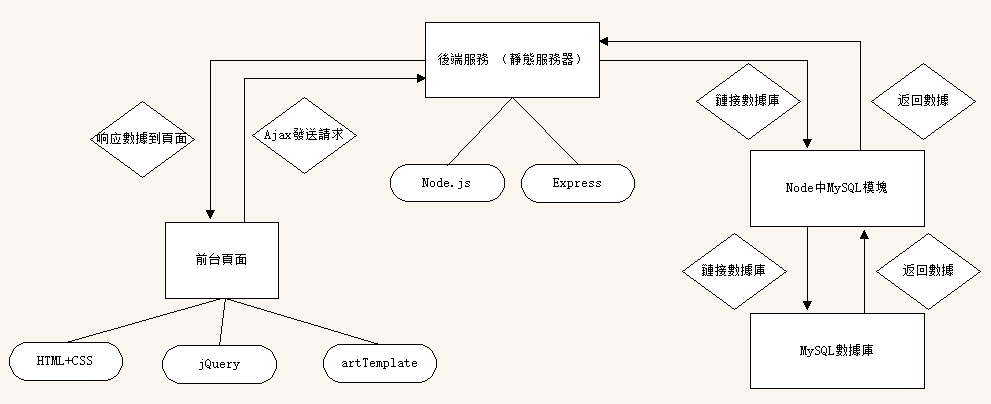
本階段目標：

1. 使用Node/Express搭建靜態服務器，提供後端服務
2. 知道Node.js不同模塊的作用
3. 使用MySQL數據庫存儲數據
4. 使用artTemplate模板引擎，高效、快速渲染數據
5. 使用Ajax請求後端接口，對數據進行增刪改查操作
6. Git項目版本管理工具
7. GitHub項目代碼託管平台

技術點之間的聯繫



Ajax介紹

## 1 客戶端與服務器

在上網過程中，存放和對外提供資源的電腦，叫做服務器。

服務器本質是一台電腦，只不過其性能要比個人電腦高很多。

在上網過程中，負責獲取和消費資源的電腦，叫做客戶端。個人電腦中，可以通過安裝瀏覽器的形式，訪問服務器對外提供的各種資源。

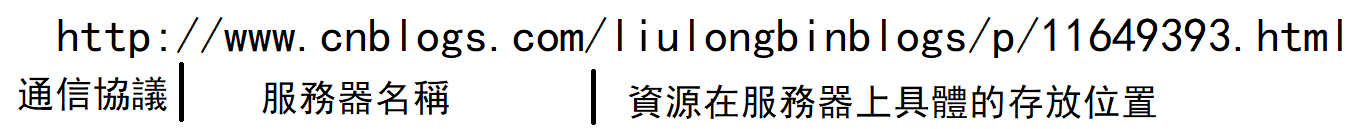
## 2 URL地址

URL英文全稱是Uniform Resource Location，中文叫統一資源定位符，用於標識互聯網上每個資源的唯一存放位置。

瀏覽器只有通過URL地址，才能正確定位資源的存放位置，從而成功訪問到對應的資源。

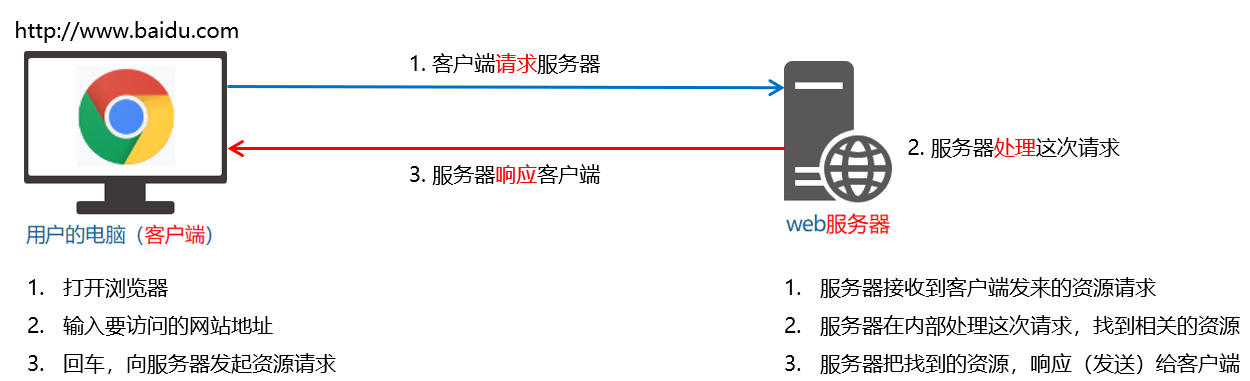
URL地址組成部分：

1. 客戶端與服務器之間的通信協議
2. 存有該資源的服務器名稱
3. 資源在服務器上具體的存放位置



## 3 分析網頁的打開過程

### （1）圖解這個過程



注意：

① 客戶端與服務器之間的通信過程，分為請求-處理-響應三個步驟。

② 網頁中的每一個資源，都是通過請求-處理-響應的方式從服務器獲取回來的。

### （2）用瀏覽器的開發者工具分析通信過程

① 打開Chrome瀏覽器

② 按Ctrl + Shift + I 或者F12打開Chrome的開發者工具

③ 切換到Network面板

④ 選中Doc標籤頁

⑤ 刷新頁面，分析客戶端與服務器的通信過程

## 4 服務器提供的資源

### （1）務器對外提供了哪些資源

除了文字內容，image圖片，audio音頻，video視頻以外，網頁中的數據也是服務器對外提供的一種資源，例如股票數據、個行業排行榜等。

數據是網頁的靈魂。

HTML是網頁的骨架。

CSS是網頁的顏值

JavaScript是網頁的行為。

骨架、顏值、行為都為數據服務。

### （2）網頁中如何請求數據？

如果要在網頁中請求服務器上的數據資源，需要用到XMLHttpRequest對象。

XMLHttpRequest（簡稱xhr）是瀏覽器提供的js成員，通過它，可以請求服務器上的數據資源。

最簡單的用法 var xhrObj = new XMLHttpRequest()

