前言

1、这么门课难吗?

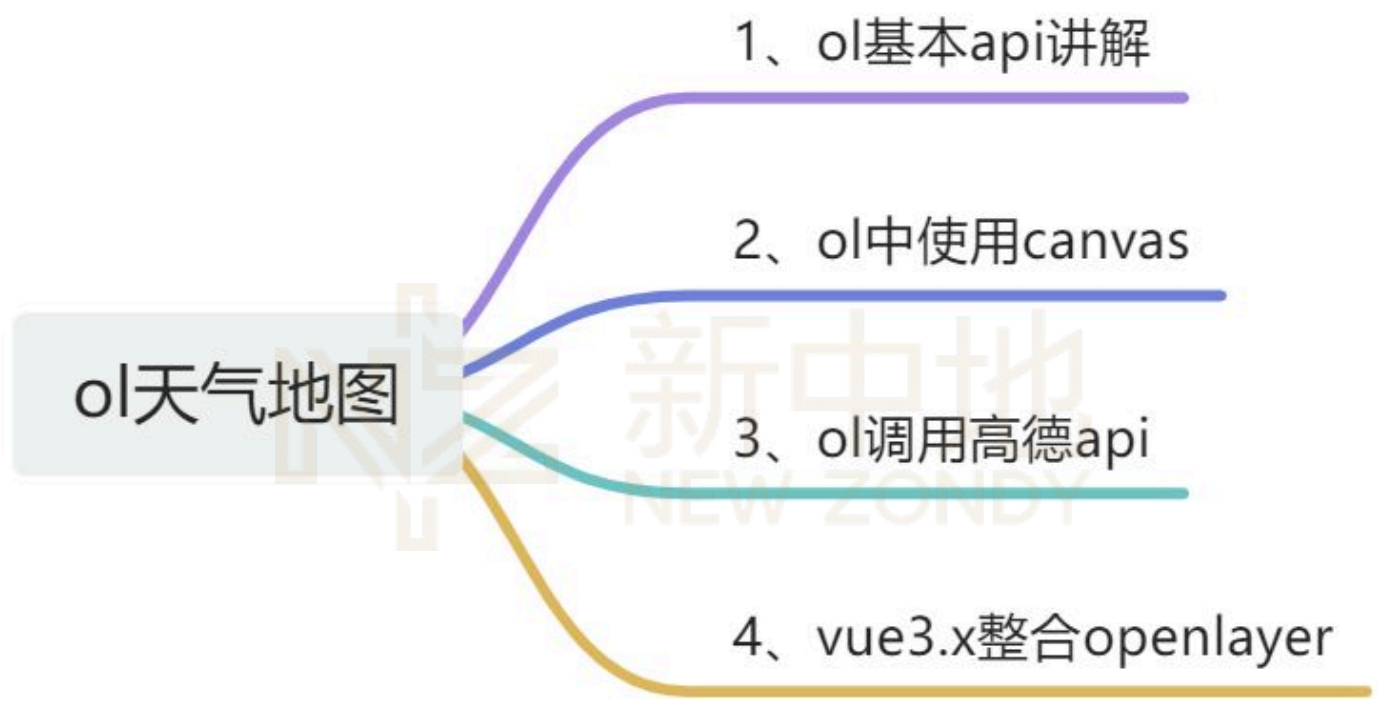
> 前置知识

1、需要大家掌握基础的 `HTML-CSS-JavaScript`

2、会vscode的简单操作

2、这门课讲什么

> 大纲



3、能够学到什么

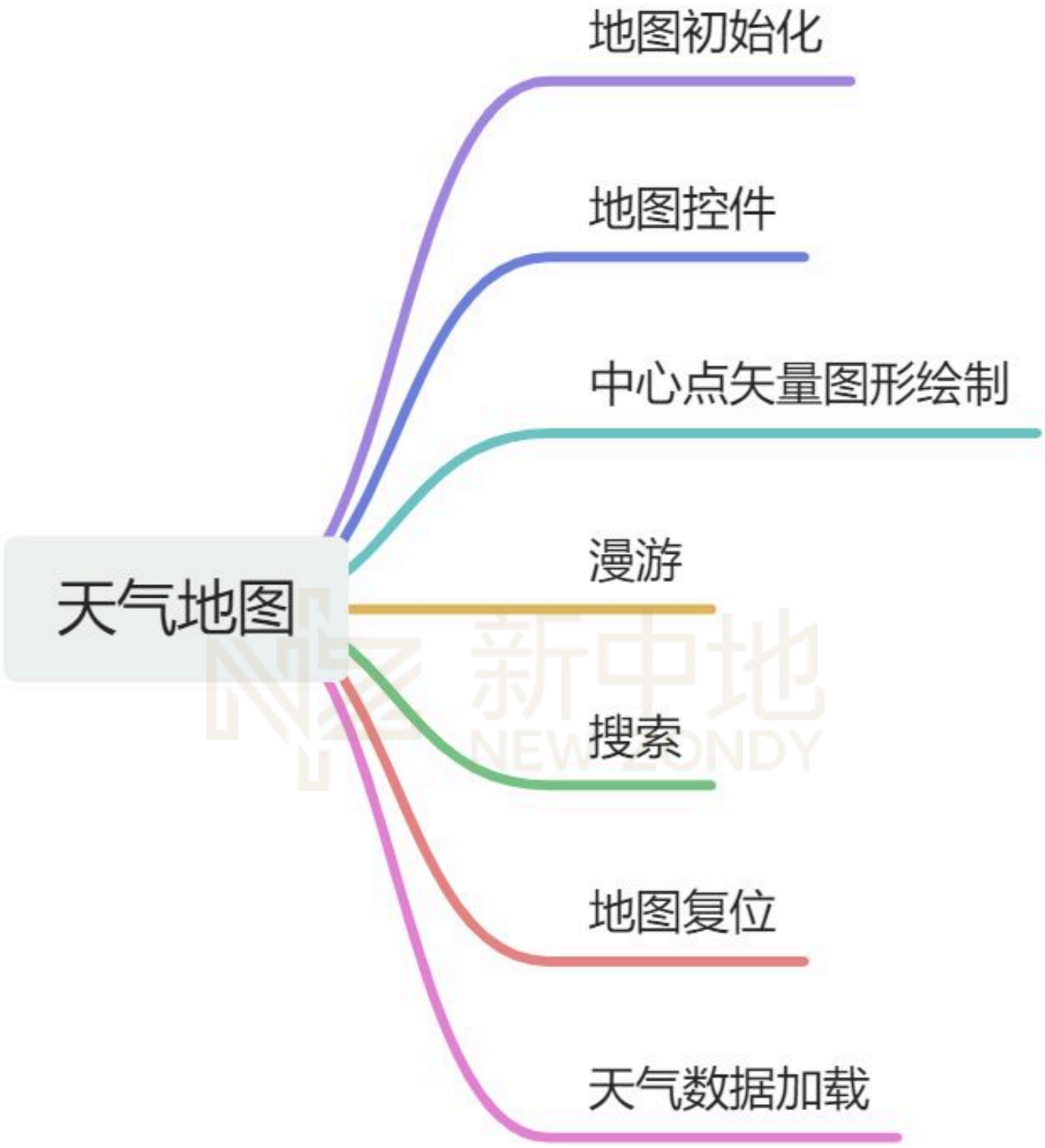
> 本课程的目的

1、通过本课程掌握ol-api的调用
2、使用ol结合高德api实现 `数据展示`
3、在之后的开发中，能够结合自己的项目，使用ol实现一些定制化的需求

> 项目功能及演示

https://animpen.com/run/pen/uRPdzh?tid=TnF0Gg0J

专注GIS开发　更多教程资料免费领取 请添加：gis2210



第一章 Openlayer基础

1、WebGIS简介

专注GIS开发　更多教程资料免费领取 请添加：gis2210

GIS的核心概念

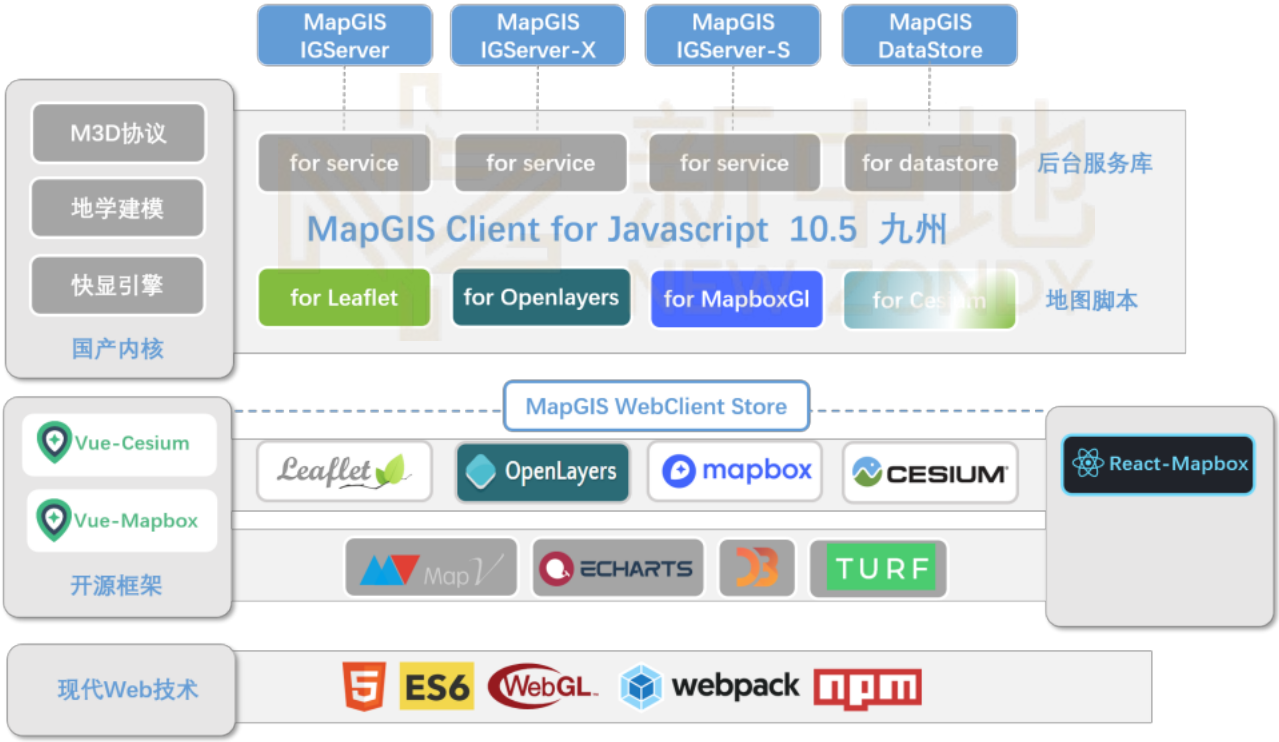
GIS（Geographic Information System）是一种用于存储、分析、管理和展示地理数据的计算机系统。以下是 GIS 中的一些核心概念：

