# 

# Useful Tools for Web Mapping

## **Introduction**

Maps are nice - and sometimes necessary - for many websites. But creating beautiful, usable, accurate maps can be tricky. There is a thicket of concepts, tools and data sources to navigate.

Here’s our quick guide to some of the amazing tools are out there to help web developers work with spatial data.

As a note - here we’ll present resources roughly in line with the path spatial data takes from its origin to a user’s browser from the perspective of a full stack web developer: collect - manipulate - analyse - store - access - visualize. Also, this list is not exhaustive! Loads of great tools exist - this is more of a windscreen tour.

## **Collect**

Data comes from somewhere, and spatial data is no different. Exactly *how* spatial data is captured and created is beyond the scope of this post - all we need to know is that **raster images** and **vector features** can be downloaded or fetched from several reliable, authoritative sources.

[**OS Maps API**](https://osdatahub.os.uk/docs/wmts/overview). The Maps API serves detailed and scalable raster backdrop maps in four colour palettes - ‘Roads’, ‘Outdoor’, ‘Light’ and ‘Leisure’. The service is offered in Web Map Tile Service (WMTS) and ZXY protocols, and can be connected to maps built with Leaflet, Mapbox GL JS, OpenLayers, and other libraries.

[**OS Features API**](https://github.com/johnx25bd/os-blogs/blob/master/drafts/useful-tools-for-web-mapping). Our Features API lets users request detailed vector features along with rich attribution metadata in various formats. Data can be requested using HTTP requests via curl, JavaScript APIs (like fetch, d3.json, axios), Python tools like urllib.request and requests, etc.

[**OS Vector Tile API**](https://osdatahub.os.uk/docs/vts/overview). We serve vector tiles via our Vector Tile API, offering scalable, customisable and lightweight mapping data for users to visualise with various mapping libraries.

## **Data Prep**

Data prep is often a major step in creating a web app. Working with spatial data can be especially tricky, especially for developers less familiar. These tools can help speed up the job.

### **Format Conversion**

Often spatial (vector) data is downloaded from sources as shapefiles - but very often web applications and JavaScript libraries are designed to work with GeoJSON, an open standard that describes geographic features and non-spatial attributes. These tools enable conversion between various spatial data formats.

[**mapshaper.org**](https://mapshaper.org/). A handy in-browser tool for working with spatial data. Users upload .shp, GeoJSON, and other data formats, then can manipulate the attributes and geometries. The Export function lets users select the output format. Very useful for quick manipulations, especially with smaller datasets.

[**QGIS**](https://qgis.org/en/site/). QGIS is an open source desktop GIS (geographic information system) program. With Q, users can load, visualize, manipulate and export vector and raster data - including into GeoJSON and other formats.

[**GDAL**](https://gdal.org/). The Geospatial Data Abstraction Library really deserves to stand on its own - it is an incredibly powerful tool to work with both raster and vector data. Many geospatial tools are built on top of GDAL (including QGIS). With the library developers can manipulate spatial data in a very sophisticated way - but it is quite a technical tool to use.

[**toGeoJSON**](https://github.com/mapbox/togeojson). A quick JS library to convert KML and GPX to GeoJSON on the command line, in with Node.js or in the browser. From Mapbox.

[**TopoJSON**](https://github.com/topojson). Vector datasets can be quite large - making websites slower and the web developer’s life more difficult. TopoJSON helps by reducing the size of GeoJSON files by efficiently describing line segments (arcs) so the same lines don’t appear twice in the dataset. Note - to use TopoJSON you’ll need to also use the [TopoJSON client](https://github.com/topojson/topojson-client/blob/master/README.md#topo2geo), to convert back to GeoJSON.

## **Data Management**

Again, web developers tend to work with GeoJSON - so our focus will be managing these files.

[**GeoJSONLint**](http://geojsonlint.com/). For checking validity of GeoJSON objects.

[**GeoJSON-Validation**](https://www.npmjs.com/package/geojson-validation). A npm module to check validity of GeoJSON. Especially useful for validating user-uploaded files.

[**geojson-vt**](https://github.com/mapbox/geojson-vt). Create vector tiles from GeoJSON data efficiently on the fly.

## **Back End**

Spatial datasets require specialized databases to efficiently store and access. Most notably, spatial queries enable developers to access records based on some spatial dimension - selecting points that are contained within a polygon, for example, or lines that intersect another line.

