background-color: var(--primary-color);
23   box-shadow: 0px 3px 6px var(--shadow-color);
24 }
25
26 .input {
27   width: 1px;

```

```

28     flex-grow: 1;
29     margin: 0 0 0 0.5em;
30     font-size: 200%;
31     text-align: right;
32     margin: 5px;
33     padding: 0;
34     border: 0;
35     outline: 0;
36     background-color: var(--primary-color);
37     color: var(--secondary-color);
38 }
39
40 .inputError {
41     width: 1px;
42     flex-grow: 1;
43     margin: 0 0 0 0.5em;
44     font-size: 200%;
45     text-align: right;
46     margin: 5px;
47     padding: 0;
48     border: 0;
49     outline: 0;
50     background-color: var(--primary-color);
51     color: red;
52 }
53
54 .input::-webkit-inner-spin-button,
55 .inputError::-webkit-inner-spin-button {
56     margin: 0;
57     -webkit-appearance: none;
58 }
59
60 .button {
61     color: var(--secondary-color);
62     font-weight: bold;
63     margin: 0 0.5em;
64     padding: 1em;
65     width: 30%;
66     outline: none;
67     background-color: var(--primary-color);
68     border-radius: 10px;

```

```

69     box-shadow: 0px 3px 6px var(--shadow-color);
70     text-align: center;
71     cursor: pointer;
72     border: none;
73     transition: all 0.3s ease;
74 }
75
76 @media (hover: hover) {
77     .button:hover {
78         background: rgba(0, 0, 0, 0.05);
79     }
80 }
81
82 .button:disabled {
83     color: gray;
84     background: rgb(245, 245, 245);
85 }
86
87 @media screen and (max-width: 992px) {
88     .betContainer {
89         width: 50%;
90     }
91 }
92
93 @media screen and (max-width: 600px) {
94     .betContainer {
95         width: 70%;
96     }
97
98     .betContainer h4 {
99         font-size: 75%;
100    }
101
102     .betContainer input {
103         font-size: 125%;
104     }
105 }

```



```

1 .handContainer {
2     color: black;
3     display: flex;
4     align-items: center;

```



```

5     flex-direction: column;
6     margin: 0.5em;
7 }
8
9 .cardContainer {
10     display: flex;
11     justify-content: center;
12     flex-wrap: wrap;
13 }
14
15 .title {
16     text-align: center;
17 }
18
19 @media screen and (max-width: 600px) {
20     .title {
21         font-size: 150%;
22     }
23 }

```

```

1 .statusContainer {
2     display: flex;
3     justify-content: center;
4     background-color: var(--background-color);
5 }
6
7 .status {
8     display: flex;
9     align-items: center;
10    justify-content: center;
11    margin: 0.5em 0.5em 0.5em 1em;
12    padding: 1em;
13    width: 60%;
14    background: var(--primary-color);
15    border-radius: 10px;
16    box-shadow: 0px 3px 6px var(--shadow-color);
17 }
18
19 .balance {
20     display: flex;
21     align-items: center;
22     justify-content: center;

```

```

23   margin: 0.5em 1em 0.5em 0.5em;
24   padding: 1em;
25   width: 30%;
26   background: var(--primary-color);
27   border-radius: 10px;
28   box-shadow: 0px 3px 6px var(--shadow-color);
29 }
30
31 .value {
32   color: var(--secondary-color);
33   text-align: center;
34 }
35
36 @media screen and (max-width: 992px) {
37   .value {
38     font-size: 150%;
39   }
40 }
41
42 @media screen and (max-width: 600px) {
43   .value {
44     font-size: 115%;
45   }
46 }

```

```

1  import React from "react"
2  import styles from "../styles/Card.module.css"
3  import { clubsSymbol, spadesSymbol } from "../Localization"
4
5  type CardProps = {
6    value: string
7    suit: string
8    hidden: boolean
9  }
10
11  const Card: React.FC<CardProps> = ({ value, suit, hidden })
    => {
12    const getColor = () => {
13      if (suit === spadesSymbol || suit === clubsSymbol) {
14        return styles.black
15      } else {
16        return styles.red

```

