JavaScript API 设计原则

API (/lib/tag/API) javascript (/lib/tag/javascript) 2016-03-30 16:02:31 发布

前段时间组织优化我们的原生模块 API（iOS、Android 模块封装成 JavaScript 接口），于是学习了几篇 JavaScript API 设计的文章，尽管是旧文，但受益匪浅，这里记录一下。

好的 API 设计：在自描述的同时，达到抽象的目标。

设计良好的 API ，开发者可以快速上手，没必要经常抱着手册和文档，也没必要频繁光顾技术支持社区。

流畅的接口

方法链 **(https://en.wikipedia.org/wiki/Method_chaining)**：流畅易读，更易理解

```
//常见的 API 调用方式：改变一些颜色，添加事件监听
var elem = document.getElementById("foobar");
elem.style.background = "red";
elem.style.color = "green";
elem.addEventListener('click', function(event) {
  alert("hello world!");
}, true);

//（设想的）方法链 API
DOMHelper.getElementById('foobar')
  .setStyle("background", "red")
  .setStyle("color", "green")
  .addEvent("click", function(event) {
    alert("hello world");
  });
```

设置和获取操作，可以合二为一；方法越多，文档可能越难写

```
var $elem = jQuery("#foobar");

//setter
$elem.setCss("background", "green");
//getter
$elem.getCss("color") === "red";

//getter, setter 合二为一
$elem.css("background", "green");
$elem.css("color") === "red";
```

一致性

相关的接口保持一致的风格，一整套 API 如果传递一种熟悉和舒适的感觉，会大大减轻开发者对新工具的适应性。

命名这点事：既要短，又要自描述，最重要的是保持一致性

> “There are only two hard problems in computer science: cache-invalidation and naming things.”
> “在计算机科学界只有两件头疼的事：缓存失效和命名问题”
> — Phil Karlton

选择一个你喜欢的措辞，然后持续使用。选择一种风格，然后保持这种风格。

阅读目录

处理参数

回调（callbacks）

事件（events）

需要考虑大家如何使用你提供的方法，是否会重复调用？为何会重复调用？你的 API 如何帮助开发者减少重复的调用？
接收map映射参数，回调或者序列化的属性名，不仅让你的 API 更干净，而且使用起来更舒服、高效。

jQuery 的 `css()` 方法可以给 DOM 元素设置样式：

```
jQuery("#some-selector")
  .css("background", "red")
  .css("color", "white")
  .css("font-weight", "bold")
  .css("padding", 10);
```

这个方法可以接受一个 JSON 对象：

```
jQuery("#some-selector").css({
  "background" : "red",
  "color" : "white",
  "font-weight" : "bold",
  "padding" : 10
});

//通过传一个 map 映射绑定事件
jQuery("#some-selector").on({
  "click" : myClickHandler,
  "keyup" : myKeyupHandler,
  "change" : myChangeHandler
});

//为多个事件绑定同一个处理函数
jQuery("#some-selector").on("click keyup change", myEventHandler);
```

处理类型

定义方法的时候，需要决定它可以接收什么样的参数。我们不清楚人们如何使用我们的代码，但可以更有远见，考虑支持哪些参数类型。

```
//原来的代码
DateInterval.prototype.days = function(start, end) {
  return Math.floor((end - start) / 86400000);
};

//修改后的代码
DateInterval.prototype.days = function(start, end) {
  if (!(start instanceof Date)) {
    start = new Date(start);
  }
  if (!(end instanceof Date)) {
    end = new Date(end);
  }

  return Math.floor((end.getTime() - start.getTime()) / 86400000);
};
```

加了短短的6行代码，我们的方法强大到可以接收 `Date` 对象，数字的时间戳，甚至像 `Sat Sep 08 2012 15:34:35 GMT+0200 (CEST)` 这样的字符串

如果你需要确保传入的参数类型（字符串，数字，布尔），可以这样转换：

```
function castaway(some_string, some_integer, some_boolean) {
  some_string += "";
  some_integer += 0; // parseInt(some_integer, 10) 更安全些
  some_boolean = !!some_boolean;
}
```

