On HashMaps and Implementations

Use case

Hashes and Hashing functions

A hash function f maps a given input: $f(i) \rightarrow h$, i to a hash h. $f(i)$ has to always compute to h for the same i, otherwise the map would store values with the same key at different locations. To keep map access $O(1)$ and map insert $O(n)$, the hash function computes to an integer. This integer is then used to index into an the underlying array, instead of iterating a list or an array until the desired key is found.
Lets take a look at some common hashing applications: Java hashes strings by summing the characters of the string, while each is xored with the length minus the index of the character [1].

```
var s = "Hello World";
var h = 0;
for (int i = 0; i < s.length(); i++) {
    h += s.codePointAt(i) * 31
        ^ (s.length() - (i+1));
}
```

We will use a similar but better algorithm for hashing our key strings: fnv1-a.

Performance and the load factor

Dealing with Collisions

Naiive Implementation

[1] String.hashCode()