

## GIS, GPS坐标转换

WGS-84: 是国际标准, GPS坐标 (Google Earth使用、或者GPS模块)

GCJ-02: 中国坐标偏移标准, Google Map、高德、腾讯使用

BD-09: 百度坐标偏移标准, Baidu Map使用, 这个是百度地图自己在GCJ-02自己再加密了一层

```
//WGS-84 to GCJ-02  
GPS.gcj_encrypt();
```

```
//GCJ-02 to WGS-84 粗略  
GPS.gcj_decrypt();
```

```
//GCJ-02 to WGS-84 精确(二分极限法)
```

```
// var threshold = 0.000000001; 目前设置的是精确到小数点后9位, 这个值越小, 越精确, 但是javascript中, 浮点运算本身就不太精确, 九位在GPS里也偏差不大了
```

```
GSP.gcj_decrypt_exact();
```

```
//GCJ-02 to BD-09  
GPS.bd_encrypt();
```

```
//BD-09 to GCJ-02  
GPS.bd_decrypt();
```

```
//求距离
```

```
GPS.distance();
```

示例:

```
document.write("GPS: 39.933676862706776,116.35608315379092<br />");  
var arr2 = GPS.gcj_encrypt(39.933676862706776, 116.35608315379092);  
document.write("中国:" + arr2['lat']+"," +arr2['lon']+''<br />');  
var arr3 = GPS.gcj_decrypt_exact(arr2['lat'], arr2['lon']);  
document.write('逆算:' + arr3['lat']+"," +arr3['lon']+'' 需要和第一行相似 (目前是小数点后9位相等) ');
```

标签: [jQuery](#) [GPS](#) [PHP](#)

代码片段 (2)[\[全屏查看所有代码\]](#)

1. [代码][\[JavaScript\]代码](#)

```
?
```



1  
2  
3  
4  
5  
6  
7  
8  
9  
1  
0  
1  
1  
1  
2  
1  
3  
1  
4  
1  
5  
1  
6  
1  
7  
1  
8  
1  
9  
2  
0  
2  
1  
2  
2  
2  
3  
2  
4  
2  
5  
2  
6  
2  
7  
2  
8  
2  
9  
3  
0  
3  
1  
3  
2  
3  
3  
3  
3  
4  
3  
5  
3  
6  
3  
7  
3  
8  
3  
9  
4  
0  
4  
1  
4  
2  
4  
3  
4  
4  
4  
5  
4  
6  
4  
7  
4  
8  
4  
9  
5

```

0
5
1
5
2 var GPS = {
5   PI : 3.14159265358979324,
3   x_pi : 3.14159265358979324 * 3000.0 / 180.0,
5   delta : function(lat, lon) {
4     // Krasovsky 1940
5     //
6     // a = 6378245.0, 1/f = 298.3
7     // b = a * (1 - f)
8     // ee = (a^2 - b^2) / a^2;
9     var a = 6378245.0; // a: 卫星椭球坐标投影到平面地图坐标系的投影因子。
10    var ee = 0.00669342162296594323; // ee: 椭球的偏心率。
11    var dLat = this.transformLat(lon - 105.0, lat - 35.0);
12    var dLon = this.transformLon(lon - 105.0, lat - 35.0);
13    var radLat = lat / 180.0 * this.PI;
14    var magic = Math.sin(radLat);
15    magic = 1 - ee * magic * magic;
16    var sqrtMagic = Math.sqrt(magic);
17    dLat = (dLat * 180.0) / ((a * (1 - ee)) / (magic * sqrtMagic) * this.PI);
18    dLon = (dLon * 180.0) / (a / sqrtMagic * Math.cos(radLat) * this.PI);
19    return {'lat': dLat, 'lon': dLon};
20  },
21
22  //WGS-84 to GCJ-02
23  gcj_encrypt : function(wgsLat, wgsLon) {
