|  |
| --- |
| [Type the company name] |
| [Type the document title] |
| [Type the document subtitle] |
|  |
| **[Type the author name]** |
| **[Pick the date]** |

|  |
| --- |
| [Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.] |

Contents

[1. Introduction to the training establishment 3](#_Toc417174369)

[2. Training Experience 4](#_Toc417174370)

[2.2. Fast Track Training Project 5](#_Toc417174371)

[2.2.1. Project Description – IgViz (Interactive Generic Visualization Library) 5](#_Toc417174372)

[2.2.2. Getting Started – Inception Phase 5](#_Toc417174373)

[2.2.3. Packing up - Elaboration Phase 6](#_Toc417174374)

[2.2.4. Setting off - Construction Phase 8](#_Toc417174375)

[2.2.5. Winding Up – Transition phase 13](#_Toc417174376)

[2.3. Working with the product teams 15](#_Toc417174377)

[2.3.1. Project Idea - Analytics Dashboard 15](#_Toc417174378)

[2.3.2. Dashboard Architecture Design 15](#_Toc417174379)

[Figure 2.1 - d3js Logo 3](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165239)

[Figure 2.2 – Samples from d3js.org 3](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165240)

[Figure 2.3 – Modeled dataset sample 4](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165241)

[Figure 2.4 – Milestone plan for the fast-track training project 5](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165242)

[Figure 2.5 – JavaScript Code Segment of *chartConfig* Variable 6](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165243)

[Figure 2.6 – JavaScript Code Segment of *igviz.plot* function 6](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165244)

[Figure 2.7 – Line chart drawn using an unsorted data set 7](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165245)

[Figure 2.8 – JavaScript code segment for sorting a data set 7](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165246)

[Figure 2.9 – JavaScript code segment for appending path to data points 7](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165247)

[Figure 2.10 – Line chart with linear interpolation mode 8](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165248)

[Figure 2.11 – Line chart with cardinal interpolation mode 8](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165249)

[Figure 2.12 – Line chart with Step-before interpolation mode 8](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165250)

[Figure 2.13 – HTML menu for selecting 8](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165251)

[Figure 2.14 – JavaScript code segment for loading google geochart 9](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165252)

[Figure 2.15 – JavaScript code segment for drawing the map on an html div 9](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165253)

[Figure 2.16 – JavaScript code segment for setting options for the google chart 9](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165254)

[Figure 2.17 –Regions Chart example 10](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165255)

[Figure 2.18 –Markers Chart example 10](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165256)

[Figure 2.19 –Terrain Map example 11](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165257)

[Figure 2.20 –Miscellaneous chart types provided by IgViz 11](file:///D:\Academic\Internship%20Aca\Training-Report\Untitled%201.docx#_Toc417165258)

# Introduction to the training establishment

## Prologue - WSO2 Inc

WSO2 (Web Services Oxygen) is a 100% open source middleware company. As the name implies, WSO2 facilitates varied number of web services which provides enterprise solutions in the web space. Perhaps WSO2 is the only company that provides a comprehensive, open source ecosystem platform for this subject.



##### Figure 2.3 – WSO2 Company Logo

WSO2 has branches in US (Mountain View, CA and Bloomington, IN), UK (London), and Sri Lanka (Colombo 03), Mountain View branch being the Headquarters. Product development is mainly carried out here in Sri Lanka. WSO2 opened two new offices in Jaffna and Maradana Trace Expert City. They will be moving the office in Palm Grove, Colombo 03 anytime soon to the Trace branch.

## History of WSO2

Initially, the activities of the company focused on providing open source software solutions to the professional software developers. Currently, it focuses on development of software projects based on Service Oriented Architecture which is generally known as SOA.

WSO2 was found by Dr. Sanjiva Weerawarana and Paul Fremantle in August 2005. Initially, WSO2 was named as ‘Serendib System’, which later turned into ‘WSO2’ to address a request from an investor.

