**Математические операции:**

- eval (); // используется для выполнения строк, которые могут содержать как математические выражения

- Math.pow(7, 2); // Возведение в степень (49)

**Операции с числами:**

- floor() // Округление вниз

- random() // Генерация случайного числа

**Диалоговые окна:**

- alert() // Окно сообщения

- prompt(text, defaultText) // Вывод вопроса и получение ответа через ввод пользователя

**Работа со строками:**

- str.split('').reverse().join(''); // Разбиение, реверс и объединение строки

- num.toString(); // Преобразование числа в строку

- str.slice() // Извлечение части строки

- substring() // удаления заданного частя

**Массивы:**

- reduce() // Уменьшение массива до единственного значения

- map() // Преобразование каждого элемента массива

- splice(start, deleteCount, item1, item2, /\* …, \*/ itemN) // Изменение содержимого массива

- sort() // Для сортировки массива можно написать функцию внутри

- pop() // Удаляет последний элемент из массива и возвращает его значение.

- localeCompare() //

- forEach() // Выполнение функции для каждого элемента массива

- some() проверяет, удовлетворяет ли какой-либо элемент массива условию, заданному в передаваемой функции.

**Работа с числами:**

- toFixed() // Округление числа до указанного количества цифр после точки

**Проверка наличия элемента:**

- includes() // Проверка наличия элемента в массиве

**Конкатенация массивов:**

- concat() // Объединение двух массивов

**Фильтрация массива:**

- filter() // Создание нового массива с элементами, прошедшими проверку

**Работа с обьектами:**  
- Object.entries() используется для создания массива из элементов, состоящих из пар ключ-значение, представленных в виде массивов. Каждый элемент массива результата имеет вид [ключ, значение]. Этот метод пригодится, когда вам нужно работать с элементами объекта, используя методы массивов, такие как map, filter, reduce, и так далее.

* Object.fromEntries() преобразует список пар ключ-значение в объект.
* **Object.values()** возвращает массив значений перечисляемых свойств объекта в том же порядке что и цикл for...in. Разница между циклом и методом в том, что цикл перечисляет свойства и из цепочки прототипов.

**Поиск элемента:**

**-** str.match(regexp) //возвращает получившиеся совпадения при сопоставлении строки с регулярным выражением.

- find() // Поиск элемента в массиве

**Интервалы:**

- setInterval() // Повторное выполнение кода через определенный интервал времени

- clearInterval() // Остановка выполнения кода, запущенного с помощью setInterval()

**Работа со строками:**

- indexOf() // Поиск позиции подстроки в строке

- lastIndexOf() // Поиск позиции последнего вхождения подстроки в строке

- trim() // Удаление пробельных символов в начале и конце строки

- toUpperCase() // Преобразование строки в верхний регистр

- toLowerCase() // Преобразование строки в нижний регистр

- repeat() // Повторение строки указанное количество раз

- localeCompare() // возвращает число, где должна находиться эта строка при сортировке

**Генерация случайных чисел:**

- random() // Генерация случайного числа

- Умножение на число для получения случайного значения до указанного числа

**Проверка строк:**

- startsWith() // Проверка начала строки

- endsWith() // Проверка конца строки

- includes() // Проверка наличия подстроки в строке

- isNaN() // Проверка на NaN

**Замена подстрок:**

-replace(то что, на это)

- replaceAll() // Замена всех вхождений подстроки в строке

**Добавление символов:**

- padStart() // Добавление символов в начало строки

- padEnd() // Добавление символов в конец строки

**Работа с элементами строки:**

- charAt() // Получение символа строки по индексу

**Циклы:**

- do while // Цикл с проверкой условия после выполнения тела цикла

- for // Цикл с предустановленным количеством итераций

**Spread оператор:**

- ... // Распаковка элементов массива или объекта

**Регулярные выражения:**

- var pattern = /^[a-zA-Z0-9]+$/; // Проверка на наличие алфанумерических символов

**JavaScriptв веб-разработке:**

**Стрелочные функции (Arrow Functions):**

- () => {} // Сокращенный синтаксис для объявления функций

**Шаблонные строки (Template Literals):**

- `` Текст ${выражение} `` // Позволяют встраивать выражения в строки

**Деструктуризация (Destructuring):**

- const { prop1, prop2 } = obj; // Извлечение значений из объекта или массива

**Fetch API:**

- fetch(url) // Позволяет делать HTTP-запросы и обрабатывать ответы

**Promise:**

- new Promise((resolve, reject) => { /\* код \*/ }) // Позволяет работать с асинхронными операциями

**async/await:**

- async function name() { await /\* асинхронная операция \*/ } // Синтаксис для работы с асинхронным кодом более линейно