24    if(this.outOfChina(wgsLat, wgsLon))
25      return {'lat': wgsLat, 'lon': wgsLon};
26
27    var d = this.delta(wgsLat, wgsLon);
28    return {'lat': wgsLat + d.lat,'lon': wgsLon + d.lon};
29  },
30  //GCJ-02 to WGS-84
31  gcj_decrypt : function(gcjLat, gcjLon) {
32    if(this.outOfChina(gcjLat, gcjLon))
33      return {'lat': gcjLat, 'lon': gcjLon};
34
35    var d = this.delta(gcjLat, gcjLon);
36    return {'lat': gcjLat - d.lat, 'lon': gcjLon - d.lon};
37  },
38  //GCJ-02 to WGS-84 exactly
39  gcj_decrypt_exact : function(gcjLat, gcjLon) {
40    var initDelta = 0.01;
41    var threshold = 0.00000001;
42    var dLat = initDelta, dLon = initDelta;
43    var mLat = gcjLat - dLat, mLon = gcjLon - dLon;
44    var pLat = gcjLat + dLat, pLon = gcjLon + dLon;
45    var wgsLat, wgsLon, i = 0;
46    while(1) {
47      wgsLat = (mLat + pLat) / 2;
48      wgsLon = (mLon + pLon) / 2;
49      var tmp = this.gcj_encrypt(wgsLat, wgsLon)
50      dLat = tmp.lat - gcjLat;
51      dLon = tmp.lon - gcjLon;
52      if((Math.abs(dLat) < threshold) && (Math.abs(dLon) < threshold))
53        break;
54
55      if(dLat > 0) pLat = wgsLat; else mLat = wgsLat;
56      if(dLon > 0) pLon = wgsLon; else mLon = wgsLon;
57
58      if(++i > 10000) break;
59    }
60    //console.log(i);
61    return {'lat': wgsLat, 'lon': wgsLon};
62  },
63  //GCJ-02 to BD-09
64  bd_encrypt : function(gcjLat, gcjLon) {
65    var x = gcjLon, y = gcjLat;
66    var z = Math.sqrt(x * x + y * y) + 0.00002 * Math.sin(y * this.x_pi);
67    var theta = Math.atan2(y, x) + 0.000003 * Math.cos(x * this.x_pi);
68    bdLon = z * Math.cos(theta) + 0.0065;
69    bdLat = z * Math.sin(theta) + 0.006;
70    return{'lat': bdLat,'lon': bdLon};
71  },
72  //BD-09 to GCJ-02
73  bd_decrypt : function(bdLat, bdLon) {
74    var x = bdLon - 0.0065, y = bdLat - 0.006;
75    var z = Math.sqrt(x * x + y * y) - 0.00002 * Math.sin(y * this.x_pi);
76    var theta = Math.atan2(y, x) - 0.000003 * Math.cos(x * this.x_pi);
77    var gcjLon = z * Math.cos(theta);
78    var gcjLat = z * Math.sin(theta);
79    return {'lat': gcjLat, 'lon': gcjLon};
80  },
81  //WGS-84 to Web mercator
82  //mercatorLat -> y mercatorLon -> x
83  mercator_encrypt : function(wgsLat, wgsLon) {
84    var x = wgsLon * 20037508.34 / 180.;
85    var y = Math.log(Math.tan((90. + wgsLat) * this.PI / 360.)) / (this.PI / 180.);