### （3）資源的請求方式

客戶端請求服務器時，請求的方式有很多種，最常見的兩種請求方式分別為get和post請求。

* get請求通常用於獲取服務端資源（向服務器要資源）

如根據URL地址，從服務器獲取HTML文件、CSS文件、JS文件、圖片文件、數據資源等。

* post請求通常用於向服務器提交數據（往服務器發送資源）

如登錄時向服務器提交的登錄信息、註冊時向服務器提交的註冊信息、添加用戶時向服務器提交的用戶信息等各種數據的提交操作。

## 5 了解Ajax

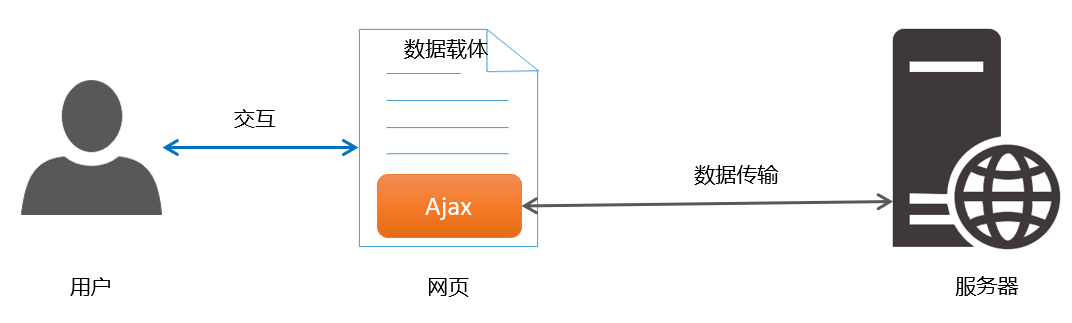
### （1）什麼是Ajax

Ajax全稱是Asynchronous JavaScript And XML （異步JavaScript和XML）。

通俗的理解：在網頁中利用XMLHttpRequest對象和服務器進行數據交互的方式，就是Ajax。

### （2）為什麼要學Ajax

之前所學的技術，只能把網頁做得更美觀漂亮，或者添加一些動畫效果。而Ajax可以讓我們輕鬆實現網頁與服務器之間的數據交互。

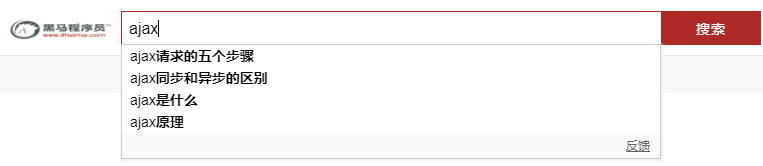


### （3）Ajax的典型應用場景

用戶名檢測：註冊用戶時，通過Ajax的形式，動態檢測用戶名是否被佔用。



搜索提示：當輸入搜索關鍵字時，通過Ajax的形式，動態加載搜索提示列表。



數據分頁顯示：當點擊頁碼值的時候，通過Ajax的形式，根據頁碼值動態刷新表格的數據。



數據的增刪改查：數據的添加、刪除、修改、查詢操作，都需要通過Ajax的形式來實現數據的交互。



## 6 jQuery中的Ajax

### （1）概述

瀏覽器中提供的XMLHttpRequest用法比較複雜，所以jQuery對XMLHttpRequest進行了封裝，提供了一系列Ajax相關的函數，極大地降低了Ajax的使用難度。

jQuery中發起Ajax請求最常用的三個方法如下：

$.get() 向服務器拿數據

$.post() 向服務器提交數據

$.ajax() 既可以向服務器拿數據又可以向服務器提交數據

### （2）$.get()

jQuery中$.get()函數的功能單一，專門用來發起get請求，從而將服務器上的資源請求到客戶端來進行使用。

語法：

$.get(url, [data], [callback])

第1個參數是必填，後面兩個參數被中括號[ ]括起來，是選填的。

|  |  |  |  |
| --- | --- | --- | --- |
| 參數名 | 參數類型 | 是否必填 | 說明 |
| url | string | 是 | 要請求的資源地址 |
| data | object | 否 | 請求資源期間要攜帶的參數 |
| callback | function | 否 | 請求成功時的回調函數 |

發起不帶參數的請求

    <script>

      $.get("http://www.liulongbin.top:3006/api/getbooks", function (*res*) {

        console.log(res); //res是服務器返回的數據

      });

    </script>



發起帶參數的請求

    <script>

      $.get(

        "http://www.liulongbin.top:3006/api/getbooks",

        { id: 1 },

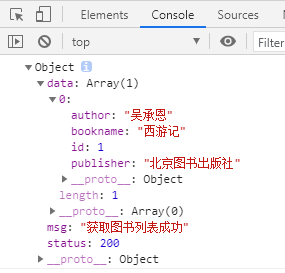
        function (*res*) {

          console.log(res); //res是服務器返回的數據

        }

      );

    </script>



### （3）$.post()

jQuery中$.post()函數的功能單一，專門用來發起post請求，從而向服務器提交數據。

$.post()函數的語法如下：

$.post(url, [data], [callback])

其中三個參數各自代表的含義如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 參數名 | 參數類型 | 是否必填 | 說明 |
| url | string | 是 | 要提交數據的地址 |
| data | object | 否 | 要提交的數據 |
| callback | function | 否 | 數據提交成功時的回調函數 |

    <script>

      $.post(

        "http://www.liulongbin.top:3006/api/addbook", //請求的URL地址

        {

          bookname: "星之繼承者",

          author: "詹姆斯·P·霍根",

          publisher: "新星出版社",

        }, //提交的數據

        function (*res*) {

          //回調函數

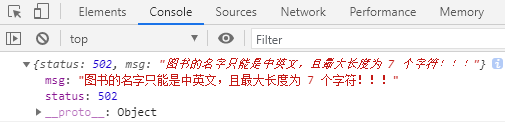
          console.log(res); //res是服務器返回的數據

        }

      );

    </script>







### （4）$.ajax()

相比于$.get()和$.post()，$.ajax()是一個功能比較綜合的函數，它允許我們對Ajax請求進行更詳細的配置。

$.ajax()函數的基本語法如下：

$.ajax({

type: ‘’, //請求的方式，例如GET或者POST （全部大寫或者全部小寫都正確）

url: ‘’ , //請求的URL地址

data: { }, //請求要攜帶的數據

success: function(res) { } //請求成功之後的回調函數

})

    <script>

      $.ajax({

        type: "GET", //請求的方式，例如GET或者POST

        url: "http://www.liulongbin.top:3006/api/getbooks", //請求的URL地址

        data: { id: 1 }, //請求要攜帶的數據。如果要返回全部數據，這個data可省略

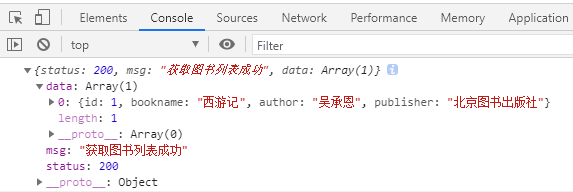
        success: function (*res*) { //請求成功之後的回調函數

          console.log(res); //res是服務器返回的數據

        },

      });

    </script>



    <script>

      $.ajax({

        type: "POST", //請求的方式，例如GET或者POST

        url: "http://www.liulongbin.top:3006/api/addbook", //請求的URL地址

        data: {

          //要提交給服務器的數據

          bookname: "星之繼承者",

          author: "詹姆斯·P·霍根",

          publisher: "新星出版社",

        },

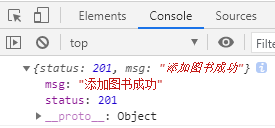
        success: function (*res*) {

          console.log(res); //res是服務器返回的數據

        }, //請求成功之後的回調函數

      });

