基础参数

type - 基本层类型

类型：Number，默认：0

layer提供了5种层类型。可传入的值有：0（信息框，默认）1（页面层）2（iframe层）3（加载层）4（tips层）。 若你采用layer.open({type: 1})方式调用，则type为必填项（信息框除外）

title - 标题

类型：String/Array/Boolean，默认：'信息'

title支持三种类型的值，若你传入的是普通的字符串，如title :'我是标题'，那么只会改变标题文本；若你还需要自定义标题区域样式，那么你可以title: ['文本', 'font-size:18px;']，数组第二项可以写任意css样式；如果你不想显示标题栏，你可以title: false

content - 内容

类型：String/DOM/Array，默认：''

content可传入的值是灵活多变的，不仅可以传入普通的html内容，还可以指定DOM，更可以随着type的不同而不同。譬如：

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. /!\*
2. 如果是页面层
3. \*/
4. layer.open({
5. type: 1,
6. content: '传入任意的文本或html' //这里content是一个普通的String
7. });
8. layer.open({
9. type: 1,
10. content: $('#id') //这里content是一个DOM
11. });
12. //Ajax获取
13. $.post('url', {}, function(str){
14. layer.open({
15. type: 1,
16. content: str //注意，如果str是object，那么需要字符拼接。
17. });
18. });
19. /!\*
20. 如果是iframe层
21. \*/
22. layer.open({
23. type: 2,
24. content: 'http://sentsin.com' //这里content是一个URL，如果你不想让iframe出现滚动条，你还可以content: ['http://sentsin.com', 'no']
25. });
26. /!\*
27. 如果是用layer.open执行tips层
28. \*/
29. layer.open({
30. type: 4,
31. content: ['内容', '#id'] //数组第二项即吸附元素选择器或者DOM
32. });

skin - 样式类名

类型：String，默认：''

skin不仅允许你传入layer内置的样式class名，还可以传入您自定义的class名。这是一个很好的切入点，意味着你可以借助skin轻松完成不同的风格定制。目前layer内置的skin有：layui-layer-lanlayui-layer-molv，未来我们还会选择性地内置更多，但更推荐您自己来定义。以下是一个自定义风格的简单例子

code[layui.code](http://www.layui.com/doc/modules/code.html)

2. //单个使用
3. layer.open({
4. skin: 'demo-class'
5. });
6. //全局使用。即所有弹出层都默认采用，但是单个配置skin的优先级更高
7. layer.config({
8. skin: 'demo-class'
9. })
10. //CSS
11. body .demo-class .layui-layer-title{background:#c00; color:#fff; border: none;}
12. body .demo-class .layui-layer-btn{border-top:1px solid #E9E7E7}
13. body .demo-class .layui-layer-btn a{background:#333;}
14. body .demo-class .layui-layer-btn .layui-layer-btn1{background:#999;}
15. …
16. 加上body是为了保证优先级。你可以借助Chrome调试工具，定义更多样式控制层更多的区域。

你也可以[去查看layer皮肤制作说明](http://layer.layui.com/skin.html#publish)

area - 宽高

类型：String/Array，默认：'auto'

在默认状态下，layer是宽高都自适应的，但当你只想定义宽度时，你可以area: '500px'，高度仍然是自适应的。当你宽高都要定义时，你可以area: ['500px', '300px']

offset - 坐标

类型：String/Array，默认：'auto'

默认垂直水平居中。但当你只想定义top时，你可以offset: '100px'。当您top、left都要定义时，你可以offset: ['100px', '200px']。除此之外，你还可以定义offset: 'rb'，表示右下角。其它的特殊坐标，你可以自己计算赋值。

icon - 图标。信息框和加载层的私有参数

类型：Number，默认：-1（信息框）/0（加载层）

信息框默认不显示图标。当你想显示图标时，默认皮肤可以传入0-6如果是加载层，可以传入0-2。如：

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //eg1
2. layer.alert('酷毙了', {icon: 1});
3. //eg2
4. layer.msg('不开心。。', {icon: 5});
5. //eg3
6. layer.load(1); //风格1的加载

btn - 按钮

类型：String/Array，默认：'确认'