```

```
5   y = y * 20037508.34 / 180.;  
6   return {'lat': y, 'lon': x};  
7   /*  
8    if ((Math.abs(wgsLon) > 180 || Math.abs(wgsLat) > 90))  
9     return null;  
10   var x = 6378137.0 * wgsLon * 0.017453292519943295;  
11   var a = wgsLat * 0.017453292519943295;  
12   var y = 3189068.5 * Math.log((1.0 + Math.sin(a)) / (1.0 - Math.sin(a)));  
13   return {'lat': y, 'lon': x};  
14   /**/  
15 },  
16 // Web mercator to WGS-84  
17 // mercatorLat -> y mercatorLon -> x  
18 mercator_decrypt : function(mercatorLat, mercatorLon) {  
19   var x = mercatorLon / 20037508.34 * 180.;  
20   var y = mercatorLat / 20037508.34 * 180.;  
21   y = 180 / this.PI * (2 * Math.atan(Math.exp(y * this.PI / 180.)) - this.PI / 2);  
22   return {'lat': y, 'lon': x};  
23   /*  
24    if (Math.abs(mercatorLon) < 180 && Math.abs(mercatorLat) < 90)  
25      return null;  
26    if ((Math.abs(mercatorLon) > 20037508.3427892) || (Math.abs(mercatorLat) > 20037508.3427892))  
27      return null;  
28    var a = mercatorLon / 6378137.0 * 57.295779513082323;  
29    var x = a - (Math.floor((a + 180.0) / 360.0) * 360.0);  
30    var y = (1.5707963267948966 - (2.0 * Math.atan(Math.exp((-1.0 * mercatorLat) / 6378137.0)))) * 57.2957795  
31    13082323;  
32    return {'lat': y, 'lon': x};  
33   /**/  
34 },  
35 // two point's distance  
36 distance : function(latA, lonA, latB, lonB) {  
37   var earthR = 6371000.;  
38   var x = Math.cos(latA * this.PI / 180.) * Math.cos(latB * this.PI / 180.) * Math.cos((lonA - lonB) * this.PI / 180.);  
39   var y = Math.sin(latA * this.PI / 180.) * Math.sin(latB * this.PI / 180.);  
40   var s = x + y;  
41   if(s > 1) s = 1;  
42   if(s < -1) s = -1;  
43   var alpha = Math.acos(s);  
44   var distance = alpha * earthR;  
45   return distance;  
46 },  
47 outOfChina : function(lat, lon) {  
48   if(lon < 72.004 || lon > 137.8347)  
49     return true;  
50   if(lat < 0.8293 || lat > 55.8271)  
51     return true;  
52   return false;  
53 },  
54 transformLat : function(x, y) {  
55   var ret = -100.0 + 2.0 * x + 3.0 * y + 0.2 * y * y + 0.1 * x * y + 0.2 * Math.sqrt(Math.abs(x));  
56   ret += (20.0 * Math.sin(6.0 * x * this.PI) + 20.0 * Math.sin(2.0 * x * this.PI)) * 2.0 / 3.0;  
57   ret += (20.0 * Math.sin(y * this.PI) + 40.0 * Math.sin(y / 3.0 * this.PI)) * 2.0 / 3.0;  
58   ret += (160.0 * Math.sin(y / 12.0 * this.PI) + 320 * Math.sin(y * this.PI / 30.0)) * 2.0 / 3.0;  
59   return ret;  
60 },  
61 transformLon : function(x, y) {  
62   var ret = 300.0 + x + 2.0 * y + 0.1 * x * x + 0.1 * x * y + 0.1 * Math.sqrt(Math.abs(x));  
63   ret += (20.0 * Math.sin(6.0 * x * this.PI) + 20.0 * Math.sin(2.0 * x * this.PI)) * 2.0 / 3.0;  
64   ret += (20.0 * Math.sin(x * this.PI) + 40.0 * Math.sin(x / 3.0 * this.PI)) * 2.0 / 3.0;  
65   ret += (150.0 * Math.sin(x / 12.0 * this.PI) + 300.0 * Math.sin(x / 30.0 * this.PI)) * 2.0 / 3.0;  
66   return ret;  
67 }  
68 };
```

```
1  
2  
7  
1  
2  
8  
1  
2  
9  
1  
3  
0  
1  
3  
1  
1  
3  
2  
1  
3  
3  
1  
3  
4  
1  
3  
5  
1  
3  
6  
1  
3  
7  
1  
3  
8  
1  
3  
9  
1  
4  
0  
1  
4  
1  
1  
4  
2  
1  
4  
3  
1  
4  
4  
1  
4  
5  
1  
4  
6  
1  
4  
7  
1  
4  
8  
1  
4  
9
```

