

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет “Радиотехнический”
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языков программирования»

Отчет по Домашнему Заданию

Выполнил:
студент группы РТ5-31Б:
Филатов И. В.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Москва, 2025 г.

Web-приложение погоды, которое:

- Получает данные из открытой OpenWeather API
- Отображает погодные условия с визуальными эффектами
- Сохраняет последний запрос в LocalStorage
- Обработывает ошибки пользовательского ввода

Использовано императивное, процедурное, асинхронное программирование.

- Императивное: Прямая манипуляция DOM, изменение состояния приложения.
- Процедурное: Использование процедур (функций), разделение логики.
- Асинхронное: Использование Promise с fetch(), цепочки .then().

Текст программы

consts.js:

// Каждая функция является чистой, то есть всегда возвращают новый DOM элемент,
// не меняя внешнее состояние. То есть детерминированные.

```
export function getClouds() {  
  // Создание DOM-элемента  
  const cloud1 = document.createElement("img");  
  // Установка атрибутов элемента  
  cloud1.id = "weather_visibility";  
  cloud1.className = "weather__header-clouds";  
  cloud1.src = "Clouds.png";  
  const cloud2 = document.createElement("img");  
  cloud2.id = "weather_visibility";  
  cloud2.className = "weather__header-clouds-2";  
  cloud2.src = "Clouds.png";  
  
  // Возврат значения  
  return [cloud1, cloud2];  
}
```

```
export function getRain(){  
  const rain = document.createElement("img");  
  rain.id = "weather_visibility";  
  rain.className = "weather__header-rain";  
  rain.src = "rain.gif";
```

```
  return rain;  
}
```

```
export function getClear(){  
  const blik = document.createElement("img");  
  blik.id = "weather_visibility";  
  blik.className = "weather__header-blik";  
  blik.src = "blik.png";
```

```
  return blik;  
}
```

```
export function getSnow(){  
  const snow = document.createElement("img");  
  snow.id = "weather_visibility";  
  snow.className = "weather__header-snow";  
  snow.src = "snow2.gif";
```

```
  return snow;  
}
```

```

}
export function getFog(){
  const fog = document.createElement("img");
  fog.id = "weather_visibility";
  fog.className = "weather__header-fog";
  fog.src = "fog.gif";

  return fog;
}

```

script.js

// Импорт конкретных функций из модуля

```

import { getClouds } from "./consts.js";
import { getRain } from "./consts.js";
import { getClear } from "./consts.js";
import { getSnow } from "./consts.js";
import { getFog } from "./consts.js";

```

// Использование открытого API

```

const api = {
  key: "f8434dffb8f56a03b56ef99f44a6f862",
  baseUrl: "https://api.openweathermap.org/data/2.5/",
};

```

// Использование localStorage браузера

```

if (localStorage.length > 3) {
  document.querySelector( // выбор объекта
    ".weather__main-location-city"
  ).innerText = `${localStorage.getItem("city")}, ${localStorage.getItem(
    "country"
  )}`;
  document.querySelector(".weather__main-location-date").innerText =
    localStorage.getItem("date");
  document.querySelector(".weather__main-info-temp").innerText =
    localStorage.getItem("temperature");
  document.querySelector(".weather__main-info-weather").innerText =
    localStorage.getItem("weather");
  document.querySelector(".weather__main-info-hi-low").innerText =
    localStorage.getItem("hi-low");
  // получение информации из объекта и изменение контента в элементе

```

```

  if (localStorage.getItem("weather") === "Clouds") {
    document.querySelector(".weather__header").prepend(getClouds()[0]); // добавление
элемента
    document.querySelector(".weather__header").prepend(getClouds()[1]);
  }
  if (localStorage.getItem("weather") === "Rain"){
    document.querySelector(".weather").prepend(getRain());
  }
  if (localStorage.getItem("weather") === "Clear"){
    document.querySelector(".weather").prepend(getClear());
  }

```

```

    }
    if (localStorage.getItem("weather") === "Snow"){
        document.querySelector(".weather").prepend(getSnow());
    }
    if (localStorage.getItem("weather") === "Fog"){
        document.querySelector(".weather").prepend(getFog());
    }
    if (localStorage.getItem("weather") === "Mist"){
        document.querySelector(".weather").prepend(getFog());
    }
    }
    const search = document.querySelector(".weather__header-input-search");
    // const button = document.querySelector(".weather__header-input-button");
    const setQuery = (event) => {
        if (event.keyCode == 13) {
            getResults(search.value); // получение контента с input field
        }
    };
    const gif = document.createElement("img");
    gif.className = "gif";
    gif.src = "loading-loader.gif"; // загрузка пока идет асинхронный процесс
    gif.width = 70;

    const city_not_exist = document.createElement("div");
    city_not_exist.className = "message";
    const getResults = (query) => {
        document.querySelectorAll("#weather_visibility").forEach((element) => {
            document.querySelector("#weather_visibility").remove();
        });
        document.querySelector(".weather__header").append(gif);
        fetch(`${api.baseurl}weather?q=${query}&units=metric&APPID=${api.key}`) //
        асинхронный запрос
        .then((weather) => {
            return weather.json(); // парсинг в JSON
        })
        .then((weather) => { // then -> только после успешного прошлого then
            displayResults(weather);
            if (weather.weather[0].main === "Clouds") {
                document.querySelector(".weather__header").prepend(getClouds()[0]);
                document.querySelector(".weather__header").prepend(getClouds()[1]);
            }
            else if (weather.weather[0].main === "Rain"){
                document.querySelector(".weather").prepend(getRain());
            }
            else if (weather.weather[0].main === "Clear"){
                document.querySelector(".weather").prepend(getClear());
            }
            else if (weather.weather[0].main === "Snow"){
                document.querySelector(".weather").prepend(getSnow());
            }
            else if (weather.weather[0].main === "Fog"){
                document.querySelector(".weather").prepend(getFog());
            }
        });
    };