The initial investments offered by several volunteers paved a new path to the company, which made the company to widen and establish globally. With huge efforts and dedication, they managed to release the carbon platform which is known as ‘WSO2 Carbon’ by today. WSO2 Carbon is an SOA [*middleware*](http://en.wikipedia.org/wiki/Middleware) platform from WSO2. All the WSO2 products have been built on top of the WSO2 Carbon middleware platform.

The company which started with very few developers has now turned out to a massive software development company where more than 450 developers are employed. Opening new branches and hiring developers and trainees in an accelerated rate provides proof for the letter statement.

## Comparison

Almost all of the IT companies in Sri Lanka, and even in the whole South Asian region, focus on developing software outsourced by other non-IT companies. WSO2, in contrast, is a product development company, similar to the business model of Microsoft or Oracle in that aspect. Nevertheless WSO2 is 100% open source unlike the latter two. In fact WSO2 has only one enterprise ready version for every product, unlike most other open source companies. They have no hidden features or agenda.

## Products and Services

The main advantage, of all WSO2 products, is being built on a common basis - WSO2 Carbon, which is a modular, reconfigurable, flexible OSGi-oriented architecture. This creates a rock solid sound foundation for the construction of large-scale middleware applications and for integration with existing applications.

Currently WSO2 provides 19 products and one development environment plug-in for WSO2 product development, namely

* WSO2 Carbon
* WSO2 API Manager
* WSO2 Application Server
* WSO2 Business Activity Monitor
* WSO2 Business Process Server
* WSO2 Business Rules Server
* WSO2 Cloud Gateway
* WSO2 Complex Event Processor
* WSO2 Data Services Server
* WSO2 Elastic Load Balancer
* WSO2 Enterprise Mobility Manager
* WSO2 Enterprise Service Bus
* WSO2 Enterprise Store
* WSO2 Governance Registry
* WSO2 Identity Server
* WSO2 Message Broker
* WSO2 Storage Server
* WSO2 Task Server
* WSO2 User Engagement Server
* WSO2 Developer Studio

All of these products are free. So from where does all the money come from? That is one of the most FAQs asked from interested people. The answer is, via services. People with a lot of money, but less time, tend to buy WSO2 product related services opening a huge stream of income to the company.

## WSO2 Vision

WSO2s vision is based on *four pillars of innovation,* as they say. The idea is that they are trying to disrupt the competing vendors through a new approach, which is often described by four interesting statements.

### Reinvent the Technology

In my words, WSO2 is doing it right. The concept is that, if you do something from scratch, you can make a better version. This is not the same as *reinventing the wheel*. Middleware industry is still not perfect. Therefore doing the same thing from scratch enables WSO2 to find better approaches as well as avoid failure approaches. Final result is a model that fits perfectly to the need, while adapting to changes and opening doors for new and better technologies.

### Reinvent the Business Relationship

Most of the businesses in this world are based on the customers’ assets. The price of a product is usually decided looking at the buyers’ wallet. But this is not what WSO2 does. They have a fixed price for everything. If someone wants to buy their support, the price is fixed. Yet everything is free if the person is unable to pay the price. Still, the free version has all the advanced features as the paid version. This is one of the biggest challenges that WSO2s competitors face, disrupting the not so better structure.

Another point I realized while working for the company is that the company’s profit is not proportional to the number of workers. This is what enables them to expand faster and enable employees to enjoy better salaries and increments.

### Reinvent the Support Model

WSO2 does not have a separate set of support engineers. Support is given by the same people who develop stuff. This ensures that the person who comes for support knows all the underlying connections. WSO2 provides hot fixes, patches, and service packs to keep the installations running smoothly.

In my opinion, what WSO2 believe is that a satisfied customer is way better than a short term profit they might gain by having a cheap support system.

### Create a Great Place to Work

The internal structure of the company is very simple and has a flat hierarchy. A suggestion from the CEO has the same value as one from an intern. This allows all the good choices to get together in a pool from which the best one will pop out eventually. However there are job titles given based on the employees performance.

