Ace editor 调研文档

一、简介

1、什么是 Ace editor

Ace(Ajax.org Cloud9 Editor)是一个用 JavaScript 编写的可嵌入代码编辑器。它与 Sublime，Vim 和 TextMate 等本地编辑器的功能和性能相匹配。它可以轻松地嵌入任何网页和 JavaScript 应用程序中。

2、Ace logo

3、特征

- 超过 110 种语言的语法高亮显示 (可以导入 TextMate/Sublime Text.tmlanguage 文件)
- 超过 20 个主题（可以导入 TextMate/Sublime Text .tmtheme 文件)
- 自动缩进和缩进
- 可选的命令行

- 处理大型文档（四百万行似乎是极限！
- 完全可定制的键绑定，包括 vim 和 Emacs 模式
- 搜索并替换为正则表达式
- 突出显示匹配的括号
- 在软标签和真实标签之间切换
- 显示隐藏字符
- 使用鼠标拖放文本
- 换行
- 代码折叠
- 多个光标和选择
- 实时语法检查器（目前为 JavaScript/CoffeeScript/CSS/XQuery）
- 剪切、复制和粘贴功能

4、发展历程

Skywriter/Bespin 和 Ace 开始作为两个独立的项目，都旨在建立一个不妥协的 Web 的代码编辑器组件。Bespin 最初是作为 Mozilla Labs 基于<canvas>标签，而 Ace 是 Cloud9 IDE 的编辑器组件，并使用 DOM 进行渲染。在 2010 年 JSConf.eu 柏林发布 Ace 之后，Skywriter 团队决定将 Ace 与简化的 Ace 合并。 Skywriter 的插件系统和 Skywriter 的一些插件系统的版本扩展点。所有这些更改都已合并回 Ace 现在，它取代了 Skywriter。Cloud9 IDE 和 Mozilla 都在积极开发和维护 Ace。

二、快速开始

1、简单使用

本地引用，ace editor 在 GitHub 上下载即可， 注意要下载已经编译过的版本；

下载地址：GitHub - ajaxorg/ace-builds: Packaged version of Ace code editor

```
<div id="editor" style="height: 500px; width: 500px">let hello = 'hello world'</div>
<script src="/ace-builds-master/src/ace.js" type="text/javascript"></script>
<script>
  var editor = ace.edit("editor");
  editor.setTheme("ace/theme/monokai");
  editor.session.setMode("ace/mode/javascript");
</script>
```

2、设置主题

要设置主题，请为编辑器配置要使用的主题路径。主题文件将按需加载：

```
  editor.setTheme("ace/theme/twilight");
```

3、设置编程语言模式

默认情况下，编辑器支持纯文本模式。所有其他语言模式都作为单独的模块提供，按需加载，如下所示：

```
  editor.session.setMode("ace/mode/javascript");
```

4、编辑器状态

Ace 将所有编辑器状态（选择，滚动位置等）保留在 editor.session 中，这对于制作可切换式编辑器非常有用：

```
var EditSession = require("ace/edit_session").EditSession
var js = new EditSession("some js code")
var css = new EditSession(["some", "css", "code here"])

// 要将文档加载到编辑器中，只需这样调用
editor.setSession(js)
```

5、在项目中配置 ace

```
// 将代码模式配置到 ace 选项
ace.edit(element, {
  mode: "ace/mode/javascript",
  selectionStyle: "text"
})
// 使用 setOptions 方法一次设置多个选项
editor.setOptions({
  autoScrollEditorIntoView: true,
  copyWithEmptySelection: true,
});
// 单独设置 setOptions 方法
editor.setOption("mergeUndoDeltas", "always");
// 一些选项也直接设置，例如：
editor.setTheme(...)
// 获取选项设置值
editor.getOption("optionName");
// 核心 Ace 组件包括（editor, session, renderer, mouseHandler）
setOption(optionName, optionValue)
setOptions({
  optionName: optionValue
})
getOption(optionName)
getOptions()
```

常规操作

- 设置和获取内容：

```
editor.setValue("the new text here"); // 或 session.setValue
editor.getValue(); // 或 session.getValue
```

