**Table of Contents**

[Table of Contents](http://cege0052.cege.ucl.ac.uk/cege0043tutorials/tutorials/week4/practical4-step4.html#toc0)

[Using JSDoc to Document Code](http://cege0052.cege.ucl.ac.uk/cege0043tutorials/tutorials/week4/practical4-step4.html#toc1)

[Step 1 - Running the JSDoc Command](http://cege0052.cege.ucl.ac.uk/cege0043tutorials/tutorials/week4/practical4-step4.html#toc2)

[Step 2 - Creating Documentation for a Simple Function](http://cege0052.cege.ucl.ac.uk/cege0043tutorials/tutorials/week4/practical4-step4.html#toc3)

[Step 3 - Creating Documentation for a Function with Parameters](http://cege0052.cege.ucl.ac.uk/cege0043tutorials/tutorials/week4/practical4-step4.html#toc4)

[Step 4 - Creating Documentation for a Function that Returns a Value](http://cege0052.cege.ucl.ac.uk/cege0043tutorials/tutorials/week4/practical4-step4.html#toc5)

[Step 5 - Creating Documentation via GitHub Actions (not recommended)](http://cege0052.cege.ucl.ac.uk/cege0043tutorials/tutorials/week4/practical4-step4.html#toc6)

**Using JSDoc to Document Code**

This practical will provide some hints and tips on how to use JSDoc to document your JavasScript functions.   The resulting documentation is part of the overall documentation of your codebase, which also includes any in-code comments, any comments you make when you COMMIT to GitHub and any documentation in the GitHub repository - e.g. a README file.

NB - the tasks below should be repeated for EVERY GitHub branch and every JavaScript function write, all through this module

**Step 1 - Running the JSDoc Command**

Some of the sample code in the repository already has JSDoc-formatted documentation comments.   You need to compile (create) this documentation by installing and using the JSDoc code.

1. In GitHub, create a branch in the APP repository called *documentation*

2. Log into your server, and clone your code

3. At the command line and use the cd command to change directory to your repository.  Depending on where you are storing your code, the command will be something like

cd /home/<<your CS username>>/code/cege0043-apps-23-24-<<your github user name goes here>>

4. Check to see if the *documentation* directory already exists in your repository, by using the *ls* command to show the files in the repository directory.

5. If the documentation directory does not exist, then create it as follows

mkdir documentation

6.Use the following command to install JSDoc

npm install jsdoc --save

7. Make sure that the jsdoc.conf.json file exists by typing ls to see the content of the directory

8. Run the following command to create the documentation

./node\_modules/jsdoc/jsdoc.js -c ./jsdoc.conf.json --verbose

This command uses the instructions we have placed in the jsdoc.conf.json file to compile the documentation.  Those instructions specifically tell JSDoc to only create documentation for files with a .js extension - i.e. only document the JavaScript files, and also to exclude any files in *node\_modules* (as otherwise we'd be documenting all the NodeJS libraries we are using)

7.   If the run is succesful, you will see messages as shown below

A screenshot of a computer screen

Description automatically generated

8.  The documentation can be accessed via a browser, by typing in the following URL

http://<<your CS computer name>>/app/documentation/index.html

9.  Documentation will appear as follows.  You can click on a function name on the list to see the detailed documentation for that function

A screenshot of a computer

Description automatically generated

10.  Every time you create documentation, new files will be added to the codebase.  Make sure you use git add -A ,  git commit -am "xxxx" and git push to save the files to GitHub.

**Step 2 - Creating Documentation for a Simple Function**

Use the @function and @description tags to create the required documentation.  This example also shows how you can use HTML tags (in this case <br> for a line break) to format the documentation

/\*\*

\* @function showLeaflet

\*

\* @description show the div with the cesium map loaded

\* <br>no close button at the top as the user can swap back to 2D from the menu

\*

\*/

function showLeaflet() {

showDiv('mapWrapper');

}

**Step 3 - Creating Documentation for a Function with Parameters**

The @param tag allows you to create specific references to the parameters that are required when calling a function).    I

It is important to specify the data type of the parameter as part of this doocumentation, using { } brackets.