1. 坐标系：地理数据通常以某种坐标系的形式存储，如经纬度坐标系或投影坐标系。
2. 地理数据：GIS 中的数据是具有地理上的位置信息的数据，如地图、地形、道路、建筑物、人口数据等。
3. 地图投影：地图投影是将三维地球表示为二维地图的过程。
4. 叠加分析：叠加分析是在一张地图上查看多个图层的分析，以探究其相互关系。
5. 空间分析：空间分析是指使用地理数据分析空间关系，如距离、面积和方位等。
6. 地理信息数据库：GIS 的核心是地理信息数据库，用于存储和管理地理数据。
7. 可视化：GIS 的目的之一是提供可视化，以方便人们更好地理解地理数据。

这些概念是 GIS 应用的核心，对于任何涉及 GIS 的项目都需要对这些概念有所了解。

1、WebGIS开发框架简介

http://develop.smaryun.com/#/total/core



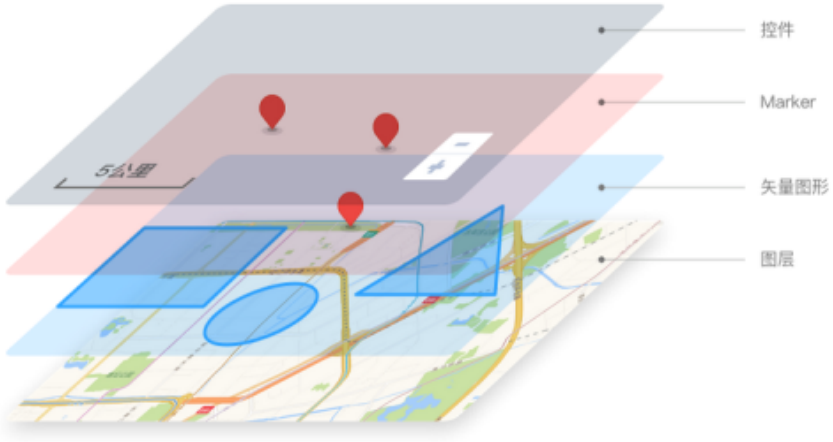
2、Openlayers

> 概念简单，入手难度相对较低

> 以高德地图为例

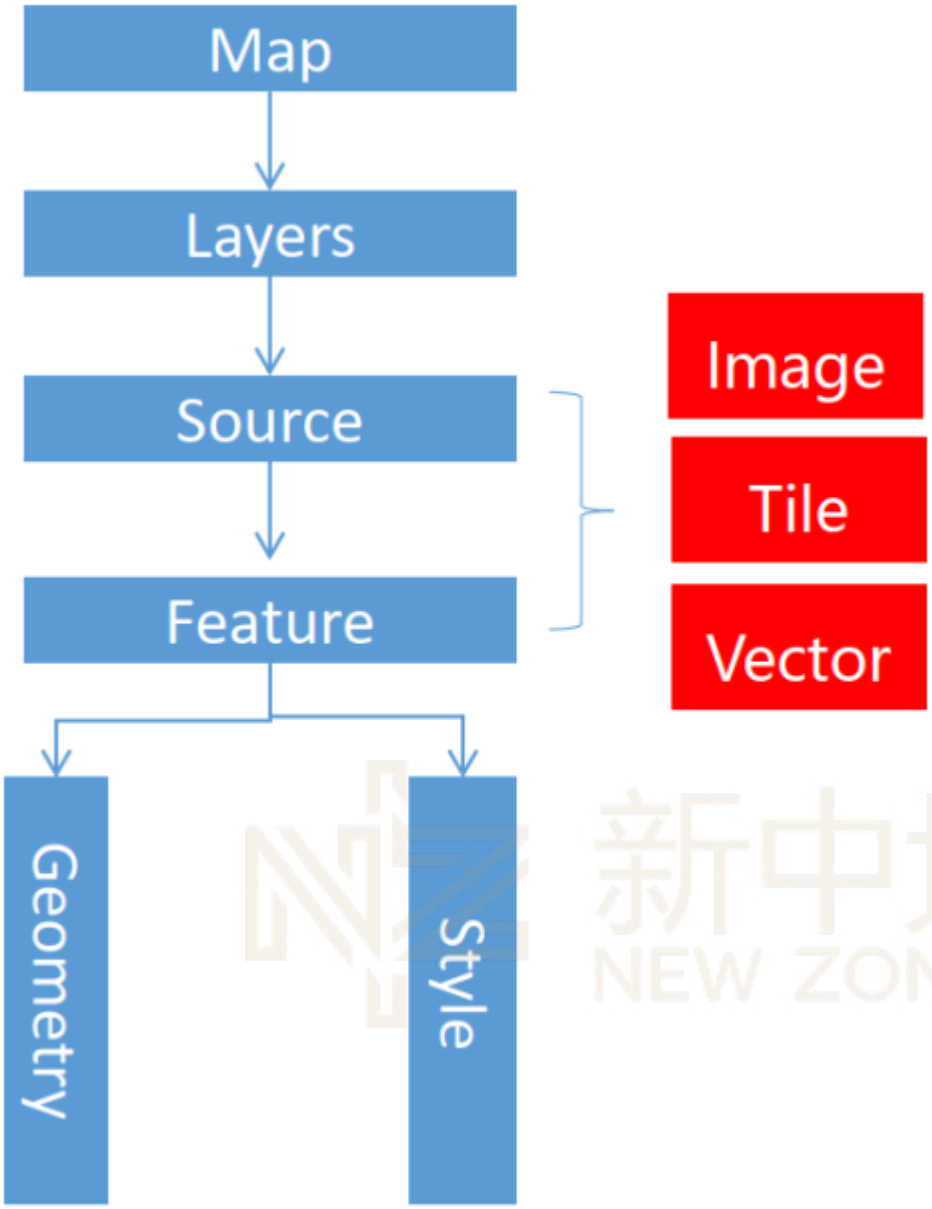
https://ditu.amap.com/

地图组成结构



https://openlayers.org/

专注GIS开发　更多教程资料免费领取 请添加：gis2210



Openlayers的核心概念:

1、一张 Map 是由很多 Layer 图层组成的。
2、一个 Layer 对应一个 Source 矢量数据源
3、一个 Source 由很多 Feature 组成
4、 Feature 是 Geometry 和 Style 组成

https://animpen.com/

OpenLayers是一个开源的地图显示库，它是基于JavaScript语言实现的。下面是OpenLayers的核心概念：

1. Map: 地图是一个可视化的容器，在其中显示地图内容。

2. Layer: 图层是地图中显示的一个独立的**数据集**。OpenLayers支持多种类型的图层，如影像图层，矢量图层，瓦片图层等。
3. Source是 图层**数据的来源**，表示图层**显示的内容**。

”Source”是一个抽象的概念，它是用来获取图层数据的。OpenLayers支持多种不同类型的数据源，如：

1. Image: 影像图像数据源，适用于显示静态的影像图像数据。
2. Tile: 瓦片数据源，适用于显示瓦片数据，例如在线地图服务。
3. Vector: 矢量数据源，适用于显示矢量数据，例如GeoJSON，KML等。

每个数据源都有其特定的配置选项，例如瓦片数据源需要指定瓦片地址模板，而矢量数据源需要指定矢量数据的URL。
通过选择不同的数据源类型，并配置相应的参数，可以实现多种类型的图层显示效果。因此，”source”在OpenLayers中是一个很重要的概念，它决定了图层显示的内容。

4. View: 视图是地图的显示范围，包括地图的中心点，缩放级别，显示角度等。
5. Projection: 投影是一种将地理坐标映射到平面坐标系上的方法。OpenLayers支持多种投影系统，如WGS 84，Web Mercator等。
6. Feature: 特征是地图上一个独立的显示对象，如点，线，面等。
7. Style: 样式是控制特征显示的方式的方法，如颜色，形状，大小等。

这些概念是OpenLayers的核心概念，通过灵活的配置和扩展，可以实现复杂的地图显示效果。

3、项目依赖

```
https://animpen.com/
https://cdn.baomitu.com/
https://lbs.amap.com/api/webservice/guide/create-project/get-key
```

4、色卡

色块	例子
#00B96B	更新
#6528e0	Read the docs
#dc3545	$red #dc3545
#fd7e14	$orange #fd7e14
#0d6efd	$blue #0d6efd