信息框模式时，btn默认是一个确认按钮，其它层类型则默认不显示，加载层和tips层则无效。当您只想自定义一个按钮时，你可以btn: '我知道了'，当你要定义两个按钮时，你可以btn: ['yes', 'no']。当然，你也可以定义更多按钮，比如：btn: ['按钮1', '按钮2', '按钮3', …]，按钮1和按钮2的回调分别是yes和cancel，而从按钮3开始，则回调为btn3: function(){}，以此类推。如：

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //eg1
2. layer.confirm('纳尼？', {
3. btn: ['按钮一', '按钮二', '按钮三'] //可以无限个按钮
4. ,btn3: function(index, layero){
5. //按钮【按钮三】的回调
6. }
7. }, function(index, layero){
8. //按钮【按钮一】的回调
9. }, function(index){
10. //按钮【按钮二】的回调
11. });
12. //eg2
13. layer.open({
14. content: 'test'
15. ,btn: ['按钮一', '按钮二', '按钮三']
16. ,yes: function(index, layero){
17. //按钮【按钮一】的回调
18. },btn2: function(index, layero){
19. //按钮【按钮二】的回调
20. },btn3: function(index, layero){
21. //按钮【按钮三】的回调
22. }
23. ,cancel: function(){
24. //右上角关闭回调
25. }
26. });

closeBtn - 关闭按钮

类型：String/Boolean，默认：1

layer提供了两种风格的关闭按钮，可通过配置1和2来展示，如果不显示，则closeBtn: 0

shade - 遮罩

类型：String/Array/Boolean，默认：0.3

即弹层外区域。默认是0.3透明度的黑色背景（'#000'）。如果你想定义别的颜色，可以shade: [0.8, '#393D49']；如果你不想显示遮罩，可以shade: 0

shadeClose - 是否点击遮罩关闭

类型：Boolean，默认：false

如果你的shade是存在的，那么你可以设定shadeClose来控制点击弹层外区域关闭。

time - 自动关闭所需毫秒

类型：Number，默认：0

默认不会自动关闭。当你想自动关闭时，可以time: 5000，即代表5秒后自动关闭，注意单位是毫秒（1秒=1000毫秒）

id - 用于控制弹层唯一标识

类型：String，默认：空字符

设置该值后，不管是什么类型的层，都只允许同时弹出一个。一般用于页面层和iframe层模式

shift - 动画

类型：Number，默认：0

从1.9开始，我们的出场动画全部采用CSS3。这意味着除了ie6-9，其它所有浏览器都是支持的。目前shift可支持的动画类型有0-6

maxmin - 最大最小化。

类型：Boolean，默认：false

该参数值对type:1和type:2有效。默认不显示最大小化按钮。需要显示配置maxmin: true即可

fix - 固定

类型：Boolean，默认：true

即鼠标滚动时，层是否固定在可视区域。如果不想，设置fix: false即可

scrollbar - 是否允许浏览器出现滚动条

类型：Boolean，默认：true

默认允许浏览器滚动，如果设定scrollbar: false，则屏蔽

maxWidth - 最大宽度

类型：，默认：360

当area: 'auto'时，maxWidth的设定才有效。

zIndex - 层叠顺序

类型：，默认：19891014（贤心生日 0.0）

一般用于解决和其它组件的层叠冲突。

move - 触发拖动的元素

类型：String/DOM/Boolean，默认：'.layui-layer-title'

默认是触发标题区域拖拽。如果你想单独定义，指向元素的选择器或者DOM即可。如move: '.mine-move'。你还配置设定move: false来禁止拖拽

moveType - 拖拽风格

类型：Number，默认：0

默认的拖拽风格正如你所见到的，会有个过度的透明框。但是如果你不喜欢，你可以设定moveType: 1切换到传统的拖拽模式

moveOut - 是否允许拖拽到窗口外

类型：Boolean，默认：false

默认只能在窗口内拖拽，如果你想让拖到窗外，那么设定moveOut: true即可

moveEnd - 拖动完毕后的回调方法

类型：Function，默认：null

默认不会触发moveEnd，如果你需要，设定moveEnd: function(){}即可。

tips - tips方向和颜色

类型：Number/Array，默认：2

tips层的私有参数。支持上右下左四个方向，通过1-4进行方向设定。如tips: 3则表示在元素的下面出现。有时你还可能会定义一些颜色，可以设定tips: [1, '#c00']

