Code Nation

{CODENATION}™

What's the problem with this?

```
{
    "_id" : ObjectId("5c73de12d707f623115d0313"),
    "name" : "your name",
    "email" : "yourName@mail.com",
    "password" : "mysupersecretpassword",
    "__v" : 0
}
```

Storing passwords securely

Learning outcomes:

understand how to store passwords securely using bcrypt

Hashing and encryption

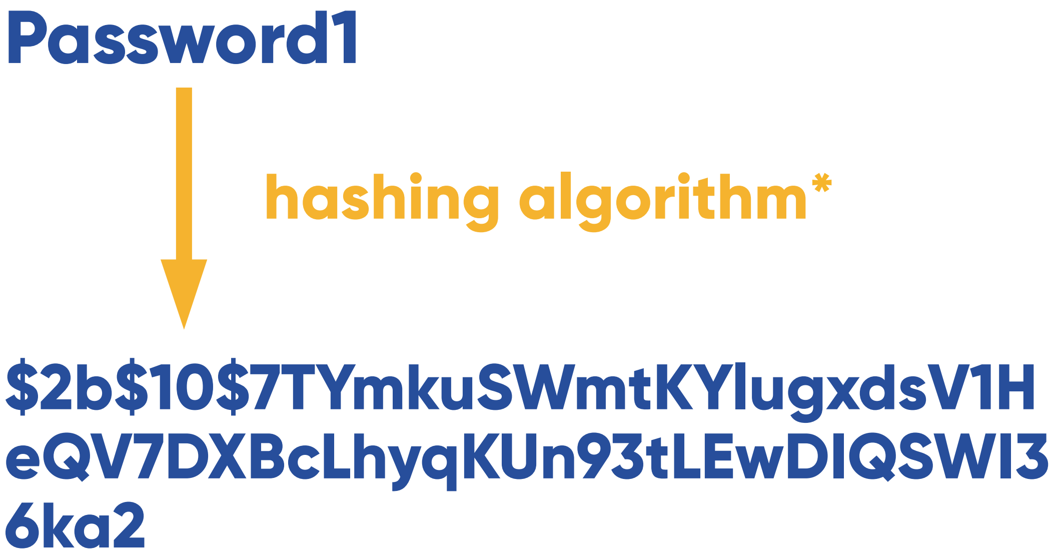
Designed to be compared

Designed to be reversed

What do we mean by this?

Which would be better for our use case? Why?

Hashing process



Password1

hashing algorithm*

$2b$10$7TYmkuSWmtKYlugxdsV1HeQV7DXBcLhyqKUn93tLEwDIQSWI36ka2

* more on the bcrypt algo: https://auth0.com/blog/hashing-in-action-understanding-bcrypt/

Lets break it down

cost/rounds

$2b$10$7TYmkuSWmtKYlugxdsV1HeQV7DXBcLhyqKUn93tLEwDIQSWI36ka2

prefix

salt & hashed password

prefixes

- Specify the algorithm for the hash
- bcrypt got $2$
- (a, b, x, y) added after the 2 for different versions*

*More detailed explanation of the versions: https://stackoverflow.com/questions/15733196/where-2x-prefix-are-used-in-bcrypt

Other prefixes

- **1 - MD5**
- **5 - SHA 256**
- **6 - SHA 512**

Cost/rounds

- The higher the cost, the more hashing rounds* are done
- Number of hashing rounds = 2^{10} by default ($2b$10$...)
- Makes the hash harder to brute force