2、开发环境配置

1、VSCode安装

1) 安装

我们写代码需要安装代码编辑工具, 目前比较好用的推荐VSCode, 免费而功能强大

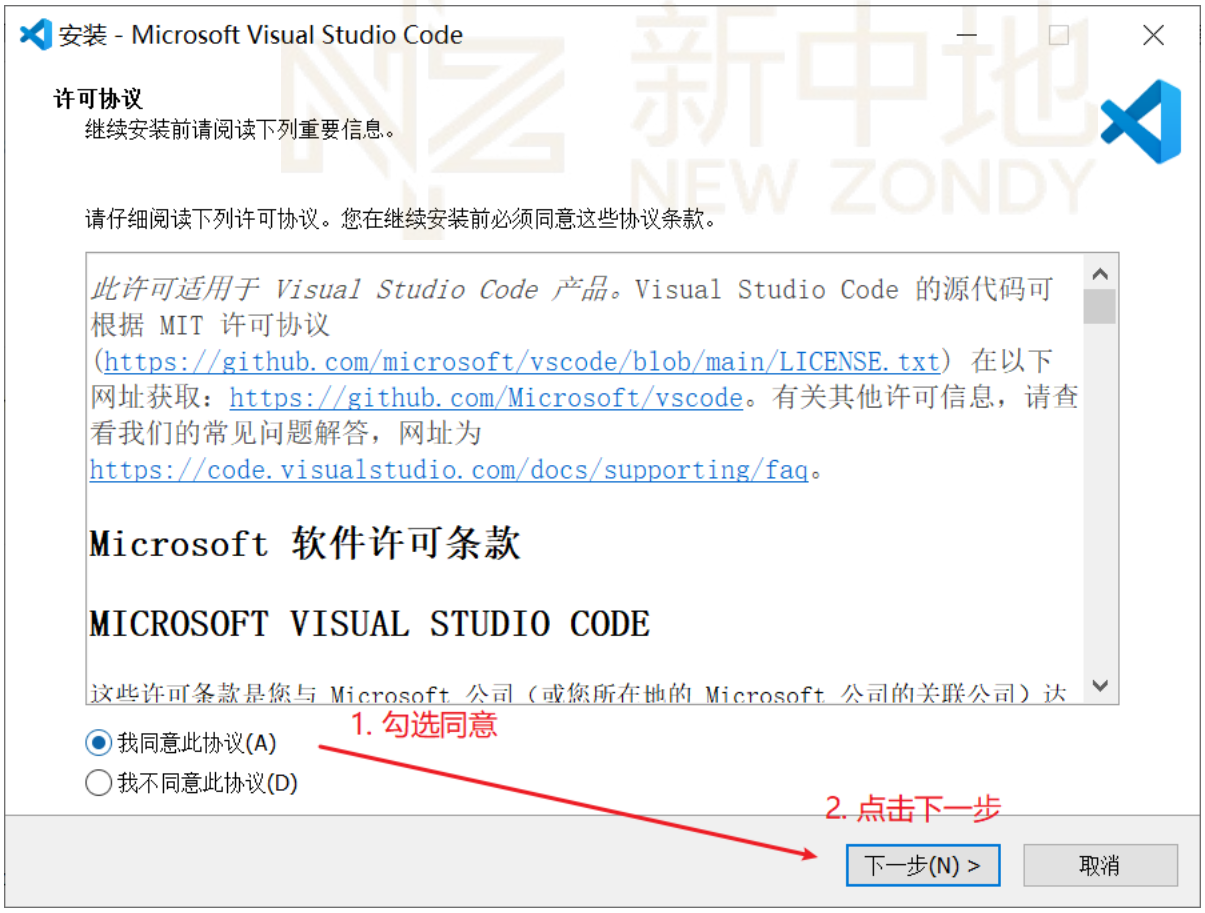
官网地址

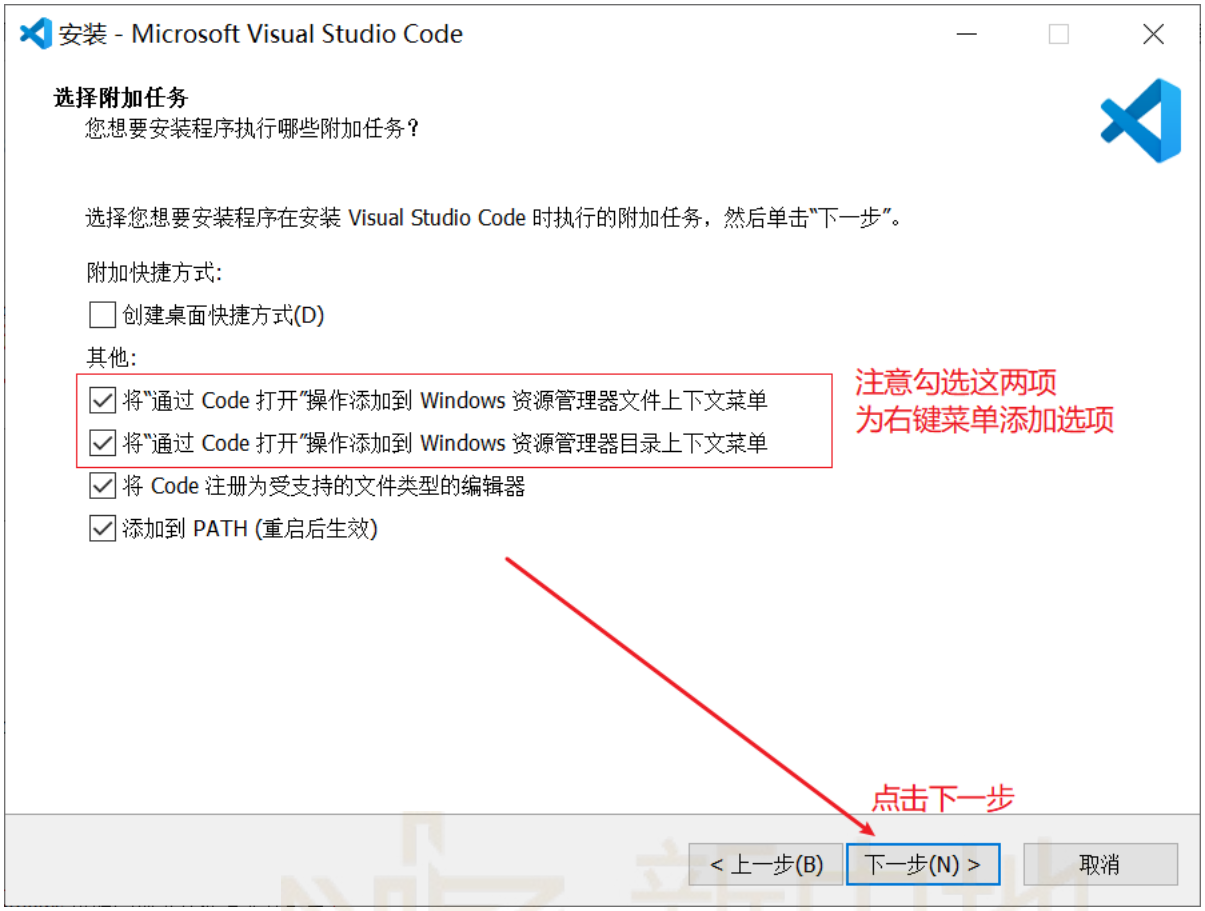
https://code.visualstudio.com/
微盘

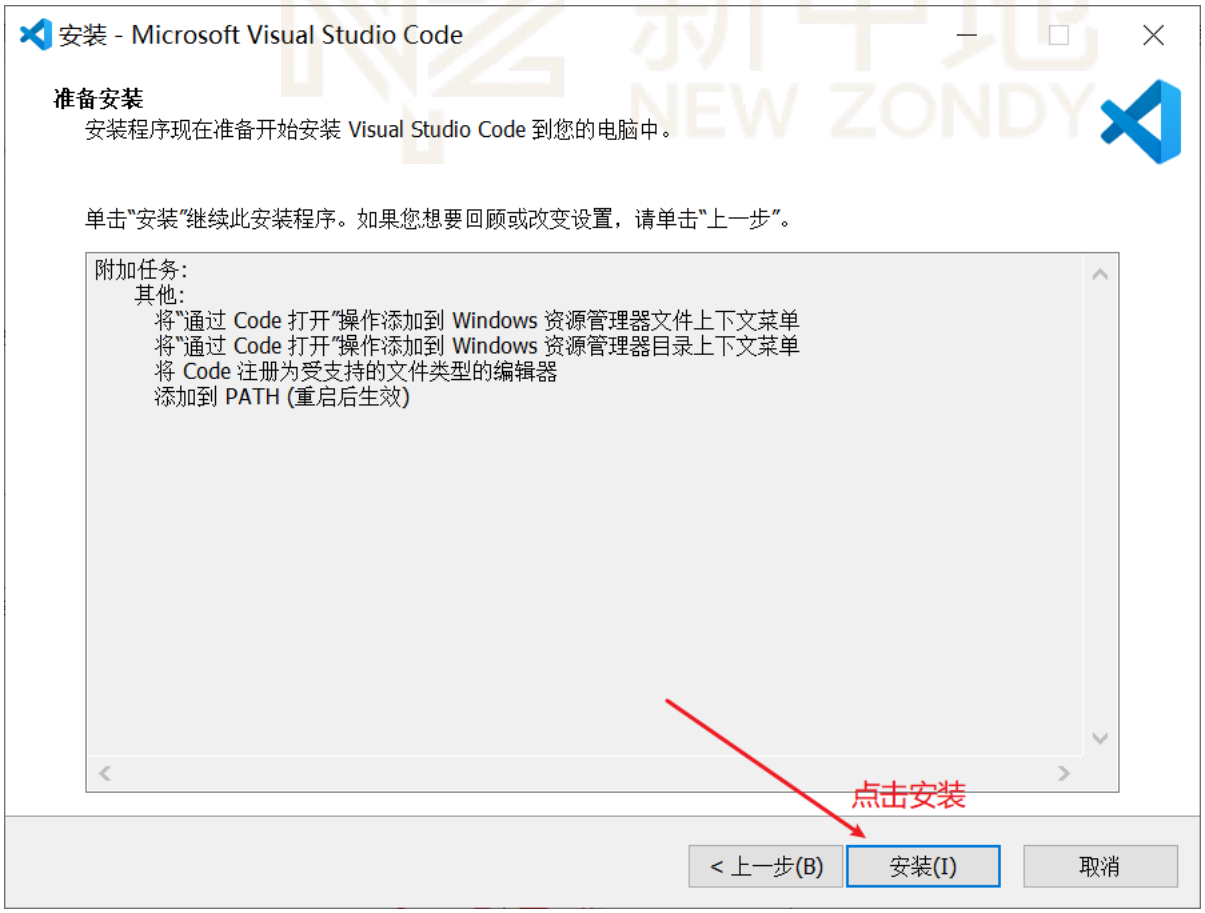
https://share.weiyun.com/sTBO8KRo

> 安装过程

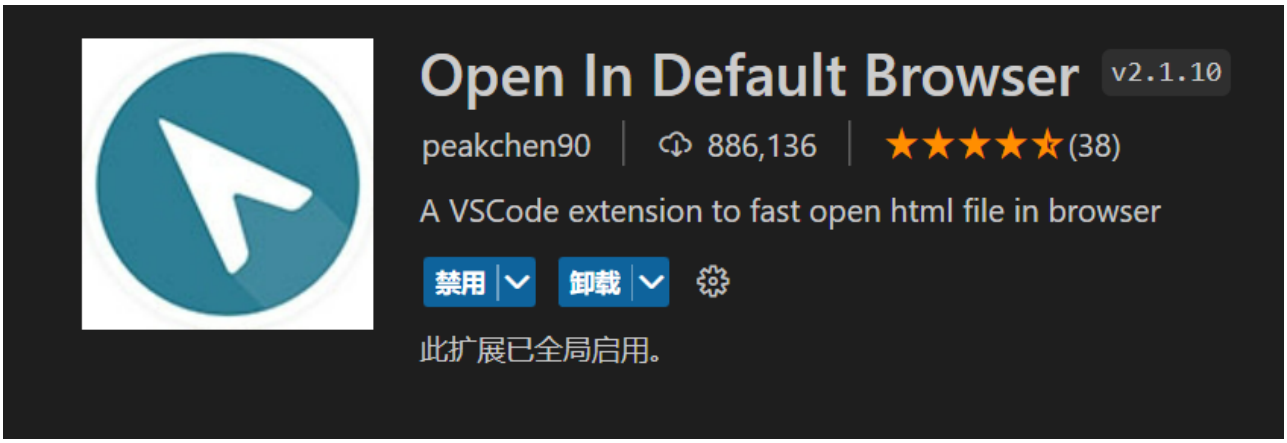
运行安装包 `VSCodeSetup-x64-1.65.2.exe`







2) 配置插件



2、 浏览器

做为代码的运行和测试环境, 我们需要安装一个浏览器, 这里推荐大家使用

- chrome

https://www.google.com/intl/zh-CN/chrome/

3、初始化地图

代码演示

https://animpen.com/run/pen/xMS3V8?tid=TID_kvKiz4

1、导入第三方依赖

https://cdn.baomitu.com/

```
<link rel="stylesheet" href="https://lib.baomitu.com/ol3/4.6.5/ol.css">
<script src="https://lib.baomitu.com/ol3/4.6.5/ol.js"></script>
```

2、初始化地图

https://zmnet.github.io/2018/05/28/OpenLayers2/

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="https://lib.baomitu.com/ol3/4.6.5/ol.css">
    <script src="https://lib.baomitu.com/ol3/4.6.5/ol.js"></script>
    <style>
      *{margin:0;padding:0}
      #map{
        width:100vw;
        height: 100vh;
```


专注GIS开发　更多教程资料免费领取 请添加：gis2210

```
    }
  </style>
 </head>

 <body>
  <div id="map">
  </div>
  <script>
   const gaode = new ol.layer.Tile({
    title: "高德地图",
    source: new ol.source.XYZ({
     url: 'http://wprd0{1-4}.is.autonavi.com/appmaptile?lang=zh_cn&size=1&style=7&x={x}&y={y}&z={z}',
     wrapX: false
    })
   });
   const map = new ol.Map({
    target: "map",
    layers: [
     gaode
    ],
    view:new ol.View({
     center:[114.30,30.50],