WSO2 does not force its employees to work until they get exhausted. According to Sanjeeva, the CEO of the company, employees must be there not only physically, but also mentally. Leaves are not recorded. Food, fun activities, leisure time activities and etc are provided without any boundaries. However, at the end of the day, employees do not waste time on just fun activities. They work with their maximum potential with a peace in their minds.

WSO2 makes the employees so good, that they become good enough to leave the company at anytime. On the same time WSO2 treat its employees even better, so they will stay and contribute for its further development.

# Training Experience

## Fast Track Training Project

After completing the *Fast-Track* training period successfully, we made up groups to proceed with our fast track training project. Me, Fawsan, and Fasna made up a group of three members and selected a project from the “WSO2-Redmine”. We scheduled a meeting with Mr. Srinath Perera to discuss about the project details. He inspired us with the project idea and gave the approval for us to continue working on the project.

### Project Description – IgViz (Interactive Generic Visualization Library)

WSO2-Products, despite of having very powerful and efficient functionalities, did not consist of a killer mechanism to visualize data. This could lead potential customers to underestimate the eminent productivity of their products. Therefore a generic web tool (a *widget* as we may address) for visualizing data, without expertise knowledge on the subject, was a vital requirement. Our task, therefore, was to create a generic JavaScript library which can be used by open-source developers to draw charts with minimal effort.

### Getting Started – Inception Phase

The library will be a simple JavaScript library from a high level point of view. However doing this without the aid of external libraries is discouraged due to 3 *mundane* facts,

* Limited time frame
* Having to *reinvent the wheel*
* Neglecting the best open-source practices.

Nevertheless we have to be extra careful when selecting external libraries for a software project. We must ensure that the resource is trustworthy and able to perform the expected task. Considering all above mentioned facts, we agreed upon using **d3js** to lay the foundation for our tool.

#### Why *we* chose d3js over other similar libraries?

The term 'd3' stands for *data driven documents*. It is a kind of low level JavaScript library for manipulating documents based on data. So to answer the question,

* It provides several reusable and advanced functionalities, which would have consumed a lot of time, if attempted to write from scratch.
* It has gained a good reputation over time, even from experts, for its strongly written code base.
* A lot of samples are available for new developers to grasp the d3 environment.
* Neatly written documentations are also available.



##### Figure 2.1 - d3js Logo



##### Figure 2.2 – Samples from d3js.org

During this time period, I mainly focused on learning the core technologies that will be used for this project. *The-NewBoston'*s free video tutorial series (www.thenewboston.com) on JavaScript was pretty helpful. Additionally I followed few more videos to learn the d3 basics for svg manipulation.

Adapting to the JavaScript development environment was not much of a challenge as it is a scripting language used for web development which does not require any additional configurations to be installed. The only requirements were a *text editor* and a *web browser*.

The plan we made during the **Inception phase** of the fast track project can be summarized as follows,

* Project Idea : Build a generic tool for chart visualization
* Proposed Core Technologies: JavaScript, d3js, WebStorm IDE, Sublime Text IDE.
* Expected Delivery Date: 23/01/2015
* Proposed Core Deliverable(s): A generic JavaScript library
* Version Controlling: git via github

### Packing up - Elaboration Phase

After the initial stage, we started designing the tool. Considering the requirement explained in an earlier discussion, we identified the need to implement the following chart types in the initial version of the tool.

* Single Number diagram: A chart which summarizes the data set into a single number representation.

In our case, we will show the average, maximum, and minimum value.

* **Line Chart Diagram:** A two dimensional chart which shows the change in one dimension against a

unit change in the other as a continuous line.

* Bubble Chart: A chart that displays three dimensions of data in a two dimensional plain.
* Table: A Straight forward representation of the complete data set with custom styles.
* **Map Diagram:** A diagram which shows data specific to a location on the world map.
* Bar Chart: A chart that represents the correlation of data of one discrete variable against another

discrete/continuous variable using rectangle bars.

We discussed and analyzed above mentioned chart types giving thought to how each would be implemented without losing consistency and extensibility. We had to come up with some mechanism to accomplish these as well as other common non functional requirements.