```

17     }
18   }
19
20   const getCard = () => {
21     if (hidden) {
22       return <div className={styles.hiddenCard} />
23     } else {
24       return (
25         <div className={styles.card}>
26           <div className={getColor()}>
27             <h1 className={styles.value}>{value}</h1>
28             <h1 className={styles.suit}>{suit}</h1>
29           </div>
30         </div>
31       )
32     }
33   }
34
35   return <>{getCard()}</>
36 }
37
38 export default Card

```

```

1 import React, { useState, useEffect } from "react"
2 import styles from "../styles/Controls.module.css"
3 import {
4   amountButton,
5   betButton,
6   hitButton,
7   resetButton,
8   standButton,
9 } from "../Localization"
10
11 type ControlsProps = {
12   balance: number
13   gameState: number
14   buttonState: any
15   betEvent: any
16   hitEvent: any
17   standEvent: any
18   resetEvent: any
19 }

```

```

20
21 const Controls: React.FC<ControlsProps> = ({
22     balance,
23     gameState,
24     buttonState,
25     betEvent,
26     hitEvent,
27     standEvent,
28     resetEvent,
29 }) => {
30     const [amount, setAmount] = useState(10)
31     const [inputStyle, setInputStyle] = useState(styles.input)
32
33     useEffect(() => {
34         validation()
35     }, [amount, balance])
36
37     const validation = () => {
38         if (amount > balance) {
39             setInputStyle(styles.inputError)
40             return false
41         }
42         if (amount < 0.01) {
43             setInputStyle(styles.inputError)
44             return false
45         }
46         setInputStyle(styles.input)
47         return true
48     }
49
50     const amountChange = (e: any) => {
51         setAmount(e.target.value)
52     }
53
54     const onBetClick = () => {
55         if (validation()) {
56             betEvent(Math.round(amount * 100) / 100)
57         }
58     }
59
60     const getControls = () => {

```

```

61     if (gameState === 0) {
62         return (
63             <div className={styles.controlsContainer}>
64                 <div className={styles.betContainer}>
65                     <h4>{amountButton}</h4>
66                     <input
67                         autoFocus
68                         type="number"
69                         value={amount}
70                         onChange={amountChange}
71                         className={inputStyle}
72                     />
73                 </div>
74                 <button onClick={() => onBetClick()} className={
75                     styles.button}>
76                     {betButton}
77                 </button>
78             </div>
79         )
80     } else {
81         return (
82             <div className={styles.controlsContainer}>
83                 <button
84                     onClick={() => hitEvent()}
85                     disabled={buttonState.hitDisabled}
86                     className={styles.button}
87                 >
88                     {hitButton}
89                 </button>
90                 <button
91                     onClick={() => standEvent()}
92                     disabled={buttonState.standDisabled}
93                     className={styles.button}
94                 >
95                     {standButton}
96                 </button>
97                 <button
98                     onClick={() => resetEvent()}
99                     disabled={buttonState.resetDisabled}
100                    className={styles.button}

```

```

101         {resetButton}
102     </button>
103 </div>
104 )
105 }
106 }
107
108 return <>{getControls()}</>
109 }
110
111 export default Controls

```

```

1 import React from 'react';
2 import styles from '../styles/Hand.module.css';
3 import Card from '../Card';
4
5 type HandProps = {
6   title: string,
7   cards: any[]
8 };
9
10 const Hand: React.FC<HandProps> = ({ title, cards }) => {
11   const getTitle = () => {
12     if (cards.length > 0) {
13       return (
14         <h1 className={styles.title}>{title}</h1>
15       );
16     }
17   }
18   return (
19     <div className={styles.handContainer}>
20       {getTitle()}
21       <div className={styles.cardContainer}>
22         {cards.map((card: any, index: number) => {
23           return (
24             <Card key={index} value={card.value} suit={card.suit}
25               hidden={card.hidden} />
26           );
27         })}
28       </div>
29     </div>

```

```

30 }
31
32 export default Hand;

1 import React from 'react';
2 import styles from '../styles/Status.module.css';
3
4 type StatusProps = {
5   message: string,
6   balance: number
7 };
8
9 const Status: React.FC<StatusProps> = ({ message, balance })
10   => {
11   return (
12     <div className={styles.statusContainer}>
13       <div className={styles.status}>
14         <h1 className={styles.value}>{message}</h1>
15       </div>
16       <div className={styles.balance}>
17         <h1 className={styles.value}>${balance}</h1>
18       </div>
19     </div>
20   );
21 }
22 export default Status;