## 2. [代码]**[PHP]**代码

?



1  
2  
3  
4  
5  
6  
7  
8  
9  
1  
0  
1  
1  
1  
2  
1  
3  
1  
4  
1  
5  
1  
6  
1  
7  
1  
8  
1  
9  
2  
0  
2  
1  
2  
2  
2  
3  
2  
4  
2  
5  
2  
6  
2  
7  
2  
8  
2  
9  
3  
0  
3  
1  
3  
2  
3  
3  
3  
3  
4  
3  
5  
3  
6  
3  
7  
3  
8  
3  
9  
4  
0  
4  
1  
4  
2  
4  
3  
4  
4  
4  
5  
4  
6  
4  
7  
4  
8  
4  
9  
5

```

0
5
1
5
2
5
3
5
4
5
5
5
7
5
8
5
9
6
0
6 <?php
1 class GPS {
2     private $PI= 3.14159265358979324;
3     private $x_pi= 0;
4
5     public function __construct()
6     {
7         $this->x_pi = 3.14159265358979324 * 3000.0 / 180.0;
8     }
9
10 //WGS-84 to GCJ-02
11 public function gcj_encrypt($wgsLat, $wgsLon) {
12     if($this->outOfChina($wgsLat, $wgsLon))
13         return array('lat'=> $wgsLat, 'lon'=> $wgsLon);
14
15     $d= $this->delta($wgsLat, $wgsLon);
16     return array('lat'=> $wgsLat+ $d['lat'], 'lon'=> $wgsLon+ $d['lon']);
17 }
18
19 //GCJ-02 to WGS-84
20 public function gcj_decrypt($gcjLat, $gcjLon) {
21     if($this->outOfChina($gcjLat, $gcjLon))
22         return array('lat'=> $gcjLat, 'lon'=> $gcjLon);
23
24     $d= $this->delta($gcjLat, $gcjLon);
25     return array('lat'=> $gcjLat- $d['lat'], 'lon'=> $gcjLon- $d['lon']);
26 }
27
28 //GCJ-02 to WGS-84 exactly
29 public function gcj_decrypt_exact($gcjLat, $gcjLon) {
30     $initDelta= 0.01;
31     $threshold= 0.00000001;
32     $dLat= $initDelta; $dLon= $initDelta;
33     $mLat= $gcjLat- $dLat; $mLon= $gcjLon- $dLon;
34     $pLat= $gcjLat+ $dLat; $pLon= $gcjLon+ $dLon;
35     $wgsLat= 0; $wgsLon= 0; $i= 0;
36     while(TRUE) {
37         $wgsLat= ($mLat+ $pLat) / 2;
38         $wgsLon= ($mLon+ $pLon) / 2;
39         $tmp= $this->gcj_encrypt($wgsLat, $wgsLon);
40         $dLat= $tmp['lat'] - $gcjLat;
41         $dLon= $tmp['lon'] - $gcjLon;
42         if((abs($dLat) < $threshold) && (abs($dLon) < $threshold))
43             break;
44
45         if($dLat> 0) $pLat= $wgsLat; else $mLat= $wgsLat;
46         if($dLon> 0) $pLon= $wgsLon; else $mLon= $wgsLon;
47
48         if(++$i> 10000) break;
49     }
50     //console.log(i);
51     return array('lat'=> $wgsLat, 'lon'=> $wgsLon);
52 }
53
54 //GCJ-02 to BD-09
55 public function bd_encrypt($gcjLat, $gcjLon) {
56     $x= $gcjLon; $y= $gcjLat;
57     $z= sqrt($x* $x+ $y* $y) + 0.00002 * sin($y* $this->x_pi);
58     $theta= atan2($y, $x) + 0.00003 * cos($x* $this->x_pi);
59     $bdLon= $z* cos($theta) + 0.0065;
60     $bdLat= $z* sin($theta) + 0.006;
61     return array('lat'=> $bdLat, 'lon'=> $bdLon);
62 }
63
64 //BD-09 to GCJ-02
65 public function bd_decrypt($bdLat, $bdLon)
66 {
67     $x= $bdLon- 0.0065; $y= $bdLat- 0.006;
68     $z= sqrt($x* $x+ $y* $y) - 0.00002 * sin($y* $this->x_pi);
69     $theta= atan2($y, $x) - 0.00003 * cos($x* $this->x_pi);
70     $gcjLon= $z* cos($theta);
71     $gcjLat= $z* sin($theta);
72     return array('lat'=> $gcjLat, 'lon'=> $gcjLon);
73 }