处理 undefined

为了使你的 API 更健壮，需要鉴别是否真正的 `undefined` 值被传递进来，可以检查 `arguments` 对象：

```
function testUndefined(expecting, someArgument) {
  if (someArgument === undefined) {
    console.log("someArgument 是 undefined");
  }
  if (arguments.length > 1) {
    console.log("然而它实际是传进来的");
  }
}

testUndefined("foo");
// 结果: someArgument 是 undefined
testUndefined("foo", undefined);
// 结果:  someArgument 是 undefined , 然而它实际是传进来的
```

给参数命名

```
event.initMouseEvent(
  "click", true, true, window,
  123, 101, 202, 101, 202,
  true, false, false, false,
  1, null);
```

Event.initMouseEvent (https://developer.mozilla.org/en-US/docs/DOM/event.initMouseEvent) 这个方法简直丧心病狂，不看文档的话，谁能说出每个参数是什么意思？

给每个参数起个名字，赋个默认值，可好

```
event.initMouseEvent(
  type="click",
  canBubble=true,
  cancelable=true,
  view=window,
  detail=123,
  screenX=101,
  screenY=202,
  clientX=101,
  clientY=202,
  ctrlKey=true,
  altKey=false,
  shiftKey=false,
  metaKey=false,
  button=1,
  relatedTarget=null);
```

ES6, 或者 Harmony 就有 默认参数值 (http://wiki.ecmascript.org/doku.php?id=harmony:parameter_default_values) 和 rest 参数 (http://wiki.ecmascript.org/doku.php?id=harmony:rest_parameters) 了。

参数接收 JSON 对象

与其接收一堆参数，不如接收一个 JSON 对象：

```
function nightmare(accepts, async, beforeSend, cache, complete, /* 等28个参数 */) {
  if (accepts === "text") {
    // 准备接收纯文本
  }
}

function dream(options) {
  options = options || {};
  if (options.accepts === "text") {
    // 准备接收纯文本
  }
}
```

调用起来也更简单了：

```
nightmare("text", true, undefined, false, undefined, /* 等28个参数 */);

dream({
  accepts: "text",
  async: true,
  cache: false
});
```

参数默认值

参数最好有默认值，通过 jQuery.extend() (http://api.jquery.com/jQuery.extend/%20%20,%20_.extend%28) http://underscorejs.org/#extend (http://underscorejs.org/#extend)) 和 Protoype 的 Object.extend (http://api.prototypejs.org/language/Object/extend/) ，可以覆盖预设的默认值。

```
var default_options = {
  accepts: "text",
  async: true,
  beforeSend: null,
  cache: false,
  complete: null,
  // …
};

function dream(options) {
  var o = jQuery.extend({}, default_options, options || {});
  console.log(o.accepts);
}

dream({ async: false });
// prints: "text"
```

扩展性

回调（callbacks）

通过回调， API 用户可以覆盖你的某一部分代码。把一些需要自定义的功能开放成可配置的回调函数，允许 API 用户轻松覆盖你的默认代码。

API 接口一旦接收回调，确保在文档中加以说明，并提供代码示例。

事件（events）

事件接口最好见名知意，可以自由选择事件名字，避免与原生事件 (https://developer.mozilla.org/en-US/docs/DOM/DOM_event_reference) 重名。

处理错误

不是所有的错误都对开发者调试代码有用：

```
// jQuery 允许这么写
$(document.body).on('click', {});

// 点击时报错
//   TypeError: ((p.event.special[l.origType] || {}).handle || l.handler).apply is not
a function
//   in jQuery.min.js on Line 3
```

这样的错误调试起来很痛苦，不要浪费开发者的时间，直接告诉他们犯了什么错：

```
if (Object.prototype.toString.call(callback) !== '[object Function]') { // 看备注
  throw new TypeError("callback is not a function!");
}
```

> 备注： `typeof callback === "function"` 在老的浏览器上会有问题， `object` 会当成个 `function` 。

可预测性

好的 API 具有可预测性，开发者可以根据例子推断它的用法。

Modernizr's 特性检测 (http://modernizr.com/docs/#howitworks) 是个例子：

a) 它使用的属性名完全与 HTML5、CSS 概念和 API 相匹配

b) 每一个单独的检测一致地返回 true 或 false 值

```
// 所有这些属性都返回 'true' 或 'false'
Modernizr.geolocation
Modernizr.localstorage
Modernizr.webworkers
Modernizr.canvas
Modernizr.borderradius
Modernizr.boxshadow
Modernizr.flexbox
```