1 import React, { useState, useEffect } from "react"
2 import Status from "../Status"
3 import Controls from "../Controls"
4 import Hand from "../Hand"
5 import jsonData from "../deck.json"
6 import {
7   Message,
8   allCardsAreDrawn,
9   clubsSymbol,
10  dealerHand,
11  diamondsSymbol,
12  heartsSymbol,
13  spadesSymbol,
14  userHand,
15 } from "../Localization"

```

```

16
17 const App: React.FC = () => {
18   enum GameState {
19     bet ,
20     init ,
21     userTurn ,
22     dealerTurn ,
23   }
24
25   enum Deal {
26     user ,
27     dealer ,
28     hidden ,
29   }
30
31   const data = JSON.parse(JSON.stringify(jsonData.cards))
32   const [deck , setDeck]: any[] = useState(data)
33
34   const [userCards , setUserCards]: any[] = useState([])
35   const [userScore , setUserScore] = useState(0)
36   const [userCount , setUserCount] = useState(0)
37
38   const [dealerCards , setDealerCards]: any[] = useState([])
39   const [dealerScore , setDealerScore] = useState(0)
40   const [dealerCount , setDealerCount] = useState(0)
41
42   const [balance , setBalance] = useState(100)
43   const [bet , setBet] = useState(0)
44
45   const [gameState , setGameState] = useState(GameState.bet)
46   const [message , setMessage] = useState(Message.bet)
47   const [buttonState , setButtonState] = useState({
48     hitDisabled: false ,
49     standDisabled: false ,
50     resetDisabled: true ,
51   })
52
53   useEffect(() => {
54     if (gameState === GameState.init) {
55       drawCard(Deal.user)
56       drawCard(Deal.hidden)

```



```

57     drawCard(Deal.user)
58     drawCard(Deal.dealer)
59     setGameState(GameState.userTurn)
60     setMessage(Message.hitStand)
61   }
62 }, [gameState])
63
64 useEffect(() => {
65   calculate(userCards, setUserScore)
66   setUserCount(userCount + 1)
67 }, [userCards])
68
69 useEffect(() => {
70   calculate(dealerCards, setDealerScore)
71   setDealerCount(dealerCount + 1)
72 }, [dealerCards])
73
74 useEffect(() => {
75   if (gameState === GameState.userTurn) {
76     if (userScore === 21) {
77       buttonState.hitDisabled = true
78       setButtonState({ ...buttonState })
79     } else if (userScore > 21) {
80       bust()
81     }
82   }
83 }, [userCount])
84
85 useEffect(() => {
86   if (gameState === GameState.dealerTurn) {
87     if (dealerScore >= 17) {
88       checkWin()
89     } else {
90       drawCard(Deal.dealer)
91     }
92   }
93 }, [dealerCount])
94
95 const resetGame = () => {
96   console.clear()
97   setDeck(data)

```

```

98
99     setUserCards ([])
100    setUserScore (0)
101    setUserCount (0)
102
103    setDealerCards ([])
104    setDealerScore (0)
105    setDealerCount (0)
106
107    setBet (0)
108
109    setGameState (GameState.bet)
110    setMessage (Message.bet)
111    setButtonState ({
112        hitDisabled: false,
113        standDisabled: false,
114        resetDisabled: true,
115    })
116 }
117
118 const placeBet = (amount: number) => {
119     setBet (amount)
120     setBalance (Math.round ((balance - amount) * 100) / 100)
121     setGameState (GameState.init)
122 }
123
124 const drawCard = (dealType: Deal) => {
125     if (deck.length > 0) {
126         const randomIndex = Math.floor (Math.random () * deck.
127             length)
128         const card = deck[randomIndex]
129         deck.splice (randomIndex, 1)
130         setDeck ([... deck])
131         switch (card.suit) {
132             case "spades":
133                 dealCard (dealType, card.value, spadesSymbol)
134                 break
135             case "diamonds":
136                 dealCard (dealType, card.value, clubsSymbol)
137                 break
138             case "clubs":

```