Therefore we planned to create a *JSON* skeleton to which a data set will be modeled into, before sending to draw a chart.

"dataTable": {

"metadata":{

"names":["Country","Area","GDP","Inflation","Life.expect","Military","Pop.growth"],

"types":['C', 'N', 'N', 'N', 'N', 'N', 'N','N']

},

"data": [

["Austria", 83871, 41600, 3.5, 79.91, 0.8, 0.03],

["Belgium", 30528, 37800, 3.5, 79.65, 1.3, 0.06],

["Bulgaria", 110879, 13800, 4.2, 73.84, 2.6, -0.8],

["Croatia", 56594, 18000, 2.3, 75.99, 2.39, -0.09],

["Czech Republic", 78867, 27100, 1.9, 77.38, 1.15, -0.13]

]

}

##### Figure 2.3 – Modeled dataset sample

Fig 2.3 shows how the ‘types’ section labels the columns defined by the ‘names’. ‘C’ stands for *categorical* and ‘N’ stands for *numerical*. This categorization will come in handy when we have to suggest suitable chart types according to the selected columns. The ‘data’ section contains a matching set of data to the metadata section defined earlier.

Considering the above design constraints and the complexity of each chart type, we divided the work among ourselves. Shown below is the milestone plan we created to present to the project stake holders.

#### Milestone Plan for the project

We prepared an on-line milestone plan for the project according to design considerations and sent it via email to Mr. Samissa for getting the approval. Shown below is the accepted milestone plan according t which we continued the project.



##### Figure 2.4 – Milestone plan for the fast-track training project

### Setting off - Construction Phase

We created a repository for the project in *git* as planned and each member forked a copy from the main repository. I was assigned earlier to implement the line chart and the map diagram. I started my task with the line chart diagram. However while working on our individual tasks, we had to implement the frame for the library to which individual charts would be *plugged in* after they are implemented.

#### Skeleton JavaScript file with the shared primitive functions

Seven functions were implemented in this file to accomplish the following sub tasks

* Plot Chart
* Create Scales
* Create Axes
* Configure Points
* Configure Point Labels
* Re Draw Clicked
* Create Form

If we examine the js functions written for above, the one for “Re-Draw-Clicked” plays a special role. This contains a variable called *chartConfigs* which defines the dimensions needed to draw the charts. The code segment for *chartConfigs* is shown below,

var chartConfig = {

"title": "Title",

"xLog": false,

"yLog": false,

"xAxisData": getValue('xAxis'),

"yAxisData": getValue('yAxis'),

"yAxis2Data": getValue('yAxis2'),

"yAxis3Data": getValue('yAxis3'),

"mapLocation": getValue('mapLocation'),

"pointColor": getValue('pointColor'),

"pointSize": getValue('pointSize'),

"pointLabel": 0,

"chartWidth": 600,

"chartHight": 400,

"padding": 60,

"chartType": targetChartId.replace("#", "")

}

##### Figure 2.5 – JavaScript Code Segment of *chartConfig* Variable

The plot function is implemented to call the correct function to plot a chart. The key parameter for selecting the chart type is extracted from the variable mentioned above which will be passed to the plot function as a parameter.

igViz.plot = function (divId, chartConfig) {

if ("scatter" == chartConfig.chartType) {

drawScatterPlot(divId, chartConfig, this.dataTable)

} else if ("bar" == chartConfig.chartType) {

drawBarChart(divId, chartConfig, this.dataTable)

} else if ("singleNumber" == chartConfig.chartType) {

drawSingleNumberDiagram(divId, chartConfig, this.dataTable)

} else if ("map" == chartConfig.chartType) {

drawMapDiagram(divId, chartConfig, this.dataTable)

} else if ("lineChart" == chartConfig.chartType) {

drawLineChart(divId, chartConfig, this.dataTable)

}

else {console.error("Unknown chart type " + chartConfig.chartType);return;}

}

##### Figure 2.6 – JavaScript Code Segment of *igviz.plot* function

#### Line Chart Diagram Implementation

The main concepts, sub tasks, and challenges I came across while this implementation phase can be listed as follows,

* Sorting the data set with respect to the X-axis coordinates
* Mapping coordinates according to a scale
* Connecting coordinates and *interpolation*
* Coloring each line chart
* Appending labels

Sorting the data set is a vital requirement. This makes sure that the coordinates are connected in correct order. Unlike in a situation where we would connect the dots with our hand, this causes the path to be a scribble if unsorted. Shown below is a case in which the X coordinates are not sorted properly.

