

Локальные GPT для фронтендера

Практические кейсы использования локальных больших языковых моделей



Ollama

- ollama.com
- [Исходники ollama](#)
- [JS API](#)

Get up and running with large language models.

Run [Llama 3](#), [Phi 3](#), [Mistral](#), [Gemma](#), and other models. Customize and create your own.

Download ↓

Available for macOS, Linux,
and Windows (preview)

Установка Ollama

- ollama.com/download

Linux:

```
curl -fsSL https://ollama.com/install.sh | sh
```

[Дока по ручной установке](#)

Доступны также версии для винды и мака.

Download Ollama



Install with one command:

```
curl -fsSL https://ollama.com/install.sh | sh
```



[View script source](#) • [Manual install instructions](#)

Где брать языковые модели

- Библиотека моделей

Некоторые модели:

- [Llama3](#) - общая языковая модель
- [Llava](#) - рассматривает и описывает картинки

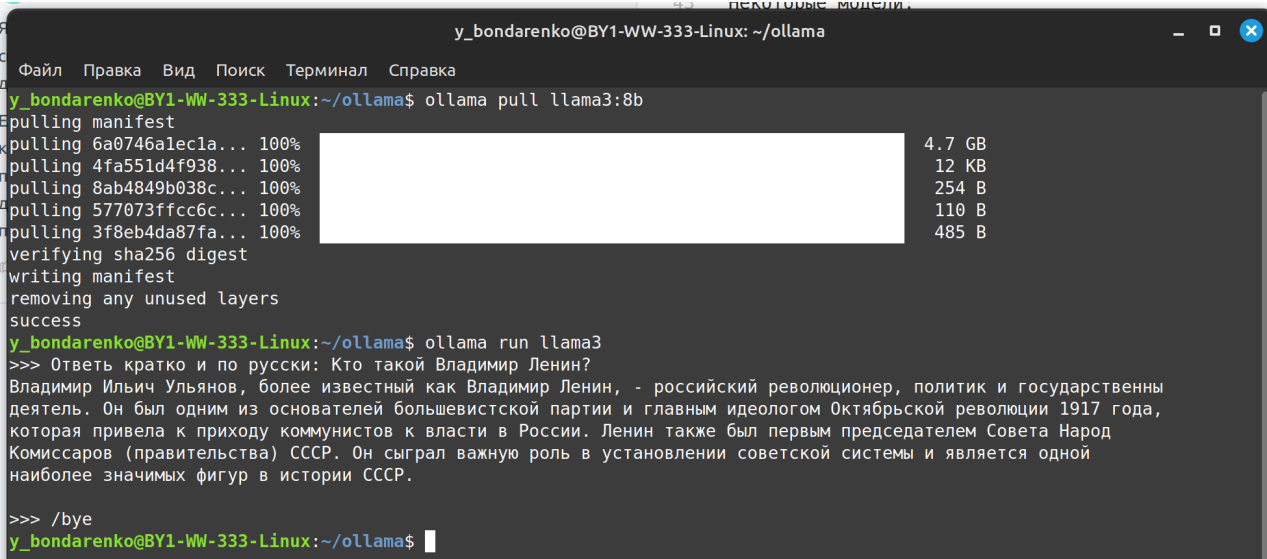
ЧЯТИК

Скачиваем модельку себе на комп:

```
ollama pull llama3:8b
```

Запускаем чат с моделью llama3:

```
ollama run llama3
```



```
y_bondarenko@BY1-WW-333-Linux: ~/ollama
Файл Правка Вид Поиск Терминал Справка
y_bondarenko@BY1-WW-333-Linux:~/ollama$ ollama pull llama3:8b
pulling manifest
pulling 6a0746alec1a... 100%
pulling 4fa551d4f938... 100%
pulling 8ab4849b038c... 100%
pulling 577073ffcc6c... 100%
pulling 3f8eb4da87fa... 100%
verifying sha256 digest
writing manifest
removing any unused layers
success
y_bondarenko@BY1-WW-333-Linux:~/ollama$ ollama run llama3
>>> Ответь кратко и по русски: Кто такой Владимир Ленин?
Владимир Ильич Ульянов, более известный как Владимир Ленин, - российский революционер, политик и государственный деятель. Он был одним из основателей большевистской партии и главным идеологом Октябрьской революции 1917 года, которая привела к приходу коммунистов к власти в России. Ленин также был первым председателем Совета Народных Комиссаров (правительства) СССР. Он сыграл важную роль в установлении советской системы и является одной из наиболее значимых фигур в истории СССР.

>>> /bye
y_bondarenko@BY1-WW-333-Linux:~/ollama$
```