     zoom:12,
     projection:'EPSG:4326'
    })
   })
  </script>
 </body>
</html>
```


3、地图参数解释

> 初始化地图的代码只有一句

```
new ol.Map({})
```

- 其他的代码只是地图的用于设置地图的参数

1、参数 `target`

```
target: 'map'
```

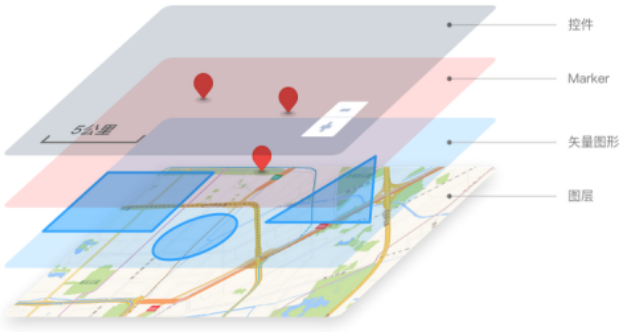
制定初始化的地图设置到 `html` 页面上的哪一个DOM元素上

2、参数 `layers`

通过这个参数的名字我可以推断一个 `map` 可以设置多个 `layer` ，图层是构成openlayers的基本单位，地图是由多个layer组成的，这种设计类似于Photoshop里面的图层，多个图层是可以叠加的，在最上面的会覆盖下面的。

专注GIS开发　更多教程资料免费领取 请添加：gis2210

地图组成结构



3、参数 view

视图是设置地图的显示范围，包括中心点，放大级别，坐标

```
view: new ol.View({
      center: [114.30, 30.50],
      zoom: 12,
      projection: 'EPSG:4326'
})
```

EPSG:4326 是一个在 GIS（地理信息系统）中使用的坐标参考系（Coordinate Reference System）代码。它表示一个地理坐标系，即使用经纬度来表示地理位置。

EPSG代码是由 European Petroleum Survey Group 分配的，它是一个用于统一管理坐标参考系的代码。4326代码是在 WGS 84（世界大地测量系统）椭球体模型的基础上定义的。

EPSG:4326 常被用于在网络上传输地理位置信息，如在 Web 地图服务和地理位置 API 等。

EPSG:4326 的经纬度范围是：经度范围在 −180° 到 180° 之间，纬度范围在 −90° 到 90° 之间。

> **总结**

通过以下代码解读，我可以得出一个结论，一个 ol 的地图，主要由 layer 和 view 组成。layer 可以有多个，view只能设置一个。

4、EPSG:4326和 EPSG:3857的区别

EPSG:4326 和 EPSG:3857 是两个常用的坐标参考系代码，用于在 GIS 中表示地理位置。它们的主要区别如下：

- EPSG:4326 表示一个地理坐标系，使用经纬度来表示地理位置，通常用于地理位置的显示和制图。
- EPSG:3857 表示一个 Web 墨卡托坐标系，使用平面墨卡托投影来表示地理位置。

因此，两个坐标系的主要区别在于它们使用的坐标系统不同：EPSG:4326 使用的是经纬度，而 EPSG:3857 使用的是平面墨卡托投影。

EPSG:3857 在在线地图中被广泛使用，因为它能够在 Web 地图上提供更高的精度和更好的分辨率。然而，EPSG:4326 在网络上传输地理位置信息时被更多地使用，因为它使用的是标准的地理坐标系。

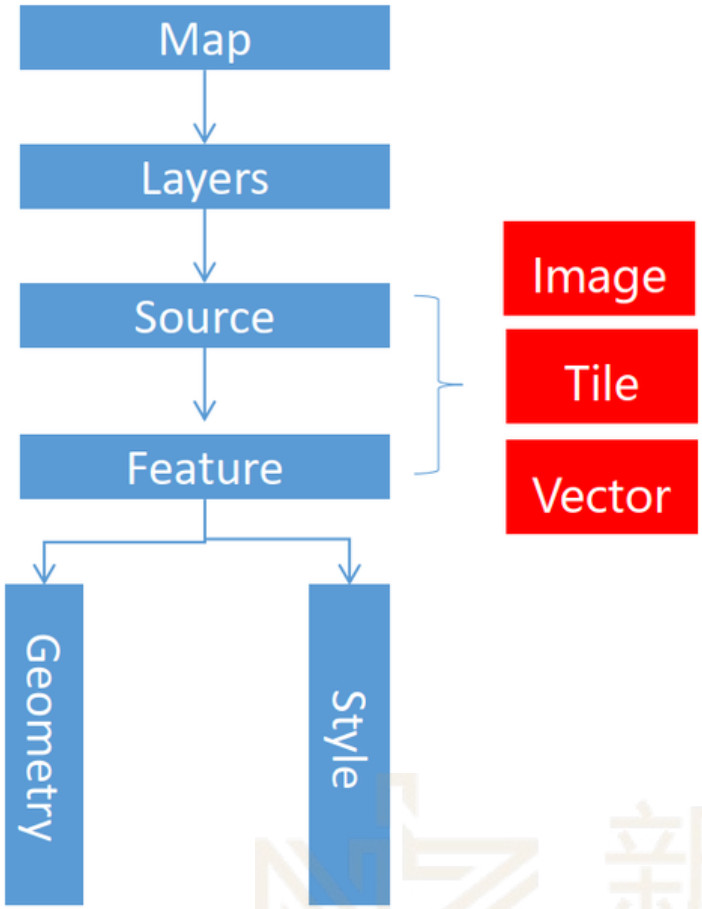
总的来说，选择使用哪个坐标系取决于你的应用需求：如果需要高精度和分辨率，选择 EPSG:3857；如果需要标准的地理坐标系，选择 EPSG:4326

4、地图控件

https://animpen.com/pen/uRPdzh