    </script>



## 7 接口

### （1）接口的概念

使用Ajax請求數據時，被請求的URL地址，就叫做數據接口（簡稱接口）。同時，每個接口必須有請求方式（GET或者POST）。

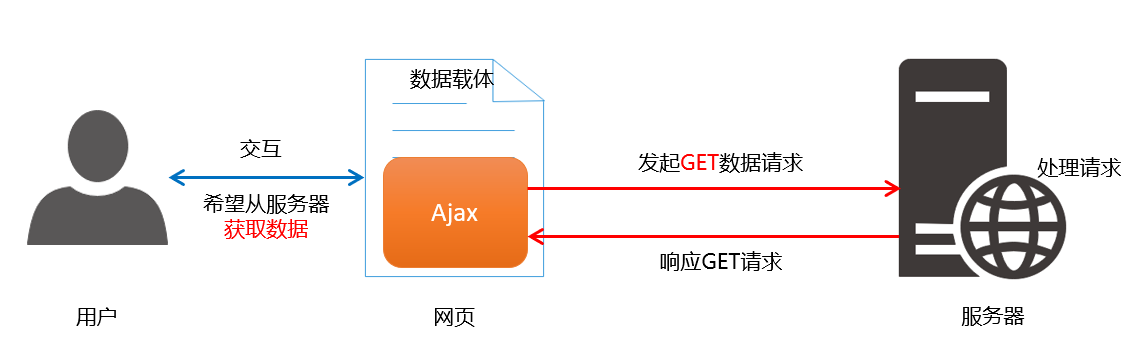
例如：

<http://www.liulongbin.top:3006/api/getbooks> 獲取圖書列表的接口（GET請求）

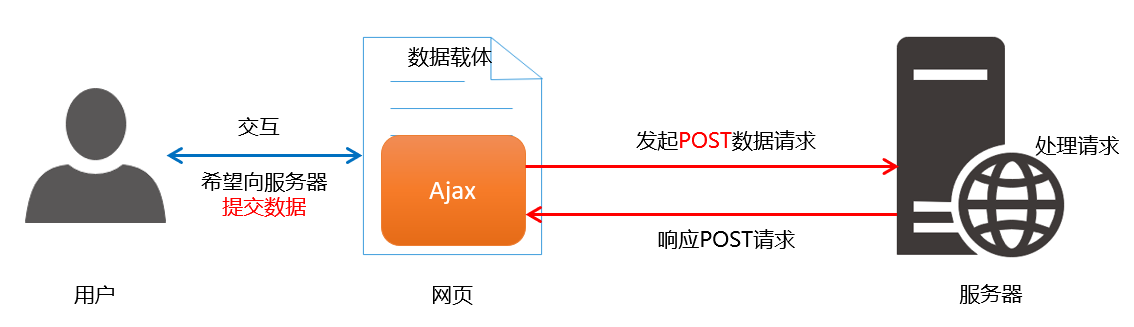
<http://www.liulongbin.top:3006/api/addbook> 添加圖書的接口（POST請求）

