# 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

```
?
```

```
var GPS = {
    PI : 3.14159265358979324,
    x_pi : 3.14159265358979324 * 3000.0 / 180.0,
    delta : function(lat, lon) {
        // Krasovsky 1940
        //
        // a = 6378245.0, 1/f = 298.3
        // b = a * (1 - f)
        // ee = (a^2 - b^2) / a^2;
        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。
        var ee = 0.00669342162296594323; //  ee: 椭球的偏心率。
        var dLat = this.transformLat(lon - 105.0, lat - 35.0);
        var dLon = this.transformLon(lon - 105.0, lat - 35.0);
        var radLat = lat / 180.0 * this.PI;
        var magic = Math.sin(radLat);
        magic = 1 - ee * magic * magic;
        var sqrtMagic = Math.sqrt(magic);
        dLat = (dLat * 180.0) / ((a * (1 - ee)) / (magic * sqrtMagic) * this.PI);
        dLon = (dLon * 180.0) / (a / sqrtMagic * Math.cos(radLat) * this.PI);
        return {'lat': dLat, 'lon': dLon};
    },

    //WGS-84 to GCJ-02
    gcj_encrypt : function(wgsLat, wgsLon) {
        if(this.outOfChina(wgsLat, wgsLon))
            return {'lat': wgsLat, 'lon': wgsLon};

        var d = this.delta(wgsLat, wgsLon);
        return {'lat': wgsLat + d.lat,'lon': wgsLon + d.lon};
    },
    //GCJ-02 to WGS-84
    gcj_decrypt : function(gcjLat, gcjLon) {
        if(this.outOfChina(gcjLat, gcjLon))
            return {'lat': gcjLat, 'lon': gcjLon};

        var d = this.delta(gcjLat, gcjLon);
        return {'lat': gcjLat - d.lat, 'lon': gcjLon - d.lon};
    },
    //GCJ-02 to WGS-84 exactly
    gcj_decrypt_exact : function(gcjLat, gcjLon) {
        var initDelta = 0.01;
        var threshold = 0.000000001;
        var dLat = initDelta, dLon = initDelta;
        var mLat = gcjLat - dLat, mLon = gcjLon - dLon;
        var pLat = gcjLat + dLat, pLon = gcjLon + dLon;
        var wgsLat, wgsLon, i = 0;
        while(1) {
            wgsLat = (mLat + pLat) / 2;
            wgsLon = (mLon + pLon) / 2;
            var tmp = this.gcj_encrypt(wgsLat, wgsLon)
            dLat = tmp.lat - gcjLat;
            dLon = tmp.lon - gcjLon;
            if((Math.abs(dLat) < threshold) && (Math.abs(dLon) < threshold))
                break;

            if(dLat > 0) pLat = wgsLat; else mLat = wgsLat;
            if(dLon > 0) pLon = wgsLon; else mLon = wgsLon;

            if(++i > 10000) break;
        }
        //console.log(i);
        return {'lat': wgsLat, 'lon': wgsLon};
    },
    //GCJ-02 to BD-09
    bd_encrypt : function(gcjLat, gcjLon) {
        var x = gcjLon, y = gcjLat;
        var z = Math.sqrt(x * x + y * y) + 0.00002 * Math.sin(y * this.x_pi);
        var theta = Math.atan2(y, x) + 0.000003 * Math.cos(x * this.x_pi);
        bdLon = z * Math.cos(theta) + 0.0065;
        bdLat = z * Math.sin(theta) + 0.006;
        return{'lat': bdLat,'lon': bdLon};
    },
    //BD-09 to GCJ-02
    bd_decrypt : function(bdLat, bdLon) {
        var x = bdLon - 0.0065, y = bdLat - 0.006;
        var z = Math.sqrt(x * x + y * y) - 0.00002 * Math.sin(y * this.x_pi);
        var theta = Math.atan2(y, x) - 0.000003 * Math.cos(x * this.x_pi);
        var gcjLon = z * Math.cos(theta);
        var gcjLat = z * Math.sin(theta);
        return {'lat': gcjLat, 'lon': gcjLon};
    },