- 获取选定的文本：

```
editor.session.getTextRange(editor.getSelectionRange());
```

- 在光标处插入：

```
editor.insert("Something cool");
```

- 获取当前光标所在的行和列：

```
editor.selection.getCursor();
```

- 转到某一行：

```
editor.gotoLine(lineNumber);
```

- 获取总行数：

```
editor.session.getLength();
```

- 设置默认标签大小：

```
editor.getSession().setTabSize(4);
```

- 使用软标签：

```
editor.getSession().setUseSoftTabs(true);
```

- 设置字体大小：

```
document.getElementById('editor').style.fontSize='12px';
```

- 切换自动换行：

```
editor.getSession().setUseWrapMode(true);
```

- 设置行高亮显示：

```
editor.setHighlightActiveLine(false);
```

- 设置打印边距可见性：

```
editor.setShowPrintMargin(false);
```

- 设置编辑器为只读：

```
editor.setReadOnly(true);  // false 为可编辑
```

窗口自适应

编辑器仅在 resize 事件触发时时调整自身大小。要想以其他方式调整编辑器 div 的大小，并

且需要调整编辑器大小，请使用以下命令：

```
editor.resize()
```

在代码中搜索

主要的 ACE 编辑器搜索功能在 search.js.中定义。以下选项可用于搜索参数：

- needle: 要查找的字符串或正则表达式
- backwards: 是否从当前光标所在的位置向后搜索。默认为“false”
- wrap: 当搜索到达结尾时，是否将搜索返回到开头。默认为“false
- caseSensitive: 搜索是否应该区分大小写。默认为“false”
- wholeWord: 搜索是否只匹配整个单词。默认为“false”
- range: 搜索匹配范围，要搜索整个文档则设置为空
- regExp: 搜索是否为正则表达式。默认为“false”
- start: 开始搜索的起始范围或光标位置
- skipCurrent: 是否在搜索中包含当前行。默认为“false”

下面是一个如何在编辑器对象上设置搜索的示例：

```
editor.find('needle', {
  backwards: false,
  wrap: false,
  caseSensitive: false,
  wholeWord: false,
  regExp: false
});
editor.findNext();
editor.findPrevious();
```

这是执行替换的方法：

```
editor.find('foo');
editor.replace('bar');
```

全部替换：

```
editor.replaceAll('bar');
```

事件监听

change

```
editor.getSession().on('change', callback);
```

changeSelection

```
editor.getSession().selection.on('changeSelection', callback);
```

changeCursor

```
editor.getSession().selection.on('changeCursor', callback);
```

添加新的命令和绑定

将指定键绑定并分配给自定义功能：

```
editor.commands.addCommand({
    name: 'myCommand',
    bindKey: { win: 'Ctrl-M', mac: 'Command-M' },
    exec: function (editor) {
      //...
    }
 });
```

配置模式和主题的动态加载

默认情况下，ace 通过查找 ace.js 的脚本节点来获取动态加载的 url。 如果 ace.js 没有加载单独的脚本标签，这不起作用，在这种情况下，需要显式设置 url

```
ace.config.set("basePath", "https://url.to.a/folder/that/contains-ace-modes");
```

单独一个模块的路径可以配置为：

```
ace.config.setModuleUrl("ace/theme/textmate", "url for textmate.js");
```

当将 ace 与 webpack 一起使用时，可以使用

```
require("ace-builds/webpack-resolver");
```

使用撤消管理器

要将下一次编辑的撤消增量与前一次编辑分组，请将“mergeUndoDeltas”设置为 true

```
editor.session.mergeUndoDeltas = true;
editor.session.insert({ row: 0, column: 0 }, Date() + "");
```

要启动新的撤消组，请使用“markUndoGroup”方法

```
editor.insertSnippet("a$0b");
editor.session.markUndoGroup();
editor.insertSnippet("x$0y");
```

三、配置项

以下是目前所支持的主要选项的列表。除非另有说明，否则选项值皆为布尔值，可以通过 editor.setOption 来设置。

1、editor 选项

选项名	值类型	默认值	可选值	备注
selectionStyle	String	text	line|text	选中样式
highlightActiveLine	Boolean	true	-	高亮当前行
highlightSelectedWord	Boolean	true	-	高亮选中文本
readOnly	Boolean	false	-	是否只读
cursorStyle	String	ace	ace|slim|smooth|wide	光标样式