### （2）分析接口的請求過程

通過GET方式請求接口的過程



通過POST方式請求接口的過程



### （3）接口測試工具

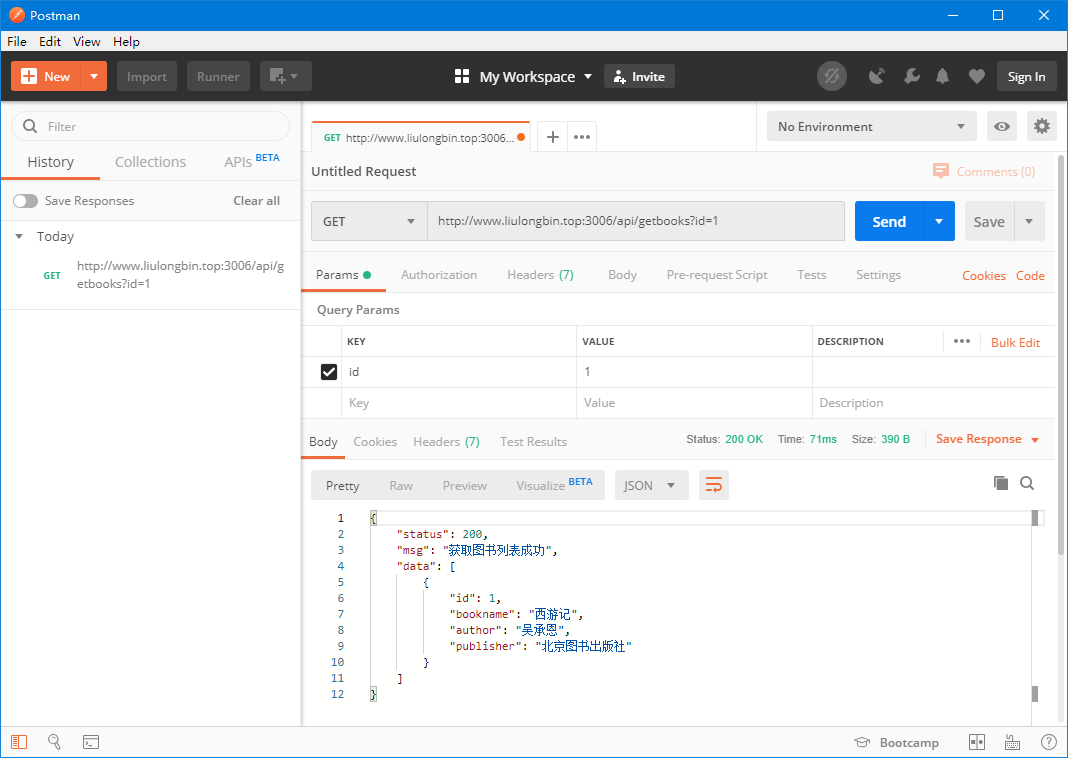
為了驗證接口能否被正常訪問，我們常常需要使用接口測試工具來對數據接口進行檢測。

好處：接口測試工具讓我們在不寫任何代碼的情況下，對接口進行調用和測試。

PostMan是個非常優秀的接口測試工具

訪問PostMan的官方下載網址：<https://www.getpostman.com/downloads/> ，下載所需的安裝程序後，直接安裝即可。

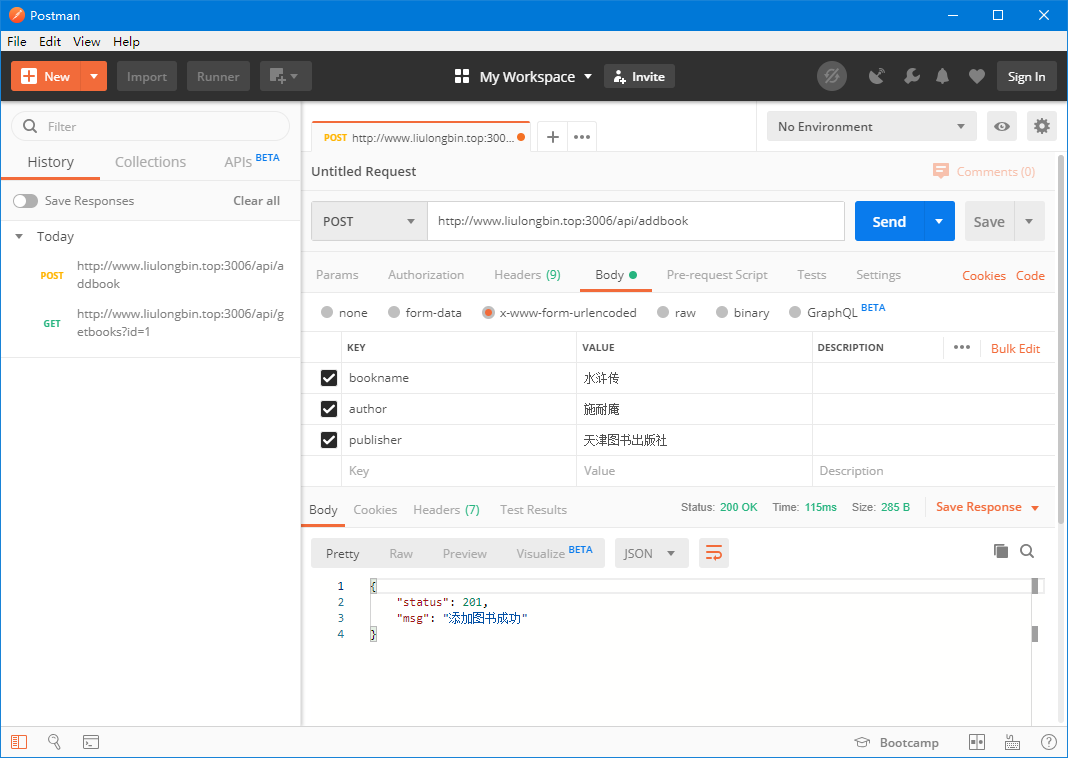
使用PostMan測試GET接口



步驟：

1. 選擇請求方式GET
2. 填寫URL地址
3. 填寫參數
4. 點擊Send按鈕發起GET請求
5. 查看服務器響應的結果

使用PostMan測試POST接口



1. 選擇請求方式POST
2. 填寫URL地址
3. 選擇Body面板，勾選數據格式x-www-form-urlencoded
4. 填寫要發送到服務器的數據
5. 點擊send按鈕發起post請求
6. 查看服務器響應的結果

### （4）接口文檔

接口文檔，即接口說明文檔，它是我們調用接口的依據。好的接口文檔包含了對接口URL、參數以及輸出內容的說明。我們參照接口文檔就能方便地知道接口的作用，以及如何調用接口。

接口文檔的組成部分：

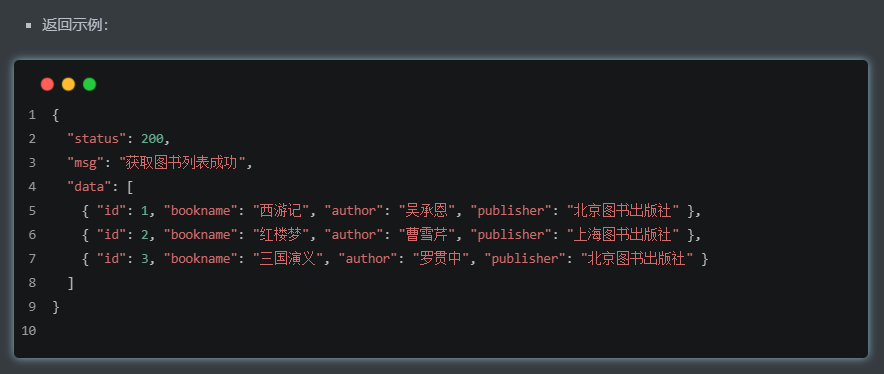
接口文檔可以包含很多信息，也可以按需進行精簡。一個合格的接口文檔，應該包含以下6項內容：

1. 接口名稱：用來標識各個接口的簡單說明，如登錄接口，獲取圖書列表接口等。
2. 接口URL：接口的調用地址。
3. 調用方式：如GET，POST。
4. 參數格式：接口需要傳遞的參數，每個參數必須包含參數名稱、參數類型、是否必選、參數說明這4項內容。
5. 響應格式：接口的返回值的詳細描述，一般包含數據名稱、數據類型、說明3項內容。
6. 返回示例（可選）：通過對象的形式，例舉服務器返回數據的結構。

接口文檔示例：







## 8 案例-圖書管理

### （1）渲染UI結構



### （2）案例用到的庫個插件

用到的CSS庫：bootstrap.css

用到的JavaScript庫：jquery.js

用到的vs code 插件：Bootstrap 3 Snippets 或者 Bootstrap 4 Snippets

## 9 案例-聊天機器人

分析

實現步驟：

1. 梳理案例的代碼結構
2. 將用戶輸入的內容渲染到聊天窗口
3. 發起請求獲取聊天消息
4. 將機器人的聊天內容轉換為语音
5. 通過<audio>播放語音
6. 使用回車鍵發送消息

梳理案例的代碼結構：

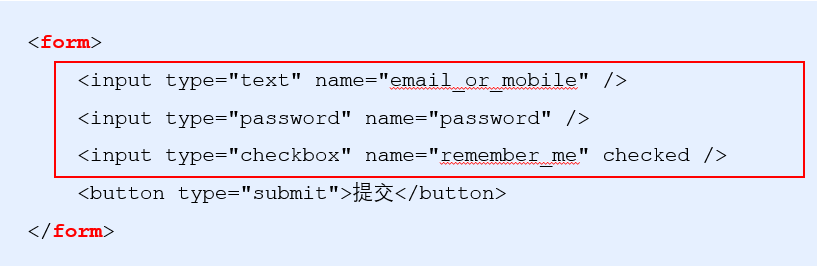
1. 梳理頁面的UI佈局
2. 將業務代碼抽離到chat.js中
3. 了解resetui()函數的作用：向聊天區域發送了新內容後，滾動條可以被自動拉到頁面最底部。

Form表單和模板引擎

## 1 form表單的基本使用

### （1）什麼是表單

表單在網頁中主要負責採集數據。HTML中的<form>標籤，用於採集用戶輸入的信息，並通過<form>標籤的提交操作，把採集到的信息提交到服務器端進行處理。



### （2）表單的組成部分

表單由三個基本部分組成：

* 表單標籤<form></form>
* 表單域：包含了文本框、密碼框、隱藏域、多行文本框、複選框、單選框、下拉選擇框和文件上傳框等。
* 表單按鈕

### （3）<form>標籤的屬性

<form>標籤的屬性是用來規定如何把採集到的數據發送到服務器。

|  |  |  |
| --- | --- | --- |
| **屬性** | **值** | **描述** |
| action | URL地址 | 規定當提交表單時，向何處發送表單數據， |
| method | get或post | 規定以何種方式把表單數據提交到action URL。 |
| enctype | application/x-www-form-urlencoded  發送前編碼所有字符（默認）。  multipart/form-data  不對字符編碼。  使用包含文件上傳控件的表單時，必須使用該值。  text/plain  空格轉換為“+”，但不對特殊字符編碼。（很少用） | 規定在發送表單數據之前如何對其進行編碼 |
| target | \_blank 在新窗口中打開  \_self 默認，在相同的框架中打開  \_parent 在父框架集中打開（很少用）  \_top 在整個窗口中打開（很少用）  *framename 在指定的框架中打開（很少用）* | 規定在何處打開 action URL |