```
/* 视图跳转控件 */
const ZoomToExtent = new ol.control.ZoomToExtent({
      extent: [110, 30, 160, 30],
})
map.addControl(ZoomToExtent)
/* 放大缩小控件 */
const zoomslider = new ol.control.ZoomSlider();
map.addControl(zoomslider)
//全屏控件
const fullScreen = new ol.control.FullScreen();
map.addControl(fullScreen)
```

5、设置矢量图形



> 绘图步骤

1、通过几何信息和样式信息构建要素
2、将要素添加到 `矢量数据源`
3、将矢量数据源添加到 `矢量图层`
4、将 `矢量图层` 添加到地图容器

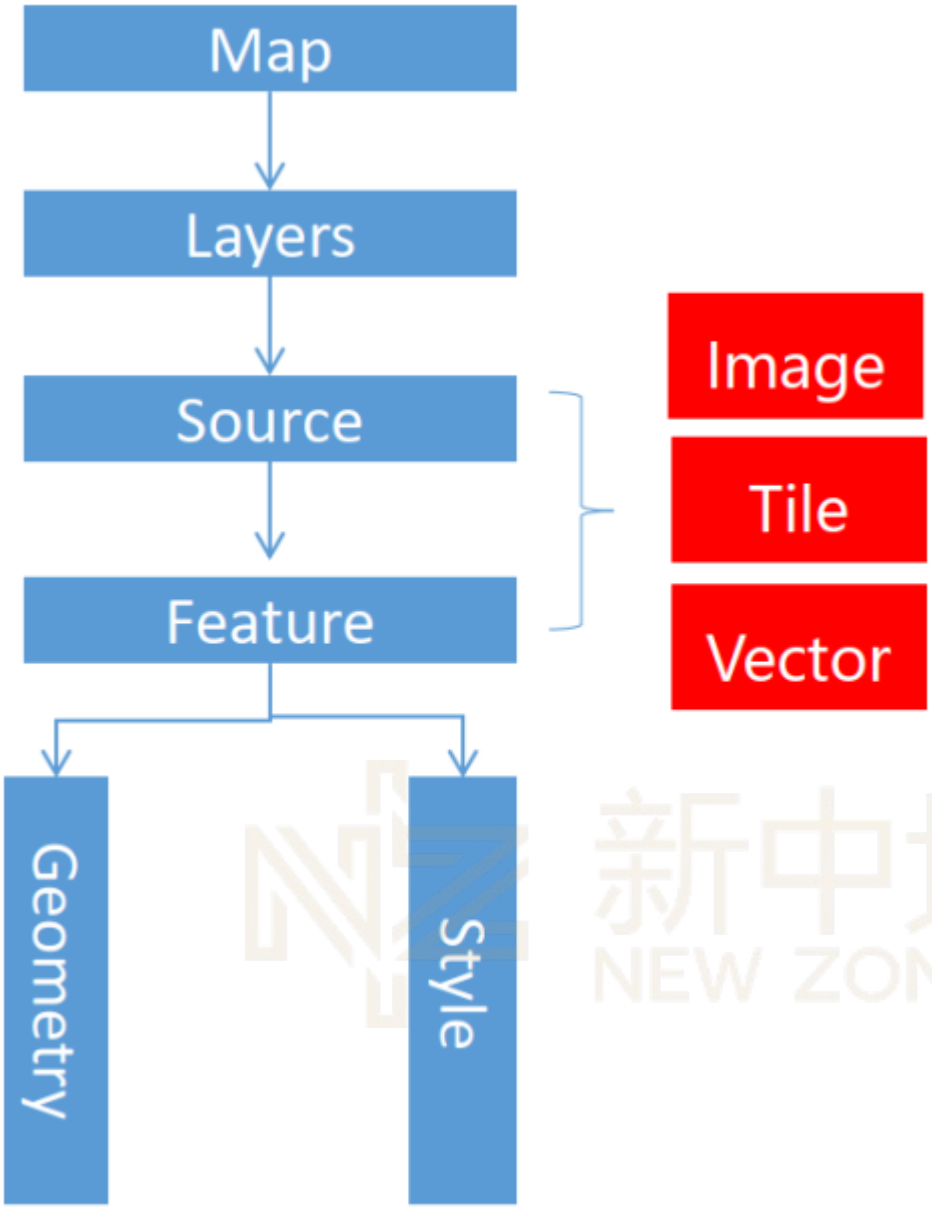
> 代码

```
/* 1、构建要素 */
var point = new ol.Feature({
      geometry: new ol.geom.Point([114.30, 30.50])
})
let style = new ol.style.Style({
      image: new ol.style.Circle({
            radius: 10,
            fill: new ol.style.Fill({
                  color: ”#ff2d51”
            }),
            stroke: new ol.style.Stroke({
                  color: ”#333”,
                  width: 2
            })
```

```
        })
})
point.setStyle(style);
/* 2、将要素添加到矢量数据源 */
const source = new ol.source.Vector({
        features: [point]
})
/* 3、将矢量数据源添加到矢量图层 */
const layer = new ol.layer.Vector({
        source
})
/* 4、将矢量图层添加到地图容器 */
map.addLayer(layer)
```

6、加载geojson数据

https://openlayers.org/en/latest/examples/geojson.html



1、geojson定义

> geojson 数据是矢量数据，是包含地理信息的json数据，格式是以key:value的形式存在的。后缀以 geojson 结尾

2、geojson设置一个点要素

2-1、设置点要素

https://animpen.com/pen/7A01Qa

```
var data = {
    type: ”FeatureCollection”,
    features: [
        {
            type: ”Feature”,
            geometry: {
                type: ”Point”,
                coordinates: [114.30, 30.50]
            }
        }
    ]
}
var source = new ol.source.Vector({
    /* 将geojson数据设置给实例数据源 */
    features: new ol.format.GeoJSON().readFeatures(data)
})
var layer = new ol.layer.Vector({
    source
})
map.addLayer(layer);
```

> 设置样式

```
const style = new ol.style.Style({
    image: new ol.style.Circle({
        radius: 8,
        fill: new ol.style.Fill({
            color: ”#ff2d51”
        }),
        stroke: new ol.style.Stroke({
            color: ’#333’,
            width: 2
        })
    }),

})
layer.setStyle(style);
```

专注GIS开发　更多教程资料免费领取 请添加：gis2210

2-2、设置线

https://animpen.com/pen/slNXPH

```
var data = {
    type: "FeatureCollection",
    features: [
        // {
        //     type: "Feature",
        //     geometry: {
        //         type: "Point",
        //         coordinates: [114.30, 30.50]
        //     }
        // },
        {
            type: "Feature",
            geometry: {
                type: "LineString",
                coordinates: [[114.30, 30.50], [116, 30.50]]
            }
        }
    ]
}
//设置样式
const style = new ol.style.Style({
    //边线颜色
    stroke: new ol.style.Stroke({
        color: '#ff2d51',
        width: 4
    })

})
```

2-3、设置 Polygon 区

https://animpen.com/pen/-jl-_m

```
var data = {
      type: "FeatureCollection",
      features: [
            {
                  type: "Feature",
                  geometry: {
                        type: "Polygon",
                        coordinates: [[[114.30, 30.50], [116, 30.50], [116, 30]]]
                  }
            }
      ]
}
//设置样式
const style = new ol.style.Style({
      //边线颜色
      stroke: new ol.style.Stroke({
            color: '#ff2d51',
            width: 2
      }),
      //设置填充色
      fill: new ol.style.Fill({
            color: "rgba(50, 50, 50, 0.3)"
      })

})
```

3、加载本地 geojson 文件的数据

```
{
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": [114.30, 30.50]
            }
        }
    ]
}
```

```
const source = new ol.source.Vector({
    url: './data/map.geojson',
    format: new ol.format.GeoJSON()
})
const layer = new ol.layer.Vector({
    source
})
map.addLayer(layer)
```

> 一定要以服务的方式去加载，否则报错

> 给图层设置样式

```
const style = new ol.style.Style({
      image: new ol.style.Circle({
            radius: 8,
            fill: new ol.style.Fill({
                  color: '#ff2d51'
            }),
            stroke: new ol.style.Stroke({
                  color: '#333',
                  width: 2
            })
      })
})
layer.setStyle(style)
```


4、加载网络数据

https://animpen.com/pen/C_4WLb

https://datav.aliyun.com/portal/school/atlas/area_selector

```
const source = new ol.source.Vector({
    url: 'https://geo.datav.aliyun.com/areas_v3/bound/geojson?code=420100',
    format: new ol.format.GeoJSON()
})
const layer = new ol.layer.Vector({
    source
})
map.addLayer(layer)
```

> 设置样式

```
const style = new ol.style.Style({
      //填充色
      fill: new ol.style.Fill({
            color: 'rgba(50,50,50,0.4)'
      }),
      //边线颜色
      stroke: new ol.style.Stroke({
            color: '#ff2d5180',
            width: 2
      })
})
layer.setStyle(style)
```

7、地图事件及漫游

1、地图点击事件

```
map.on("click",(evt)=>{
      var position = evt.coordinate;
      console.log(position)
})
```

点击事件在地图上设置点

```
//...map
var source = new ol.source.Vector({

});
var layer = new ol.layer.Vector({
      source
})
map.addLayer(layer);
map.on("click", (evt) => {
      var position = evt.coordinate;
      var point = new ol.Feature({
            geometry: new ol.geom.Point(position)
      })
      source.addFeature(point)
})
```

2、漫游

核心API `map.getView.animate()`

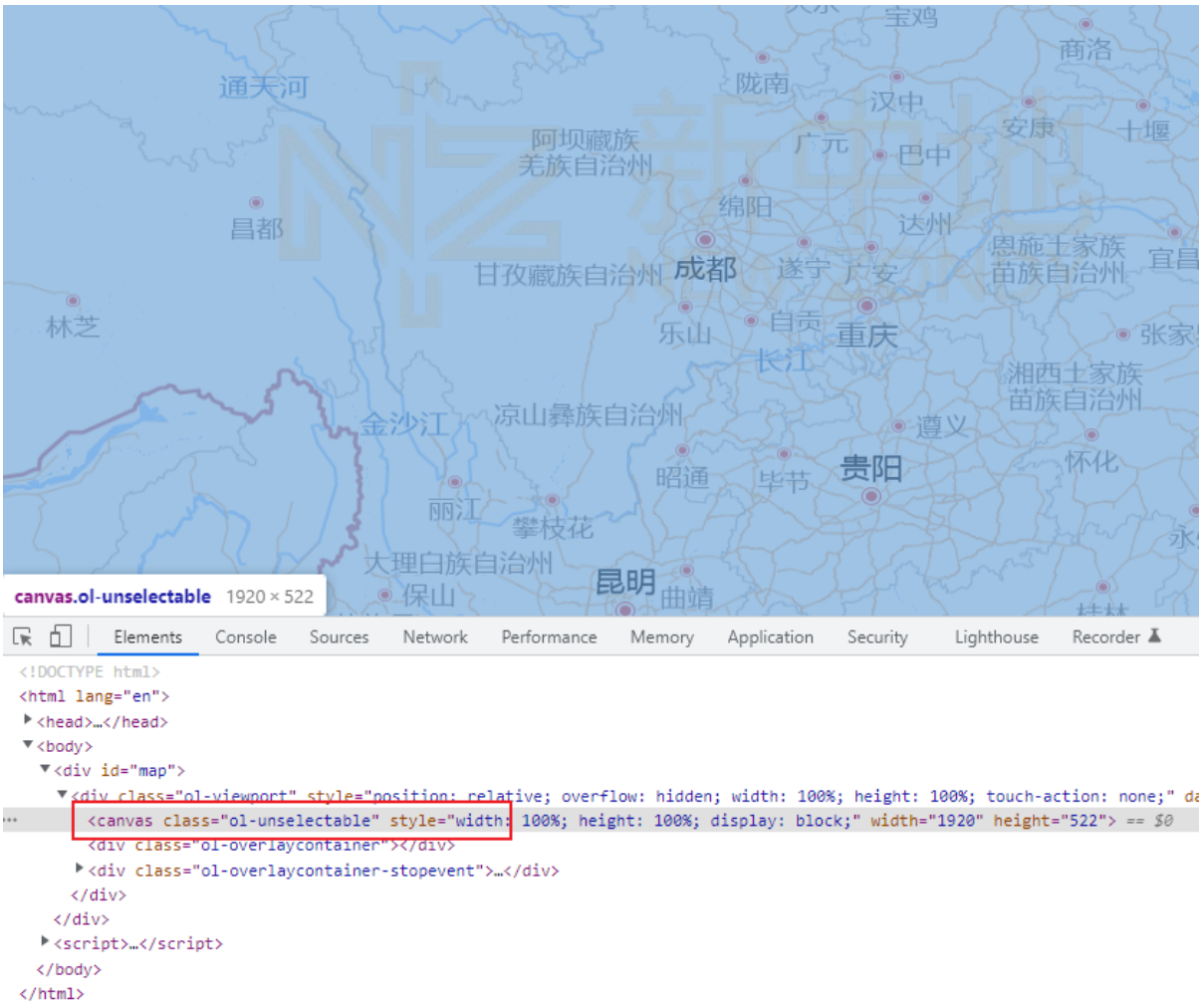
点击什么地方飞行到什么地方

