Touch API 文档

标签 移动端手势 Touch.js手势库 移动设备手势识别与事件库

> 在开发移动端的应用中会使用到很多的手势操作，例如一指拖动、两指旋转等等，为了方便开放者快速集成这些手势，在Clouda中内置了事件和手势库`Library.touch`，下面将详细的介绍如何使用Library.touch。

---

touch.js

Touch.js是移动设备上的手势识别与事件库, 由百度云Clouda团队维护，也是在百度内部广泛使用的开发工具.

Touch.js的代码已托管于github并开源，希望能帮助国内更多的开发者学习和开发出优秀的App产品.

Touch.js手势库专为移动设备设计, 请在Webkit内核浏览器中使用.

极速CDN

```
<script src="http://code.baidu.com/touch-0.2.14.min.js"></s
cript>
```

touch.config

```
语法:  touch.config
```

对手势事件库进行全局配置。

参数描述：

- config为一个对象

```
touch.config={
    tap: true,                  //tap类事件开关，默认为true
    doubleTap: true,          //doubleTap事件开关，默认为true
    hold: true,                 //hold事件开关，默认为true
    holdTime: 650,              //hold时间长度
    swipe: true,                //swipe事件开关
    swipeTime: 300,             //触发swipe事件的最大时长
    swipeMinDistance: 18,       //swipe移动最小距离
    swipeFactor: 5,             //加速因子，值越大变化速率越快
    drag: true,                 //drag事件开关
    pinch: true,                //pinch类事件开关
}
```

touch.on

```
语法: touch.on(element, types, options, callback)
```

绑定指定元素的事件。

参数描述：

- element: 元素对象或选择器。
- types: 事件的类型, 可接受多个事件以空格分开，支持原生事件的透传, 支持的一些事件类型有:

pinchstart	**双指缩放动作开始**
pinchend	**双指缩放动作结束**
pinch	**双指缩放事件**
pinchin	**双指向里缩小**
pinchout	**双指向外放大**
rotateleft	**向左旋转**
rotateright	**向右旋转**
rotate	**旋转事件**
swipestart	**单指滑动动作开始**
swiping	**单指滑动事件**
swipeend	**单指滑动动作结束**
swipeleft	**单指向左滑动**
swiperight	**单指向右滑动事件**
swipeup	**单指向上滑动**
swipedown	**单指向下滑动**
swipe	**单指滑动事件**
drag	**单指向左右拖动**
hold	**单指按住不放事件**
tap	**单指点击**

doubletap	单指双击

例如旋转实例如下：

```
var angle = 30;
touch.on('#rotation .target', 'touchstart', function(ev){
    ev.startRotate();
    ev.originEvent.preventDefault();
    ev.originEvent.stopPropagation();
});
touch.on('#rotation .target', 'rotate', {interval: 10}, fun
ction(ev){
var totalAngle = angle + ev.rotation;
if(ev.fingerStatus === 'end'){
    angle = angle + ev.rotation;
}

this.style.webkitTransform = 'rotate(' + totalAngle + 'de
g)';
});
```

更多使用实例请查看http://code.baidu.com/

- options(可选): 目前可配置的参数为:

```
{
    //采样频率
    interval: 10,//性能参数，值越小，实时性越好， 但性能可能略
差， 值越大， 性能越好。遇到性能问题时，可以将值设大调优，建议值设
置为10。
    //swipe加速度因子（swipe事件专用）
    swipeFactor: 5 //(int: 1-10)值越大，速率更快。
}
```

- callback: 事件处理函数， 该函数接受的参数为一个gesture event object, 可访问的属性有：
    - originEvent //触发某事件的原生对象
    - type //事件的名称
    - rotation //旋转角度

- scale //缩放比例
- direction //操作的方向属性
- fingersCount //操作的手势数量
- position //相关位置信息, 不同的操作产生不同的位置信息。
- distance //swipe类两点之间的位移
- distanceX //swipe类事件x方向的位移
- distanceY //swipe类事件y方向的位移
- angle //swipe类事件触发时偏移角度
- factor //swipe事件加速度因子
- startRotate //启动单指旋转方法，在某个元素的touchstart触发时调用。

touch.live

```
语法：touch.live(selector, types, options, callback)
```

使用方法基本上与on相同，live的第一个参数只接受`css3选择器`。通过`live()`方法附加的事件处理程序适用于匹配选择器的当前及未来的元素（比如由脚本创建的新元素）

touch.off

```
语法：touch.off(element,types,callback)
```

解除某元素上的事件绑定。

参数描述：

- element：元素对象或选择器
- types：事件的类型
- callback：时间处理函数

示例

1. Rotate旋转

```
//rotation
var angle = 0;
touch.on('#target', 'touchstart', function(ev){
    ev.startRotate();
    ev.preventDefault();
});

touch.on('#target', 'rotate', function(ev){
    var totalAngle = angle + ev.rotation;
    if(ev.fingerStatus === 'end'){
        angle = angle + ev.rotation;
    }
    this.style.webkitTransform = 'rotate(' + totalAngle +
'deg)';
});
```

2. Scale缩放

```
var target = document.getElementById("target");
target.style.webkitTransition = 'all ease 0.05s';

touch.on('#target', 'touchstart', function(ev){
    ev.preventDefault();
});

var initialScale = 1;
var currentScale;

touch.on('#target', 'pinchend', function(ev){
    currentScale = ev.scale - 1;
    currentScale = initialScale + currentScale;
    currentScale = currentScale > 2 ? 2 : currentScale;
    currentScale = currentScale < 1 ? 1 : currentScale;
    this.style.webkitTransform = 'scale(' + currentScale +
')';
    log("当前缩放比例为:" + currentScale + ".");
});

touch.on('#target', 'pinchend', function(ev){
    initialScale = currentScale;
});
```

3. tap, doubletap和hold

```
touch.on('#target', 'hold tap doubletap', function(ev){
    //console.log(ev.type);
});
```

4. 向左, 向右swipe滑动

```
touch.on('#target', 'touchstart', function(ev){
    ev.preventDefault();
});

var target = document.getElementById("target");
target.style.webkitTransition = 'all ease 0.2s';

touch.on(target, 'swiperight', function(ev){
    this.style.webkitTransform = "translate3d(" + rt + "p
x,0,0)";
    log("向右滑动.");
});

touch.on(target, 'swipeleft', function(ev){
    log("向左滑动.");
    this.style.webkitTransform = "translate3d(-" + this.off
setLeft + "px,0,0)";
});
```

5. drag 抓取并移动目标

```
touch.on('#target', 'touchstart', function(ev){
    ev.preventDefault();
});

var target = document.getElementById("target");
var dx, dy;

touch.on('#target', 'drag', function(ev){
    dx = dx || 0;
    dy = dy || 0;
    log("当前x值为:" + dx + ", 当前y值为:" + dy +".");
    var offx = dx + ev.x + "px";
    var offy = dy + ev.y + "px";
    this.style.webkitTransform = "translate3d(" + offx +
"," + offy + ",0)";
});

touch.on('#target', 'dragend', function(ev){
    dx += ev.x;
    dy += ev.y;
});
```

识别原生事件

```
touch.on('#target', 'touchstart touchmove touchend', functi
on(ev){
    console.log(ev.type);
});
```