基础知识

浏览器天生具发送 HTTP 请求的能力，比如在在址栏输入内容，提交 FORM 表单等。本章来学习通过 JS 程序来管理 HTTP 请求的能力。
使用 JS 脚本发送 HTTP 请求，不会带来页面的刷新，所以用户体验非常好。

#XMLHttpRequest

使用 XMLHttpRequest 发送请求，是一种存在很久的方案。现代浏览器支持使用 fetch 的异步请求方式，fetch 基于 promise 异步操作体验更好。

#基本使用

使用 XMLHttpRequest 发送请求需要执行以下几步

1. 使用 new XMLHttpRequest 创建 xhr 对象
2. xhr.open 初始化请求参数
3. xhr.send 发送网络请求
4. xhr.onload 监听请求结果
5. xhr.onerror 请求中断等错误发生时的处理

#响应类型

通过设置 `xhr.responseType` 对响应结果进行声明，来对结果自动进行处理。
下面是可以使用的响应类型

类型	说明
text	响应结果为文本
json	响应内容为 JSON，系统会自动将结果转为 JSON 对象
blob	二进制数据响应
document	XML DOCUMENT 内容

#响应结果

xhr.onload 用于处理响应完毕后的处理
使用以下属性来处理结果

- xhr.status 为 HTTP 状态码 如 404/422/403 等，当为 200 时为正确响应
- xhr.statusText HTTP 状态码内容，200 时为 ok,404 为 Not Found
- xhr.response 服务器端响应的内容

#使用示例

```
const xhr = new XMLHttpRequest()
xhr.timeout = 5000
xhr.open('GET', 'hd.php')
xhr.send()
xhr.onload = function () {
    if (xhr.status == 200) {
        console.log(xhr.response)
    } else {
        console.log(`${xhr.status}:${xhr.statusText}`)
    }
}
xhr.onerror = function (error) {
    console.log(error)
}

```

#发送表单

下面来使用 XMLHttpRequest 发送 POST 请求

#后台服务

下面创建 hd.php 后台脚本（你可以使用你喜欢的后台脚本进行测试）

```
<?php
echo $_POST['title'];

```

然后启动服务器

```
php -S localhost:8000

```

现在可以在浏览器中访问 http://localhsot:8080/hd.php 请求 hd.php

#前端异步请求

```
<form action="" id="form">
    <input type="text" name="title" />
    <input type="submit" />
</form>

<script>
    const form = document.getElementById('form')
    form.addEventListener('submit', function () {
        //阻止默认提交行为
        event.preventDefault()

        post('hd.php', new FormData(this))
    })
    function post(url, data) {
        const xhr = new XMLHttpRequest()
        xhr.open('POST', url)
        xhr.send(data)
        xhr.onload = () => {
            if (xhr.status == 200) {
                console.log(xhr.response)
            } else {
                console.log(`${xhr.status}:${xhr.statusText}`)
            }
        }
    }
</script>

```

#封装请求类

下面结合 Promise 来构建一个 XHR 异步处理类，使异步请求操作的变得更简单。

```
class HD {
  options = {
    responseType: 'json',
  }
  (method = 'GET', url, data = null, options) {
    this.method = method
    this.url = url
```

```
    this.data = this.formatData(data)
    Object.assign(this.options, options)
  }
  formatData(data) {
    if (typeof data != 'object' || data == null) data = {}
    let form = new FormData()
    for (const [name, value] of Object.entries(data)) {
      form.append(name, value)
    }

    return form
  }
  static get(url, options) {
    return new this('GET', url, null, options).xhr()
  }
  static post(url, data, options) {
    return new this('POST', url, data, options).xhr()
  }
  xhr() {
    return new Promise((resolve, reject) => {
      const xhr = new XMLHttpRequest()
      xhr.open(this.method, this.url)
      xhr.responseType = this.options.responseType
      xhr.send(this.data)
      xhr.onload = function () {
        if (xhr.status != 200) {
          reject({ status: xhr.status, error: xhr.statusText })
        } else {
          resolve(xhr.response)
        }
      }
      xhr.onerror = function (error) {
        reject(error)
      }
    })
  }
}

```

使用 HD.get 静态方法发送 GET 请求

```
HD.get('1.php', {
    responseType: 'text',
  }).then((response) => {
    console.log(response)
  })

```

使用 HD.post 静态方法发送 POST 请求

```
HD.post('2.php', data, {
  responseType: 'json',
}).then((response) => {
  console.log(response)
})

```

#FETCH

FETCH 是 JS 升级后提供的更简便的网络请求的操作方法，内部使用 Promise 来完成异步请求。

- response.json()接收 JSON 类型数据
- response.text()接收 TEXT 类型数据
- response.blog()接收 Blog 二进制数据

#请示步骤

使用 fetch 方法发送异步请求需要分以下两步操作

#响应头解析

第一步对服务器返回的响应头进行解析，会接到 Response 类创建的对象实例，里面包含以下属性。

- status:HTTP 状态码
- ok:状态码为 200-299 时为 true 的布尔值

#响应内容解析

第二步对返回的保存在 response.body 中的响应结果进行解析，支持了以下几种方式对结果进行解析。

- response.json()接收 JSON 类型数据
- response.text()接收 TEXT 类型数据
- response.blog()接收 Blog 二进制数据

> 以上方法不能同时使用，因为使用一个方法后数据已经被处理，其他方法就不可以操作了

#实例操作

下面来体验使用 fetch 发送请求

#后台服务

下面创建 hd.php 后台脚本（你可以使用你喜欢的后台脚本进行测试）

```
<?php
$articles = [
    ['name' => '后盾人'],
    ['name' => 'hdcms.com'],
    ['name' => 'houdunren.com']
];
echo json_encode($articles);

```

然后启动服务器

```
php -S localhost:8000

```

现在可以在浏览器中访问 http://localhsot:8080/hd.php 请求 hd.php

#发送请求

下面使用 FETCH 发送 GET 请求

```
fetch(`1.php`)
.then(response => {
    return response.json()
})
.then(articles => {
    console.log(articles)
})

```

因为 fetch 结果是 promise 所以也可以使用 async/await 操作

```

```

```
async function query() {
  const response = await fetch(`1.php`)
  const articles = await response.json()
  console.log(articles)
}
query()

```

#POST

发送 POST 请求需要设置请求头 Request header

#发送请求

- 发送的 JSON 类型需要设置请求头为

application/json;charset=utf-8

```
async function post() {
    const response = await fetch(`hd.php`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json;charset=utf-8',
        },
        body: JSON.stringify({ name: '后盾人' }),
    })
    if (response.ok) {
        const articles = await response.json()
        console.log(articles)
    }
}
post()

```

#后台响应

因为前台发送的是非表单数据，而是 JSON 字符串所以后台使用 `php://input` 来接收数据

```
<?php
echo file_get_contents('php://input');
```

```
3
```