tipsMore - 是否允许多个tips

类型：Boolean，默认：false

允许多个意味着不会销毁之前的tips层。通过tipsMore: true开启

success - 层弹出后的成功回调方法

类型：Function，默认：null

当你需要在层创建完毕时即执行一些语句，可以通过该回调。success会携带两个参数，分别是当前层DOM当前层索引。如：

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. layer.open({
2. content: '测试回调',
3. success: function(layero, index){
4. console.log(layero, index);
5. }
6. });

yes - 确定按钮回调方法

类型：Function，默认：null

该回调携带两个参数，分别为当前层索引、当前层DOM对象。如：

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. layer.open({
2. content: '测试回调',
3. yes: function(index, layero){
4. //do something
5. layer.close(index); //如果设定了yes回调，需进行手工关闭
6. }
7. });

cancel - 取消和关闭按钮触发的回调

类型：Function，默认：null

该回调同样只携带当前层索引一个参数，无需进行手工关闭。如果不想关闭，return false即可，如 cancel: function(index){ return false; } 则不会关闭；

end - 层销毁后触发的回调

类型：Function，默认：null

无论是确认还是取消，只要层被销毁了，end都会执行，不携带任何参数。

full/min/restore -分别代表最大化、最小化、还原 后触发的回调

类型：Function，默认：null

携带一个参数，即当前层DOM

内置方法

layer.config(options) - 初始化全局配置

这是一个可以重要也可以不重要的方法，重要的是，它的权利真的很大，尤其是在模块化加载layer时，你会发现你必须要用到它。它不仅可以配置一些诸如路径、加载的模块，甚至还可以决定整个弹层的默认参数。而说它不重要，是因为多数情况下，你会发现，你似乎不是那么十分需要它。但你真的需要认识一下这位伙计。

如果您是采用seajs或者requirejs加载layer，你需要执行该方法来完成初始化的配置。比如：

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. layer.config({
2. path: '/res/layer/' //layer.js所在的目录，可以是绝对目录，也可以是相对目录
3. });
4. //这样的话，layer就会去加载一些它所需要的配件，比如css等。
5. //当然，你即便不用seajs或者requirejs，也可以通过上述方式设定路径

如果你是采用<script src="?a.js&layer.js">这种合并的方式引入layer，那么您需要在script标签上加一个自定义属性merge="true"。如：

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. <script src="?a.js&layer.js" merge="true">
2. 这样的话，layer就不会去自动去获取路径，但你需要通过以下方式来完成初始化的配置
3. layer.config({
4. path: '/res/layer/' //layer.js所在的目录，可以是绝对目录，也可以是相对目录
5. });

但layer.config的作用远不止上述这样。它还可以配置层默认的基础参数，如：

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. layer.config({
2. shift: 1, //默认动画风格
3. skin: 'layui-layer-molv' //默认皮肤
4. …
5. });
6. //除此之外，extend还允许你加载css。这对于layer的第三方皮肤有极大的帮助，如：
7. layer.config({
8. extend: [
9. 'skin/layer-ext-myskin/style.css' //layer-ext-myskin即是你拓展的皮肤文件
10. ]
11. });
12. //扩展css的规范：您必须在你扩展中的css文件加上这段
13. html #layui\_layer\_skinlayer-ext-myskinstylecss{display:none; position: absolute; width:1989px;}
14. 规则就是：html #layui\_layer\_skin{文件夹名}{文件名}css
15. skin名以文件夹名为标准。

layer.ready(path, callback) - 初始化就绪

由于我们的layer内置了轻量级加载器，所以你根本不需要单独引入css等文件。但是加载总是需要过程的。当你在页面一打开就要执行弹层时，layer.ready()会是一个不错的帮手。它也可以做一些layer.config可以做的事，比如指向layer.js所在目录。但是如果你已经通过layer.config配置了path，你在使用layer.ready时，是不需要path的，如：