```

```

    }
    else if (weather.weather[0].main === "Mist"){
        document.querySelector(".weather").prepend(getFog());
    }
    document.querySelector(".gif").remove();
    search.value = "";
    document.querySelector(".message").remove();

    })
    .catch((err) => { // обработка ошибки (нет ответа от сервера / некорректный ввод)
        if (search.value !== "") {
            city_not_exist.innerText = `City '${search.value}' doesn't exist or openweather API has
broken`;
            search.value = "";
            document.querySelector(".weather__header").append(city_not_exist);
        }
        document.querySelector(".gif").remove();
    });
};

const displayResults = (weather) => {
    let city = document.querySelector(".weather__main-location-city");
    city.innerText = `${weather.name}, ${weather.sys.country}`;

    let now = new Date();
    let date = document.querySelector(".weather__main-location-date");
    date.innerText = dateBuilder(now);

    let temp = document.querySelector(".weather__main-info-temp");
    temp.innerHTML = `${Math.round(weather.main.temp)}B°C`;

    let weather_el = document.querySelector(".weather__main-info-weather");
    weather_el.innerText = weather.weather[0].main;
    // console.log(weather.weather[0].main);
    // if (weather.weather[0].main === "Clouds") {
    //     console.log("yep");
    // }
    // if (weather.weather[0].main === "Rain") {
    //     console.log("yyyyeeep");
    // }
    let hilow = document.querySelector(".weather__main-info-hi-low");
    hilow.innerText = `${Math.round(weather.main.temp_min)}B°C / ${Math.round(
        weather.main.temp_max
    )}B°C`;

    // запись в localStorage браузера
    localStorage.setItem("city", weather.name);
    localStorage.setItem("country", weather.sys.country);
    localStorage.setItem("date", dateBuilder(now));
    localStorage.setItem("temperature", `${Math.round(weather.main.temp)}B°C`);
    localStorage.setItem("weather", weather.weather[0].main);
    localStorage.setItem(

```

```
    "hi-low",  
    `${Math.round(weather.main.temp_min)}B°c / ${Math.round(  
        weather.main.temp_max  
    )}B°c`  
);  
};
```

```
search.addEventListener("keypress", setQuery); // использование обработчика событий  
нажатия на кнопку  
// button.addEventListener("click", getResults(search.value));
```

```
// функция построителя даты
```

```
function dateBuilder(d) {  
    let months = [  
        "January",  
        "February",  
        "March",  
        "April",  
        "May",  
        "June",  
        "July",  
        "August",  
        "September",  
        "October",  
        "November",  
        "December",  
    ];  
    let days = [  
        "Sunday",  
        "Monday",  
        "Tuesday",  
        "Wednesday",  
        "Thursday",  
        "Friday",  
        "Saturday",  
    ];  
  
    let day = days[d.getDay()];  
    let date = d.getDate();  
    let month = months[d.getMonth()];  
    let year = d.getFullYear();  
  
    return `${day} ${date} ${month} ${year}`;  
}
```

Результаты

Search for a city...

Magnitogorsk, RU
Thursday 18 December 2025

-3°C

Snow
-4°C / -3°C

A weather card for Magnitogorsk, RU. The background is a dark, starry night sky with silhouettes of palm trees and a city skyline. At the top is a search bar. Below it, the city name and date are displayed. The temperature is shown in large white text as -3°C. Below that, the weather condition 'Snow' is written in italics, followed by a range of -4°C / -3°C.

Search for a city...

Moscow, RU
Friday 19 December 2025

1°C

Clouds
1°C / 2°C

A weather card for Moscow, RU. The background is a blue sky with white clouds and silhouettes of palm trees and a city skyline. At the top is a search bar. Below it, the city name and date are displayed. The temperature is shown in large white text as 1°C. Below that, the weather condition 'Clouds' is written in italics, followed by a range of 1°C / 2°C.

Search for a city...

Yakutsk, RU
Friday 19 December 2025

-46°C

Fog
-46°C / -46°C

A weather card for Yakutsk, RU. The background is a dark, foggy sky with silhouettes of palm trees and a city skyline. At the top is a search bar. Below it, the city name and date are displayed. The temperature is shown in large white text as -46°C. Below that, the weather condition 'Fog' is written in italics, followed by a range of -46°C / -46°C.

Search for a city...

City 'ошибка' doesn't exist or openweather API has broken

Miami, US
Friday 19 December 2025

26°C

Clouds
26°C / 27°C

A weather card for Miami, US. The background is a dark sky with silhouettes of palm trees and a city skyline. At the top is a search bar. Below it, a red error message is displayed: 'City 'ошибка' doesn't exist or openweather API has broken'. Below the error message, the city name and date are displayed. The temperature is shown in large white text as 26°C. Below that, the weather condition 'Clouds' is written in italics, followed by a range of 26°C / 27°C.