```
map.on("click", (evt) => {
      var position = evt.coordinate;
      map.getView().animate({
            center: position,
            zoom: 10
      })
})
```

第二章 Canvas

本章节讲解Canvas如何结合 `Openlayer` 使用，首先我们讲解Canvas的绘图基础。

我们初始化地图的时候可以看见，实际上Openlayer的地图就是用Canvas实现绘制的。



1、Canvas绘制基本

参考教程：https://github.com/827652549/CanvasStudy

1、概念

什么是canvas

`HTML5 <canvas>` 元素用于图形的绘制，区别于css，它的绘制通过 `JavaScript` 来完成绘制。

`<canvas>` 标签只是 `图形容器` ，您必须使用脚本来绘制图形

Canvas API主要聚焦于2D图形。同样使用<canvas>元素的 `WebGL API` 则用于绘制硬件加速的2D和3D图形。

2、绘制矩形

```
<canvas id="canvas" width="200" height="200"></canvas>
<script>
        /* canvas */
        var canvas = document.getElementById("canvas");
        /* canvas
        getContext('2d')HTML5,
         */
```

```
        const ctx = canvas.getContext("2d");
        /* fillStyle cssbackground-color */
        ctx.fillStyle= "#6528e0";
        /* fillRect(x,y,width,height) */
        ctx.fillRect(10,10,100,100)
</script>
```

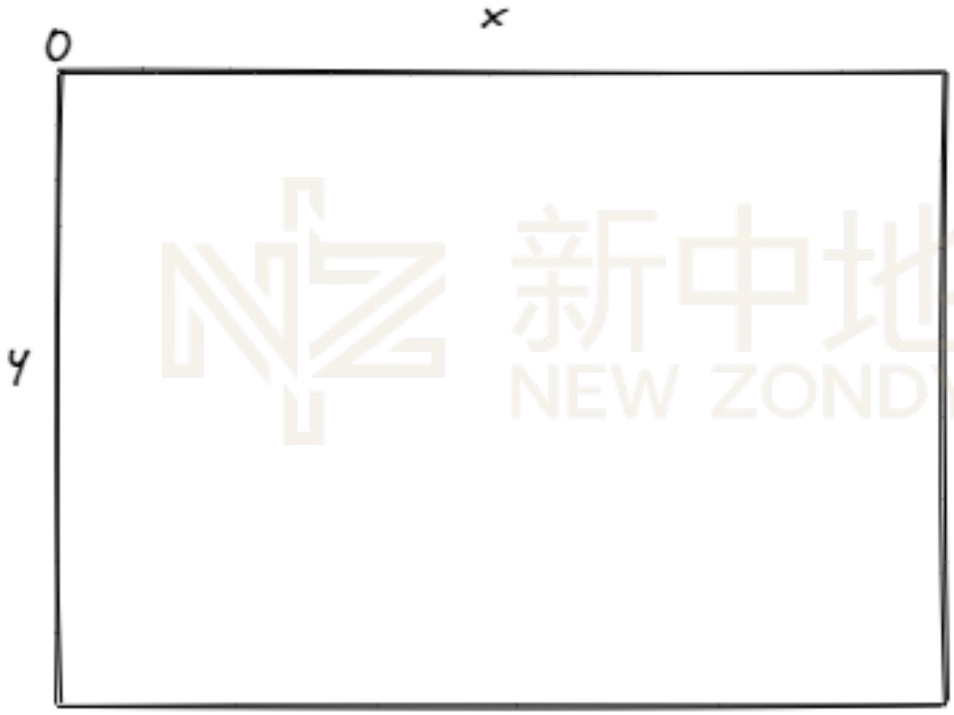
3、canvas中的坐标

canvas 是一个二维网格。

canvas 的左上角坐标为 (0,0)

上面的 fillRect 方法拥有参数 (0,0,100,,100)。

意思是：在左上角开始 (0,0)的位置，绘制100*100的图形

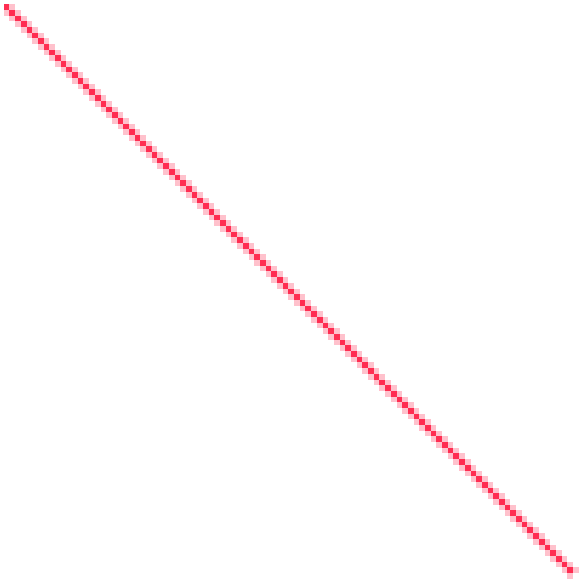


4、路径

在Canvas上画线，我们将使用以下两种方法：

- moveTo(*x,y*) 定义线条开始坐标
- lineTo(*x,y*) 定义线条结束坐标

绘制线条我们必须使用到 ”link” 的 `stroke()` 方法,执行绘制。




```
<canvas id="canvas" width="200" height="200"></canvas>
<script>
        /* canvas */
        var canvas = document.getElementById("canvas");
        /* canvas
        getContext('2d')HTML5,
         */
        const ctx = canvas.getContext("2d");
       /*  */
       ctx.moveTo(0,0);
       ctx.lineTo(100,100);
       ctx.strokeStyle = "#ff2d51";
       ctx.stroke();
</script>
```

5、绘制圆 arc

```
<canvas id="canvas" width="200" height="200"></canvas>
<script>
        /* canvas */
        var canvas = document.getElementById("canvas");
        /* canvas */
        const ctx = canvas.getContext("2d");
        ctx.beginPath();
```

```
        ctx.arc(100,100,50,0,Math.PI*2);
        ctx.closePath();
        ctx.fillStyle= "#6528e0";
        ctx.fill();
</script>
```

2、Arc绘制圆

绘制弧线

arc(x, y, r, startAngle, endAngle, anticlockwise): 以(x, y) 为圆心，以r 为半径，从 startAngle 弧度开始到endAngle弧度结束。anticlosewise 是布尔值，true 表示逆时针，false 表示顺时针(默认是顺时针)。

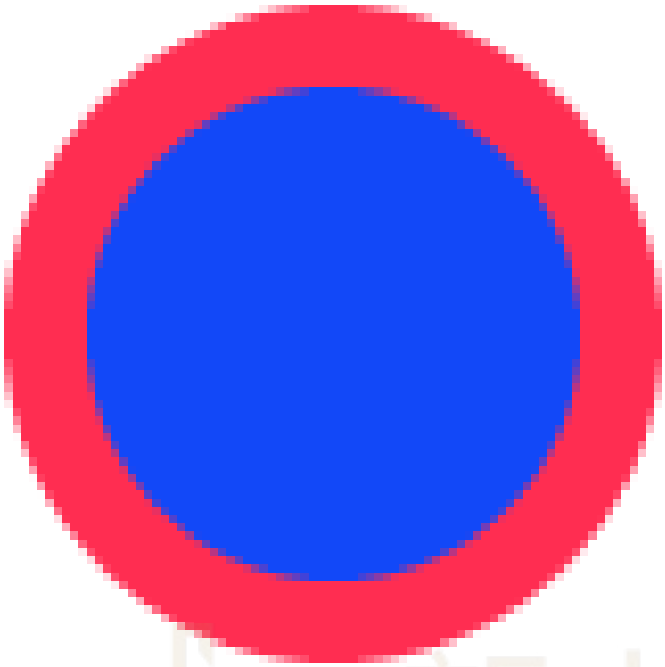
注意：

1. 这里的度数都是弧度。
2. 0 弧度是指的 x 轴正方向。radians=(Math.PI/180)*degrees //角度转换成弧度

1、绘制圆

```
function draw() {
    const ctx = canvas.getContext("2d");
    ctx.beginPath();
    /* false-[],true- */
    ctx.arc(50, 50, 50, 0, Math.PI * 2, false)
    ctx.stroke();
}
draw();
```

2、绘制两个圆