#### ① action

action 屬性用來規定當提交表單時，向何處發送表單數據。

action 屬性的值應該是後端提供的一個 URL 地址，這個 URL 地址專門負責接收表單提交過來的資料。

當 <form> 表單在未指定 action 屬性值的情況下，action 的默認值為當前頁面的 URL 地址。

注意：當提交表單後，頁面會立即跳轉到 action 屬性指定的 URL 地址，同時將提交的表單數據以字符串的形式放到URL地址後面。

#### ② method

method 屬性用來規定以何種方式把表單數據提交到 action URL。

它的可選值有兩個，分別是 get 和 post。

默認情況下，method 的值為 get，表示通過URL地址的形式，把表單數據提交到 action URL。

如果以POST方式提交，是以隱藏的方式提交數據，URL地址欄中不會顯示提交的數據。

注意：

get 方式適合用來提交少量的、簡單的數據。

post 方式適合用來提交大量的、複雜的、或包含文件上傳的數據。

在實際開發中，<form> 表單的 post 提交方式用的最多，很少用 get。例如登錄、註冊、添加資料等表單操作，都需要使用 post 方式來提交表單。

#### ③ enctype

enctype 屬性用來規定在發送表單數據之前如何對數據進行編碼。

它的可選值有三個，默認情況下，enctype 的值為 application/x-www-form-urlencoded，表示在發送前編碼所有的字符。

#### ④ target

target 屬性用來規定在何處打開 action URL。

它的可選值有5個，默認情況下，target 的值是 \_self，表示在相同的框架中打開 action URL。

### （4）表單的同步提交及缺點

#### ① 什麼是表單的同步提交

通過點擊 submit 按鈕，觸發表單提交的操作，從而使頁面跳轉到 action URL 的行為，叫做表單的同步提交。

#### ② 表單同步提交的缺點

<form>表單同步提交後，整個頁面會發生跳轉，跳轉到 action URL 所指向的地址，用戶體驗很差。

<form>表單同步提交後，頁面之前的狀態和數據會丟失。

#### ③ 表單同步提交的缺點的解決方案

表單只負責採集數據，Ajax 負責將數據提交到服務器。

## 2 通過Ajax提交表單數據

### （1）監聽表單提交事件

在jQuery中，可以使用如下兩種方式，監聽到表單的提交事件：

    <script>

      $("#form1").submit(function (*e*) {

        alert("submit監聽到了表單的提交事件");

      });

      $("#form2").on("submit", function (*e*) {

        alert("on submit監聽到了表單的提交事件");

      });

    </script>

### （2）阻值表單默認提交行為

當監聽到表單的提交事件以後，可以調用事件對象的 event.preventDefault() 函數，來阻止表單的提交和頁面的跳轉，示例代碼如下：

    <script>

      $("#form1").submit(function (*e*) {

        e.preventDefault();

      });

      $("#form2").on("submit", function (*e*) {

        e.preventDefault();

      });

    </script>

### （3）快速獲取表單中的數據

為了簡化表單中數據的獲取操作，jQuery 提供了 serialize() 函數，其語法格式如下：

$(selector).serialize()

serialize() 函數的好處：可以一次性獲取到表單中的所有的數據。

注意：在使用 serialize() 函數快速獲取表單數據時，必須為每個表單元素添加 name 屬性！

  <body>

    <form id="form1">

      <input type="text" name="username" />

      <input type="password" name="password" />

      <button type="submit">提交</button>

    </form>

    <script src="./jquery.mini.js"></script>

    <script>

      $("#form1").on("submit", function (*e*) {

        e.preventDefault();

        console.log($("#form1").serialize());

        // 调用的结果：

        // username=用户名的值&password=密码的值

      });

    </script>

  </body>

## 3 案例 – 評論列表

步驟：

1. 渲染UI結構
2. 獲取評論列表
3. 發表評論
   1. 把包裹發表評論區域的div改造成form表單，並且為form表單添加ID=formAddCmt
   2. 為評論人的input和評論內容的textarea添加name屬性
   3. 監聽form表單的submit事件

    // $('#formAddCmt')獲取到的是jQuery對象，得把它轉換成一個原生的DOM對象才能使用原生form表單的函數reset()，轉換方法：在這個jQuery對象後面加個[0]

    $('#formAddCmt')[0].reset()

## 4 模板引擎的基本概念

### （1）渲染UI結構時遇到的問題

在上述評論列表的案例中，通過拼接字符串的形式來渲染UI結構。

* 如果UI結構比較複雜，則拼接字串的時候需要格外注意引號之前的嵌套。
* 且一旦需求發生變化，修改起來也非常麻煩。

這些問題可以通過模板引擎解決。

### （2）什麼是模板引擎

模板引擎，可以根據程序員指定的模板結構和數據，自動生成一個完整的HTML頁面。



### （3）模板引擎的好處

* 減少了字符串的拼接操作，降低犯錯風險
* 使代碼結構更清晰更易於閱讀
* 使代碼更易於維護

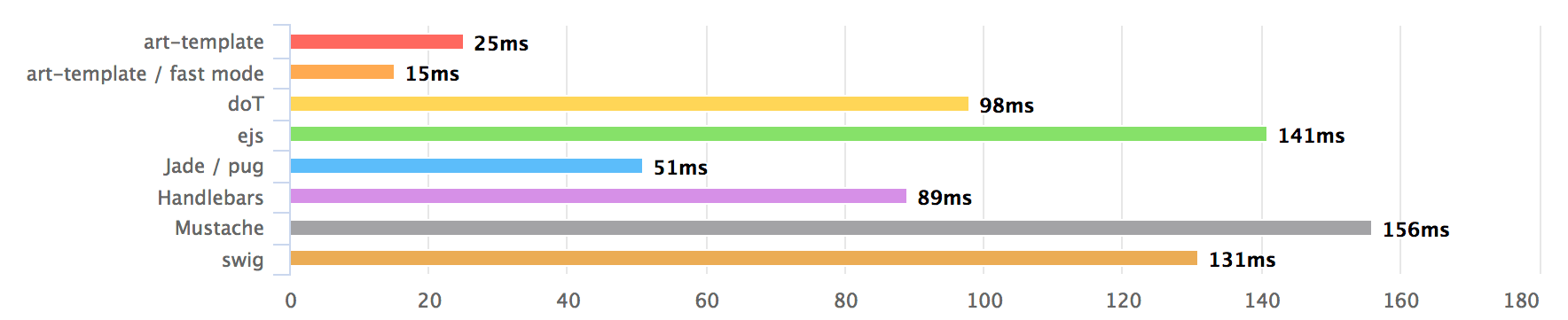
## 5 art-template模板引擎

### （1）art-template簡介

art-template 是一個簡約、超快的模板引擎。中文官網首頁為 <http://aui.github.io/art-template/zh-cn/index.html>