    //WGS-84 to Web mercator
    //mercatorLat -> y mercatorLon -> x
    mercator_encrypt : function(wgsLat, wgsLon) {
        var x = wgsLon * 20037508.34 / 180.;
        var y = Math.log(Math.tan((90. + wgsLat) * this.PI / 360.)) / (this.PI / 180.);
```

```
        y = y * 20037508.34 / 180.;
        return {'lat': y, 'lon': x};
        /*
        if ((Math.abs(wgsLon) > 180 || Math.abs(wgsLat) > 90))
            return null;
        var x = 6378137.0 * wgsLon * 0.017453292519943295;
        var a = wgsLat * 0.017453292519943295;
        var y = 3189068.5 * Math.log((1.0 + Math.sin(a)) / (1.0 - Math.sin(a)));
        return {'lat' : y, 'lon' : x};
        //*/
    },
    // Web mercator to WGS-84
    // mercatorLat -> y mercatorLon -> x
    mercator_decrypt : function(mercatorLat, mercatorLon) {
        var x = mercatorLon / 20037508.34 * 180.;
        var y = mercatorLat / 20037508.34 * 180.;
        y = 180 / this.PI * (2 * Math.atan(Math.exp(y * this.PI / 180.)) - this.PI / 2);
        return {'lat': y, 'lon': x};
        /*
        if (Math.abs(mercatorLon) < 180 && Math.abs(mercatorLat) < 90)
            return null;
        if ((Math.abs(mercatorLon) > 20037508.3427892) || (Math.abs(mercatorLat) > 20037508.3427892))
            return null;
        var a = mercatorLon / 6378137.0 * 57.295779513082323;
        var x = a - (Math.floor(((a + 180.0) / 360.0)) * 360.0);
        var y = (1.5707963267948966 - (2.0 * Math.atan(Math.exp((-1.0 * mercatorLat) / 6378137.0)))) * 57.2957795
13082323;
        return {'lat' : y, 'lon' : x};
        //*/
    },
    // two point's distance
    distance : function(latA, lonA, latB, lonB) {
        var earthR = 6371000.;
        var x = Math.cos(latA * this.PI / 180.) * Math.cos(latB * this.PI / 180.) * Math.cos((lonA - lonB) * this
.PI / 180);
        var y = Math.sin(latA * this.PI / 180.) * Math.sin(latB * this.PI / 180.);
        var s = x + y;
        if(s > 1) s = 1;
        if(s < -1) s = -1;
        var alpha = Math.acos(s);
        var distance = alpha * earthR;
        return distance;
    },
    outOfChina : function(lat, lon) {
        if(lon < 72.004 || lon > 137.8347)
            return true;
        if(lat < 0.8293 || lat > 55.8271)
            return true;
        return false;
    },
    transformLat : function(x, y) {
        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));
        ret += (20.0 * Math.sin(6.0 * x * this.PI) + 20.0 * Math.sin(2.0 * x * this.PI)) * 2.0 / 3.0;
        ret += (20.0 * Math.sin(y * this.PI) + 40.0 * Math.sin(y / 3.0 * this.PI)) * 2.0 / 3.0;
        ret += (160.0 * Math.sin(y / 12.0 * this.PI) + 320 * Math.sin(y * this.PI / 30.0)) * 2.0 / 3.0;
        return ret;
    },
    transformLon : function(x, y) {
        var ret = 300.0 + x + 2.0 * y + 0.1 * x * x + 0.1 * x * y + 0.1 * Math.sqrt(Math.abs(x));
        ret += (20.0 * Math.sin(6.0 * x * this.PI) + 20.0 * Math.sin(2.0 * x * this.PI)) * 2.0 / 3.0;
        ret += (20.0 * Math.sin(x * this.PI) + 40.0 * Math.sin(x / 3.0 * this.PI)) * 2.0 / 3.0;
        ret += (150.0 * Math.sin(x / 12.0 * this.PI) + 300.0 * Math.sin(x / 30.0 * this.PI)) * 2.0 / 3.0;
        return ret;
    }
};
```

```
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

