EXPEDITION JSQL

JSON Query Language

EXPEDITION JSQL

JSON Query Language

Mission Crew

Akhilesh Mantripragada
Commander a.k.a Project Manager

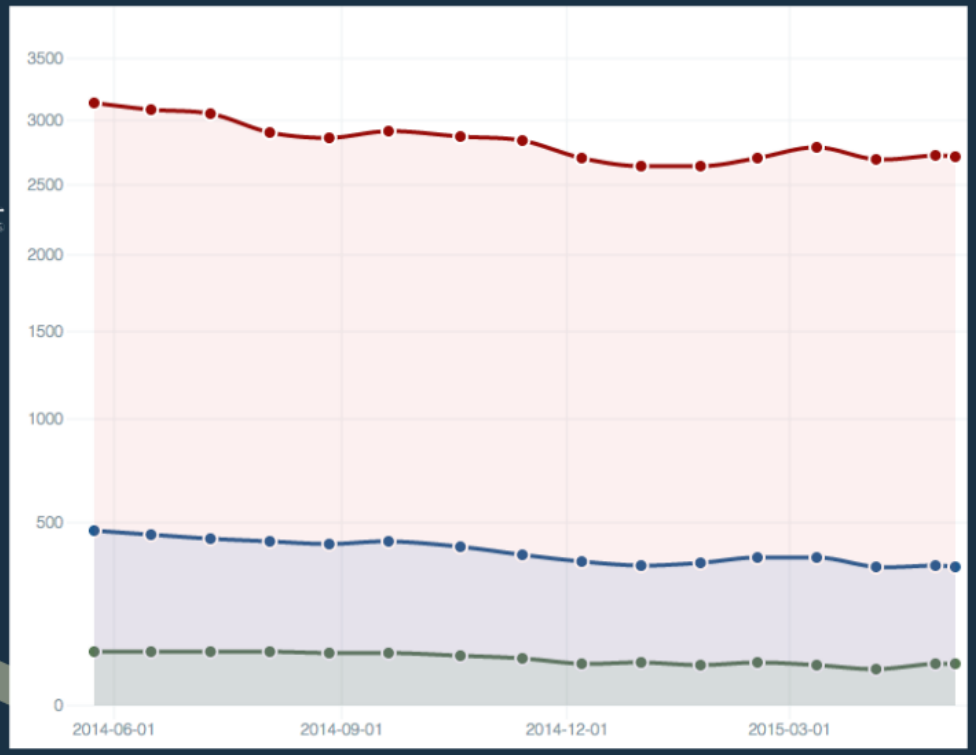
Seth Mishan
Mission Specialist a.k.a Language Guru

Abhyuday Polineni
Flight Specialist a.k.a System Architect

Rahul Gaur
Flight Engineer a.k.a System Integrator

Natasha Kenkre
X a.k.a Tester

JSON's Popularity



JSQL Motivation

Get a JSON Object from a file or URL

```
JsonList jsonArray = readJsonListFromFile(“path/to/file.txt”)
```