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //页面一打开就执行弹层
2. layer.ready(function(){
3. layer.msg('很高兴一开场就见到你');
4. });

我是华丽的酱油：介绍完上面两位引导者，接下来我们真正的主角闪亮登场了。此处应有掌声 ^,^

layer.open(options) - 原始核心方法

基本上是露脸率最高的方法，不管是使用哪种方式创建层，都是走layer.open()，创建任何类型的弹层都会返回一个当前层索引，上述的options即是基础参数，另外，该文档统一采用options作为基础参数的标识例子：

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. var index = layer.open({
2. content: 'test'
3. });
4. //拿到的index是一个重要的凭据，它是诸如layer.close(index)等方法的必传参数。

噢，请等等，上面这位主角的介绍篇幅怎么看怎么都觉得跟它的地位不符，作者在文档中只给了它如此可怜的一块地？？这是因为，它真的已经大众得不能再大众了，你真正需要了解的，是它的内部器官，即上面一大篇幅介绍的各种基础参数。 ←\_←

layer.alert(content, options, yes) - 普通信息框

它的弹出似乎显得有些高调，一般用于对用户造成比较强烈的关注，类似系统alert，但却比alert更灵便。它的参数是自动向左补齐的。通过第二个参数，可以设定各种你所需要的基础参数，但如果你不需要的话，直接写回调即可。如

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //eg1
2. layer.alert('只想简单的提示');
3. //eg2
4. layer.alert('加了个图标', {icon: 1}); //这时如果你也还想执行yes回调，可以放在第三个参数中。
5. //eg3
6. layer.alert('有了回调', function(index){
7. //do something
9. layer.close(index);
10. });

layer.confirm(content, options, yes, cancel) - 询问框

类似系统confirm，但却远胜confirm，另外它不是和系统的confirm一样阻塞你需要把交互的语句放在回调体中。同样的，它的参数也是自动补齐的。

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //eg1
2. layer.confirm('is not?', {icon: 3, title:'提示'}, function(index){
3. //do something
5. layer.close(index);
6. });
7. //eg2
8. layer.confirm('is not?', function(index){
9. //do something
11. layer.close(index);
12. });

layer.msg(content, options, end) - 提示框

我在源码中有了相对较大的篇幅来定制了这个msg，目的是想将其打造成露脸率最高的提示框。而事实上我的确也在大量地使用它。因为它简单，而且足够得自觉，它不仅占据很少的面积，而且默认还会3秒后自动消失所有这一切都决定了我对msg的爱。因此我赋予了她许多可能在外形方面，它坚持简陋的变化，在作用方面，他坚持零用户操作。而且它的参数也是机会自动补齐的。

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //eg1
2. layer.msg('只想弱弱提示');
3. //eg2
4. layer.msg('有表情地提示', {icon: 6});
5. //eg3
6. layer.msg('关闭后想做些什么', function(){
7. //do something
8. });
9. //eg
10. layer.msg('同上', {
11. icon: 1,
12. time: 2000 //2秒关闭（如果不配置，默认是3秒）
13. }, function(){
14. //do something
15. });

layer.load(icon, options) - 加载层

type:3的深度定制。load并不需要你传太多的参数，但如果你不喜欢默认的加载风格，你还有选择空间。icon支持传入0-2如果是0，无需传。另外特别注意一点：load默认是不会自动关闭的，因为你一般会在ajax回调体中关闭它。

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //eg1
2. var index = layer.load();
3. //eg2
4. var index = layer.load(1); //换了种风格
5. //eg3
6. var index = layer.load(2, {time: 10\*1000}); //又换了种风格，并且设定最长等待10秒
7. //关闭
8. layer.close(index);

layer.tips(content, follow, options) - tips层

type:4的深度定制。也是我本人比较喜欢的一个层类型，因为它拥有和msg一样的低调和自觉，而且会智能定位，即灵活地判断它应该出现在哪边。默认是在元素右边弹出

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //eg1
2. layer.tips('只想提示地精准些', '#id');
3. //eg 2
4. $('#id').on('click', function(){
5. var that = this;
6. layer.tips('只想提示地精准些', that); //在元素的事件回调体中，follow直接赋予this即可
7. });
8. //eg 3
9. layer.tips('在上面', '#id', {
10. tips: 1
11. });