****

##### Figure 2.7 – Line chart drawn using an unsorted data set

I solved this issue using the following handy function to sort the data set.

dataSet.sort(function (a, b) {

return a.data[xAxisID] - b.data[xAxisID];

});

##### Figure 2.8 – JavaScript code segment for sorting a data set

This *algorithm* sorts the data set *in place***.** The logic underneath is similar to *bubble sort*. Comparison is done internally by evaluating the sign of the subtraction. JavaScript language provides several such functions which can be used to replace complex code segments.

When everything else is setup correctly, the code segment for appending the path to the svg can be executed. Interpolation and line coloring can be done in the same code segment if the programmer knows how to play with his stuff. The code segment I wrote is shown below.

graph.append("path")

.attr("class", "line")

.attr("d", function (d) {

return line.interpolate(mode)(d.values);

})

.style("stroke", function (d, i) {

return getColor(i % 3)

});

##### Figure 2.9 – JavaScript code segment for appending path to data points

The interpolation *mode* in the above code segment can be one of linear, basis, step-before, step-after, cardinal or monotonous.

Shown below are chartsdrawn in 3 such interpolation modes



##### Figure 2.10 – Line chart with linear interpolation mode

##### Figure 2.11 – Line chart with cardinal interpolation mode



##### Figure 2.12 – Line chart with Step-before interpolation mode

I created a sub menu to select chart configurations (i.e. - columns, colors and interpolation mode) for drawing the chart. Shown below is a snapshot of the menu.



##### Figure 2.13 – HTML menu for selecting

**chart configurations**

#### Map Diagram Implementation

A major issue I faced while implementing this chart type was d3 not supporting advanced functions for map manipulation. The abstract map provided by d3 is very basic. The maximum possible zoom-level is country level. That means that users cannot visualize a data set local to a country like Sri Lanka.

Therefore I had to import another library to lay the foundation for the Map-diagram. After some research, I came across two possibilities

* Google visualization charts
* D3 based Data Maps

Considering the ease of use and the list of supporting functionalities, Google visualization charts library appeared to be a better candidate.

However, one major con of using google-charts was identified as the inability to draw the chart offline. But considering the features and the use case of the IgViz library, this trade off was evaluated as profitable.

**Important code quotations**

The LoadMap() function is responsible for fetching the chart definitions from google. This google.load(…) function can be seen as a common feature in all kinds of google charts. It should be noted that this does not load the map; it is responsible only for loading the chart definitions.

function LoadMap() {

google.load('visualization', '1', {'packages': ['map', 'geochart']});

}

##### Figure 2.14 – JavaScript code segment for loading google geochart

After loading the definitions, the following code segment will draw the map.

var chart = new google.visualization.GeoChart(document.getElementById('chart\_div'));

chart.draw(data, options);

##### Figure 2.15 – JavaScript code segment for drawing the map on an html div

In the above function, the ‘options’ parameter can be modified to set custom map configurations. Shown below is an example that I have set in the IgViz library.

var options = {

region: regionO,

displayMode: mode,

colorAxis: {colors: ['red', 'blue']},

magnifyingGlass: {enable: true, zoomFactor: 3.0},

enableRegionInteractivity: true

};

##### Figure 2.16 – JavaScript code segment for setting options for the google chart

In the above variable, the displayMode can be one of

* Regions Chart
* Markers Chart
* Terrain Map
* Normal Map

Regions Chart

This chart type is used when the data table contains countries in the location field. If this chart type is selected in a case where the location field contains cities, google will map the city name with a country and then draw the chart. Shown below is a *Regions Chart* generated from our IgViz library. The styles are set from the *options* variable (see Fig. 2.16).