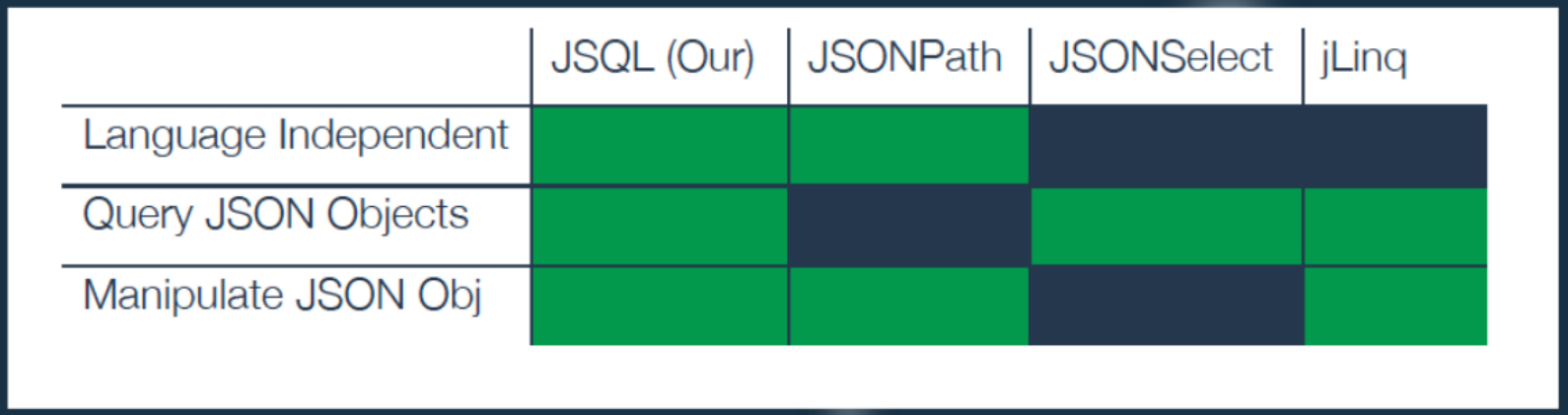
Being able to Query a JSON

```
JsonList selectFirstName = jsonList.Select("FirstName");
JsonList jsonList = jsonList.Where("FirstName = Seth");
```