```
?
```

```php
<?php
class GPS {
    private $PI = 3.14159265358979324;
    private $x_pi = 0;

    public function __construct()
    {
        $this->x_pi = 3.14159265358979324 * 3000.0 / 180.0;
    }
    //WGS-84 to GCJ-02
    public function gcj_encrypt($wgsLat, $wgsLon) {
        if($this->outOfChina($wgsLat, $wgsLon))
            return array('lat' => $wgsLat, 'lon' => $wgsLon);

        $d = $this->delta($wgsLat, $wgsLon);
        return array('lat' => $wgsLat + $d['lat'],'lon' => $wgsLon + $d['lon']);
    }
    //GCJ-02 to WGS-84
    public function gcj_decrypt($gcjLat, $gcjLon) {
        if($this->outOfChina($gcjLat, $gcjLon))
            return array('lat' => $gcjLat, 'lon' => $gcjLon);

        $d = $this->delta($gcjLat, $gcjLon);
        return array('lat' => $gcjLat - $d['lat'], 'lon' => $gcjLon - $d['lon']);
    }
    //GCJ-02 to WGS-84 exactly
    public function gcj_decrypt_exact($gcjLat, $gcjLon) {
        $initDelta = 0.01;
        $threshold = 0.000000001;
        $dLat = $initDelta; $dLon = $initDelta;
        $mLat = $gcjLat - $dLat; $mLon = $gcjLon - $dLon;
        $pLat = $gcjLat + $dLat; $pLon = $gcjLon + $dLon;
        $wgsLat = 0; $wgsLon = 0; $i = 0;
        while(TRUE) {
            $wgsLat = ($mLat + $pLat) / 2;
            $wgsLon = ($mLon + $pLon) / 2;
            $tmp = $this->gcj_encrypt($wgsLat, $wgsLon);
            $dLat = $tmp['lat'] - $gcjLat;
            $dLon = $tmp['lon'] - $gcjLon;
            if((abs($dLat) < $threshold) && (abs($dLon) < $threshold))
                break;

            if($dLat > 0) $pLat = $wgsLat; else $mLat = $wgsLat;
            if($dLon > 0) $pLon = $wgsLon; else $mLon = $wgsLon;

            if(++$i > 10000) break;
        }
        //console.log(i);
        return array('lat' => $wgsLat, 'lon' => $wgsLon);
    }
    //GCJ-02 to BD-09
    public function bd_encrypt($gcjLat, $gcjLon) {
        $x = $gcjLon; $y = $gcjLat;
        $z = sqrt($x * $x + $y * $y) + 0.00002 * sin($y * $this->x_pi);
        $theta = atan2($y, $x) + 0.000003 * cos($x * $this->x_pi);
        $bdLon = $z * cos($theta) + 0.0065;
        $bdLat = $z * sin($theta) + 0.006;
        return array('lat' => $bdLat,'lon' => $bdLon);
    }
    //BD-09 to GCJ-02
    public function bd_decrypt($bdLat, $bdLon)
    {
        $x = $bdLon - 0.0065; $y = $bdLat - 0.006;
        $z = sqrt($x * $x + $y * $y) - 0.00002 * sin($y * $this->x_pi);
        $theta = atan2($y, $x) - 0.000003 * cos($x * $this->x_pi);
        $$gcjLon = $z * cos($theta);
        $gcjLat = $z * sin($theta);
        return array('lat' => $gcjLat, 'lon' => $gcjLon);
    }
```