选项名	值类型	默认值	可选值	备注
mergeUndoDeltas	String|Boolean	false	always	合并撤销
behavioursEnabled	Boolean	true	-	启用行为
wrapBehavioursEnabled	Boolean	true	-	启用换行
autoScrollEditorIntoView	Boolean	false	-	启用滚动
copyWithEmptySelection	Boolean	true	-	复制空格
useSoftTabs	Boolean	false	-	使用软标签
navigateWithinSoftTabs	Boolean	false	-	软标签跳转
enableMultiselect	Boolean	false	-	选中多处

2、renderer 选项

选项名	值类型	默认值	可选值	备注
hScrollBarAlwaysVisible	Boolean	false	-	纵向滚动条始终可见
vScrollBarAlwaysVisible	Boolean	false	-	横向滚动条始终可见
highlightGutterLine	Boolean	true	-	高亮边线
animatedScroll	Boolean	false	-	滚动动画
showInvisibles	Boolean	false	-	显示不可见字符
showPrintMargin	Boolean	true	-	显示打印边距
printMarginColumn	Number	80	-	设置页边距
printMargin	Boolean|Number	false	-	显示并设置页边距

选项名	值类型	默认值	可选值	备注
fadeFoldWidgets	Boolean	false	-	淡入折叠部件
showFoldWidgets	Boolean	true	-	显示折叠部件
showLineNumbers	Boolean	true	-	显示行号
showGutter	Boolean	true	-	显示行号区域
displayIndentGuides	Boolean	true	-	显示参考线
fontSize	Number|String	inherit	-	设置字号
fontFamily	String	inherit		设置字体
maxLines	Number	-	-	至多行数
minLines	Number	-	-	至少行数
scrollPastEnd	Boolean|Number	0	-	滚动位置
fixedWidthGutter	Boolean	false	-	固定行号区域宽度
theme	String	-	-	主题引用路径，例如 "ace/theme/textmate

3、mouseHandler 选项

选项名	值类型	默认值	可选值	备注
scrollSpeed	Number	-	-	滚动速度
dragDelay	Number	-	-	拖拽延时
dragEnabled	Boolean	true	-	是否启用拖动
focusTimout	Number	-	-	聚焦超时

选项名	值类型	默认值	可选值	备注
tooltipFollowsMouse	Boolean	false	-	鼠标提示

4、session 选项

选项名	值类型	默认值	可选值	备注
firstLineNumber	Number	1	-	起始行号
overwrite	Boolean	-	-	重做
newLineMode	String	auto	auto|unix|windows	新开行模式
useWorker	Boolean	-	-	使用辅助对象
useSoftTabs	Boolean	-	-	使用软标签
tabSize	Number	-	-	标签大小
wrap	Boolean	-	-	换行
foldStyle	String	-	markbegin|markbeginend|manual	折叠样式

5、扩展选项

选项名	值类型	默认值	可选值	备注
enableBasicAutocompletion	Boolean	-	-	启用基本自动完成
enableLiveAutocompletion	Boolean	-	-	启用实时自动完成
enableSnippets	Boolean	-	-	启用代码段
enableEmmet	Boolean	-	-	启用 Emmet

选项名	值类型	默认值	可选值	备注
useElasticTabstops	Boolean	-	-	使用弹性制表位

四、扩展

基于 vue3+vite 使用 ace-builds 封装 ace-editor

安装

```
npm install ace-builds --save-dev
//引入 ace 报错需要安装
npm install vue-loader-v16 -D
```

封装

webpack 环境必备：import "ace-builds/webpack-resolver";