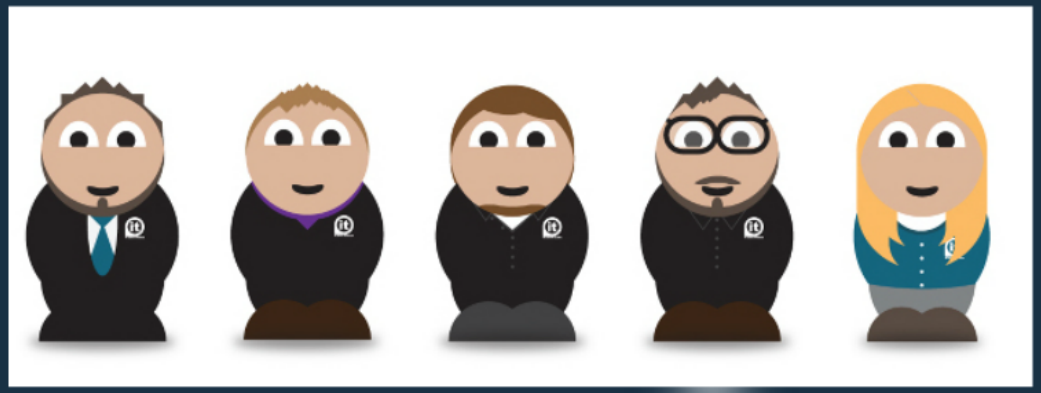
Being able to manipulate JSONs

Union, Intersection, Add, Subtract

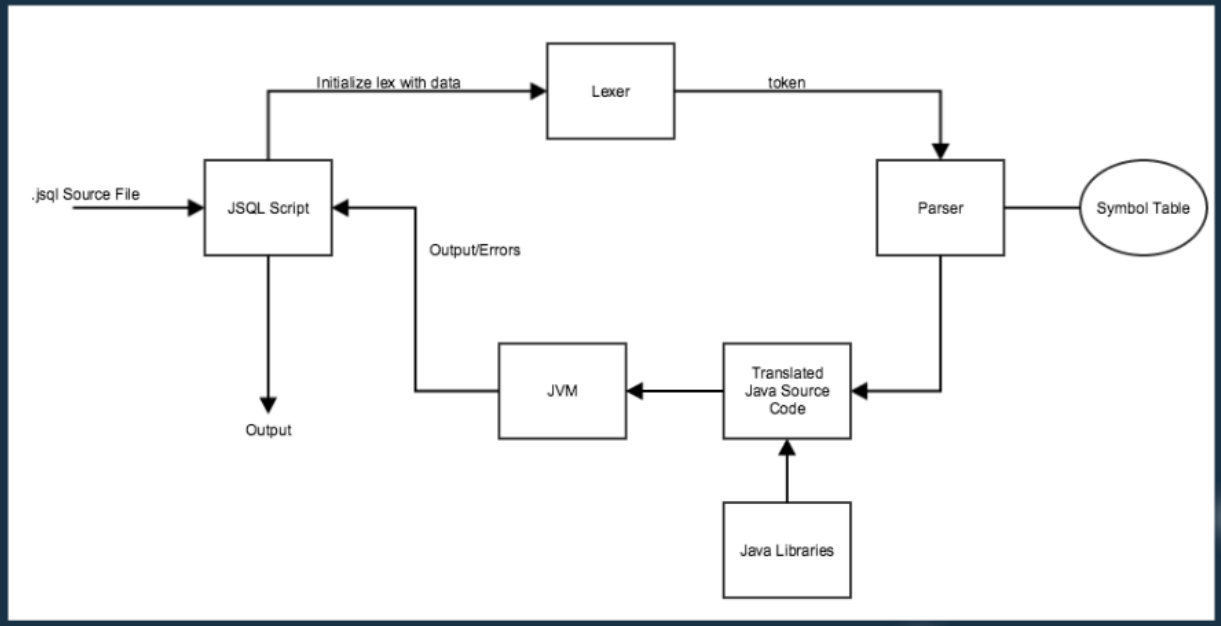
Existing JSON Related Languages



Project Management



ARCHITECTURE






Syntactic Construct

JSON Operators

<> Union
>< Intersection
<< Addition
>> Subtraction

JSON Types

Number
Boolean
Json
Json List
String

Basic Operators

+ Addition
- Subtraction
* Multiplication
/ Division
> Greater than
< Less than
% Modulo

Example

Adds fields from object Account 2 to Account 1 that do not exist in object Account 1

account1 = { "Name": { "First":"Abhyuday", "Last":"Polineni" }, "Username": "Abhi", “Password": “password”, "Age" : 25, "Courses": ["CV", “PLT"] }
account2 = { "Name": { "First":"Abhyuday", "Last":"Polineni" }, "Username": "Abhyuday", “Password": “password2”, “City” : “Hyderabad”}

account1 << account2 = { "Name": { "First":"Abhyuday", "Last":"Polineni" },
"Username": "Abhi", “Password": “password”, "Age" :25, "Courses": ["CV", “PLT"] , “City”: “Hyderabad”}

Syntactic Construct

JSON LIST Queries

SELECT	JsonList selectFirstName = jsonList.Select("FirstName");
ORDER BY	JsonList sortAsc = jsonList.OrderBy("FirstName", “Ascending”);
WHERE	JsonList jsonList = jsonList.Where("FirstName = Seth");
LIMIT	JsonList limitArray = jsonList.Limit(5);

Translation

JSQL

```
void main(){
        json obj = “{ “message” = “hello world” }”;

        Number a = 1;
        Number b = 2;
        print a+b;

        if (a>1) {
                print "Print to console";
        }

}
```

JAVA

```
Class JSQLClass{
        public static void main(String args[]){
                JsonWorker obj = new
                JsonWorker("{"message" = "hello
                world"}");

                double a = 1;
                double b = 2;
                System.out.println(a+b);

                if( a > 1) {
                        System.out.println("Print to
                        console");
                }
```

JAVA IMPLEMENTATION

-Java was used as a target language whose compiler we relied on instead of building our own
-Wrote classes in Java which simulated the functionality of our language's objects
-Relied on Java in-part for error-handling
-Json-Simple is a Java library that we used to generate json objects and parse raw strings into jsons