依赖于开发者已熟悉的概念也可以达到可预测的目的。

jQuery's 选择器语法 (http://api.jquery.com/category/selectors/) 就是一个显著的例子，CSS1-CSS3 的选择器可直接用于它的 DOM 选择器引擎。

```
$("#grid") // Selects by ID
$("ul.nav > li") // All LIs for the UL with class "nav"
$("ul li:nth-child(2)") // Second item in each list
```

比例协调

好的 API 并不一定是小的 API，API 的体积大小要跟它的功能相称。

比如 Moment.js (http://momentjs.com/) ，著名的日期解析和格式化的库，可以称之为均衡，它的 API 既简洁又功能明确。

像 Moment.js 这样特定功能的库，确保 API 的专注和小巧非常重要。

编写 API 文档

软件开发最艰难的任务之一是写文档，实际上每个人都恨写文档，怨声载道的是没有一个好用的文档工具。

以下是一些文档自动生成工具：

- YUIDoc (http://yui.github.com/yuidoc/) (requires Node.js, npm)
- JsDoc Toolkit (https://github.com/p120ph37/node-jsdoc-toolkit) (requires Node.js, npm)
- Markdox (https://github.com/cbou/markdox) (requires Node.js, npm)
- Dox (https://github.com/visionmedia/dox) (requires Node.js, npm)
- Docco (http://jashkenas.github.com/docco/) (requires Node.js, Python, CoffeeScript)
- JSDuck (https://github.com/senchalabs/jsduck) (reqires Ruby, gem)

- JSDoc 3 (https://github.com/jsdoc3/jsdoc) (requires Java)
  最重要的是：确保文档跟代码同步更新。

> 参考资料：
>
> - 好的 API 设计 (http://reeze.cn/2014/02/07/what-makes-a-good-api/)
> - Designing Better JavaScript APIs (http://www.smashingmagazine.com/2012/10/09/designing-javascript-apis-usability/)
> - Secrets of Awesome JavaScript API Design (http://webstandardssherpa.com/reviews/secrets-of-awesome-javascript-api-design/)

来源 http://www.codeceo.com/article/javascript-api-design.html

同类热门经验

1. Node.js 初体验 (/lib/view/open1326870121968.html)
2. JavaScript开发规范要求 (/lib/view/open1352263831610.html)
3. 使用拖拉操作来自定义网页界面布局并保存结果 (/lib/view/open1325064347889.html)
4. Nodejs入门学习，nodejs web开发入门，npm、express、socket配置安装、nodejs聊天室开发 (/lib/view/open1329050007640.html)
5. 利用HTML5同时上传多个文件 - resumable.js (/lib/view/open1327591300671.html)
6. nide：一个不错的Node.js开发工具IDE (/lib/view/open1325834128750.html)

相关文档 — 更多 (http://www.open-open.com/doc)

- 使用DocsBuilder生成JavaScript API文档.doc (http://www.open-open.com/doc/view/e21dfe278d9c42069f8a1e8a7c76e0d3)
- 建立基于ArcGIS Server的JavaScript API 和Flex API的地图应用.pdf (http://www.open-open.com/doc/view/9f8ccdc39805409495789fa93b4981b2)
- Google Maps JavaScript API V3 参考.docx (http://www.open-open.com/doc/view/9e86c9979ead4bfdb3a52437c9f9ed1e)
- ArcGIS API for Javascript 开发教程.pdf (http://www.open-open.com/doc/view/d49d8b6604d740eda89701d1d5054d67)
- Google Maps JavaScript API V3 中文版.pdf (http://www.open-open.com/doc/view/4b6d72c0a5c0492b98b6f2cdc88c41e8)
- JavaScript 中文帮助手册(API).chm (http://www.open-open.com/doc/view/6b52b03bb3434e6bbc53ef153382fd31)
- 百度地图 JavaScript API v2.0 开发指南.pdf (http://www.open-open.com/doc/view/4a2084a38b5f4a44bf1abee51663f89a)
- 高德地图JavaScript API 个性化地图开发利器.pdf (http://www.open-open.com/doc/view/7012bc18d9244d0780d5fe0b5415d9fc)
- 天地图 JavaScript API接口文档V2.1.3 .pdf (http://www.open-open.com/doc/view/37fcda6d8e9d4a1eb1ce4a0d9b13394d)
- FusionCharts主要API接口.docx (http://www.open-open.com/doc/view/f39e2dbe144b4abb91c28d471b6ad7d9)
- Mapbar 地图 API 帮助文档.chm (http://www.open-open.com/doc/view/63f148c324d64b1ea3d84d713bd127d8)
- JavaScript高级程序设计(JavaScript for Web Developers).pdf

