- H5其他API
  - 离线缓存（应用程序缓存–Application Cache）
    - 通过创建 cache manifest 文件，可以轻松地创建 web 应用的离线版本
    - 离线缓存的三个好处
      - 1、离线浏览
      - 2、速度更快
        - 已缓存资源加载得更快
      - 3、减少服务器负载
        - 浏览器将只从服务器下载更新过或更改过的资源
      - 除了IE其他浏览器都支持
    - 离线缓存的原理
      - 1、发起请求,检测到需要加载的 js css 等文件标识
      - 2、首先会在本地检索是否有这些标识的文件存在,有的话,还需要对比本地的是否和网络端的一致,一致就直接读本地,返回状态码304;否则从网络端重新缓存,覆盖本地缓存
      - 3、本地没有，则从网络进行请求；存到本地
    - 搭建离线缓存的流程
      - 1、<html manifest=”xxx.manifest”>
      - 2、创建xxx.manifest文件，编写缓存代码
CACHE MANIFEST（缓存标识）
v 1.9

CACHE:　（需要缓存的文件css和js等文件）
./css/test.css
./js/test.js

NETWORK:（不需要缓存的文件，每次都需要从服务器获取的）
#*　　（标识除了以上文件以外，都是需要重新获取的）
./css/network.css （也可以指定某一些文件）

FALLBACK:（请求失败后返回的页面）
./404.html
      - 3、需要刷新页面两次 可以看到效果
  - Geolocation（浏览器地理位置）
    - H5的地理位置接口提供了一个可以准确知道浏览器用户当前位置的方法
    - 地理位置跟踪属于用户的隐私，可以通过浏览器设置是否同意共享地理位置。
    - 判断浏览器是否支持地理位置接口

```
if (navigator.geolocation) {
  // 想干嘛就干嘛
}
```

navigator.geolocation.getCurrentPosition(successCallback, errorCallback, options);

其中，successCallback为方法成功的回调，此参数必须；
errorCallback为方法失败时候的回调，此参数可选；
option参数为额外参数，也是可选参数，对象。
option参数支持三个可选参数API，
为：enableHighAccuracy, timeout, maximumAge.

1. enableHighAccuracy参数表示是否高精度可用，为Boolean类型，默认为false，如果开启，响应时间会变慢，同时，在手机设备上会用掉更多的流量，也就是money了。
2. timeout参数表示等待响应的最大时间，默认是0毫秒，表示无穷时间。
3. maximumAge表示应用程序的缓存时间。单位毫秒，默认是0，意味着每次请求都是立即去获取一个全新的对象内容。

通过getCurrentPosition和watchPosition这两个方法可以获取用户的地理位置，这两个方法参数一致，支持三个参数

当用户的位置被成功返回的时候，会返回在一个位置对象中，该对象包括一些属性，具体见下
coords.latitude　纬度数值
coords.longitude经度数值
coords.altitude　参考椭球之上的高度
coords.accuracy 精确度
coords.altitudeAccuracy　高度的精确度
coords.heading　设备正北顺时针前进的方位
coords.speed　设备外部环境的移动速度(m/s)
timestamp　当位置捕获到时的时间戳

当获取用户的位置返回失败的时候，会返回一个失败的对象，我们可以通过失败对象的状态吗来描述失败的原因。
通过error.code 的值来判断
1 : ’位置服务被拒绝’,
2 : ’获取不到位置信息’,
3 : ’获取信息超时’

getCurrentPosition方法属于一次性取用户的地理位置信息，而watchPosition方法则不停地取用户的地理位置信息，不停地更新用户的位置信息,watchPosition方法可以通过watchPosition方法停掉（停止不断更新用户地理位置信息），方法就是传递watchPosition方法返回的watchID了（类似于定时器的timerId）

  - webworker
    - 通过webworker可以创建一个子线程，我们会把耗时操作（下载等）放在子线程中去执行，把更新UI的代码放在主线程执行
    - 创建一个子线程

```
var worker =new Worker(”js/worker.js”);
```

在子线程的JS中，调用postMessage方法返回数据给主线程

    - 监听回调接受消息，更新UI代码

```
worker.onmessage =function(evt){
  console.log(evt.data);
}
```

    - 终止子线程的执行

```
worker.terminate()
```