There are a number of [standard parameter types](https://jsdoc.app/tags-type) in JSDoc that you can use - but the system also allows you to specifcy your own types as in this example.

/\*\*

\* @function

\* @param {GeoJSON} datasource - the geojson layer

\* @param {JSON} feature -the data with the style information. Includes the layer\_colour, the layer\_transparency

\* @description style the points added via geoJSON. Default outline colour of black is used.

\*/

function stylePoints(datasource,feature){

var entities = datasource.entities.values;

let fillColor = Cesium.Color.fromCssColorString(feature.properties.layer\_colour).withAlpha(feature.properties.layer\_transparency);

let outlineColor = Cesium.Color.fromCssColorString('black');

for (var i = 0; i < entities.length; i++) {

var entity = entities[i];

entity.billboard = undefined;

entity.point = new Cesium.PointGraphics({

color: fillColor,

outlineColor: outlineColor,

outlineWidth: 5,

pixelSize: 20

});

}

}

The above code also illustrates the flexibility of JSDoc - you don't need to specify the function name - the @function tag will tell JSDoc to get the function name from the next function declaration in the text.

**Step 4 - Creating Documentation for a Function that Returns a Value**

The @return parameter allows you to specify the data type and description of values returned by a function.

/\*\*

\* @function calculateDistance

\*

\* @description given coordinates and a unit, calculate the distance between the points

\* code adapted from https://www.htmlgoodies.com/beyond/javascript/calculate-the-distance-between-two-points-in-your-web-apps.html [accessed 24 Dec 2023]

\*

\* @param {double} lat1 - the latitude of the first point

\* @param {double} lon1 - the longitude of the first point

\* @param {double} lat2 - the latitude of the second point

\* @param {double} lon2 - the longitude of the second point

\* @param {String} unit - the unit for disance - by deafult the distance is in miles, using K gives kilometers, and N gives nautical mines

\*

\* @returns {number} dist - the distance between the provided coordinates

\*/

function calculateDistance(lat1, lon1, lat2, lon2, unit) {

let radlat1 = Math.PI \* lat1/180

let radlat2 = Math.PI \* lat2/180

let radlon1 = Math.PI \* lon1/180

let radlon2 = Math.PI \* lon2/180

let theta = lon1-lon2

let radtheta = Math.PI \* theta/180

let subAngle = Math.sin(radlat1) \* Math.sin(radlat2) + Math.cos(radlat1) \* Math.cos(radlat2) \* Math.cos(radtheta);

subAngle = Math.acos(subAngle)

subAngle = subAngle \* 180/Math.PI // convert the degree value returned by acos back to degrees from radians

let dist = (subAngle/360) \* 2 \* Math.PI \* 3956; // ((subtended angle in degrees)/360) \* 2 \* pi \* radius where radius is 3956 miles

if (unit=="K") { dist = dist \* 1.609344 } // convert miles to km

if (unit=="N") { dist = dist \* 0.8684 } // convert miles to nautical miles

return dist

}

**Step 5 - Creating Documentation via GitHub Actions (not recommended)**

If necessary, you can use the pre-configured GitHub Action to create the documentation once your code is pushed to GitHub.    To do this

2. Commit all your code files to GitHub and push them to the repository

2. Delete ALL the files from your server.

3. On the GitHub website, click ACTIONS and select the Documentation - JSDoc and Markdown action

A screenshot of a computer

Description automatically generated

4. Click Run Workflow to run the action

5. You will see new/updated documentation files in the documentation directory

NB: This approach to documentation creation is not recommended as it does not give you the option to check that the resulting documentation is correctly formatted and contains the required information, and also means that if you have not deleted all your code from your server, you could end up with server code that is not syncrhonised with the GitHub code (resuting in the need to pull code onto the server and merge it manually)