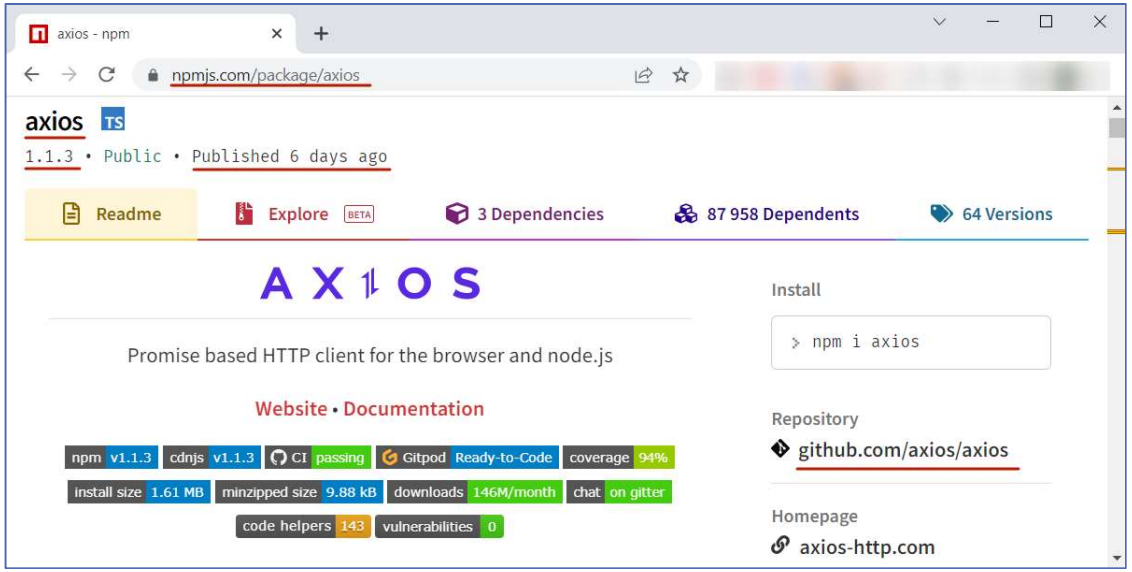
ПРОГРАММИРОВАНИЕ В INTERNET

AXIOS

Axios =

это пакет для разработки HTTP-клиента на основе Promise, работающего и в браузере, и в среде Node.js.

На стороне сервера он использует нативный node.js http-модуль, тогда как на стороне клиента (браузер) он использует XMLHttpRequests.



документация

GET-запрос с query-параметрами

axios(config)

```
const axios = require('axios');

// let query = require('qs');
// let parms = query.stringify({ x: 34, y: 12, s: 'xxx' });

let parms = new URLSearchParams({ x: 34, y: 12, s: 'xxx' }).toString();

function httpRequest() {
    return axios({
        url: `http://localhost:5000/?${parms}`,
        method: 'get'
    });
}

httpRequest()
    .then(response => {
        console.log(response.data);
    })
    .catch(error => {
        console.error(error)
    })
```

Так как axios возвращает Promise, одним из способов обработки результата являются обработчики then(), catch(), finally()

Это можно сделать явно указав параметры, предварительно преобразовав их в строку соответствующего формата с помощью qs.stringify(), или URLSearchParams, или можно использовать свойство params в options.

Вместо модуля qs можно использовать:
– stringify(…) => new URLSearchParams(…).toString()
– parse(…) => new URLSearchParams(…).get(…)

```
PS D:\Материалы\cwp_07> node 07-01c
x = 34; y = 12
```

GET-запрос с query-параметрами

axios(config)

```
const axios = require('axios');

async function httpRequest() {
    try {
        let response = await axios({
            url: `http://localhost:5000/`,
            method: 'get',
            params: { x: 34, y: 12, s: 'xxx' }
        });
        console.log(response.data);
    }
    catch (error) {
        console.error(error)
    }
}

httpRequest();
```

Также промис возвращаемый axios можно обработать конструкцией async/await. Однако, для обработки ошибок необходимо оборачивать в try-catch.

