Google Maps API

JS maps API
July 2, 2017

Basic

• Load google maps API:
• Getting started
    - Get API keys: cloud.google.com
• Initmap
    - function initMap() {}
    - - Initiates the map object and other maps api objects. Runs when loading the page
• Styles
    - var styles = []
    - - Styles array for google maps map object
    - - See *url* for styles reference
    - - See *url* for examples
• map
    - var map = new google.maps.Map(*html element*){
        • zoom: int,
        • center: {lat: num1, lng: num2},
        • styles: styles,
        • etc: see *url* for more options
    - }
    - - The map object is the object for the map in the dom
• InfoWindow
    - var InfoWindow = new google.maps.InfoWindow();
• making markers
    - 1. init arrays for locations and markers
        • var markers = [];
        • var locations = [{title: 'Some string', location:{lat:123, lng:321}}, ...]
    - 2. Iterate over locations and get the data from each loc
    - 3. Make marker for each loc:
        • var marker = new google.maps.Marker({position:*from loc*, title: *from loc*, icon: *google.maps.MarkerImage*, id: int})
    - 4. Push marker to markers array: markers.push(marker)
    - 5. Add listeners to marker object
    - 6. When you wish to display markerobjects: marker.setMap(map)
• bounds
    - 1. var bounds = new google.maps.LatLngBounds();
    - 2. iterate over locations -> bounds.extend(position)
    - 3. map.fitBounds(bounds)

Libraries

- Loading libs
  - **.../maps/api/js?libraries=geometry,drawing,places,visualization=3=XXX...**
- Geometry
  - Library for calculating geometric values on earth; eg. distance/area. Useful for doing earth calculations
  - Examples:
    - geometry.spherical.computeHeading(*userposition*, *locationposition*) -> Heading to location => use in streetview
    - geometry.poly.containsLocation(position,polygon) -> returns bool, useful for just displaying markers in a polygon
    - geometry.spherical.computeArea(*array of latlngs*) -> Calculates area
- Drawing
  - Graphical interface for users to draw polygons, tects, polylines, circles, and markers on a map
  - Use:
    - var drawingManager = google.maps.drawing.DrawingManager(){args}
      - see *url for arguments*
    - drawingManager.setMap(map)
    - drawingManager.addListener('overlaycomplete', function(event){polygon = event.overlay, ...})
    - Do stuff to the polygon object after this. Eg calculate area with geometry lib
- Places
  - Lib for searching for places within a defined area. eg points of interest
  - 1. var service = new google.maps.places.PlacesService(map);
  - 2. service.getDetails({placeId: marker.id}, function(place, status){})
  - 3 –> Check status === google.maps.places.PLacesServiceStatus.OK –> place.attribute
  - 4. Check that an attribute exist before accessing it
- Autocomplete
  - Part of places library
  - Provides autocomplete when searching for places
  - 1. var xAutocomplete = new google.maps.places.Autocomplete(domElement-input_text)
  - 2. xAutocomplete.bindTo(arg, map)
- Visualization
  - Provides heatmaps for visual representation of data

Geocoding

- Getting started
  - Comes with google maps js api
  - var geocoder = new google.maps.Geocoder();
  - geocoder.geocode({args:rejfhesgj}, function(result, status){})
    - if success => status === google.maps.Geocoder.status.OK
    - result => array of results as json
    - to see results => var str = JSON.stringify(results[0], null, 4)

Webservices	• **See developers.google.com/maps/web-services** • Distance matrix - GET .../maps/api/distancematrix/json?origins=*address*=*array of addresses seperated by ,* more options • Options: mode=transit_mode=*eg.train,driving,bicycling*_time=*secs since 1 jan 1970* • see /maps/documentation/distance-matrix • Directions - GET ../maps/api/directions/json?origin=*address*=*address* more options • Options: travelMode, waypoints, transit_routing_preferences, optimizeWaypoints • see: https://developers.google.com/maps/documentation/javascript/directions • Limits - https://developers.google.com/maps/documentation/javascript/directions#UsageLimits • Other useful apis - Nearby search • Gets list of nearby places from center and radius - Text search • Search without specified location by using text query - Radar • Like nearby search, except more places and less specified information • Timezone api - maps/api/timezone/json?location=lat,lng=1jan1970timesince=87ut6958069032495 • Geolocating - Useful for machines which doesnt have geolocation built in - Websevice to pass info from celltowers/wifinodes to get location
Roads API	• Avaliable only as a webservice • Snap to roads - Takes up to 100 points and snaps the points to a road - args: Interpolate returns smooths out the returned points • Speed limits - Rakes path(list og latlngs) or placeIds and returns speed limit + snapped points
Other API features	• API Use - See console to monitor api use - QPD = Queries per day - QPS = Queries per sec - In console you may limit the api use - There is a premium key: • You have to have premium on closed sites • You must have premium on paid services • JS documentation on - **https://developers.google.com/maps/documentation/javascript/tutorial** • List of various apis: - ...

Project req

- Interface
  - Responsive design, works across platforms
- Functionality
  - Geocoding and filters
  - Displays all desired markers by default and filtered subset when filter is applied (eg searching -> filtering)
  - Infowindow when markers are clicked
  - Listeners on markers
- App architecture
  - MVVM model
  - Use of Knockout JS
- API use
  - Use of google maps API and atleast one other API
  - AJAX requests that fail results in visible error
- Location details
  - At least 5 markers
  - Loads additional data from 3rd party api
  - Application runs without errors
  - Functionality is presented in usable and responsive manner
- Documentation
  - README file
  - Comments are present and explain longer code procedures
  - Code is good -> See JS styleguide
- Plan
  - devenv
    - NodeJS + gulp
    - atom
  - Features
    - MVVM with Knockout JS
    - München as the city
    - Use of the apis