[**PostGIS**](https://postgis.net/). PostgreSQL, with the PostGIS extension, is a commonly used relational database for spatial data. Well tested, large community, SQL.

[**SpatiaLite**](https://www.gaia-gis.it/fossil/libspatialite/index). Like PostGIS, SpatiaLite extends SQLite to support spatial queries.

[**Mapbox**](https://www.mapbox.com/). With Mapbox, users can upload spatial data, which is stored in a way that it can be served to a browser. With Mapbox Studio users can create custom map styles for their location data.

[**Carto**](https://carto.com). Carto is a location intelligence platform with tools to ingest, enrich, analyse, visualise and integrate spatial data.

[**MongoDB**](https://docs.mongodb.com/manual/geospatial-queries/). This NoSQL database supports geospatial queries.

[**GeoDjango**](https://docs.djangoproject.com/en/3.0/ref/contrib/gis/). For developers running a Django back end, GeoDjango extends the framework to work with geographic data. Designed to connect to a geographic database like PostGIS or SpatiaLite.

[**NodeJS**](https://nodejs.org/). Node seamlessly can work with spatial data, connecting to PostGIS and MongoDB instances. The runtime environment benefits from having access to the multitude of JavaScript libraries developed for working with location data.

## **Front End**

[**Leaflet**](https://leafletjs.com/). Leaflet is “a JavaScript library for interactive maps”. The library handles raster and vector tiles and enables web developers to customize styling and interactivity - on desktop and mobile devices. A standard for web mappers.

[**Mapbox GL JS**](https://docs.mapbox.com/mapbox-gl-js/api/). Mapbox GL JS lets web developers build customizable, interactive vector maps, rendered using WebGL. This gives developers the option the customize styling and offers a smooth, impressive user experience, including 3D effects. GL JS fits into the Mapbox ecosystem.

[**OpenLayers**](https://github.com/johnx25bd/os-blogs/blob/master/drafts/useful-tools-for-web-mapping). OpenLayers is another library for creating dynamic, interactive maps in the browser. The library handles both raster and vector tiles and can visualize spatial data from various formats, like GeoJSON, KML, GML and others.

[**ArcGIS API for JavaScript**](https://developers.arcgis.com/javascript/). With Esri’s ArcGIS API for JavaScript developers can build location experiences in 2D and 3D, using dynamic styling based on data.

[**d3.js**](https://d3js.org/). Data-Driven Documents (D3) is an incredibly powerful library for working with data in the browser. The library excels as a way to create interactive geographic maps and visualizations - supported by a large community and range of example code snippets.

[**Proj4js**](http://proj4js.org/). A very useful JavaScript library for transforming coordinates between coordinate systems, including datum transformations.

## **Data Analysis**

[**Turf.js**](https://turfjs.org/). Geospatial analysis in JavaScript. Turf provides a suite of functions to analyse vector geospatial features and work with coordinates and coordinate arrays.

[**geotiff.js**](https://geotiffjs.github.io/). A JavaScript library for parsing and visualizing TIFF (raster) files, including raw raster data.

[**Geoblaze**](http://geoblaze.io/). Extending geotiff.js, Geoblaze enables users to analyse and visualize raster data in the browser or in NodeJS.

[**Observable**](https://observablehq.com). Observable is a web-based notebook for exploring and visualizing data. Powerful - well worth a look.

## **Design / Cartography**

### **Colour Pickers**

[**Colorbrewer**](https://colorbrewer2.org/). Created by a cartographer who has extensively researched how to use colour on maps, Colorbrewer provides various colour palettes for map designers.

[**Adobe Color**](https://color.adobe.com/create). A tool to generate colour palettes, including various schemes and hex code outputs.

[**OS Colour Palette**](https://github.com/OrdnanceSurvey/GeoDataViz-Toolkit/tree/master/Colours). Ordnance Survey cartographers have created a colour scheme to be used on OS maps, made available in the OS GeoDataViz toolkit on Github.

### **Iconography**

[**Mapbox Maki**](https://labs.mapbox.com/maki-icons/). From Mapbox, Maki is a set of vector icons specifically for map designers - beautiful, with lots of icons you don’t find elsewhere.

[**Font Awesome**](https://fontawesome.com/). Another vector icon pack - with a free option.

–

Know of any more great tools for web mappers? Tweet at us [@OrdnanceSurvey](https://twitter.com/ordnancesurvey) or tag [#OSDeveloper](https://twitter.com/hashtag/osdeveloper)!