上面主要是一些弹层的调用方式，而下面介绍的是一些辅助性的方法

layer.close(index) - 关闭特定层

关于它似乎没有太多介绍的必要，唯一让你疑惑的，可能就是这个index了吧

事实上它非常容易得到。

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //当你想关闭当前页的某个层时
2. var index = layer.open();
3. var index = layer.alert();
4. var index = layer.load();
5. var index = layer.tips();
6. //正如你看到的，每一种弹层调用方式，都会返回一个index
7. layer.close(index); //此时你只需要把获得的index，轻轻地赋予layer.close即可
8. //当你在iframe页面关闭自身时
9. var index = parent.layer.getFrameIndex(window.name); //先得到当前iframe层的索引
10. parent.layer.close(index); //再执行关闭

layer.closeAll(type) - 关闭所有层

如果你很懒，你不想去获取index你只想关闭。那么closeAll真的可以帮上你。如果你不指向层类型的话，它会销毁掉当前页所有的layer层。当然，如果你只想关闭某个类型的层，那么你可以

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. layer.closeAll(); //疯狂模式，关闭所有层
2. layer.closeAll('dialog'); //关闭信息框
3. layer.closeAll('page'); //关闭所有页面层
4. layer.closeAll('iframe'); //关闭所有的iframe层
5. layer.closeAll('loading'); //关闭加载层
6. layer.closeAll('tips'); //关闭所有的tips层

layer.style(index, cssStyle) - 重新定义层的样式

cssStyle允许你传入任意的css属性

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //重新给指定层设定width、top等
2. layer.style(index, {
3. width: '1000px',
4. top: '10px'
5. });

layer.title(title, index) - 改变层的标题

使用方式：layer.title('标题变了', index)

layer.getChildFrame(selector, index) - 获取iframe页的DOM

当你试图在当前页获取iframe页的DOM元素时，你可以用此方法。selector即iframe页的选择器

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. layer.open({
2. type: 2,
3. content: 'test/iframe.html',
4. success: function(layero, index){
5. var body = layer.getChildFrame('body', index);
6. var iframeWin = window[layero.find('iframe')[0]['name']]; //得到iframe页的窗口对象，执行iframe页的方法：iframeWin.method();
7. console.log(body.html()) //得到iframe页的body内容
8. body.find('input').val('Hi，我是从父页来的')
9. }
10. });

layer.getFrameIndex(windowName) - 获取特定iframe层的索引

此方法一般用于在iframe页关闭自身时用到。

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //假设这是iframe页
2. var index = parent.layer.getFrameIndex(window.name); //先得到当前iframe层的索引
3. parent.layer.close(index); //再执行关闭

layer.iframeAuto(index) - 指定iframe层自适应

调用该方法时，iframe层的高度会重新进行适应

layer.iframeSrc(index, url) - //重置特定iframe url

似乎不怎么常用的样子。使用方式：layer.iframeSrc(index, 'http://sentsin.com')

layer.setTop(layero) -置顶当前窗口

非常强大的一个方法，虽然一般很少用。但是当你的页面有很多很多layer窗口，你需要像Window窗体那样，点击某个窗口，该窗体就置顶在上面，那么setTop可以来轻松实现。它采用巧妙的逻辑，以使这种置顶的性能达到最优

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //通过这种方式弹出的层，每当它被选择，就会置顶。
2. layer.open({
3. type: 2,
4. shade: false,
5. area: '500px',
6. maxmin: true,
7. content: 'http://www.layui.com',
8. zIndex: layer.zIndex, //重点1
9. success: function(layero){
10. layer.setTop(layero); //重点2
11. }
12. });

layer.full()、layer.min()、layer.restore() - 手工执行最大小化

（这三个酱油又一次被并列 ==。）一般用于在自定义元素上触发最大化、最小化和全屏。

其他内置层

请注意，从2.3开始，无需通过layer.config来加载拓展模块。如果您是之前版本，则需通过下述方式来加载

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. layer.config({
2. extend: 'extend/layer.ext.js'
3. });

layer.prompt(options, yes) - 输入层

prompt的参数也是向前补齐的。options不仅可支持传入基础参数，还可以传入prompt专用的属性。当然，也可以不传。yes携带value 表单值index 索引elem 表单元素

