**绘制了图标，点击图标展示对应的信息**

**参考网址：**[**openlayers上添加点击事件**](https://www.cnblogs.com/ldlx-mars/p/11906992.html)

 // 监听singleclick事件,通过点击事件，获取对应点的feature图标

that.map.on("singleclick", function(e) {

            //在点击时获取像素区域

            that.mapClick(e);

          });

**判断任何一个图层中的相关要素**

/*/ 地图点击事件*

 *mapCli*ck(e) {

 *var*coordinates = *e*.coordinate;

 *var*pixel = e.pixe*l*;

 *if (*this.map.hasFe*a*tureAtPixel(pixel)) {

 *va*r feature = th*i*s.map.getFeaturesAtPixel(pixel);

 *va*r properties =feature.values\_;

 *va*r statistics =JSON.parse(properties.statistics);

 *if* (properties.t*y*peId == 100) {

 var inhtml = "*"*;

 for (let i = 0*;* i < statistics.length; i++) {

   const elemen*t* = statistics[i];

   inhtml =

     inhtml + e*l*ement.category\_name + ":" + element.number + ";";

 }

 *}*else {

 var inhtml = "*"*;

 *}*

 *//*点击图标\*\*\*\*获得图标的信*息*  提示框\*\*\*\*\*

 *va*r container = *d*ocument.getElementById("popup"); //最外层包含所有元素的div

 *va*r popupCloser *=* document.getElementById("popup-closer"); //弹出框的关闭按钮，是一个a标签

 *co*ntainer.style.*d*isplay = "block"; //提示显示

 *//* 创建一个Overlay 放*到*地图上

 *va*r overlay = ne*w* OlOverlay({

 //设置弹出框的容器

 element: conta*i*ner,

 //是否自动平移，即假如标记*在*屏幕边缘，弹出时自动平移地图使弹出框完全可见

 autoPan: true,

 autoPanAnimati*o*n: {

   duration: 25*0* //当Popup超出地图边界时，为了Popup全部可见，地图移动的速度.

 }

 *})*;

 *un*itByUnitId({ u*n*itId: properties.affiliationCompany }).then(res => {

 var that = thi*s*;

 console.log("u*n*it", res.data);

 var unit = res*.*data;

 that.popupInne*r* = properties;

 that.popupInne*r*.statistics = inhtml;

 that.popupInne*r*.armsName = unit.armsName;

 that.popupInne*r*.theaterId = unit.theaterId;

 that.popupInne*r*.theaterName = unit.theaterName;

 that.popupInne*r*.unitName = unit.unitName;

 that.$nextTick*(*() => {

   //设置overlay的*显*示位置 查询回来只设置一次。

   overlay.setP*o*sition(coordinates);

   //显示overlay

   that.map.add*O*verlay(overlay);

   // 详细信息提示框关闭*的*按钮

   popupCloser.*a*ddEventListener("click", function() {

     overlay.se*t*Position(undefined);

   });

 });

 *})*;

 *}*

 *},*

**判断指定图层中的相关要素hasFeatureAtPixel()方法**

   *// 地图单击事件*

   *mapClick*(e) {

   *this.p*opupShow = fal*s*e;

   *var co*ordinates = e.*c*oordinate;

   *var pi*xel = e.pixel;

   *// 判断某*一个图层是否包含featur*e*

   */\*\**

   *\*判断像素*上是否存在地图要素

   *\*hasF*eatureAtPixel(*p*ixel, opt\_options)

   *\* @pa*ram {\*Pixel数据}pixel

   *\* @pa*ram {\*Object 过*滤*选项} opt\_options

   *\* 过滤项*组成

   *\* @pa*ram {\*undefine*d* | function 图层过滤函数  return boolean } layerFilter

   *\* @pa*ram {\*number (*d*efaults to 0)    命中检测公差(css像素),在给定位置周围的半径内的像素将被检查 } hitToleranc

   *\* @pa*ram {\*boolean *(*defaults to true)    覆盖检测,检测在+/- 1世界范围内的几何图形，只在投影坐标系下生效 } checkWrapped \*

   *\* @re*turns 检测到的要素集 *A*rray

   *\*/*

   *var fe*atures = this.*m*ap.getFeaturesAtPixel(pixel, {

   *laye*rFilter: layer=> {

   *if* (layer.values*\_*.name == "equipment-source") {

   return true;

   *}*else {

   return false;

   *}*

   *}*

   *});*

   *},*