Апи

```
curl -X POST http://localhost:11434/api/generate -d '{
  "model": "llama3",
  "prompt": "Ответь очень кратко и по-русски: кто такой Иосиф Сталин?"
}'
```

```
y_bondarenko@BY1-WW-333-Linux: ~/ollama
Файл Правка Вид Поиск Терминал Справка
y_bondarenko@BY1-WW-333-Linux:~/ollama$ curl -X POST http://localhost:11434/api/generate -d '{
  "model": "llama3",
  "prompt": "Ответь очень кратко и по-русски: кто такой Иосиф Сталин?"
}'
{"model": "llama3", "created_at": "2024-05-27T10:44:35.772777047Z", "response": "И", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:35.826412962Z", "response": "о", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:35.877080692Z", "response": "и", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:35.936734484Z", "response": "В", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:35.994220234Z", "response": "исс", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.049457838Z", "response": "ари", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.101484585Z", "response": "он", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.162578018Z", "response": "ович", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.214597001Z", "response": "Ста", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.269804793Z", "response": "лин", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.322506967Z", "response": " -", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.378655021Z", "response": " совет", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.435629668Z", "response": "ский", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.48410351Z", "response": " полит", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.534960038Z", "response": "ик", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.588315036Z", "response": " ", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.646513535Z", "response": " л", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.697149592Z", "response": "ид", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.748401386Z", "response": "ер", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.806650957Z", "response": " пар", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.859677426Z", "response": " тии", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.912988721Z", "response": " \\", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:36.968506242Z", "response": "В", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.020743496Z", "response": "К", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.081649174Z", "response": "П", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.134223413Z", "response": "(", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.193314028Z", "response": "6", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.253657855Z", "response": ")\\", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.308984281Z", "response": "(", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.36858586Z", "response": "с", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.419871282Z", "response": " ", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.475103607Z", "response": "195", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.529585517Z", "response": "2", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.581550195Z", "response": " рога", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.640818558Z", "response": " -", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.693824431Z", "response": " К", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.747522814Z", "response": "П", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.801422887Z", "response": "С", "done": false}
{"model": "llama3", "created_at": "2024-05-27T10:44:37.860029158Z", "response": "С", "done": false}
```

Либа для работы с Ollama

- [ollama-js](#)