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //prompt层新定制的成员如下
2. {
3. formType: 1, //输入框类型，支持0（文本）默认1（密码）2（多行文本）
4. value: '', //初始时的值，默认空字符
5. maxlength: 140, //可输入文本的最大长度，默认500
6. }
7. //例子1
8. layer.prompt(function(value, index, elem){
9. alert(value); //得到value
10. layer.close(index);
11. });
12. //例子2
13. layer.prompt({
14. formType: 2,
15. value: '初始值',
16. title: '请输入值'
17. }, function(value, index, elem){
18. alert(value); //得到value
19. layer.close(index);
20. });

layer.tab(options) - tab层

tab层只单独定制了一个成员，即tab: []，这个好像没有什么可介绍的，简单粗暴看例子

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. layer.tab({
2. area: ['600px', '300px'],
3. tab: [{
4. title: 'TAB1',
5. content: '内容1'
6. }, {
7. title: 'TAB2',
8. content: '内容2'
9. }, {
10. title: 'TAB3',
11. content: '内容3'
12. }]
13. });

layer.photos(options) - 相册层

相册层，也可以称之为图片查看器。它的出场动画从layer内置的动画类型中随机展现。photos支持传入json和直接读取页面图片两种方式。如果是json传入，如下：

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. $.getJSON('/jquery/layer/test/photos.json', function(json){
2. layer.photos({
3. photos: json
4. ,shift: 5 //0-6的选择，指定弹出图片动画类型，默认随机
5. });
6. });
7. //而返回的json需严格按照如下格式：
8. code[layui.code](http://www.layui.com/doc/modules/code.html)
   1. {
   2. "title": "", //相册标题
   3. "id": 123, //相册id
   4. "start": 0, //初始显示的图片序号，默认0
   5. "data": [ //相册包含的图片，数组格式
   6. {
   7. "alt": "图片名",
   8. "pid": 666, //图片id
   9. "src": "", //原图地址
   10. "thumb": "" //缩略图地址
   11. }
   12. ]
   13. }

如果是直接从页面中获取图片，那么需要指向图片的父容器，并且你的img可以设定一些规定的属性（但不是必须的）。

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. //HTML示例
2. <div id="layer-photos-demo" class="layer-photos-demo">
3. <img layer-pid="图片id，可以不写" layer-src="大图地址" src="缩略图" alt="图片名">
4. <img layer-pid="图片id，可以不写" layer-src="大图地址" src="缩略图" alt="图片名">
5. </div>
6. //调用示例
7. layer.ready(function(){ //为了layer.ext.js加载完毕再执行
8. layer.photos({
9. photos: '#layer-photos-demo'
10. ,shift: 5 //0-6的选择，指定弹出图片动画类型，默认随机
11. });
12. });

看看一个实例呗：



第二种方式的图片查看器显然更加简单，因为无需像第一种那样返回规定的json，但是他们还是有各自的应用场景的，你可以按照你的需求进行选择。另外，photos还有个tab回调，切换图片时触发。

code[layui.code](http://www.layui.com/doc/modules/code.html)

1. layer.photos({
2. photos: json/选择器,
3. tab: function(pic, layero){
4. console.log(pic) //当前图片的一些信息
5. }
6. });

合理地设定基础参数，合理地选择内置方法，合理地运用拓展方法，合理的心态，合理地阅读，只要一切都在合理的前提下，你才会感知到layer许许多多令人愉悦的地方，***她真的是否如你所愿，取决于你对她了解的深远***。愿layer能给你的web开发带来一段美妙的旅程。相信这一版的API文档，也会给你带来全新的便捷。别忘了在线调试。