非 webpack 环境不需要引入

```
<template>
  <div class="aceEditor" ref="aceEditor"></div>
</template>

<script lang="ts" setup>
import { onBeforeUnmount, onMounted, ref, watch } from 'vue';
import * as ace from 'ace-builds';
import "ace-builds/src-noconflict/mode-javascript"; // 语言模式
import "ace-builds/src-noconflict/theme-monokai" // 主题
import "ace-builds/src-noconflict/ext-language_tools"; // 语法提示
import "ace-builds/src-noconflict/snippets/javascript"; // 语法段提示模块
const props = withDefaults(defineProps<{
  value: string,
}>(), {
})
```

```
const emits = defineEmits(['update:value']);
let editor: any = null;
const aceEditor = ref<string | Element>('')
// 编辑器默认配置项
const options = {
  theme: 'ace/theme/monokai',
  //mode: 'ace/mode/javascript',
  mode: 'ace/mode/mylang',
  tabSize: 1,
  maxLines: 25,
  minLines: 25,
  showPrintMargin: false,
  fontSize: 14,
  printMarginColumn: 20,
  useWorker: false,
  showLineNumbers: true, // 显示行号
  showGutter: true, // 显示行号区域
  highlightActiveLine: false,
  highlightSelectedWord: false, // 高亮选中文本
  readOnly: false, // 控制编辑器是否只读
  enableSnippets: true, // 启用代码段
  enableLiveAutocompletion: true, // 启用实时自动完成
  enableBasicAutocompletion: true,    // 启用基本自动完成
}
// 初始化编辑器
const initEditor = () => {
  if (editor) editor.destroy();
  // 初始化
  editor = ace.edit(aceEditor.value, options);
  // 切换自动换行
  editor.getSession().setUseWrapMode(true);
  // 支持双向绑定
  editor.setValue(props.value ? props.value : "");
  editor.on("change", () => {
    emits("update:value", editor.getValue());
  })
}
watch(
  () => props.value,
  (newProps) => {
    //解决光标移动问题
    const position = editor.getCursorPosition();
    editor.getSession().setValue(newProps);
    editor.clearSelection();
```

```
      editor.moveCursorToPosition(position);
    }
  );
  onMounted(() => {
    initEditor()
  });
  onBeforeUnmount(() => {
    editor.destroy();
  });
  </script>
  <style>
  .aceEditor {
    width: 500px;
    height: 500px;
  }
  </style>
```

页面使用

```
  <template>
      <Ace :value="value" @update:value="getVal" />
  </template>
```

自定义语法提示

ace 编辑器语法提示最基础设置如下：

```
   {
      enableBasicAutocompletion: true, //boolean 或 completer 数组
      enableLiveAutocompletion: true, //boolean 或 completer 数组
   }
```

completer，就是拥有一个 getCompletions(editor, session, pos, prefix, callback) 属性的 object，自定义语法提示可以通过 callback(null, tipsData) 回传

如果 enableBasicAutocompletion 和 enableLiveAutocompletion 的值为 completer 数组，就会覆盖编辑器默认的 completers，不推荐使用。

1、enableBasicAutocompletion 设置 enableBasicAutocompletion = true，就会在全局

commands 中增加 Autocomplete.startCommand 命令（startCommand.bindKey = “Ctrl-Space|Ctrl-Shift-Space|Alt-Space”）

2、enableLiveAutocompletion 设置 enableLiveAutocompletion = true，就会在输入内容时，弹出语法提示框，但是逻辑代码中忽略了一些情况，如删除（vscode 也存在此问题...）。所以如果交互要求变动就弹出提示的话，可以在 editor 绑定的 onChange 事件中触发命令：

```
editor.execCommand("startAutocomplete");
```

自定义提示语句

```
const tipsData = [
  // name 显示的名称；value 插入的值；score 权重（数值越大，提示越靠前）；meta 描述
  { meta: "自定义", caption: "hello", value: "hello", score: 1 },
  { meta: "自定义", caption: "world", value: "world", score: 1 },
  { meta: "自定义", caption: "ace", value: "ace editor", score: 1 },
];

let langTools = ace.require("ace/ext/language_tools");
  langTools.addCompleter({
    getCompletions: function (state, session, pos, prefix, callback) {
      console.log("prefix: ", prefix);
      if (prefix.length === 0) {
        callback(null, []);
        return;
      }
      callback(null, tipsData);
    },
  });
```

五、进阶

扩展自定语言

步骤：

1、安装 ace-builds 库

```
npm install ace-builds
```

2、引入

```
import ace from "ace-builds";
```

3、新增自定义语言模式

在 ace-builds/src-noconflict/目录下新建 mode-mylang.js 文件，新增自定义语言模式，首先需要了解如何定义一个 mode: mode 通常是一个对象，用于定义 Ace Editor 如何处理特定语言的代码。它必须包括一组规则（Rules），每个规则描述了如何处理输入文本中的单个字符序列。Ace Editor 中的规则是由 Tokenizer 对象处理的.Tokenizer 是 Ace Editor 内置的一种基于正则表达式的解析器，用于将输入文本转换为标记（Token）流。标记是 Ace Editor 中的基本元素，它们由不同类型的 token 组成，例如：keyword、comment、string 等等