Указав в объекте свойство params прикрепляются query-параметры.

```
PS D:\Материалы\cwp_07> node 07-01c
x = 34; y = 12
```

POST-запрос с телом (формат JSON)

axios(config)*

*также могут быть использованы axios(url[, config]), axios.request(config)

```
const axios = require('axios');
let query = require('querystring');

let parms = query.stringify({ x: 34, y: 12, s: 'xxx' });

async function httpRequest() {
    try {
        let response = await axios({
            url: `http://localhost:5000/`,
            method: 'post',
            //data: parms,
            data: { x: 34, y: 12, s: 'xxx' }
        });
        console.log(response.data);
    }
    catch (error) {
        console.error(error)
    }
}

httpRequest();
```

```
PS D:\Материалы\cwp_07> node 07-01c
x=34&y=12&s=xxx
PS D:\Материалы\cwp_07> node 07-01c
{ x: 34, y: 12, s: 'xxx' }
```

Можно передать информацию в тело через свойство data конфига. Можно передать параметры в строковом виде (ключ, значение) или объект. Объект впоследствии будет автоматически преобразован в JSON.

request config

```
let config = {
    url: '/user',
    method: 'get',                              // по умолчанию get
    baseURL: 'https://some-domain.com/api/',    // будет добавлен к url, если url не является абсолютным
    headers: { 'X-Requested-With': 'XMLHttpRequest' },

    params: {
        ID: 12345                               // query-параметры
    },

    data: {
        firstName: 'Fred'                       // данные в теле запроса
    },
    data: 'Country=Brasil&City=Belo Horizonte', // 2-ой вариант передачи данных в теле

    // кол-во мс до истечения времени ожидания запроса
    // истекает timeout => запрос прерывается
    timeout: 1000,                              // по умолчанию 0 (no timeout),

    responseType: 'json',                       // по умолчанию json
    responseEncoding: 'utf8',                   // по умолчанию utf-8
}
```

Здесь также можно настроить прокси, количество допустимых редиректов, аутентификацию и др.
Подробнее здесь

GET-запрос с query-параметрами

axios.get(url[, config])

```
const axios = require('axios');
let query = require('querystring');

let parms = query.stringify({ x: 34, y: 12, s: 'xxx' });

async function httpRequest() {
    try {
        // const URL1 = `http://localhost:5000/?${parms}`;
        // const response = await axios.get(URL1);
        const URL2 = `http://localhost:5000/`
        const response = await axios.get(URL2, {
            params: { x: 34, y: 12, s: 'xxx' },
            headers: { "My-Header": "text" }
        });
        console.log(response.data);
    } catch (error) {
        console.error(error);
    }
}

httpRequest();
```

При использовании сокращений url, method и data свойства не нужно указывать в config.

Для удобства псевдонимы были предоставлены для всех распространенных методов запроса. Поэтому обычно используются axios.get(), axios.post(), ...

```
axios.get(url[, config])
axios.delete(url[, config])
axios.head(url[, config])
axios.options(url[, config])
axios.post(url[, data[, config]])
axios.put(url[, data[, config]])
axios.patch(url[, data[, config]])
```

Заголовки запроса добавляются через свойство header параметра конфигурации запроса.

```
PS D:\Материалы\cwp_07> node 07-02c
x = 34; y = 12
PS D:\Материалы\cwp_07> node 07-02c
x = 34; y = 12
```

