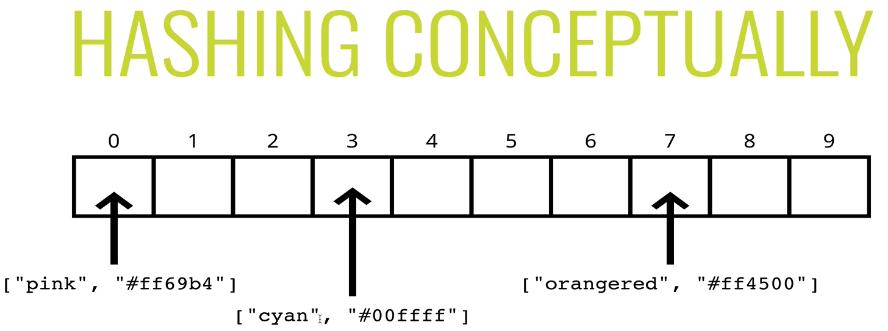
JavaScript Algorithms and Data Structures Masterclass

# Section 25: Hash Tables/Map

## What is a Hash Table?

* Used to store *key-value* pairs which are unordered
  + Very fast at **finding** values, **adding** values, **removing** values

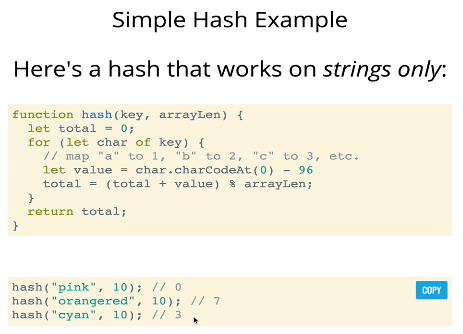
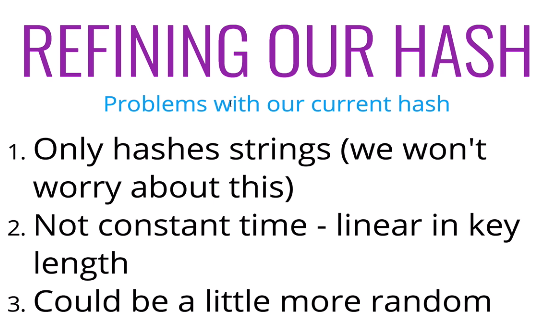
## Hash Functions

* These functions help us to convert keys into valid array indices which allows us to look up values based on their key
* For now, we will use an array to model a Hash Table
* 
  + Index0 could link to “pink” (key) ... in this case, the ‘key’ is linked to an array index

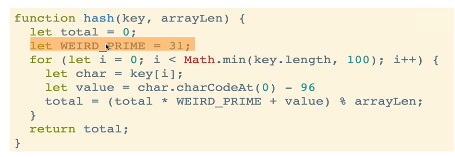
### Implementation of Hash Function

* *Characteristic of Good Hash Function*
  + Fast (i.e. constant time)
  + Doesn’t cluster outputs at specific indices, but distributes uniformly
  + Deterministic (same input => same output)

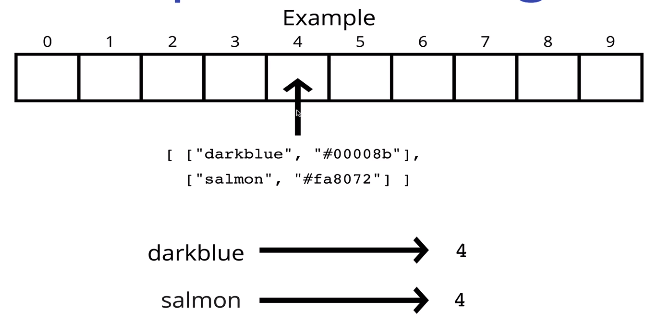
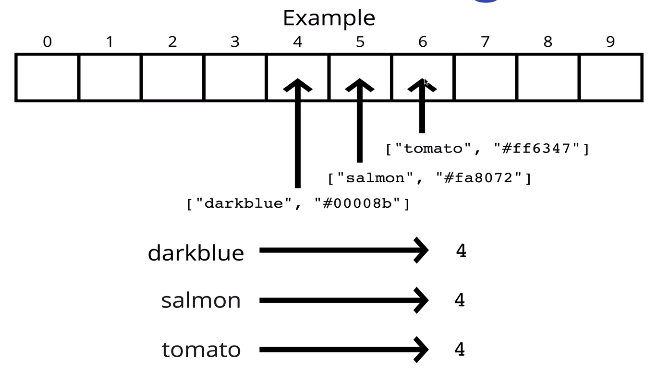
#### Basic Hash

