Understanding hash functions

Selecting transcript lines in this section will navigate to timestamp in the video

- Most associative arrays, whether they are dictionaries, maps, or hashes, are implemented using a hash table. But before we get into that, we need to understand what hashing is. Essentially, hashing is a way of taking some raw data and mixing it together to form a smaller single piece of data. This is an over-simplified analogy. But think about cooking some hash browns. In order to get the hash browns to just right, you'll need to chop up some ingredients or raw data and combine it in order to make the dish just right. In this case, the finished hash browns are the hash. But even if one ingredient is wrong then the hash would taste differently, or in our case, be a different value. We call the process of inputting the raw ingredients to produce the final hash value a hash function. The raw data goes through a hash function, or, in our analogy, the cooking process, in order to produce that final hash. It's important to remember that these hash browns, and this final hash value are a simplified reference generated from the original data, or in our analogy, ingredients. In code, we might input characters, objects, or numbers into a hash function. And the output would be an integer. Now, what's so great about hash functions? Well, almost without exception hash functions are not reversible. They are one way. This means you cannot feed the hash value into another function and get the original data back. And this is intentional. It would be like turning your hash browns into the previous ingredients. So now that we know how hashing works on a fundamental level, why would you use it? Well, let's say a bank has a database of a bunch of usernames and passwords. Then a hacker comes along and somehow gains access to the database and has access to this information. Obviously this would be very bad if the passwords were stored in plain text because now the hacker would have direct access to everyone's login information. However, as a security measure, you could store a hash of the passwords instead. Then, when someone logs in, you could put their password, the series of characters, through the hash function and see if the resulting hash value matches the hash value you have stored in the database. This is why when you try to reset your password, most websites won't send your current password through an email, but instead give you a link to reset your password. Now how does this hash function work behind the scenes? Well, imagine if a user had the password twentytwoever. A very simple hash function might take the decimal ASCII value of each character and sum them up to produce an integer. If you haven't heard the term ASCII before, it's a format for text files where each character has a single numerical representation. For example, T is represented with 116, W with 119, and so forth. In this case, the hash function would take twentytwoever, convert each letter to it's decimal ASCII value, sum them up, and output the final sum, 1463, which would be our hash value. Keep in mind that this is very simplistic and most hash functions are more complicated. With this example you might be thinking that there is more than one string that would produce the hash value 1463. And you're right. If we ordered these characters in any other way, we would still get the same hash value making this incredibly insecure. When two different inputs map to the same hash value or produce the same hash value when thrown into a hash function, we call it a collision. However, with the hash function, the same input will always produce the same output. Meaning that a single password or input will only have one hash value. You also might be thinking that this seems like encryption. But it isn't. With encryption, there's a way to decrypt the encrypted item. There is no way to unhash a hash value. In programming, we often use hash values as a way to get to or store a value at a certain location. And it's important to understand how that works as well.