相关经验 — 更多 (http://www.open-open.com/lib)

- 出色的 JavaScript API 设计秘诀 (/lib/view/open1459392850031.html)
- 测试 API 的 Javascript 控制台：API Notebook (/lib/view/open1438960055598.html)
- 介绍 JavaScript 国际化 API (/lib/view/open1418779460683.html)
- ECMAScript5新JavaScript API入门 (/lib/view/open1331175932452.html)
- Javascript API文档生成器：JSDuck (/lib/view/open1379941845070.html)
- Phantom JS - 服务器端的 JavaScript API 的 WebKit (/lib/view/open1338375583636.html)
- EasyWebSocket - 封装了 WebSocket API 的 JavaScript 库 (/lib/view/open1338513079542.html)
- 一个 Javascript RESTFUL API 库：Hello.js (/lib/view/open1410482229539.html)
- store.js - 本地存储的 JavaScript 封装 API (/lib/view/open1331779537249.html)
- JavaScript 跨平台全屏API简单包装：Screenful.js (/lib/view/open1418267243292.html)

相关讨论 — 更多 (http://www.open-open.com/solution)

- 你得学 JavaScript (http://www.open-open.com/solution/view/1318473350656)
- Web开发人员最喜爱的10款流行JavaScript库 (http://www.open-open.com/solution/view/1379903308476)
- 从中间件的历史来看移动App开发的未来 (http://www.open-open.com/solution/view/1447672613728)
- 程序员技术练级攻略 (http://www.open-open.com/solution/view/1319276210452)
- REST会是SOA的未来吗？ (http://www.open-open.com/solution/view/1324368925874)
- HTML5新手入门指南 (http://www.open-open.com/solution/view/1320764201952)
- Web开发者必备的15个非常有用的HTML5工具和资源 (http://www.open-open.com/solution/view/1376547984491)

(http://www.open-open.com/doc/view/2f14e310a9a148faa5f258c03ba87acd)
- 《JavaScript框架设计》迷你书.pdf (http://www.open-open.com/doc/view/e56b249485e6486b831bc3e253c96f39)
- JavaScript 框架设计新.pdf (http://www.open-open.com/doc/view/16220a430ed24d679968be27d5f9f4ea)
- 精通JavaScript开发(Professional JavaScript for Web Developers).pdf (http://www.open-open.com/doc/view/ceaa9dc36c774dc39073652e0cc2c43e)
- 《JavaScript高级程序设计(第2版)》(Professional JavaScript for Web Developers, 2nd Edition).pdf (http://www.open-open.com/doc/view/2dbab4a3ee2a47f9a3c142c32dd655f5)
- Web Audio API 官方文档中文版.pdf (http://www.open-open.com/doc/view/a8f3d4699731419dbf73cf9492b84931)
- 百度地图API最佳实践 贾铮.pdf (http://www.open-open.com/doc/view/ddcd114ac6904cd9b5215eee5d0f1ae7)
- ArcGIS API for JavaScript 实现点、线、面的buffer分析.doc (http://www.open-open.com/doc/view/4eb31214e89e4e01b617404e02062857)
- Google Maps JavaScript API V3 叠加层.docx (http://www.open-open.com/doc/view/4f00992e825a4b62a82bffd9b25b7616)
- 基于Ruby on Rails的JavaScript简单的 CRUD API (/lib/view/open1420352559859.html)
- 兼容现有jQuery API的轻量级JavaScript库：Zepo (/lib/view/open1396358489356.html)
- PhantomJS：基于WebKit、开源的服务器端JavaScript API (/lib/view/open1422702818580.html)
- 使用 JavaScript 来获取电池状态（Battery Status API） (/lib/view/open1339647560849.html)