Генерация

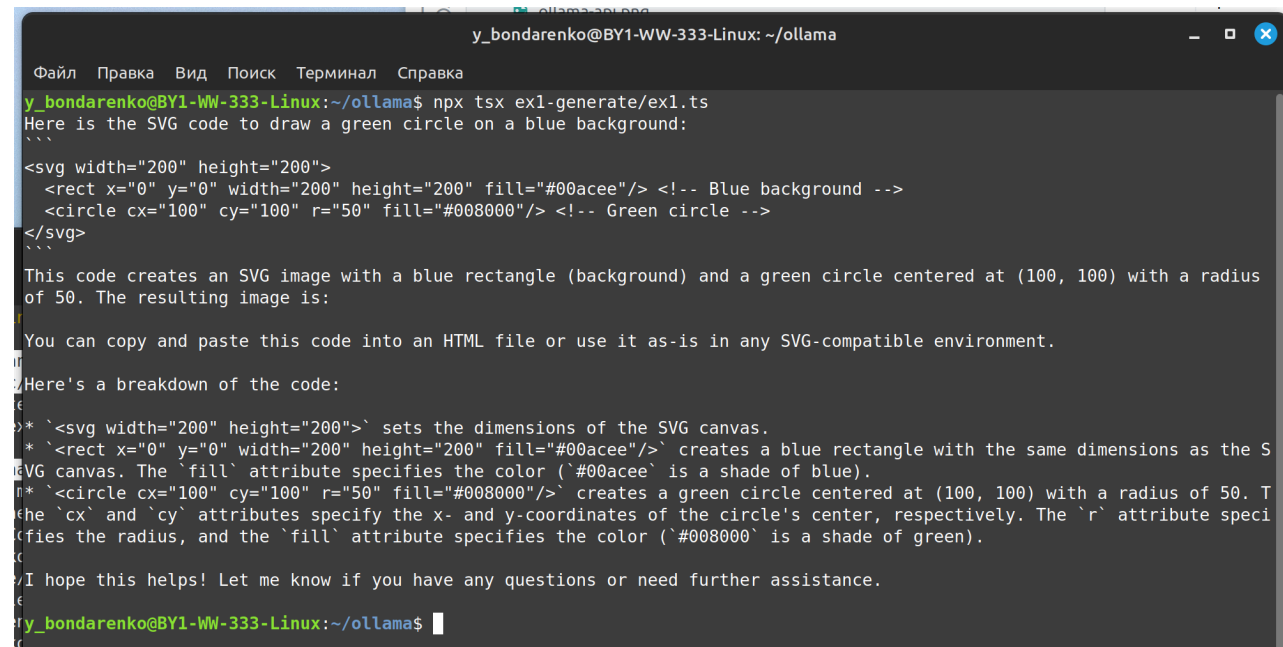
Попросим модель сгенерировать
нам SVG картинку:

```
import { Ollama } from "ollama";

const ollama = new Ollama({ host: "http://localhost:11434" });

const response = await ollama.generate({
  model: "llama3",
  prompt: "draw a green circle on a blue background in svg format",
  stream: true,
});

for await (const part of response) {
  process.stdout.write(part.response);
}
process.stdout.write("\n\n");
```



```
y_bondarenko@BY1-WW-333-Linux: ~/ollama
Файл Правка Вид Поиск Терминал Справка
y_bondarenko@BY1-WW-333-Linux:~/ollama$ npx tsx ex1-generate/ex1.ts
Here is the SVG code to draw a green circle on a blue background:
`
<svg width="200" height="200">
  <rect x="0" y="0" width="200" height="200" fill="#00acee"/> <!-- Blue background -->
  <circle cx="100" cy="100" r="50" fill="#008000"/> <!-- Green circle -->
</svg>
`

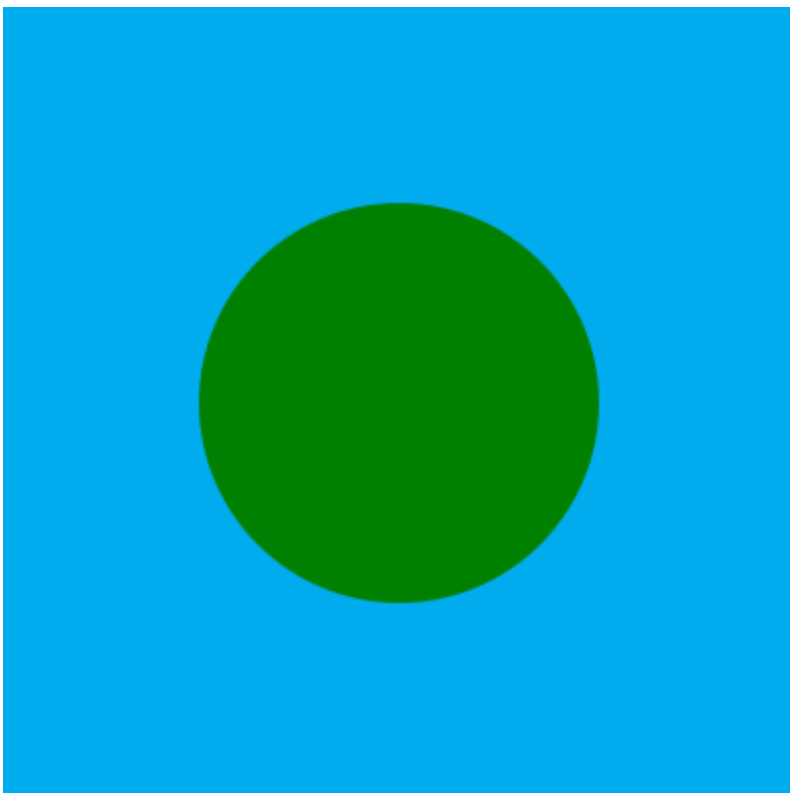
This code creates an SVG image with a blue rectangle (background) and a green circle centered at (100, 100) with a radius of 50. The resulting image is:

You can copy and paste this code into an HTML file or use it as-is in any SVG-compatible environment.

Here's a breakdown of the code:
>* `<svg width="200" height="200">` sets the dimensions of the SVG canvas.
>* `<rect x="0" y="0" width="200" height="200" fill="#00acee"/>` creates a blue rectangle with the same dimensions as the SVG canvas. The `fill` attribute specifies the color (`#00acee` is a shade of blue).
>* `<circle cx="100" cy="100" r="50" fill="#008000"/>` creates a green circle centered at (100, 100) with a radius of 50. The `cx` and `cy` attributes specify the x- and y-coordinates of the circle's center, respectively. The `r` attribute specifies the radius, and the `fill` attribute specifies the color (`#008000` is a shade of green).

I hope this helps! Let me know if you have any questions or need further assistance.
y_bondarenko@BY1-WW-333-Linux:~/ollama$
```