```

138         dealCard(dealType, card.value, diamondsSymbol)
139         break
140     case "hearts":
141         dealCard(dealType, card.value, heartsSymbol)
142         break
143     default:
144         break
145 }
146 } else {
147     alert(allCardsAreDrawn)
148 }
149 }
150
151 const dealCard = (dealType: Deal, value: string, suit:
    string) => {
152     switch (dealType) {
153         case Deal.user:
154             userCards.push({ value: value, suit: suit, hidden:
                false })
155             setUserCards([...userCards])
156             break
157         case Deal.dealer:
158             dealerCards.push({ value: value, suit: suit, hidden:
                false })
159             setDealerCards([...dealerCards])
160             break
161         case Deal.hidden:
162             dealerCards.push({ value: value, suit: suit, hidden:
                true })
163             setDealerCards([...dealerCards])
164             break
165         default:
166             break
167     }
168 }
169
170 const revealCard = () => {
171     dealerCards.filter((card: any) => {
172         if (card.hidden === true) {
173             card.hidden = false
174         }

```

```

175         return card
176     })
177     setDealerCards ([... dealerCards ])
178 }
179
180 const calculate = (cards: any[], setScore: any) => {
181     let total = 0
182     cards.forEach((card: any) => {
183         if (card.hidden === false && card.value !== "A") {
184             switch (card.value) {
185                 case "K":
186                     total += 10
187                     break
188                 case "Q":
189                     total += 10
190                     break
191                 case "J":
192                     total += 10
193                     break
194                 default:
195                     total += Number(card.value)
196                     break
197             }
198         }
199     })
200     const aces = cards.filter((card: any) => {
201         return card.value === "A"
202     })
203     aces.forEach((card: any) => {
204         if (card.hidden === false) {
205             if (total + 11 > 21) {
206                 total += 1
207             } else if (total + 11 === 21) {
208                 if (aces.length > 1) {
209                     total += 1
210                 } else {
211                     total += 11
212                 }
213             } else {
214                 total += 11
215             }

```

```

216     }
217   })
218   setScore(total)
219 }
220
221 const hit = () => {
222   drawCard(Deal.user)
223 }
224
225 const stand = () => {
226   buttonState.hitDisabled = true
227   buttonState.standDisabled = true
228   buttonState.resetDisabled = false
229   setButtonState({ ...buttonState })
230   setGameState(GameState.dealerTurn)
231   revealCard()
232 }
233
234 const bust = () => {
235   buttonState.hitDisabled = true
236   buttonState.standDisabled = true
237   buttonState.resetDisabled = false
238   setButtonState({ ...buttonState })
239   setMessage(Message.bust)
240 }
241
242 const checkWin = () => {
243   if (userScore > dealerScore || dealerScore > 21) {
244     setBalance(Math.round((balance + bet * 2) * 100) / 100)
245     setMessage(Message.userWin)
246   } else if (dealerScore > userScore) {
247     setMessage(Message.dealerWin)
248   } else {
249     setBalance(Math.round((balance + bet * 1) * 100) / 100)
250     setMessage(Message.tie)
251   }
252 }
253
254 return (
255   <>
256   <Status message={message} balance={balance} />

```

```

257     <Controls
258         balance={balance}
259         gameState={gameState}
260         buttonState={buttonState}
261         betEvent={placeBet}
262         hitEvent={hit}
263         standEvent={stand}
264         resetEvent={resetGame}
265     />
266     <Hand title={'${dealerHand} (${dealerScore})'} cards={
267         dealerCards} />
267     <Hand title={'${userHand} (${userScore})'} cards={
268         userCards} />
268 </>
269 )
270 }
271
272 export default App

```

```

1 // Status
2 export enum Message {
3     bet = "Сделайте ставку!",
4     hitStand = "Взять или оставить?",
5     bust = "Перебор!",
6     userWin = "Вы выиграли!",
7     dealerWin = "Выиграл дилер!",
8     tie = "Ничья!",
9 }
10 // Cards
11 export const spadesSymbol = "♠"
12 export const clubsSymbol = "♣"
13 export const diamondsSymbol = "♦"
14 export const heartsSymbol = "♥"
15 export const allCardsAreDrawn = "Все карты разыграны!"
16 // Hands
17 export const dealerHand = "Рука дилера"
18 export const userHand = "Ваша рука"
19 // Controls
20 export const betButton = "Сделать ставку!"
21 export const amountButton = "Сумма: "
22 export const hitButton = "Взять карту!"
23 export const standButton = "Остановиться"

```

```

24 export const resetButton = "Сбросить"