```

```

5 //WGS-84 to Web mercator
6 // $mercatorLat -> y $mercatorLon -> x
7 publicfunctionmercator_encrypt($wgsLat, $wgsLon)
8 {
9     $x= $wgsLon* 20037508.34 / 180.;
10    $y= log(tan((90. + $wgsLat) * $this->PI / 360.)) / ($this->PI / 180.);
11    $y= $y* 20037508.34 / 180.;
12    returnarray('lat'=> $y, 'lon'=> $x);
13    /*
14    if ((abs($wgsLon) > 180 || abs($wgsLat) > 90))
15        return NULL;
16    $x = 6378137.0 * $wgsLon * 0.017453292519943295;
17    $a = $wgsLat * 0.017453292519943295;
18    $y = 3189068.5 * log((1.0 + sin($a)) / (1.0 - sin($a)));
19    return array('lat' => $y, 'lon' => $x);
20    /**
21 }
22 // Web mercator to WGS-84
23 // $mercatorLat -> y $mercatorLon -> x
24 publicfunctionmercator_decrypt($mercatorLat, $mercatorLon)
25 {
26     $x= $mercatorLon/ 20037508.34 * 180.;
27     $y= $mercatorLat/ 20037508.34 * 180.;
28     $y= 180 / $this->PI * (2 * atan(exp($y* $this->PI / 180.)) - $this->PI / 2);
29     returnarray('lat'=> $y, 'lon'=> $x);
30     /*
31     if (abs($mercatorLon) < 180 && abs($mercatorLat) < 90)
32         return NULL;
33     if ((abs($mercatorLon) > 20037508.3427892) || (abs($mercatorLat) > 20037508.3427892))
34         return NULL;
35     $a = $mercatorLon / 6378137.0 * 57.295779513082323;
36     $x = $a - (floor(($a + 180.0) / 360.0)) * 360.0;
37     $y = (1.5707963267948966 - (2.0 * atan(exp((-1.0 * $mercatorLat) / 6378137.0)))) * 57.295779513082323;
38     return array('lat' => $y, 'lon' => $x);
39     /**
40 }
41 // two point's distance
42 publicfunctiondistance($latA, $lonA, $latB, $lonB)
43 {
44     $earthR= 6371000.0;
45     $x= cos($latA* $this->PI / 180.) * cos($latB* $this->PI / 180.) * cos(($lonA- $lonB) * $this->PI / 180.0);
46     $y= sin($latA* $this->PI / 180.) * sin($latB* $this->PI / 180.0);
47     $s= $x+ $y;
48     if($s> 1) $s= 1;
49     if($s< -1) $s= -1;
50     $alpha= acos($s);
51     $distance= $alpha* $earthR;
52     return$distance;
53 }
54
55 privatefunctiondelta($lat, $lon)
56 {
57     // Krasovsky 1940
58     //
59     // a = 6378245.0, 1/f = 298.3
60     // b = a * (1 - f)
61     // ee = (a^2 - b^2) / a^2;
62     // ee= 6378245.0; // a: 卫星椭球坐标投影到平面地图坐标系的投影因子。
63     $ee= 0.00669342162296594323; // ee: 椭球的偏心率。
64     $dLat= $this->transformLat($lon- 105.0, $lat- 35.0);
65     $dLon= $this->transformLon($lon- 105.0, $lat- 35.0);
66     $radLat= $lat/ 180.0 * $this->PI;
67     $magic= sin($radLat);
68     $magic= 1 - $ee* $magic* $magic;
69     $sqrtMagic= sqrt($magic);
70     $dLat= ($dLat* 180.0) / (($a* (1 - $ee)) / ($magic* $sqrtMagic) * $this->PI);
71     $dLon= ($dLon* 180.0) / ($a/ $sqrtMagic* cos($radLat) * $this->PI);
72     returnarray('lat'=> $dLat, 'lon'=> $dLon);
73 }
74
75 privatefunctionoutOfChina($lat, $lon)
76 {
77     if($lon< 72.004 || $lon> 137.8347)
78         returnTRUE;
79     if($lat< 0.8293 || $lat> 55.8271)
80         returnTRUE;
81     returnFALSE;
82 }
83
84 privatefunctiontransformLat($x, $y) {
85     $ret= -100.0 + 2.0 * $x+ 3.0 * $y+ 0.2 * $y* $y+ 0.1 * $x* $y+ 0.2 * sqrt(abs($x));
86     $ret+= (20.0 * sin(6.0 * $x* $this->PI) + 20.0 * sin(2.0 * $x* $this->PI)) * 2.0 / 3.0;
87     $ret+= (20.0 * sin($y* $this->PI) + 40.0 * sin($y/ 3.0 * $this->PI)) * 2.0 / 3.0;
88     $ret+= (160.0 * sin($y/ 12.0 * $this->PI) + 320 * sin($y* $this->PI / 30.0)) * 2.0 / 3.0;
89     return$ret;
90 }
91
92 privatefunctiontransformLon($x, $y) {
93     $ret= 300.0 + $x+ 2.0 * $y+ 0.1 * $x* $y+ 0.1 * $x* $y+ 0.1 * sqrt(abs($x));
94     $ret+= (20.0 * sin(6.0 * $x* $this->PI) + 20.0 * sin(2.0 * $x* $this->PI)) * 2.0 / 3.0;

```

```
1      $ret+= (20.0 * sin($x* $this->PI) + 40.0 * sin($x/ 3.0 * $this->PI)) * 2.0 / 3.0;
2      $ret+= (150.0 * sin($x/ 12.0 * $this->PI) + 300.0 * sin($x/ 30.0 * $this->PI)) * 2.0 / 3.0;
7      return$ret;
1
2
8
1
2
9
1
3
0
1
3
1
1
3
2
1
3
3
1
3
4
1
3
5
1
3
6
1
3
7
1
3
8
1
3
9
1
4
0
1
4
1
1
4
2
1
4
3
1
4
4
1
4
5
1
4
6
1
4
7
1
4
8
1
4
9
1
5
0
1
5
1
1
5
2
1
5
3
1
5
4
1
5
5
1
5
6
```

1	
5	
7	
1	
5	
8	
1	
5	
9	
1	
6	
0	
1	
6	
1	
1	
6	
2	
1	
6	
3	
1	
6	
4	