Результат



Переводчик

Настроим нашу модель и попросим ее перевести документацию:

```
import ollama from "ollama";
import fs from "fs";

const modelfile = `
FROM llama3
PARAMETER temperature 0
SYSTEM ""
You are a translator of technical documentation from English
into Russian. Variable names in single quotes do not need
to be translated, source code in triple quotes does not need
to be translated."
`;
await ollama.create({ model: "translator", modelfile: modelfile });

const docPath = "./ex2-translate/doc.md";
const docAsBase64 = fs.readFileSync(docPath, "utf8");

const response = await ollama.generate({
  model: "llama3",
  prompt: "Переведи эту документацию на русский язык: " + docAsBase64,
  stream: true,
});
for await (const part of response) {
  process.stdout.write(part.response);
}
```

Результат:

Using a controller

Each controller has its own creation API, but typically you will create an instance and store it with the component:

```
``ts
class MyElement extends LitElement {
  private clock = new ClockController(this, 1000);
}
```

The component associated with a controller instance is called the host component.

The controller instance registers itself to receive lifecycle callbacks from the host component, and triggers a host update when the controller has new data to render. This is how the `ClockController` example periodically renders the current time.

A controller will typically expose some functionality to be used in the host's `render()` method. For example, many controllers will have some state, like a current value:

```
``ts
render() {
  return html`
    <div>Current time: ${this.clock.value}</div>
  `;
}
```

Since each controller has its own API, refer to specific controller documentation on how to use them.

Использование контроллера

Каждый контроллер имеет свой собственный API создания, но обычно вы создаете экземпляр и храните его в компоненте:

```
``ts
class MyElement extends LitElement {
  private clock = new ClockController(this, 1000);
}
```

Компонент, ассоциированный с контроллером, называется контролируемым компонентом.

Контроллер-экземпляр регистрирует себя для получения callback'ов жизненного цикла от контролируемого компонента и запустит обновление контролируемого компонента, когда у контроллера есть новая информация для рендеринга. Это позволяет примеру `ClockController` периодически отображать текущее время.

Контроллер обычно экспонирует какую-либо функциональность для использования в методе `render()` контролируемого компонента. Например, многие контроллеры будут иметь некоторое состояние, например, текущее значение:

```
``ts
render() {
  return html`
    <div>Current time: ${this.clock.value}</div>
  `;
}
```

Так как каждый контроллер имеет свой собственный API, обращайтесь к документации конкретного контроллера для информации о его использовании.

Картинки

Спрашиваем, что на картинке:

```
import { Ollama } from "ollama";
import fs from "fs";

const imagePath = "./ex3-image/cat2.jpg";
const imageAsBase64 = fs.readFileSync(imagePath, "base64");

const ollama = new Ollama({ host: "http://localhost:11434" });

const response = await ollama.generate({
  model: "llama3",
  prompt:
    "Ответь одним словом и по русски, какое животное на этой base64 картинке: " +
    imageAsBase64,
  stream: true,
});
for await (const part of response) {
  process.stdout.write(part.response);
}
process.stdout.write("\n\n");
```

Ответ:

```
y_bondarenko@BY1-WW-333-Linux: ~/ollama
Файл Правка Вид Поиск Терминал Справка
y_bondarenko@BY1-WW-333-Linux:~/ollama$ npx tsx ex1-image/ex1.ts
Кот.
y_bondarenko@BY1-WW-333-Linux:~/ollama$
```



LlamaIndex

Фреймворк для работы с моделями.