可参考官方文档 Ace - The High Performance Code Editor for the Web (c9.io)

```
ace.define("ace/mode/mylang",
["require","exports","module","ace/lib/oop","ace/mode/text","ace/mode/custom_hig
hlig   ht_rules"], function (require, exports, module) {
  var oop = require("ace/lib/oop");
  var TextMode = require("ace/mode/text").Mode;
  var MyLangHighlightRules =
    require("ace/mode/mylang_highlight_rules").MyLangHighlightRules;
  var Tokenizer = require("ace/tokenizer").Tokenizer;

  var Mode = function () {
    this.HighlightRules = MyLangHighlightRules;
    this.$tokenizer = new Tokenizer(new MyLangHighlightRules().getRules());
  };
  oop.inherits(Mode, TextMode);
```

```
    (function() {
      // Load stylesheet  加载 css 样式设置，以便控制自定义语言关键词高亮颜色
      var dom = require("ace/lib/dom");
      dom.importCssString(exports.cssText, exports.cssClass);
    }).call(Mode.prototype);
    (function () {
      // 添加代码提示
      this.completer = {
        getCompletions: function (editor, session, pos, prefix, callback) {
          var wordList = [
            "hello",
            "world",
            "AceEditor",
            "hello world this is AceEditor",
          ];
          callback(
            null,
            wordList.map(function (word) {
              return {
                caption: word,
                value: word,
                meta: "mylang", // 自定义语言标识
              };
            })
          );
        },
      };
    }.call(Mode.prototype));
    exports.Mode = Mode;
  });
```

在上述代码中，定义了自己的语言模式 mylang，并将其继承自 TextMode。添加了代码提示语 css 样式控制器，

4、新增自定义语言高亮规则

同时，我们需要定义这种模式下的高亮规则，因此还需要定义高亮规则文件

```
ace.define(
  "ace/mode/mylang_highlight_rules",
  ["require","exports","module","ace/lib/oop","ace/mode/text_highlight_rules"],
```

```
function (require, exports, module) {
  var oop = require("ace/lib/oop");
  var TextHighlightRules =
    require("ace/mode/text_highlight_rules").TextHighlightRules;

  var MyLangHighlightRules = function () {
    // 定义高亮规则
    var keywordList = "let|const|function|world";  // 高亮关键词
    this.$rules = {
      start: [
        {
          token: "keyword",
          regex: "\\b(?:" + keywordList + ")\\b",
        },
        {
          token: "string",
          regex: '".*?"',
        },
        {
          token: "constant",
          regex: /\b(true|false|null)\b/,
        },
        {
          token: "comment",
          regex: /\/\/.*$/,
        },
        {
          token: "comment",
          start: "/\\*",
          end: "\\*/",
        },
        {
          token: "mylang",
          regex: "\\b(?:hello|world|AceEditor)\\b",
        },
      ],
    };
  };
  oop.inherits(MyLangHighlightRules, TextHighlightRules);
  exports.MyLangHighlightRules = MyLangHighlightRules;
}
);
```

在上述代码中，我们定义了模式下的高亮规则 MyLangHighlightRules，并将其继承自 TextHighlightRules，其中将(markdown mode)[https://github.com/ajaxorg/ace/blob/master/lib/ace/mode/markdown_highlight_rules.js]高亮规则进行了修改，以适应自定义语言。

5、拷贝

在 src 目录下新建 mylang.js，并拷贝一份 mode-mylang.js 源码

6、引入并设置为自定义语言模式

```
// 自定义语言
import "ace-builds/src-noconflict/mode-mylang";
import "./mylang";
...
 let options = {
      mode: "ace/mode/mylang",
      ...
    };
```

```
/* 自定义语言，匹配不同类型关键词高亮颜色 */
.ace_constant {
  color: #FF00FF;
  font-weight: bold;
}
```

至此完成自定义扩展

六、相关文档及网站

官网：Ace - The High Performance Code Editor for the Web

GitHub: GitHub - ajaxorg/ace: Ace (Ajax.org Cloud9 Editor)

Vue2 版：GitHub - chairuosen/vue2-ace-editor

Vue3 版：GitHub - CarterLi/vue3-ace-editor

在线 demo: https://ace.c9.io/build/kitchen-sink.html