1  :root {
2    --card-bg: #f2f2f2;
3    --card-shadow: rgba(0, 0, 0, 0.25);
4    --hidden-card-bg: rgb(230, 230, 230);
5    --text-black: black;
6    --text-red: red;
7  }
8
9
10 * {
11   margin: 0;
12   padding: 0;
13 }
14
15 body {
16   background: var(--background-color);
17   margin: 0;
18   font-family: 'Lexend Exa', sans-serif;
19   -webkit-font-smoothing: antialiased;
20 }

1  {
2    "cards": [
3      {
4        "value": "A",
5        "suit": "spades"
6      },
7      {
8        "value": "A",
9        "suit": "diamonds"
10     },
11     {
12       "value": "A",
13       "suit": "clubs"
14     },
15     {
16       "value": "A",
17       "suit": "hearts"
18     },
19     {
20       "value": "2",

```

```
21         "suit": "spades"
22     },
23     {
24         "value": "2",
25         "suit": "diamonds"
26     },
27     {
28         "value": "2",
29         "suit": "clubs"
30     },
31     {
32         "value": "2",
33         "suit": "hearts"
34     },
35     {
36         "value": "3",
37         "suit": "spades"
38     },
39     {
40         "value": "3",
41         "suit": "diamonds"
42     },
43     {
44         "value": "3",
45         "suit": "clubs"
46     },
47     {
48         "value": "3",
49         "suit": "hearts"
50     },
51     {
52         "value": "4",
53         "suit": "spades"
54     },
55     {
56         "value": "4",
57         "suit": "diamonds"
58     },
59     {
60         "value": "4",
61         "suit": "clubs"
```



```

62     },
63     {
64         "value": "4",
65         "suit": "hearts"
66     },
67     {
68         "value": "5",
69         "suit": "spades"
70     },
71     {
72         "value": "5",
73         "suit": "diamonds"
74     },
75     {
76         "value": "5",
77         "suit": "clubs"
78     },
79     {
80         "value": "5",
81         "suit": "hearts"
82     },
83     {
84         "value": "6",
85         "suit": "spades"
86     },
87     {
88         "value": "6",
89         "suit": "diamonds"
90     },
91     {
92         "value": "6",
93         "suit": "clubs"
94     },
95     {
96         "value": "6",
97         "suit": "hearts"
98     },
99     {
100         "value": "7",
101         "suit": "spades"
102     },

```

```

103     {
104         "value": "7",
105         "suit": "diamonds"
106     },
107     {
108         "value": "7",
109         "suit": "clubs"
110     },
111     {
112         "value": "7",
113         "suit": "hearts"
114     },
115     {
116         "value": "8",
117         "suit": "spades"
118     },
119     {
120         "value": "8",
121         "suit": "diamonds"
122     },
123     {
124         "value": "8",
125         "suit": "clubs"
126     },
127     {
128         "value": "8",
129         "suit": "hearts"
130     },
131     {
132         "value": "9",
133         "suit": "spades"
134     },
135     {
136         "value": "9",
137         "suit": "diamonds"
138     },
139     {
140         "value": "9",
141         "suit": "clubs"
142     },
143     {

```

```
144         "value": "9",
145         "suit": "hearts"
146     },
147     {
148         "value": "10",
149         "suit": "spades"
150     },
151     {
152         "value": "10",
153         "suit": "diamonds"
154     },
155     {
156         "value": "10",
157         "suit": "clubs"
158     },
159     {
160         "value": "10",
161         "suit": "hearts"
162     },
163     {
164         "value": "J",
165         "suit": "spades"
166     },
167     {
168         "value": "J",
169         "suit": "diamonds"
170     },
171     {
172         "value": "J",
173         "suit": "clubs"
174     },
175     {
176         "value": "J",
177         "suit": "hearts"
178     },
179     {
180         "value": "Q",
181         "suit": "spades"
182     },
183     {
184         "value": "Q",
```

```
185         "suit": "diamonds"
186     },
187     {
188         "value": "Q",
189         "suit": "clubs"
190     },
191     {
192         "value": "Q",
193         "suit": "hearts"
194     },
195     {
196         "value": "K",
197         "suit": "spades"
198     },
199     {
200         "value": "K",
201         "suit": "diamonds"
202     },
203     {
204         "value": "K",
205         "suit": "clubs"
206     },
207     {
208         "value": "K",
209         "suit": "hearts"
210     }
211 ]
212 }
```