# Jekyll-search.js

Jekyll-search.js是一个客户端全文搜索引擎，我们使用它来完成搜索功能。项目中有两种搜索形式，顶部菜单栏搜索和搜索页面（search.html）。

顶部菜单栏搜索实现步骤如下：

* 在搜索框中输入检索项；
* 搜索引擎在JSON文件中按检索项搜索相关内容；
* 在搜索框下方显示搜索结果，并在结果最后加上跳转到search.html的链接；

搜索页面实现步骤如下：

* 单击上方的链接，检索项以参数形式被发送到search.html；
* search.html读取参数，搜索引擎在JSON文件中按检索项搜索相关内容；
* search.html显示搜索结果；

# 顶部菜单栏搜索

以“\_includes/header.html”为例，在顶部菜单后面加上搜索相关代码，包括一个搜索框，显示搜索结果的容器，JavaScript 库（Jekyll-search.js）,和搜索引擎初始化代码，代码如下：

|  |
| --- |
| <div id="search-demo-container">  <input type="text" id="search-input" placeholder="search...">  <ul id="results-container"></ul>  </div>  <script src="/js/jekyll-search.js" type="text/javascript"></script>  <script type="text/javascript">  function appendContentSearchLink(container, condition) {  container.innerHTML += '<li><a href="/dosearch.html?queryString=' + condition + '" class="search-for"><span title="搜索‘' + condition + '’"><em>搜索‘' + condition + '’</em></span></a></li>';  }  SimpleJekyllSearch.init({  searchInput: document.getElementById('search-input'),  resultsContainer: document.getElementById('results-container'),  dataSource: '/search.json',  searchResultTemplate: '<li><a href="{url}" title="Search configuration">{title}</a></li>',  noResultsText: 'No results found.',  limit: 10,  fuzzy: true,  isLimit: true,  afterRender: appendContentSearchLink  })  </script> |

* <input type="text" id="search-input" placeholder="search...">：搜索框
* <ul id="results-container"></ul>：显示搜索结果的容器
* <script src="/js/jekyll-search.js" type="text/javascript"></script>：引入jekyll-search.js库
* 方法appendContentSearchLink：在搜索结果后加跳转到search.html的链接
* SimpleJekyllSearch.init：搜索引擎初始化
  + searchInput：定义搜索框
  + resultsContainer：定义搜索结果显示容器
  + dataSource：包含所有我们想搜索的内容的JSON文件
  + searchResultTemplate：搜索结果显示模板
  + noResultsText：搜索无结果时显示的文本
  + limit：搜索结果显示条数限制，默认10
  + fuzzy：true表示模糊搜索，false表示精确搜索，默认false
  + isLimit：true表示限制搜索结果显示条数，false表示无限制，显示所有结果，默认false
  + afterRender：搜索结果加载到页面后执行的事件

# 搜索页面search.html

以“\_includes/search.html”为例，这个页面包括一个搜索框，搜索按钮，显示搜索结果的容器，和搜索引擎初始化代码，代码如下：

|  |
| --- |
| <div id="search-results-header" >  <input type="text" name="queryString" id="query-string" value="" incremental="true" maxlength="255" />  <button type="button" id="search-content-button">Search</button>  </div>  <section>  <div>  <ul id="results-content-container"></ul>  </div>  </section>  <script type="text/javascript">  function getQueryString(name) {  var reg = new RegExp("(^|&)" + name + "=([^&]\*)(&|$)", "i");  var r = window.location.search.substr(1).match(reg);  if (r != null) return decodeURI(r[2]);  return null;  }  (function() {  var queryString = getQueryString('queryString');  if (queryString) {  document.getElementById('query-string').value = queryString;  function triggerSearch() {  document.getElementById("search-content-button").click();  };  ContentJekyllSearch.init({  searchInput: document.getElementById('query-string'),  searchButton: document.getElementById('search-content-button'),  resultsContainer: document.getElementById('results-content-container'),  dataSource: '/search.json',  searchResultTemplate: '<li><a href="{url}" title="Search configuration">{title}</a><div>{content}</div></li>',  noResultsText: 'No results found.',  fuzzy: true,  afterInit: triggerSearch  });  }  })();  </script> |

* <input type="text" name="queryString" id="query-string" value="" incremental="true" maxlength="255" />：搜索框
* <button type="button" id="search-content-button">Search</button>：搜索按钮
* <ul id="results-content-container"></ul>：显示搜索结果的容器
* 方法getQueryString：获取检索项
* 方法triggerSearch：在搜索引擎初始化后，自动触发搜索
* SimpleJekyllSearch.init：搜索引擎初始化
  + searchInput：定义搜索框
  + resultsContainer：定义搜索结果显示容器
  + dataSource：包含所有我们想搜索的内容的JSON文件
  + searchResultTemplate：搜索结果显示模板
  + fuzzy：true表示模糊搜索，false表示精确搜索
  + afterInit：在搜索引擎初始化后执行的事件

search.html和顶部菜单栏搜索的区别：

* 如果两个搜索处于同一个页面，则不需要重复引入jekyll-search.js库文件。
* 顶部搜索显示搜索结果条数有限制，需要设置初始化参数limit和isLimit；而search.html显示所有结果，isLimit默认为false，所以可以省略设置。
* 顶部搜索需要在搜索结果后加跳转到search.html的链接。
* search.html需要在跳转到该页面时立即搜索。