```
function draw() {
    const ctx = canvas.getContext("2d");
    /*  */
    ctx.beginPath();
    ctx.arc(100, 100, 40, 0, Math.PI * 2, false);
    ctx.closePath();
    ctx.fillStyle = "#ff2d51"
    ctx.fill();
    /*  */
    ctx.beginPath();
    ctx.arc(100, 100, 30, 0, Math.PI * 2, false);
    ctx.closePath();
    ctx.fillStyle = "#1248F8"
    ctx.fill();
}
draw();
```

3、绘制动画圆

1、setInterval实现

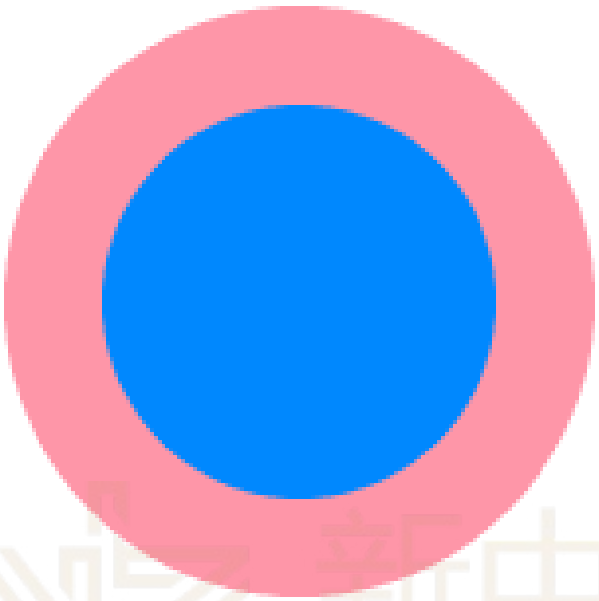
```
<body>
    <canvas id="canvas" width="200" height="200"></canvas>
    <script>
        /* canvas */
        var canvas = document.getElementById("canvas");
        const ctx = canvas.getContext("2d");
        var radius = 50;
        /*  */
        var increase = true;
        /*
        100  false
        50  true
        */
        function draw() {
            /*  */
            ctx.clearRect(0, 0, canvas.width, canvas.height);
            ctx.beginPath();
            ctx.arc(100, 100, radius, 0, Math.PI * 2, false);
            ctx.closePath();
            ctx.fillStyle = "#6528e0"
            ctx.fill();
            if (radius >= 100) {
                /* 100, */
```

```
                increase = false
            } else if (radius < 50) {
                /* 50 */
                increase = true;
            }
            if (increase) {
                radius++
            } else {
                radius--;
            }
        }
        setInterval(() => {
            draw()
        }, 20)
    </script>
</body>
```

2、setTimeout

```
<body>
      <canvas id="canvas" width="200" height="200"></canvas>
      <script>
            /* canvas */
            var canvas = document.getElementById("canvas");
            const ctx = canvas.getContext("2d");
            var radius = 50;
            /*  */
            var increase = true;
            /*
            100  false
            50  true
            */
            function draw() {
                  /*  */
                  ctx.clearRect(0, 0, canvas.width, canvas.height);
                  ctx.beginPath();
                  ctx.arc(100, 100, radius, 0, Math.PI * 2, false);
                  ctx.closePath();
                  ctx.fillStyle = "#6528e0"
                  ctx.fill();
                  if (radius >= 100) {
                        /* 100, */
                        increase = false
                  } else if (radius < 50) {
                        /* 50 */
                        increase = true;
                  }
                  if (increase) {
                        radius++
                  } else {
                        radius--;
                  }
                  setTimeout(draw, 20)
            }
            draw();
      </script>
</body>
```

4、多圈动画



```
<body>
    <canvas id="canvas" width="200" height="200"></canvas>
    <script>
        /* canvas */
        var canvas = document.getElementById("canvas");
        const ctx = canvas.getContext("2d");
        var radius = 60;
        /*  */
        var increase = true;
        /*
        100  false
        50  true
        */
        function draw() {
            /*  */
            ctx.clearRect(0, 0, canvas.width, canvas.height);
            ctx.beginPath();
            ctx.arc(100, 100, radius, 0, Math.PI * 2, false);
            ctx.closePath();
            ctx.fillStyle = "#6528e080"
            ctx.fill();
            /*  */
            ctx.beginPath();
            ctx.arc(100, 100, 50, 0, Math.PI * 2, false);
```

```
            ctx.closePath();
            ctx.fillStyle = "#0088ff"
            ctx.fill();
            if (radius >= 100) {
                /* 100, */
                increase = false
            } else if (radius < 60) {
                /* 50 */
                increase = true;
            }
            if (increase) {
                radius++
            } else {
                radius--;
            }
            setTimeout(draw, 30)
        }
        draw();
    </script>
</body>
```


5、Canvas结合Openlayer

1、openlayer-canvas

canvas 是作为样式设置给要素的。

```
var style = new ol.style.Style({
      image: new ol.style.Icon({
          img: canvas,
          imgSize: [canvas.width, canvas.height]
      })
  });
```

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="https://lib.baomitu.com/openlayers/4.6.5/ol.css">
    <script src="https://lib.baomitu.com/openlayers/4.6.5/ol.js"></script>
</head>

<body>
    <div id="map">
    </div>
    <script>
        var gaode = new ol.layer.Tile({
            title: "",
            source: new ol.source.XYZ({
                url: 'http://wprd0{1-4}.is.autonavi.com/appmaptile?lang=zh_cn&size=1&style=7&x=
{x}&y={y}&z={z}',
                wrapX: false
            })
        });
        /*  */
        var layer = new ol.layer.Vector({
            source: new ol.source.Vector()
        })
        var map = new ol.Map({
            target: "map",
            layers: [gaode, layer],
            view: new ol.View({
                projection: 'EPSG:4326',
                center: [114.30, 30.50],
                zoom: 8
            })
        })
        var center = [114.30, 30.50]
        let size = 40;
        var canvas = document.createElement('canvas');
        canvas.width = size;
        canvas.height = size;
        var ctx = canvas.getContext('2d');
        var radius = size / 4;
        var increase = true;
        function draw() {
```

```
            ctx.clearRect(0, 0, size, size);
            /*  */
            ctx.beginPath();
            ctx.arc(size / 2, size / 2, radius, 0, Math.PI * 2);
            ctx.closePath();
            ctx.fillStyle = "#ff2d5166";
            ctx.fill();
            /*  */
            ctx.beginPath();
            ctx.arc(size / 2, size / 2, 8, 0, Math.PI * 2);
            ctx.closePath();
            ctx.fillStyle = "#0088ff";
            ctx.fill();
            if (radius > 14) {
                increase = false
            }
            if (radius < 10) {
                increase = true;
            }
            if (increase) {
                radius++
            } else {
                radius--;
            }
            setTimeout(draw, 100)
            map.render();
        }
        draw();
        var style = new ol.style.Style({
            image: new ol.style.Icon({
                img: canvas,
                imgSize: [canvas.width, canvas.height]
            })
        });
        var shape = new ol.Feature({
            geometry: new ol.geom.Point(center)
        });
        var beijing = new ol.Feature({
            geometry: new ol.geom.Point([120, 40])
        })
        layer.setStyle(style)
        layer.getSource().addFeatures([shape, beijing]);
    </script>
</body>

</html>
```

2、封装

```
function setCanvas(map) {
    let size = 40;
    var canvas = document.createElement('canvas');
    canvas.width = size;
    canvas.height = size;
    var ctx = canvas.getContext('2d');
    var radius = size / 4;
    var increase = true;
    function draw() {
```

```
        ctx.clearRect(0, 0, size, size);
        /*  */
        ctx.beginPath();
        ctx.arc(size / 2, size / 2, radius, 0, Math.PI * 2);
        ctx.closePath();
        ctx.fillStyle = "#ff2d5166";
        ctx.fill();
        /*  */
        ctx.beginPath();
        ctx.arc(size / 2, size / 2, 8, 0, Math.PI * 2);
        ctx.closePath();
        ctx.fillStyle = "#0088ff";
        ctx.fill();
        if (radius > 14) {
            increase = false
        }
        if (radius < 10) {
            increase = true;
        }
        if (increase) {
            radius++
        } else {
            radius--;
        }
        setTimeout(draw, 180)
        map.render();
    }
    draw();
    return canvas;
}

//
let canvas = setCanvas(map);
var style = new ol.style.Style({
       image: new ol.style.Icon({
       img: canvas,
       imgSize: [canvas.width, canvas.height]
   })
});
```

第三章 高德API调用

1、调用高德API

在当所在的城市，设置点

1、创建key

https://lbs.amap.com/api/webservice/guide/create-project/get-key

app	e7d5c7a331302d770ea8d9b1a882a8af

2、高德api

https://restapi.amap.com/v3/

https://restapi.amap.com/v3/ip?key=${key}	ip定位
/weather/weatherInfo?city=武汉&key=${key}	天气查询

专注GIS开发　更多教程资料免费领取 请添加：gis2210

/geocode/geo?address=${city}&key=${key}	城市逆向地址解析

2、ip定位

https://restapi.amap.com/v3/

https://restapi.amap.com/v3/ip?key=${key}	ip定位

依赖库

```
<script src="https://lib.baomitu.com/jquery/2.2.4/jquery.js"></script>
<script src="https://lib.baomitu.com/Turf.js/latest/turf.min.js"></script>
```

1、ajax请求获取坐标和城市编号

```
$.ajax({
      url: `https://restapi.amap.com/v3/ip?key=${key}`
}).then(res => {
      console.log(res)
})
```

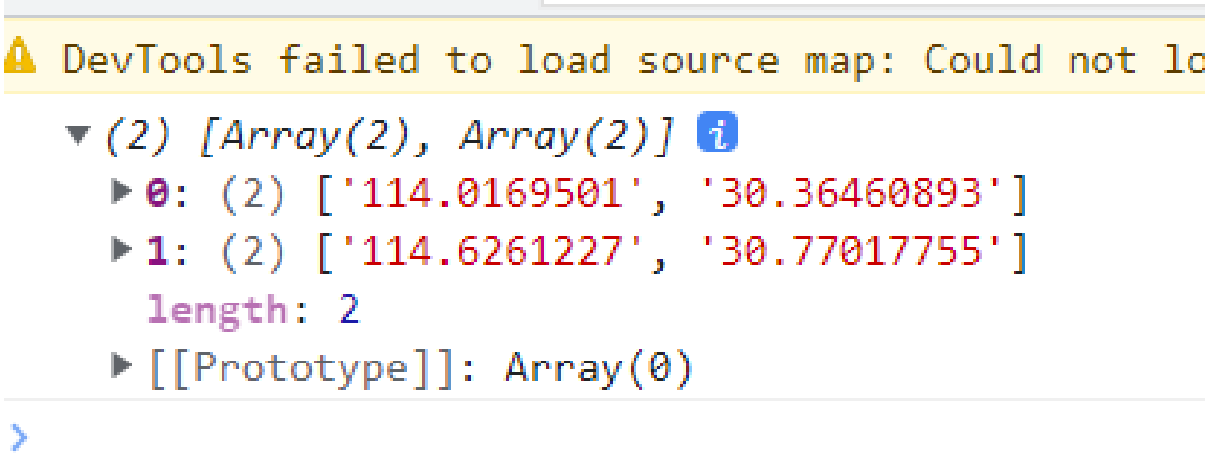
2、处理 rectangle 获取当前城市的中心点

2-1、将坐标设置二维数组

rectangle 获取的是坐下角到右上角的坐标

```
$.ajax({
      url: `https://restapi.amap.com/v3/ip?key=${key}`
}).then(res => {
      const { city, rectangle,adcode } = res;
      /*  */
      var positions = rectangle.split(";");
      var arr = [];
```

```
      positions.forEach(item => {
            arr.push(item.split(","))
      })
      console.log(arr)
})
```