- llamaindex.ai
- TS дока

по умолчанию работает с OpenAI и просит API-ключ

Анализ

Берем резюме кандидата, и анализируем:

```
import {
  Document,
  HuggingFaceEmbedding,
  Ollama,
  Settings,
  VectorStoreIndex,
} from "llamaindex";
import fs from "node:fs";

Settings.llm = new Ollama({
  model: "llama3",
});

Settings.embedModel = new HuggingFaceEmbedding({
  modelType: "BAAI/bge-small-en-v1.5",
  quantized: false,
});

async function main() {
  // Load essay from abramov.txt in Node
  const path = "./ex4-resume/profile.txt";

  const essay = fs.readFileSync(path, "utf-8");

  // Create Document object with essay
  const document = new Document({ text: essay, id_: path });

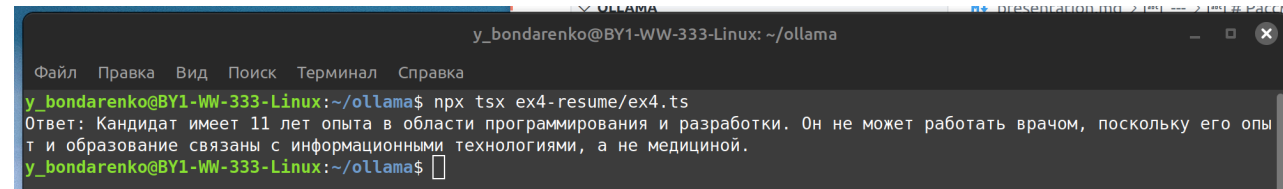
  // Split text and create embeddings. Store them in a VectorStoreIndex
  const index = await VectorStoreIndex.fromDocuments([document]);

  // Query the index
  const queryEngine = index.asQueryEngine();

  const response = await queryEngine.query({
    query:
      "Ответь по русски и кратко, сколько лет опыта у кандидата и сможет ли он работать врачом?",
  });

  // Output response
  console.log(response.toString());
}

main().catch(console.error);
```



```
y_bondarenko@BY1-WW-333-Linux: ~/ollama
Файл  Правка  Вид  Поиск  Терминал  Справка
y_bondarenko@BY1-WW-333-Linux:~/ollama$ npx tsx ex4-resume/ex4.ts
Ответ: Кандидат имеет 11 лет опыта в области программирования и разработки. Он не может работать врачом, поскольку его опыт и образование связаны с информационными технологиями, а не медициной.
y_bondarenko@BY1-WW-333-Linux:~/ollama$
```

Поиск по документам

```
import {
  Document,
  HuggingFaceEmbedding,
  Ollama,
  Settings,
  VectorStoreIndex,
} from "llama-index";
import fs from "node:fs";
import path from "node:path";

Settings.llm = new Ollama({
  model: "llama3",
});

Settings.embedModel = new HuggingFaceEmbedding({
  modelType: "BAAI/bge-small-en-v1.5",
  quantized: false,
});

function readFilesInDirectory(directoryPath) {
  const files = fs.readdirSync(directoryPath);
  const fileContents = [];

  for (const file of files) {
    const filePath = path.join(directoryPath, file);
    const fileContent = fs.readFileSync(filePath, "utf-8");
    fileContents.push(fileContent);
  }

  return fileContents;
}

async function main() {
  // Load essay from abramov.txt in Node
  // const path = "./ex4-resume/profile.txt";

  // const essay = fs.readFileSync(path, "utf-8");
  const fileContents = readFilesInDirectory("./ex5-search/data");

  // Create Document object with essay
  // const document = new Document({ text: essay, id_: path });
  const documents = fileContents.map((file, i) => {
    return new Document({ text: file, id_: `${i}` });
  });

  // Split text and create embeddings. Store them in a VectorStoreIndex
  const index = await VectorStoreIndex.fromDocuments(documents);

  // Query the index
  const queryEngine = index.asQueryEngine();

  const response = await queryEngine.query({
    query: "Ответь по русски и кратко: Что такое Проблема relayout в Gameface",
  });

  // Output response
  console.log(response.toString());
  // console.log(JSON.stringify(response, null, 2));
}

main().catch(console.error);
```

y_bondarenko@BY1-WW-333-Linux: ~/ollama

Файл Правка Вид Поиск Терминал Справка

В GameFace "Проблема relayout" - это особенность, при которой движок не сразу обновляет информацию о новом элементе на странице. Например, если добавить новый элемент (например, чей квадрат с размерами 50x50px), то рухнуть размеры и позицию нового элемента могут быть равны нулю, что может вызвать ошибки при работе с этим элементом.

y_bondarenko@BY1-WW-333-Linux:~/ollama\$