##### Figure 2.17 – Regions Chart example

When this chart type is selected, the complete world map will be shown by default. To zoom to a specific region, the region number should be selected from the *ISO 3166 Country Codes* list.

E.g. - 150: Europe, SL: Sri Lanka, US: United States

Markers Chart

This chart type can be drawn using any location on the google map as the location field. The concept is similar to a bubble chart in which we plot 3 dimensions on a two dimensional chart. The three dimensions are identified by location, point color and point size.

I have implemented this chart type in such a way that the user does not have to select the region exclusively. The region covering all the points will be selected internally.



##### Figure 2.18 – Markers Chart example

Terrain Map

This is a map rather than a chart. I have added this type for users to visualize their data set on a google map. This enables the users to use any of the general features (panning, zooming, etc) provided by the world famous google maps. Terrain Map and Normal Map are quite similar. The only difference is that normal maps do not consist of the terrains.

##### Figure 2.19 – Terrain Map example

### Winding Up – Transition phase

In this phase we merged our individual tasks into a single JavaScript file. We also created an html page to show samples of what we have accomplished so far. Types of charts provided by the library, in addition to the ones I created, are shown below.







##### Figure 2.20 – Miscellaneous chart types provided by IgViz

**Product Architecture**

The architecture of the IgViz library we implemented can be illustrated as shown in the diagram below.

IgViz Library

D3js

Google Charts

Java Script

##### Figure 2.20 – Architecture of IgViz Library

## Working with the product teams

After completing the fast track project, I was assigned to the *WSO2-Business-Analytics-Monitor* (BAM) team. Then I attended the small introductory session on which we were welcomed and advised on how to work during the coming weeks.

**Team**: WSO2 Business Analytics Monitor (BAM)

**Team Lead**: Anjana Fernando (Senior Tech Lead)

**Assigned Supervisor(s**): Anjana Fernando & Dunith Dhanushka(Senior Software Engineer)

**Acquaintances**: Tharindu Munasinghe & Mohommed Fawsan

### Project Idea - Analytics Dashboard

As advised, I started working on my first project as a BAM team member. The task was to design and implement a dashboard for BAM and *CEP* (WSO2 Complex Event Processor). The existing dashboard was complex and had numerous bugs. So the idea was to create a minimalistic dashboard targeting the two products, BAM and CEP. The dashboard should provide the following functionalities

* Login with credentials
* Create widgets
* Create new sub dashboards
* Add widgets to custom dashboards and display
* Logout

**Project Name**: Analytics Dashboard

**Deadline**: Flexible

**Project** **Team**: Dunith Dhanushka, Pubudu Dodanoda(Me), Mohommed Fawsan

### Dashboard Architecture Design

We took the following design decisions at the start of this project. These decisions got the approval from the WSO2 Architecture mail thread.

Frontend

Front end of the dashboard will be designed using the following technologies

* HTML/CSS
* JavaScript
* Bootstrap Framework
* GridsterJS
* jQuery

Backend Design and Hosting

The backend of the dashboard will be a Jax-RS web service. This means that there will be a RESTful API to which the frontend developer can send requests for data communication. This will be built into web archive file (.war) using maven and will be deployed on the WSO2 BAM server as a web-app.

Other technologies such as Jackson, gson, and JaxB may also be used.

Work Allocation

I was assigned to implement the backend while Dunith and Fawsan will proceed with the frontend development.

# Annexes

Annex. 1: WSO 2 Products Overview

Terms used: javascript , d3js, svg, version controlling, git, github, webstorm, sublime text, WSo2, IDE, text editor, web browser, widget,json,igViz, bubble sort, sort, algorithm, google, html, BAM

Gridster, bootstrap, Ajax, Rest, jax-rs, api, web service, maven, war, wso2 server, web app, Jackson, JaxB, gson, carbon, middleware,SOA,