```
    //WGS-84 to Web mercator
    //$mercatorLat -> y $mercatorLon -> x
    publicfunctionmercator_encrypt($wgsLat, $wgsLon)
    {
        $x= $wgsLon* 20037508.34 / 180.;
        $y= log(tan((90. + $wgsLat) * $this->PI / 360.)) / ($this->PI / 180.);
        $y= $y* 20037508.34 / 180.;
        returnarray('lat'=> $y, 'lon'=> $x);
        /*
        if ((abs($wgsLon) > 180 || abs($wgsLat) > 90))
            return NULL;
        $x = 6378137.0 * $wgsLon * 0.017453292519943295;
        $a = $wgsLat * 0.017453292519943295;
        $y = 3189068.5 * log((1.0 + sin($a)) / (1.0 - sin($a)));
        return array('lat' => $y, 'lon' => $x);
        //*/
    }
    // Web mercator to WGS-84
    // $mercatorLat -> y $mercatorLon -> x
    publicfunctionmercator_decrypt($mercatorLat, $mercatorLon)
    {
        $x= $mercatorLon/ 20037508.34 * 180.;
        $y= $mercatorLat/ 20037508.34 * 180.;
        $y= 180 / $this->PI * (2 * atan(exp($y* $this->PI / 180.)) - $this->PI / 2);
        returnarray('lat'=> $y, 'lon'=> $x);
        /*
        if (abs($mercatorLon) < 180 && abs($mercatorLat) < 90)
            return NULL;
        if ((abs($mercatorLon) > 20037508.3427892) || (abs($mercatorLat) > 20037508.3427892))
            return NULL;
        $a = $mercatorLon / 6378137.0 * 57.295779513082323;
        $x = $a - (floor((($a + 180.0) / 360.0)) * 360.0);
        $y = (1.5707963267948966 - (2.0 * atan(exp((-1.0 * $mercatorLat) / 6378137.0)))) * 57.295779513082323;
        return array('lat' => $y, 'lon' => $x);
        //*/
    }
    // two point's distance
    publicfunctiondistance($latA, $lonA, $latB, $lonB)
    {
        $earthR= 6371000.;
        $x= cos($latA* $this->PI / 180.) * cos($latB* $this->PI / 180.) * cos(($lonA- $lonB) * $this->PI / 180);
        $y= sin($latA* $this->PI / 180.) * sin($latB* $this->PI / 180.);
        $s= $x+ $y;
        if($s> 1) $s= 1;
        if($s< -1) $s= -1;
        $alpha= acos($s);
        $distance= $alpha* $earthR;
        return$distance;
    }

    privatefunctiondelta($lat, $lon)
    {
        // Krasovsky 1940
        //
        // a = 6378245.0, 1/f = 298.3
        // b = a * (1 - f)
        // ee = (a^2 - b^2) / a^2;
        $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。
        $ee= 0.00669342162296594323;//  ee: 椭球的偏心率。
        $dLat= $this->transformLat($lon- 105.0, $lat- 35.0);
        $dLon= $this->transformLon($lon- 105.0, $lat- 35.0);
        $radLat= $lat/ 180.0 * $this->PI;
        $magic= sin($radLat);
        $magic= 1 - $ee* $magic* $magic;
        $sqrtMagic= sqrt($magic);
        $dLat= ($dLat* 180.0) / (($a* (1 - $ee)) / ($magic* $sqrtMagic) * $this->PI);
        $dLon= ($dLon* 180.0) / ($a/ $sqrtMagic* cos($radLat) * $this->PI);
        returnarray('lat'=> $dLat, 'lon'=> $dLon);
    }

    privatefunctionoutOfChina($lat, $lon)
    {
        if($lon< 72.004 || $lon> 137.8347)
            returnTRUE;
        if($lat< 0.8293 || $lat> 55.8271)
            returnTRUE;
        returnFALSE;
    }

    privatefunctiontransformLat($x, $y) {
        $ret= -100.0 + 2.0 * $x+ 3.0 * $y+ 0.2 * $y* $y+ 0.1 * $x* $y+ 0.2 * sqrt(abs($x));
        $ret+= (20.0 * sin(6.0 * $x* $this->PI) + 20.0 * sin(2.0 * $x* $this->PI)) * 2.0 / 3.0;
        $ret+= (20.0 * sin($y* $this->PI) + 40.0 * sin($y/ 3.0 * $this->PI)) * 2.0 / 3.0;
        $ret+= (160.0 * sin($y/ 12.0 * $this->PI) + 320 * sin($y* $this->PI / 30.0)) * 2.0 / 3.0;
        return$ret;
    }

    privatefunctiontransformLon($x, $y) {
        $ret= 300.0 + $x+ 2.0 * $y+ 0.1 * $x* $x+ 0.1 * $x* $y+ 0.1 * sqrt(abs($x));
        $ret+= (20.0 * sin(6.0 * $x* $this->PI) + 20.0 * sin(2.0 * $x* $this->PI)) * 2.0 / 3.0;
```

```
        $ret+= (20.0 * sin($x* $this->PI) + 40.0 * sin($x/ 3.0 * $this->PI)) * 2.0 / 3.0;
        $ret+= (150.0 * sin($x/ 12.0 * $this->PI) + 300.0 * sin($x/ 30.0 * $this->PI)) * 2.0 / 3.0;
        return$ret;
    }
}
```