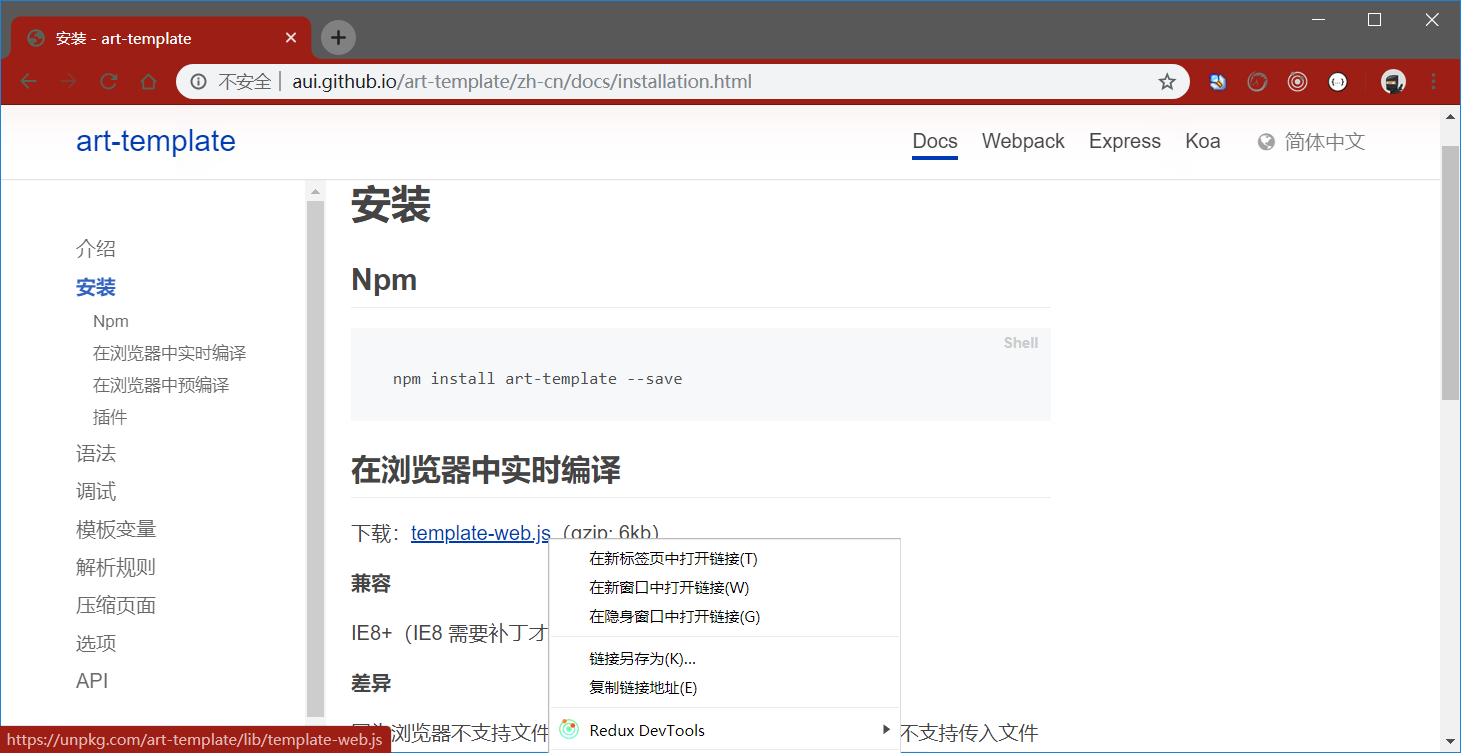
目前流行模板引擎的性能比較

（bar的長度越短，性能越好）



### （2）安裝art-template

在流覽器中訪問 <http://aui.github.io/art-template/zh-cn/docs/installation.html> 頁面，找到下載連接後，單擊鼠標右鍵，選擇“鏈接另存為”，將 art-template 下載到本地。然後通過 <script> 標籤加載到網頁上進行使用。



### （3）art-template使用步驟

1. 導入 art-template
2. 定義資料
3. 定義模板

* 普遍的script標籤沒有指定type的值，默認的是<script type=”text/javascript”></script>。意思是把這個script標籤裡面所有的值都當做JavaScript代碼去執行。
* 定義模板的時候要寫<script type=”text/html”></script>

1. 調用 template 函數
2. 渲染HTML結構

### （4）art-template標準語法

#### ① 什麼是標準語法

art-template 提供了 {{ }} 這種語法格式，在 {{ }} 內可以進行變量輸出，或循環數組等操作，這種 {{ }} 語法在 art-template 中被稱為標準語法。

#### ② 標準語法-輸出

在 {{ }} 語法中，可以進行變量的輸出、對象屬性的輸出、三元表達式輸出、邏輯或輸出、加減乘除等表達式輸出。

{{value}}

{{obj.key}}

{{obj['key']}}

{{a ? b : c}}

{{a || b}}

{{a + b}}

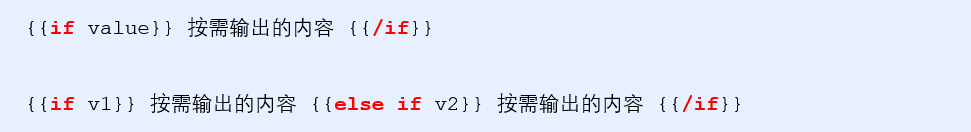
#### ③ 標準語法-原文輸出

如果要輸出的 value 值中，包含了 HTML 標籤結構，則需要使用原文輸出語法，才能保證 HTML 標籤被正常渲染。

{{**@** value }}

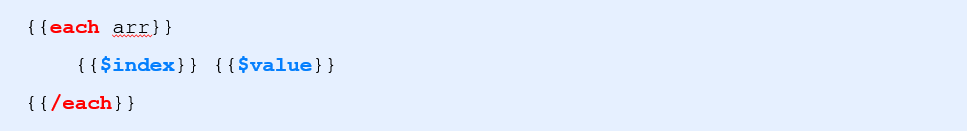
#### ④ 標準語法-條件輸出

如果要實現條件輸出，則可以在 {{ }} 中使用 if … else if … /if 的方式，進行按需輸出。



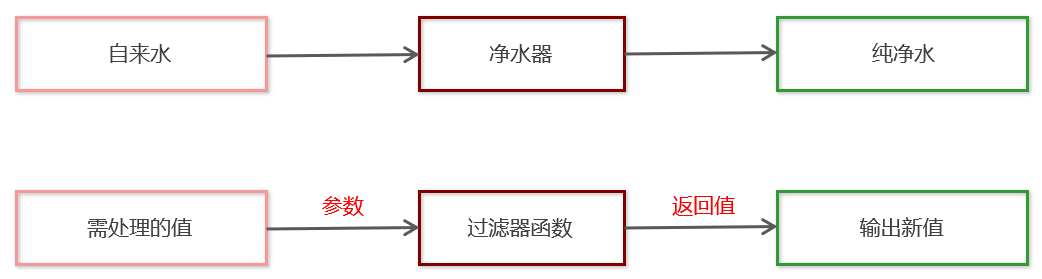
#### ⑤ 標準語法-循環輸出

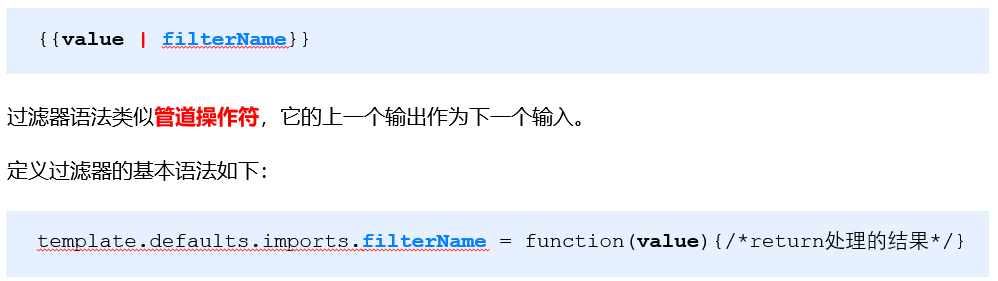
如果要實現循環輸出，則可以在 {{ }} 內，通過 each 語法循環數組，當前循環的索引使用 **$index** 進行訪問，當前的循環項使用 **$value** 進行訪問。