1. 特别说明：事件需自己绑定，以下只展现调用代码。
2. //初体验
3. layer.alert('内容')
4. //第三方扩展皮肤
5. layer.alert('内容', {
6. icon: 1,
7. skin: 'layer-ext-moon' //该皮肤由layer.seaning.com友情扩展。关于皮肤的扩展规则，[去这里查阅](http://layer.layui.com/skin.html" \l "publish" \t "_balnk)
8. })
9. //询问框
10. layer.confirm('您是如何看待前端开发？', {
11. btn: ['重要','奇葩'] //按钮
12. }, function(){
13. layer.msg('的确很重要', {icon: 1});
14. }, function(){
15. layer.msg('也可以这样', {
16. time: 20000, //20s后自动关闭
17. btn: ['明白了', '知道了']
18. });
19. });
20. //提示层
21. layer.msg('玩命提示中');
22. //墨绿深蓝风
23. layer.alert('墨绿风格，点击确认看深蓝', {
24. skin: 'layui-layer-molv' //样式类名
25. ,closeBtn: 0
26. }, function(){
27. layer.alert('偶吧深蓝style', {
28. skin: 'layui-layer-lan'
29. ,closeBtn: 0
30. ,anim: 4 //动画类型
31. });
32. });
33. //捕获页
34. layer.open({
35. type: 1,
36. shade: false,
37. title: false, //不显示标题
38. content: $('.layer\_notice'), //捕获的元素，注意：最好该指定的元素要存放在body最外层，否则可能被其它的相对元素所影响
39. cancel: function(){
40. layer.msg('捕获就是从页面已经存在的元素上，包裹layer的结构', {time: 5000, icon:6});
41. }
42. });
43. //页面层
44. layer.open({
45. type: 1,
46. skin: 'layui-layer-rim', //加上边框
47. area: ['420px', '240px'], //宽高
48. content: 'html内容'
49. });
50. //自定页
51. layer.open({
52. type: 1,
53. skin: 'layui-layer-demo', //样式类名
54. closeBtn: 0, //不显示关闭按钮
55. anim: 2,
56. shadeClose: true, //开启遮罩关闭
57. content: '内容'
58. });
59. //tips层
60. layer.tips('Hi，我是tips', '吸附元素选择器，如#id');
61. //iframe层
62. layer.open({
63. type: 2,
64. title: 'layer mobile页',
65. shadeClose: true,
66. shade: 0.8,
67. area: ['380px', '90%'],
68. content: 'mobile/' //iframe的url
69. });
70. //iframe窗
71. layer.open({
72. type: 2,
73. title: false,
74. closeBtn: 0, //不显示关闭按钮
75. shade: [0],
76. area: ['340px', '215px'],
77. offset: 'rb', //右下角弹出
78. time: 2000, //2秒后自动关闭
79. anim: 2,
80. content: ['test/guodu.html', 'no'], //iframe的url，no代表不显示滚动条
81. end: function(){ //此处用于演示
82. layer.open({
83. type: 2,
84. title: '很多时候，我们想最大化看，比如像这个页面。',
85. shadeClose: true,
86. shade: false,
87. maxmin: true, //开启最大化最小化按钮
88. area: ['893px', '600px'],
89. content: '//fly.layui.com/'
90. });
91. }
92. });
93. //加载层
94. var index = layer.load(0, {shade: false}); //0代表加载的风格，支持0-2
95. //loading层
96. var index = layer.load(1, {
97. shade: [0.1,'#fff'] //0.1透明度的白色背景
98. });
99. //小tips
100. layer.tips('我是另外一个tips，只不过我长得跟之前那位稍有些不一样。', '吸附元素选择器', {
101. tips: [1, '#3595CC'],
102. time: 4000
103. });
104. //prompt层
105. layer.prompt({title: '输入任何口令，并确认', formType: 1}, function(pass, index){
106. layer.close(index);
107. layer.prompt({title: '随便写点啥，并确认', formType: 2}, function(text, index){
108. layer.close(index);
109. layer.msg('演示完毕！您的口令：'+ pass +'<br>您最后写下了：'+text);
110. });
111. });
112. //tab层
113. layer.tab({
114. area: ['600px', '300px'],
115. tab: [{
116. title: 'TAB1',
117. content: '内容1'
118. }, {
119. title: 'TAB2',
120. content: '内容2'
121. }, {
122. title: 'TAB3',
123. content: '内容3'
124. }]
125. });
126. //相册层
127. $.getJSON('test/photos.json?v='+new Date, function(json){
128. layer.photos({
129. photos: json //格式见API文档手册页
130. ,anim: 5 //0-6的选择，指定弹出图片动画类型，默认随机
131. });
132. });