**localStorage и sessionStorage:**

- localStorage.setItem(key, value) // Сохранение данных на стороне клиента

- localStorage.getItem(key) // Получение данных из хранилища

**JSON.parse() и JSON.stringify():**

- JSON.parse(jsonString) // Преобразование строки JSON в объект JavaScript

- JSON.stringify(jsonObject) // Преобразование объекта JavaScript в строку JSON

**EventListeners:**

- element.addEventListener('click', () => { /\* код обработчика события \*/ }) // Привязка обработчика события к элементу

**ES6 Modules:**

- import { name } from './module'; // Импорт функций или объектов из других модулей

- export function name() { /\* код \*/ } // Экспорт функций или объектов для использования в других модулях

**Set и Map:**

- const set = new Set() // Коллекция уникальных значений

- const map = new Map() // Коллекция ключ-значение

**Функция setInterval():**

- setInterval(() => { /\* код \*/ }, interval) // Выполнение кода через определенный интервал времени

**Функция setTimeout():**

- setTimeout(() => { /\* код \*/ }, delay) // Запуск функции через определенное время

**Object.assign():**

- Object.assign(target, source) // Копирование свойств из одного или нескольких объектов в целевой объект

**1. Простой цикл с увеличением счетчика:**

for (let i = 0; i < 5; i++) {

console.log(i);

}

**3. Итерация по свойствам объекта:**

const obj = { a: 1, b: 2, c: 3 };

for (let key in obj) {

console.log(key + ': ' + obj[key]);

}

**4. Использование break для выхода из цикла:**

for (let i = 0; i < 10; i++) {

if (i === 5) {

break;

}

console.log(i);

}

**5. Использование continue для перехода к следующей итерации:**

for (let i = 0; i < 5; i++) {

if (i === 2) {

continue;

}

console.log(i);

}

**6. Использование цикла для выполнения асинхронных операций:**

async function fetchData() {

const urls = ['url1', 'url2', 'url3'];

for (let url of urls) {

const data = await fetch(url);

console.log(data);

}

}

**9. Использование цикла для генерации HTML элементов:**

const container = document.getElementById('container');

const items = ['item1', 'item2', 'item3'];

for (let item of items) {

container.innerHTML += `<div>${item}</div>`;

}

**10. Цикл для работы с символами строки:**

const str = 'hello';

for (let char of str) {

console.log(char);

}

**12. Использование цикла для работы с типом Map:**

const map = new Map([['a', 1], ['b', 2], ['c', 3]]);

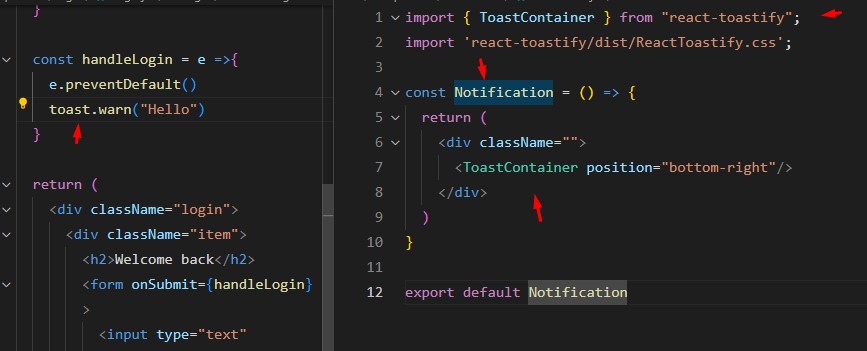
for (let [key, value] of map) {

console.log(key, value);

}



$ npm i emoji-picker-react

$ npm i react-toastify -> для предупреждения выводит алерты

$ npm i firebase

https://console.firebase.google.com/

**Netlify**

**[[redirects]]**

**from = "/\*"**

**to = "/index.html"**

**status = 200**

**НодирБек, [26.04.2024 20:30]**

**NETLIFY.toml;**

**$ npm i react-transition-group**

import { CSSTransition } from 'react-transition-group';

<CSSTransition in={showAvatar} timeout={200} classNames="my-node" unmountOnExit>

          <img src={user?.avatar || "./avatar.png"}  className="showAvatar" onClick={handleClick}/>

</CSSTransition>

**Git удаление и изменения старой ветки**[**https://youtu.be/DpdTvF2c7Hg?si=thgF-3o6mFJjnamU**](https://youtu.be/DpdTvF2c7Hg?si=thgF-3o6mFJjnamU)

**Надо отточить**

**OOP**

**Redux Toolkit**

**Firebase**

**React Query**

**useReducer – это Redux старый**

**useHistory – history.goBack(), history.goforward(), history.push(path, state)**

**useContext() – STM store**

**useCallBack – prevent unnecessary rerender**

**useMemos - render**

**Axios**

**Git**

**reactQuery**

**Async  
Promises**

**Await**

**useRef()**