*key derivation function: https://en.wikipedia.org/wiki/Key_derivation_functionz

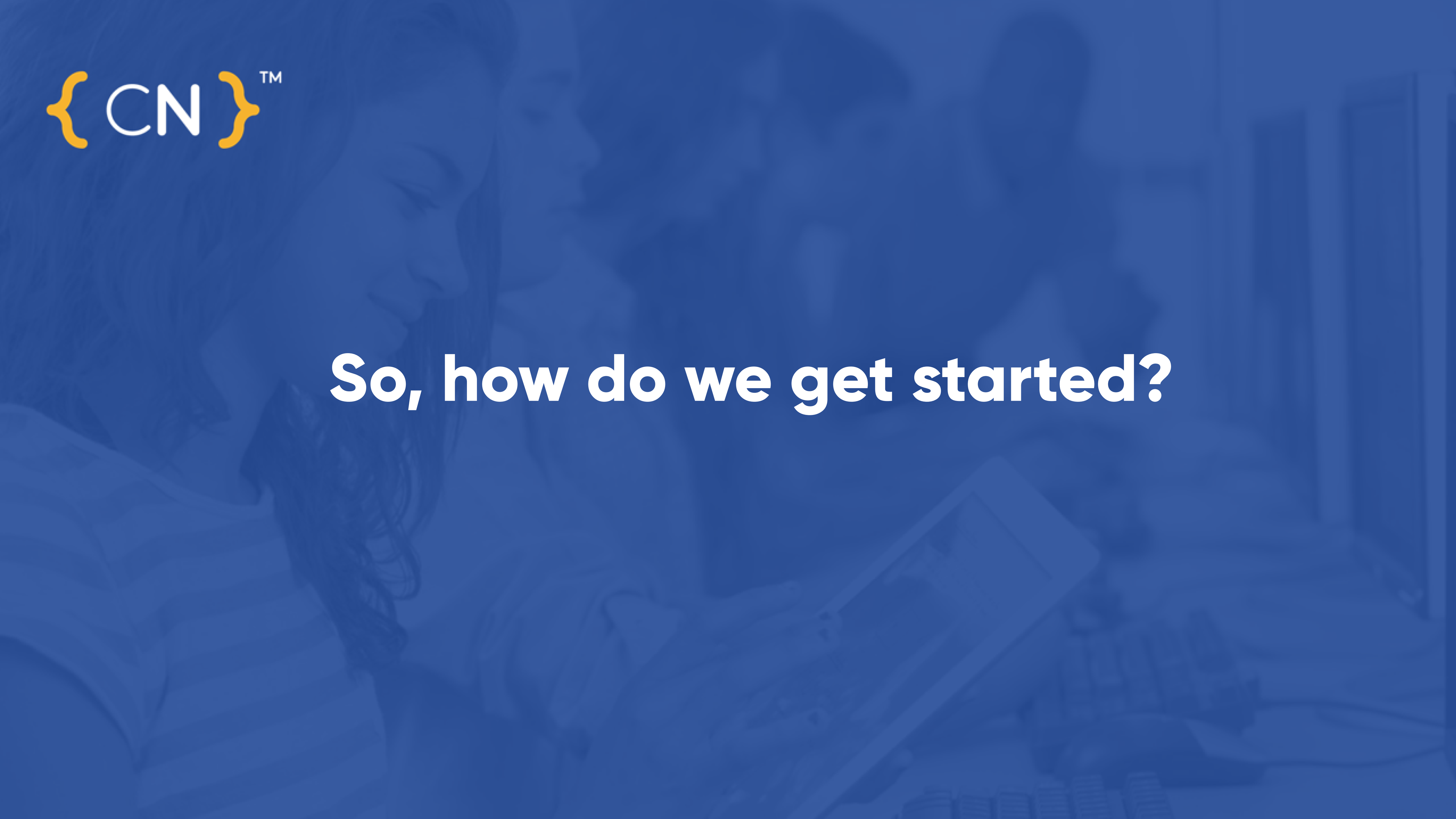
Salt

- Random data added to the password
- Without a salt, hashes of the same password would also be the same
- Protect against rainbow tables*

* rainbow tables: https://en.wikipedia.org/wiki/Rainbow_table

Rainbow tables in action

https://hashkiller.co.uk/Cracker/MD5



{CN}™

So, how do we get started?

bcrypt

https://www.npmjs.com/package/bcrypt