DEV & RUN TIME ENVIRONMENT

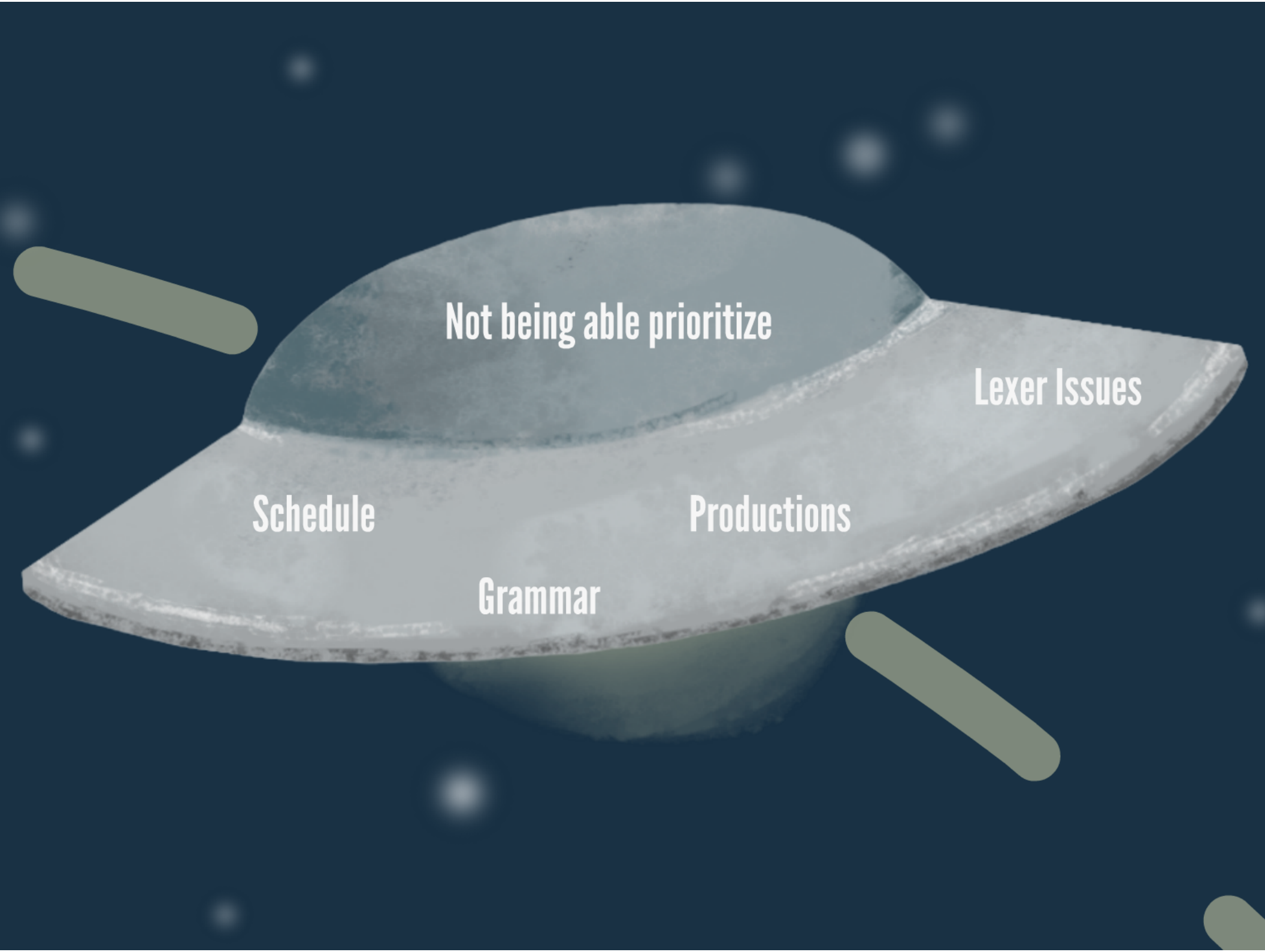
- Python Script
- Compiling JAVA Program
- Error Handling

TESTING

Testing Philosophy
- Robustness
- What could go wrong

Tests
- Operators
- JSON Operations
- In-Built Functions
- Errors
- Use Case Testing

Testing Script
- Compare pre-populated results with output


Not being able prioritize
Lexer Issues
Schedule
Productions
Grammar


JSQL
Demo

Future Explorations

- Get JSON from URL
- Distributed Query Support
- Regex Query Support

Lesson Learned

- Start Early
- Small Goals, Big Vision
- Prioritize
- Test First Development?

EXPEDITION JSQL

JSON Query Language