#### ⑥ 標準語法-過濾器

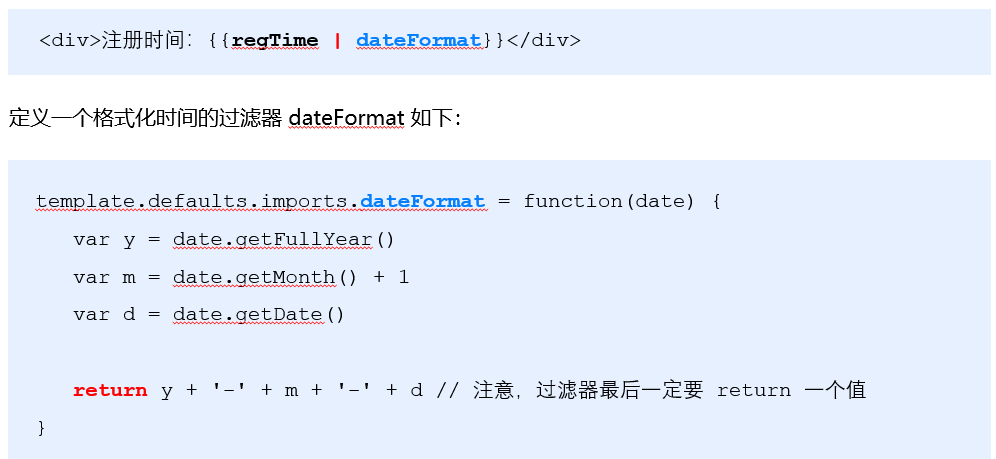
過濾器的本質是個函數





filterName可以自定義為其它的名字

例子



return的值是{{ }}最終輸出的值

## 6 模板引擎art-template案例：新聞列表

實現步驟：

1. 獲取新聞數據
2. 定義 template 模板
3. 編譯模板
4. 定義時間過濾函數
5. 定義補零函數

## 7 模板引擎的實現原理

### （1）正則與字符串操作

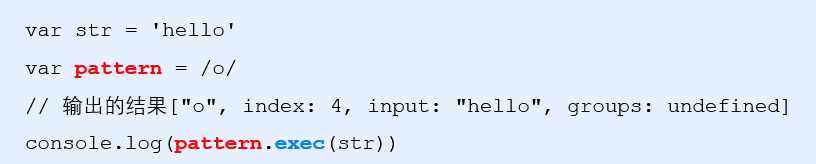
#### ① 基本語法

exec() 函數用於檢索字符串中的正則表達式的匹配。

如果字符串中有匹配的值，則返回該匹配值，否則返回 null。

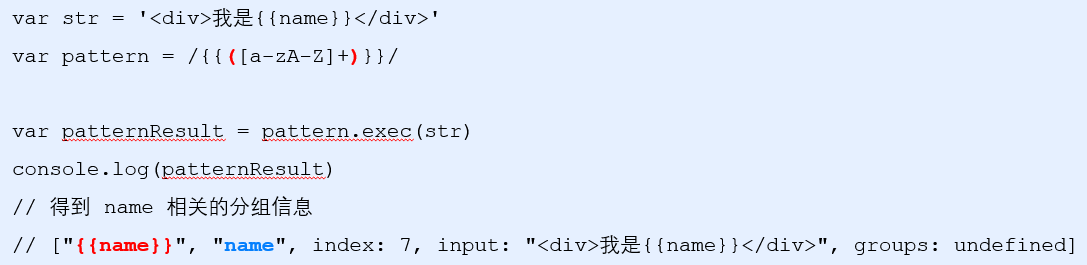
RegExpObject.**exec**(string)

示例代碼



#### ② 分組

正則表達式中()包起來的內容表示一個分組，可以通過分組來提取自己想要的內容，示例代碼如下：

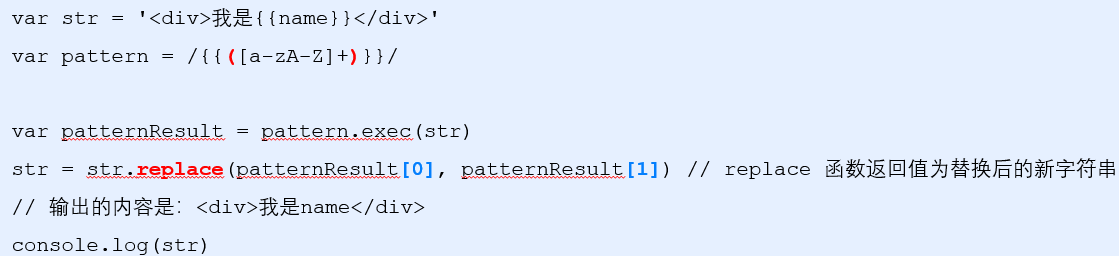


#### ③ 字符串的replace函數

replace()函數用於在字符串中用一些字符替換另一些字符，語法格式如下：



示例代碼如下：



#### ④ 多次篩選（exec）和替換（replace）

    <script>

      var str = "<div>{{name}}今年{{ age }}岁了</div>";

      var pattern = /{{\s\*([a-zA-Z]+)\s\*}}/;

      var patternResult = pattern.exec(str);

      console.log("patternResult1：");

      console.log(patternResult);

      str = str.replace(patternResult[0], patternResult[1]);

      console.log(str); // 输出 <div>name今年{{ age }}岁了</div>

      patternResult = pattern.exec(str);

      console.log("patternResult2：");

      console.log(patternResult);

      str = str.replace(patternResult[0], patternResult[1]);

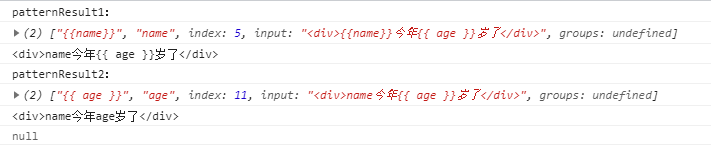
      console.log(str); // 输出 <div>name今年age岁了</div>

      patternResult = pattern.exec(str);

      console.log(patternResult); // 输出 null

    </script>

打印結果：



#### ⑤ 使用while循環來多次篩選（exec）和替換（replace）

第④裡面的多次篩選（exec）和替換（replace）有很多重複代碼，太繁瑣，可以用while循環來簡化。

    <script>

      var str = "<div>{{name}}今年{{ age }}岁了。{{ qaz}} vs {{wsx  }}</div>";

      var pattern = /{{\s\*([a-zA-Z]+)\s\*}}/;

      var patternResult = null;

      while ((patternResult = pattern.exec(str))) {

        str = str.replace(patternResult[0], patternResult[1]);

      }

      console.log(str); // 输出 <div>name今年age岁了。qaz vs wsx</div>

    </script>

#### ⑥ replace替換為數據對象的值（模板引擎的核心代碼）

    <script>

      var data = { name: "张三", age: 20 };

      var str = "<div>{{name}}今年{{ age }}岁了</div>";

      var pattern = /{{\s\*([a-zA-Z]+)\s\*}}/;

      var patternResult = null;

      while ((patternResult = pattern.exec(str))) {

        str = str.replace(patternResult[0], data[patternResult[1]]);

