# Text Data Formats for Developers


| | |
|---|---|
| **Author** | josh |
| **Date** | 2023-07-01 |


# Overview

Many software development projects require reading, writing, or interacting with data in some way, beyond simple programming logic. Perhaps a coder wants to share information about a particular hobby or area of interest. How to programmers store data in a way they can easily interact with? Data for computer programs needs to not only be human readable (to some extent), but needs to be easy for code to parse, search, and display. For large or complex data sets, it's best to use a full-fledged data engine such as MySQL (relation) or MongoDB (document/key-value based). However, in many cases a simple text file will do. Let's learn about common text data formats.

# Common Formats

## CSV (Simple Tabular Data)

One of the most simplest ways a developer can store text data is in a tabular (table) format. One of the most common formats for this is *CSV* or "Comma-Separated Value" files. These are simple tables of information stored with headers that indicate the column types, and individual rows. Each row separates its columns with a *delimiter*, a special character that indicates to the code that we're underneath the next header. Headers aren't explicitly required in the file if our code knows what each column means.

For example, let's say we want to store a jiu jitsu training log. Our columns include the date, location, and notes on any drills and sparring that we did. Our header row includes information about each column. Each subsequent row stores the data from each training session. In between each row, we have a comma.

```
Date,Location,Drills,Rolling
6/22/2023,True Believer,…,…
6/23/2023,True Believer,…,...
```

CSV files are excellent for storing simple tables. We can use any logical character such as a comma, tab, or something else to indicate the individual columns. Each newline indicates a new row in the table.

## JSON (Key, Value)

Sometimes, our data set is a bit more complex than a simple set of rows and columns. One way we can store data with more flexibility is using a key-value data format. *JSON* or "JavaScript Object Notation" is a common form developers use. With a key value store, we have the flexibility to define a bit of data with just a simple key and value, a key and a list of values, or a key and a nested set of dictionary values (objects).

Let's say for example we want to store a database with resistance training information. We have a set for bodyweight progressions and free weight movements. We can store the name of the exercise and a list of progressions (easier to harder), or the exercise category and a list of variations.

```
 "name" : "Dips",
  "progressions" : [ "Supported Dips", "Dips", "Ring Dips" ] }
```

and

```
{ "category" : "Lower Back",
  "movements" : [ "Deadlift" ]}
```

There are many possible ways to structure this data, but this gives an idea of what's possible with key-value data formats. Another such format is YAML (Yet Another Markup Language). The advantage of key-value formats such as JSON is that they allow greater flexibility in how we structure our data, while still maintaining a compact format.

## XML/HTML (Markup Languages)

Another form of text format that allows for great flexibility is something like *XML or HTML* (Markup languages). These formats are more complex to parse, but allow robust complexity in how the data is stored. HTML is the language of webpages, and contains "tags" specifying what the content should look like on the page, along with the content itself.

Let's say for example we want to store a formatted table of rock climbing grades and how they compare across systems. We will have tags that specify this is a table (<table>), the table rows (<table>), and the header and data sections (<th> and <td>).

```
<table>
     <tr><th>Yosemite</th><th>Vscale</th></tr>
     <tr><td>5.9</td><td>V0</td></tr>
</table>
```

The advantage of languages like XML and HTML is that they support complex content well (for example, webpages!) A disadvantage, however, is that this format is more difficult to parse and often takes up more storage space.

## Data and Development

This is just a short introduction to text data formats. There are a lot of ways to store information in text format, whether it be mostly words or even numerical data. These formats allow a balance between human readability, performance, and expressiveness. For other types of data, full fledged database engines or even binary formats may be better. Whichever format you choose will depend on factors such as complexity, storage space requirements, and ease of use. Different projects will have different requirements, so balancing out those factors will help determine the best choice. Happy coding!