* 
* **Problems with Simple Hash Function**
  + 

#### Improved Hash Function

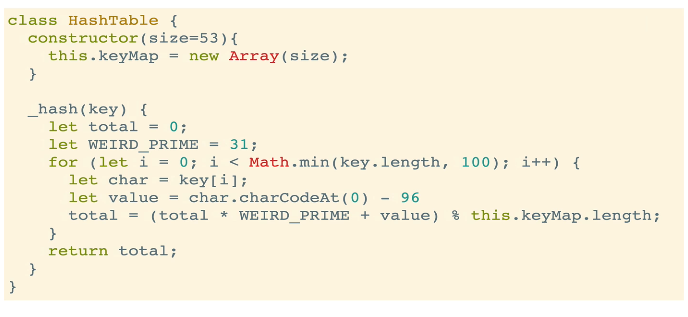
* 
* We use Prime Numbers in Hash Functions to spread out the keys more uniformly

## Handling Collisions:

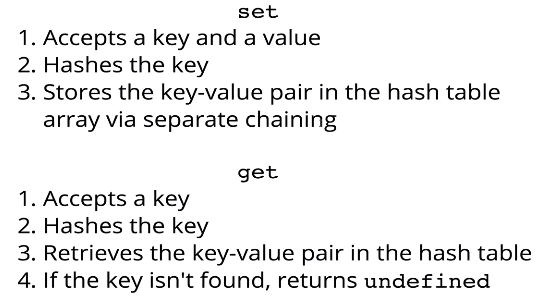
* **Separate Chaining**
  + If there is a ‘collision’ (when 2 values return the same index), we can store them together (**ex**. nested array)
    - 
* **Linear Probing**
  + IF there is a ‘collision’, we search through the array to the next empty index (ensures single key-value pairs at each index)
    - 

## Implement a Hash Table

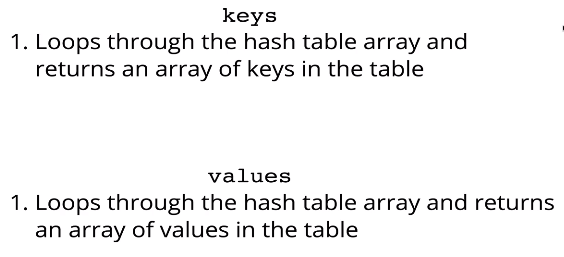
### Hash Table Class

* 

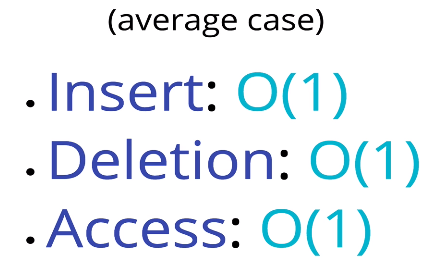
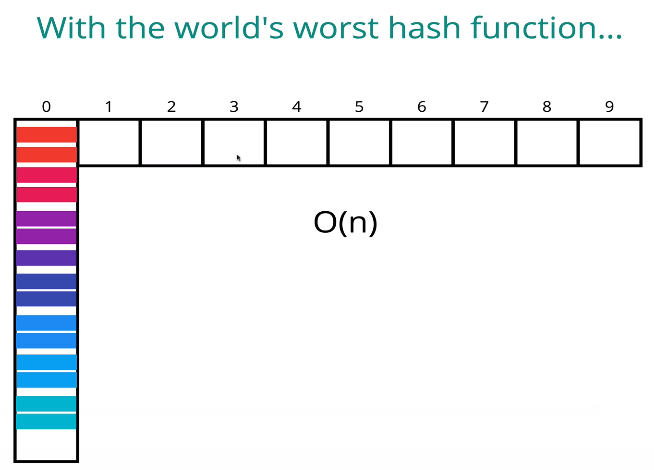
### Set/Get Functions

* 

### GetKeys/GetValues Functions

* 
  + Usually you only return unique keys/values

## Big O of Hash Tables

* 
* **Worst Case**
  + 

## Recap:

* 