2-2、使用 turf库获取 中心点

```
var p1 = turf.point(arr[0]);
var p2 = turf.point(arr[1]);
var midPoint = turf.midpoint(p1, p2)
//midPointgeojson
var center = midPoint.geometry.coordinates
console.log(midPoint)
```

2-3、 geojson 加载点

```
//geojson
const source = new ol.source.Vector({
      wrapX: false
})
const vector = new ol.layer.Vector({
      source: source,
      // style
})
map.addLayer(vector)
$.ajax(...).then(res=>{
  ...
  source.addFeatures(new ol.format.GeoJSON().readFeatures(midPoint))
})
```

3、加载 当前市 的 geojson 数据

```
$.ajax({
      url: `https://restapi.amap.com/v3/ip?key=${key}`
}).then(res => {
      const { city, rectangle, adcode } = res;
      ...
```

```
      let adJson = `https://geo.datav.aliyun.com/areas_v3/bound/${adcode}.json`
      var geoLayer = new ol.layer.Vector({
            source: new ol.source.Vector({
                  url: adJson,     //
                  format: new ol.format.GeoJSON()    //
            })
      });
      map.addLayer(geoLayer)
})
```

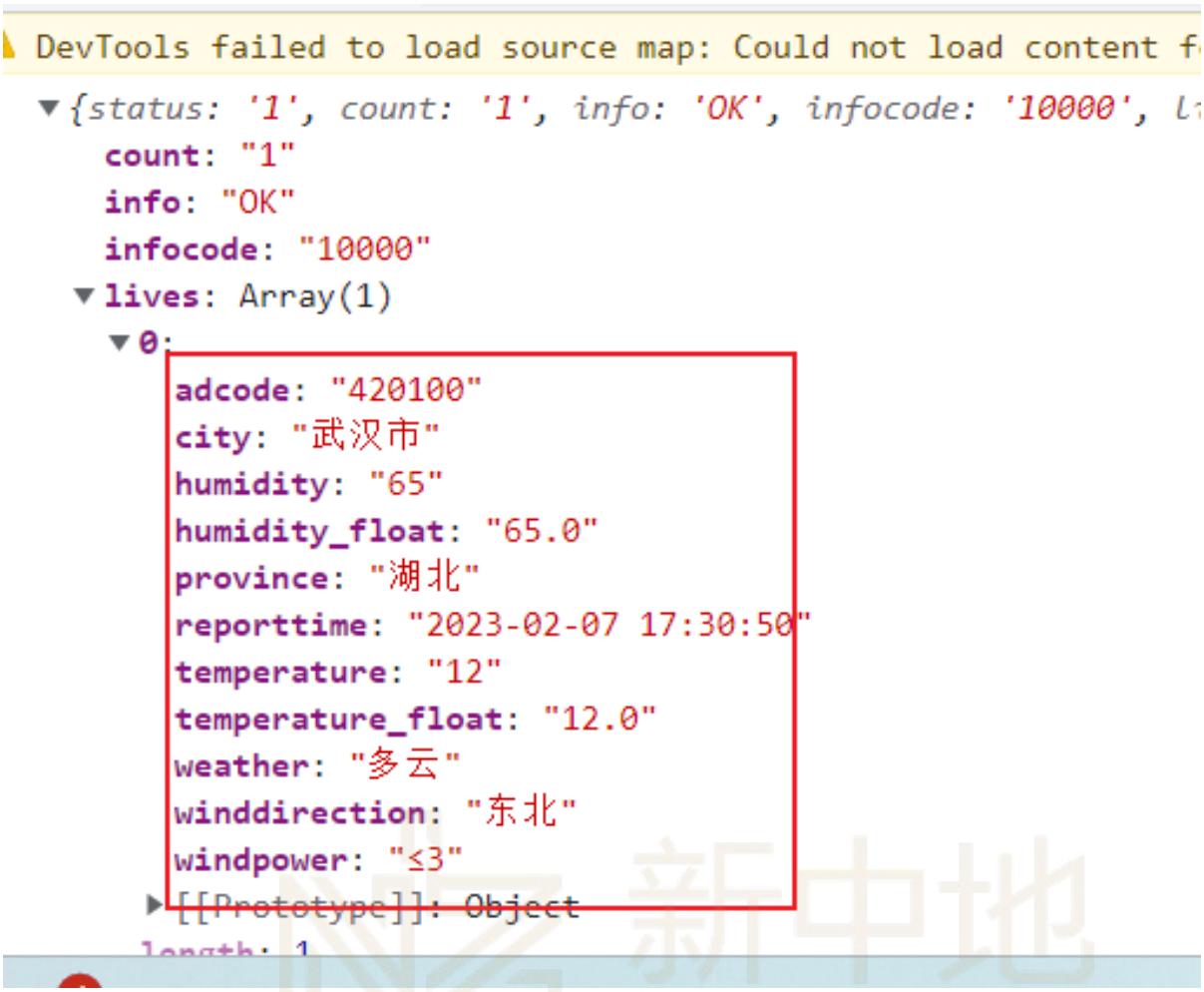
3、获取所在城市的天气数据

/weather/weatherInfo?city=武汉&key=${key}	天气查询

1、获取数据

https://animpen.com/pen/S8X5rN

```
$.ajax({
      url: `https://restapi.amap.com/v3/ip?key=${key}`
}).then(res => {
      const { city, rectangle, adcode } = res;
      var weather_url = `https://restapi.amap.com/v3/weather/weatherInfo?city=${city}&key=${key}
`;
      $.ajax({
            url: weather_url
      }).then(res => {
            console.log(res)
      })
})
```



2、渲染数据到页面

专注GIS开发　更多教程资料免费领取 请添加：gis2210


```
//2-1DOM
<div class="weather">
     <table class="weather_table">

      </table>
</div>
```

```
//
.weather {
  position: fixed;
  right: 20px;
  top: 70px;
  z-index: 100;
  border-radius: 15px;
  overflow: hidden;
  box-shadow: 0 0 3px 2px #333;
  background: #2193b0a1;
}

table,
td {
  border-collapse: collapse;

}

td {
  border: 1px solid rgba(238, 238, 238, 0.473);
}
```

```
table {
  color: #fff;
  text-align: center;
  width: 200px;
  line-height: 30px;
  border-radius: 15px;
}

//2-3DOM

$.ajax({
      url: `https://restapi.amap.com/v3/ip?key=${key}`
}).then(res => {
      const { city, rectangle, adcode } = res;
      var weather_url = `https://restapi.amap.com/v3/weather/weatherInfo?city=${city}&key=${key}
`;
      $.ajax({
            url: weather_url
      }).then(res => {
            console.log(res)
        //2-2
        var { city, weather, temperature, winddirection, windpower } = res.lives[0];
        var obj = {
              city: { val: city, name: "" },
              weather: { val: weather, name: "" },
              temperature: { val: temperature, name: "" },
              winddirection: { val: winddirection, name: "" },
              windpower: { val: windpower, name: "" }
        }
        //2-3DOM
        for(let key in obj){
               var template = `<tr>
                     <td>${obj[key].name} </td>
                     <td>${obj[key].val}</td>
                </tr>`
               $('.weather_table').append(template)
        }
      })
})
```

4、逆向地址解析

/geocode/geo?address=${city}&key=${key}	城市逆向地址解析

1、将城市写死，使用高德逆向地址解析获取当前城市的 中心点

2、执行飞行视角

```
//
...
let city =''
$.ajax({
```

```
      url: `https://restapi.amap.com/v3/geocode/geo?address=${city}&key=${key}`
}).then(res => {
      /*  */
      var center = res.geocodes[0].location.split(",");
      //
      map.getView().animate({
            center,
            zoom: 10
      })
})
```

1、复位地图

```
//DOM
<button class="reset"></button>
.reset {
      width: 200px;
      height: 50px;
      background-color: #6528e0;
      position: fixed;
      top: 30px;
      right: 30px;
      z-index: 100;
      color: #fff;
      border: none;
      border-radius: 10px;
}

.reset:hover {
      cursor: pointer;
}

const map = new ol.Map({
      target: "map",
      layers: [
            gaode
      ],
      view: new ol.View({
            center: [114.30, 30.50],
            zoom: 6,
            projection: 'EPSG:4326'
      })
})
//
const center = map.getView().getCenter();
//
$(".reset").click(() => {
      map.getView().animate({
            center,
            zoom: 6
      })
})
```

2、设置输入框，输入城市飞行到对应城市

https://animpen.com/pen/1wg_9q

第四章 Vue3.x整合ol

项目

项目01

项目开发流程

https://animpen.com/pen/uRPdzh

https://animpen.com/run/pen/uRPdzh?tid=TnF0Gg0J

参考代码

1、初始化地图

2、加载地图控件【视图跳转-放大缩小控件-比例尺-全屏控件】

3、调用高德API，根据ip获取所在的城市中心点 `rectangle` ， 城市名 `city` ， 邮编 `adcode`

4、调用turf库计算城市中心点 `center` ,将中心点使用 `canvas` 高亮显示

5、间隔5s飞行到中心点

6、将所在城市,根据邮编 `adcode` 使用 `geojson` 高亮显示

7、调用高德api,加载 `city` 的天气数据

8、复位地图

9、添加搜索功能，将搜索到的城市高亮显示

10、添加和风天气

11、移动端视配

https://widget.qweather.com/

项目02

https://share.weiyun.com/3pPavzag

专注GIS开发　更多教程资料免费领取 请添加：gis2210