1. //信息框-例1
2. layer.alert('见到你真的很高兴', {icon: 6});
3. //信息框-例2
4. layer.msg('你确定你很帅么？', {
5. time: 0 //不自动关闭
6. ,btn: ['必须啊', '丑到爆']
7. ,yes: function(index){
8. layer.close(index);
9. layer.msg('雅蠛蝶 O.o', {
10. icon: 6
11. ,btn: ['嗷','嗷','嗷']
12. });
13. }
14. });
15. //信息框-例3
16. layer.msg('这是最常用的吧');
17. //信息框-例4
18. layer.msg('不开心。。', {icon: 5});
19. //信息框-例5
20. layer.msg('玩命卖萌中', function(){
21. //关闭后的操作
22. });
23. //页面层-自定义
24. layer.open({
25. type: 1,
26. title: false,
27. closeBtn: 0,
28. shadeClose: true,
29. skin: 'yourclass',
30. content: '自定义HTML内容'
31. });
32. //页面层-佟丽娅
33. layer.open({
34. type: 1,
35. title: false,
36. closeBtn: 0,
37. area: '516px',
38. skin: 'layui-layer-nobg', //没有背景色
39. shadeClose: true,
40. content: $('#tong')
41. });
42. //iframe层-父子操作
43. layer.open({
44. type: 2,
45. area: ['700px', '450px'],
46. fixed: false, //不固定
47. maxmin: true,
48. content: 'test/iframe.html'
49. });
50. //iframe层-多媒体
51. layer.open({
52. type: 2,
53. title: false,
54. area: ['630px', '360px'],
55. shade: 0.8,
56. closeBtn: 0,
57. shadeClose: true,
58. content: '//player.youku.com/embed/XMjY3MzgzODg0'
59. });
60. layer.msg('点击任意处关闭');
61. //iframe层-禁滚动条
62. layer.open({
63. type: 2,
64. area: ['360px', '500px'],
65. skin: 'layui-layer-rim', //加上边框
66. content: ['mobile/', 'no']
67. });
68. //加载层-默认风格
69. layer.load();
70. //此处演示关闭
71. setTimeout(function(){
72. layer.closeAll('loading');
73. }, 2000);
74. //加载层-风格2
75. layer.load(1);
76. //此处演示关闭
77. setTimeout(function(){
78. layer.closeAll('loading');
79. }, 2000);
80. //加载层-风格3
81. layer.load(2);
82. //此处演示关闭
83. setTimeout(function(){
84. layer.closeAll('loading');
85. }, 2000);
86. //加载层-风格4
87. layer.msg('加载中', {
88. icon: 16
89. ,shade: 0.01
90. });
91. //打酱油
92. layer.msg('尼玛，打个酱油', {icon: 4});
93. //tips层-上
94. layer.tips('上', '#id或者.class', {
95. tips: [1, '#0FA6D8'] //还可配置颜色
96. });
97. //tips层-右
98. layer.tips('默认就是向右的', '#id或者.class');
99. //tips层-下
100. layer.tips('下', '#id或者.class', {
101. tips: 3
102. });
103. //tips层-左
104. layer.tips('左边么么哒', '#id或者.class', {
105. tips: [4, '#78BA32']
106. });
107. //tips层-不销毁之前的
108. layer.tips('不销毁之前的', '#id或者.class', {
109. tipsMore: true
110. });
111. //默认prompt
112. layer.prompt(function(val, index){
113. layer.msg('得到了'+val);
114. layer.close(index);
115. });
116. //屏蔽浏览器滚动条
117. layer.open({
118. content: '浏览器滚动条已锁',
119. scrollbar: false
120. });
121. //弹出即全屏
122. var index = layer.open({
123. type: 2,
124. content: 'http://layim.layui.com',
125. area: ['320px', '195px'],
126. maxmin: true
127. });
128. layer.full(index);
129. //正上方
130. layer.msg('灵活运用offset', {
131. offset: 't',
132. anim: 6
133. });
134. //更多例子
135. layer.msg('Hi');