      }

      console.log(str); //<div>张三今年20岁了</div>

    </script>

### （2）實現簡易的模板引擎

步驟：

1. 定義模板結構
2. 預調用模板引擎
3. 封裝 template 函數
4. 導入並使用自定義的模板引擎

    <script src="./template.js"></script>

  </head>

  <body>

    <div id="user-box"></div>

    <!-- 定义模板结构 -->

    <script type="text/html" id="tpl-user">

      <div>姓名：{{name}}</div>

      <div>年龄：{{ age }}</div>

      <div>性别：{{  gender}}</div>

      <div>住址：{{address  }}</div>

    </script>

    <script>

      // 定义数据

      var data = { name: "zs", age: 28, gender: "男", address: "北京顺义马坡" }; // 调用模板函数

      var htmlStr = template("tpl-user", data); // 渲染HTML结构

      document.getElementById("user-box").innerHTML = htmlStr;

    </script>

  </body>

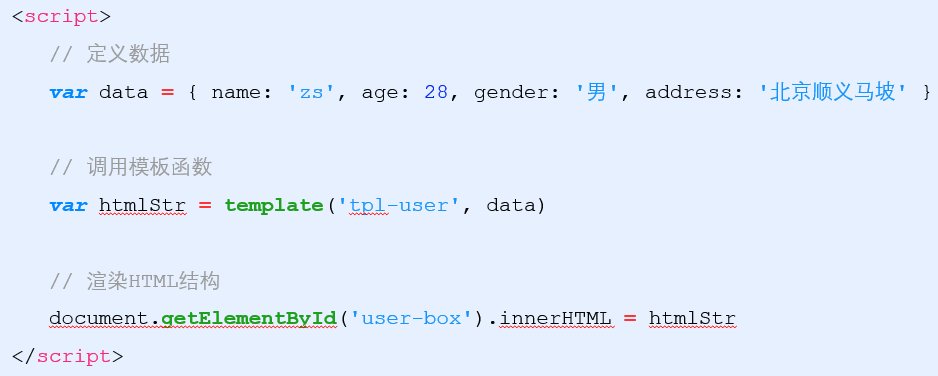
#### ① 定義模板結構

下列代碼寫在文件index.html裡



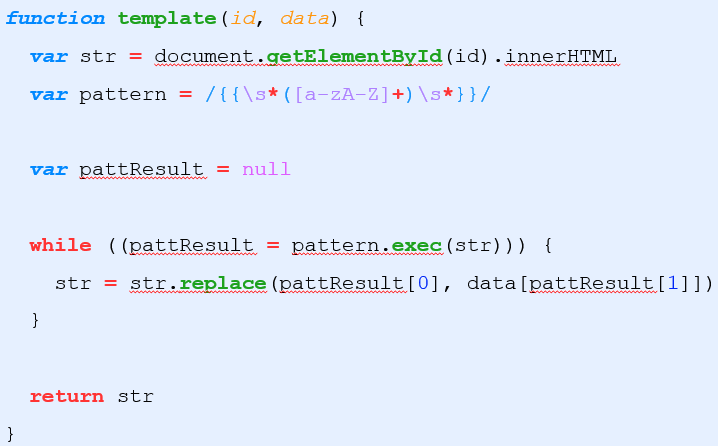
#### ② 預調用模板引擎

下列代碼寫在文件index.html裡



#### ③ 封裝 template 函數

下列函數寫在一個js文件裡，可以取名template.js。



function template(*id*, *data*) {

  var str = document.getElementById(id).innerHTML

  var pattern = /{{\s\*([a-zA-Z]+)\s\*}}/

  var pattResult = null

  while ((pattResult = pattern.exec(str))) {

    str = str.replace(pattResult[0], data[pattResult[1]])

  }

  return str

}

#### ④ 導入並使用自定義的模板引擎

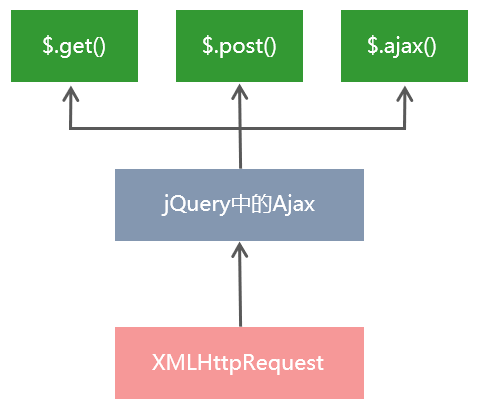
要使用的時候，把template.js引入index.html

Ajax進階

## 1 XMLHttpRequest的基本使用

## （1） 什么是XMLHttpRequest

XMLHttpRequest（簡稱 xhr）是流覽器提供的 Javascript 對象，通過它，可以請求服務器上的數據資源。之前所學的 jQuery 中的 Ajax 函數，就是基於 xhr 對象封裝出來的。



可以直接使用xhr對象發起Ajax請求。

## （2） 使用xhr發起GET請求

    <script>

      // 1. 創建 XHR 對象

      var xhr = new XMLHttpRequest();

      // 2. 調用 open 函數，指定 請求方式 與 URL地址

      xhr.open("GET", "http://www.liulongbin.top:3006/api/getbooks");

      // 3. 調用 send 函數，發起 Ajax 請求

      xhr.send();

      // 4. 監聽 onreadystatechange 事件

      xhr.onreadystatechange = function () {

        // 4.1 監聽 xhr 對象的請求狀態 readyState ；與服務器響應的狀態 status

        if (xhr.readyState === 4 && xhr.status === 200) {

          // 4.2 打印服務器響應回來的數據

          console.log(xhr.responseText);

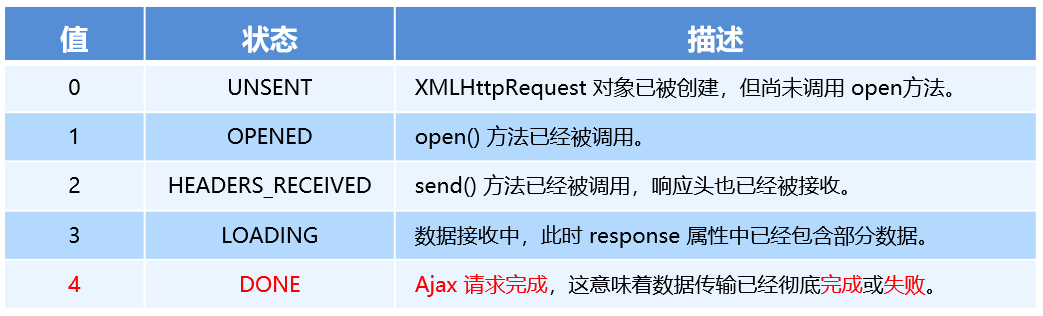
        }

      };

    </script>

#### ① readyState屬性

XMLHttpRequest對象的readyState屬性，用來表示當前Ajax請求所處的狀態。每個Ajax請求必然處於以下狀態中的一個：



#### ② 查詢字符串

在xhr.open()裡面，第2個參數是URL地址。在URL地址後面拼接的參數，叫做查詢字符串。這是鍵值對的形式。

xhr.open('GET', 'http://www.liulongbin.top:3006/api/getbooks**?id=1**')

## 2 數據交換格式

3 封裝自己的Ajax函數

4 XMLHttpRequest Level2的新特性

5 jQuery高級用法

6 axios