Свойства объекта response

```
const axios = require('axios');
let query = require('querystring');

let parms = query.stringify({ x: 34, y: 12, s: 'xxx' });

async function httpRequest() {
    try {
        // const URL1 = `http://localhost:5000/?${parms}`;
        // const response = await axios.get(URL1);
        const URL2 = `http://localhost:5000/`
        const response = await axios.get(URL2, {
            params: { x: 34, y: 12, s: 'xxx' },
            headers: { "My-Header": "text" }
        });
        console.log("Data: ", response.data);                  // информация из тела ответа
        console.log("Status code: ", response.status);         // статус код
        console.log("Status message: ", response.statusText);  // пояснение к статус коду
        console.log("Headers: ", response.headers);            // заголовки ответа
        console.log("Config: ", response.config);              // config с настройками запроса
        console.log("Request object: ", response.request);     // объект отправленного запроса; экземляр http.ClientRequest
        console.log(response.request.method);                  // метод
        console.log(response.request.path);                    // путь
        console.log(response.request.getHeader("My-Header"));  // заголовок запроса
    } catch (error) {
        console.error(error);
    }
}

httpRequest();
```

```
PS D:\Материалы\cwp_07> node 07-02c
Data:  x = 34; y = 12
Status code:  200
Status message:  OK
Headers:  {
  date: 'Fri, 16 Oct 2020 18:41:19 GMT',
  connection: 'close',
  'content-length': '14'
}
Config:  {
  url: 'http://localhost:5000/',
  method: 'get',
  headers: {
    Accept: 'application/json, text/plain, */*',
    'My-Header': 'text',
    'User-Agent': 'axios/0.20.0'
  },
  params: { x: 34, y: 12, s: 'xxx' },
  transformRequest: [ [Function: transformRequest] ],
  transformResponse: [ [Function: transformResponse] ],
  timeout: 0,
  adapter: [Function: httpAdapter],
  xsrfCookieName: 'XSRF-TOKEN',
  xsrfHeaderName: 'X-XSRF-TOKEN',
  maxContentLength: -1,
  maxBodyLength: -1,
  validateStatus: [Function: validateStatus],
  data: undefined
}
Request object:  ClientRequest {
  _events: [Object: null prototype] {
    [Symbol(kOutHeaders)]: [Object: null prototype] {
      accept: [ 'Accept', 'application/json, text/plain, */*' ],
      'my-header': [ 'My-Header', 'text' ],
      'user-agent': [ 'User-Agent', 'axios/0.20.0' ],
      host: [ 'Host', 'localhost:5000' ]
    }
  }
GET
/?x=34&y=12&s=xxx
text
PS D:\Материалы\cwp_07>
```

POST-запрос с телом (формат application/www-form-urlencoded)

axios.post(url[, data[, config]])

```
const axios = require('axios');
let query = require('querystring');

let parms = query.stringify({ x: 34, y: 12, s: 'xxx' });

async function httpRequest() {
    try {
        //let response = await axios.post('http://localhost:5000/', parms);
        let response = await axios.post('http://localhost:5000/', {data: parms});
        console.log(response.data);
    }
    catch (error) {
        console.error(error)
    }
}

httpRequest();
```

```
PS D:\Материалы\cwp_07> node .\07-01c.js
x=34&y=12&s=xxx
PS D:\Материалы\cwp_07> node .\07-01c.js
{ data: 'x=34&y=12&s=xxx' }
```

Второй аргумент в axios.post() – строка или объект, содержащий параметры POST, будет записывать в тело. Альтернатива – передавать содержимое через свойство data конфига.

POST-запрос с телом (формат JSON)

axios.post(url[, data[, config]])

```
const axios = require('axios');

let jsonm = JSON.stringify(
    {
        __comment: "Запрос. Лекция 7",
        x: 1,
        y: 2,
        s: "Сообщение",
        m: ["a", "b", "c", "d"],
        o: { "surname": "Иванов", name: "Иван" }
    }
)

async function httpRequest() {
    try {
        const config = {
            headers: { "Content-Type": "applicaton/json", "Accept": "applicaton/json" },
            data: jsonm
        };
        //const response = await axios.post('http://localhost:5000/', jsonm);
        const response = await axios.post('http://localhost:5000/', config);
        console.log(response.data);
    } catch (error) {
        console.error(error);
    }
}

httpRequest();
```

Axios чаще всего парсит ответ на основе заголовка Content-Type ответа HTTP.

JSON => JS-объект
HTML => строку

```
PS D:\Материалы\cwp_07> node 07-03c
{
  __comment: 'Ответ. Лекция 7',
  x_plus_y: 3,
  Concatenation_s_o: 'Сообщение: Иванов, Иван',
  Length_m: 4
}
```

POST-запрос с телом (формат XML)

axios.post(url[, data[, config]])

```
const axios = require('axios');
let xmlbuilder = require('xmlbuilder');
let parseString = require('xml2js').parseString;

let xmldoc = xmlbuilder.create('request').att('id', '28');
xmldoc.ele('x').att('value', '1')
    .up().ele('x').att('value', '2')
    .up().ele('m').att('value', 'a')
    .up().ele('m').att('value', 'b')
    .up().ele('m').att('value', 'c')

async function httpRequest() {
    try {
        const config = {
            headers: { "Content-Type": "text/xml", "Accept": "text/xml" },
            data: xmldoc.toString({ pretty: true })
        };
        const response = await axios.post('http://localhost:5000/', config);
        let data = parseString(response.data, (err, str) => {
            if (err) console.log('xml parse error');
            else {
                console.dir(str, { depth: null });
            }
        })

    } catch (error) {
        console.error(error);
    }
}

httpRequest();
```

```
PS D:\Материалы\cwp_07> node 07-05c
{
  response: {
    '$': { id: '33', request: '28' },
    sum: [ { '$': { element: 'x', result: '3' } } ],
    concat: [ { '$': { element: 'm', result: 'abc' } } ]
  }
}
```

Скачивание файла

```
const axios = require('axios');
const fs = require('fs');

async function getImage() {
    try {
        let config = {
            responseType: 'stream'
        };
        let response = await axios.get('http://localhost:5000/', config);
        response.data.pipe(fs.createWriteStream('image.jpg'));
    }
    catch (error) {
        console.error(error);
    }
}

getImage();
```

Можно настроить тип данных с помощью свойства responseType. По умолчанию responseType имеет значение json, что означает, что axios попытается проанализировать ответ как JSON. Однако это неверно, если вы хотите, например, скачать изображение с помощью axios.

Можно установить responseType в 'arraybuffer', чтобы получить ответ в виде ArrayBuffer, или в 'stream', чтобы получить ответ в виде потока Node.js.

Загрузка файла

```
const axios = require('axios');
const fs = require('fs');
const FormData = require('form-data'); // npm install form-data

async function uploadFile() {
    try {
        let form = new FormData();
        form.append('file', fs.createReadStream(__dirname + '/MyFile.txt'),
            { knownLength: fs.statSync(__dirname + '/MyFile.txt').size });

        let config = {
            headers: {
                ...form.getHeaders(), // возвращает Object с Content-Type и boundary
                "Content-Length": form.getLengthSync()
            }
        };

        axios.post('http://localhost:5000/', form, config);

    } catch (error) {
        console.error(error);
    }
}

uploadFile();
```

Оператор ... (spread) получает все свойства объекта, а затем перезаписывает существующие свойства теми, которые передаются, то есть по сути этот оператор **создает копию.**

Пакет formdata позволяет отправлять формы и загружать файлы на сервер.

Метод form.getHeaders() добавляет корректный заголовок Content-Type в пользовательские заголовки запроса (в том числе сгенерированный разделитель).

Метод form.getLenghtSync() синхронно возвращает значение Content-Length.

axios.all([...]) (deprecated) => Promise.all([...])

```
const axios = require('axios');

let parms1 = new URLSearchParams({ x: 34, y: 12, s: 'xxx' }).toString();
let parms2 = new URLSearchParams({ x: 24, y: 44, s: 'aaa' }).toString();

function httpRequest() {
    return Promise.all([
        axios.get(`http://localhost:5000/?${parms1}`),
        axios.get(`http://localhost:5000/?${parms2}`)
    ]);
}

httpRequest()
    .then((response) => {
        console.log(response[0].data);
        console.log(response[1].data);
    })
    .catch(error => {
        console.log(error);
    });
```

```
PS D:\Материалы\cwp_07> node 07-09c
x = 34; y = 12
x = 24; y = 44
```

Для выполнения параллельных запросов можно использовать Promise.all().