PACKAGE BCRYPT

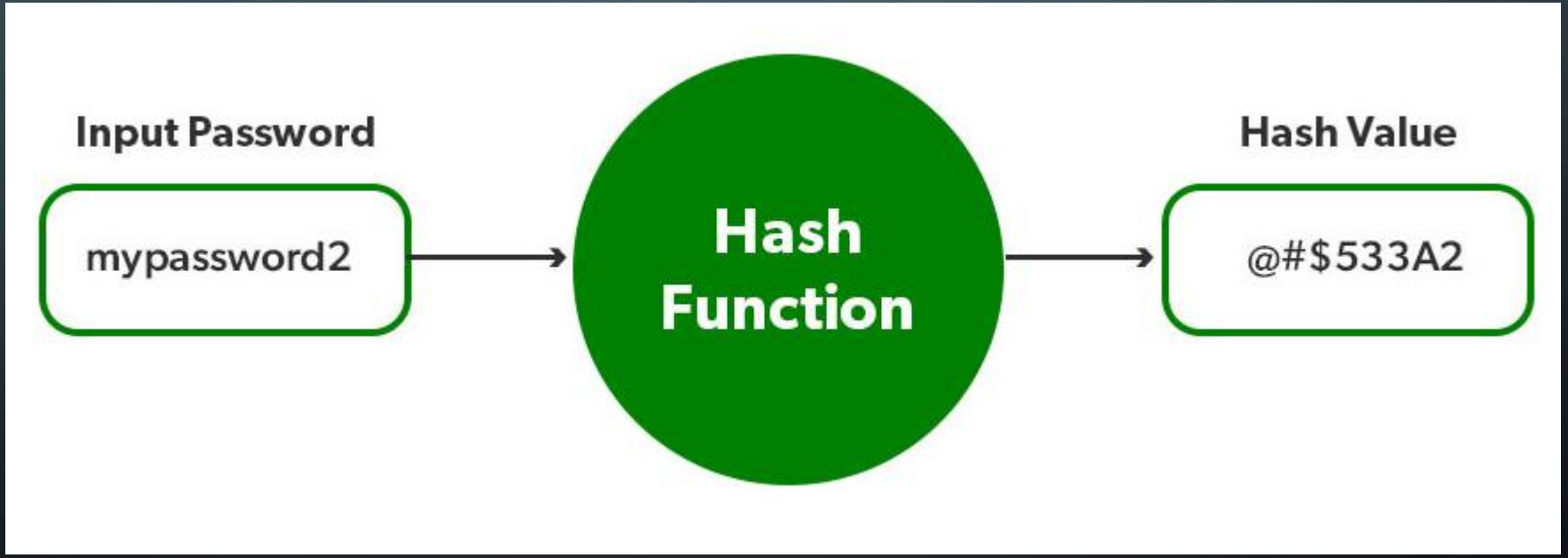
FORMATEUR : MAHDI KELLOUCH

HACHAGE EN INFORMATIQUE

- Le hachage est la transformation d'une chaîne de caractères en valeur ou en clé de longueur fixe, généralement plus courte, représentant la chaîne d'origine.
- Donc, une fonction de hachage est une fonction qui transforme une donnée quelconque en une donnée de taille fixe.
- Contrairement au cryptage qui peut être décodé pour récupérer le mot de passe original, le hachage est une fonction à sens unique qui ne peut pas être inversée une fois qu'elle est réalisée.

BCRYPT

- Le package npm bcrypt est une implémentation JavaScript de la fonction de hachage de mot de passe bcrypt qui vous permet de créer facilement un hachage à partir d'une chaîne de mots de passe.



BCRYPT FONCTIONNEMENT

- Lorsqu'un l'utilisateur soumet un mot de passe, celui-ci est haché et votre application JavaScript doit stocker le hachage dans la base de données.
- Lorsque l'utilisateur souhaite authentifier son compte, on compare le mot de passe saisi avec le hachage stocké dans votre base de données pour voir s'il correspond.

➔ La bibliothèque bcrypt facilite ce processus en vous fournissant des méthodes pour hacher et comparer les mots de passe.

INSTALLATION

- Pour commencer à utiliser la bibliothèque, vous devez l'installer avec votre gestionnaire de paquets :

npm install bcrypt

- Incluez ensuite le module dans votre code JavaScript avec import :

import bcrypt from 'bcrypt'

CRÉATION D'UN PASSWORD HASH AVEC BCRYPT

- Pour générer un mot de passe à l'aide du module bcrypt, vous devez faire appel à la méthode hash() qui accepte les trois paramètres suivants :
  - La chaîne de mots de passe que vous souhaitez hacher
  - Le nombre de tours pour sécuriser le hachage. Ce nombre est généralement compris entre 5 et 15
  - la fonction de rappel à exécuter lorsque le processus de hachage est terminé, en transmettant le message d'erreur et le résultat du hachage.

EXEMPLE DE CRÉATION D'UN HASH

- **Exemple 1 :**

```
const bcrypt = require('bcrypt');

bcrypt.hash('password',5, (err, hash) => {
    console.log(hash);
})
```

- **Exemple 2 :**

```
const bcrypt = require('bcrypt');

const hash = bcrypt.hashSync('password',5);

console.log(hash);
```

GÉNÉRATION DU SALT POUR LE HACHAGE

- Une fonction de hachage nécessite l'ajout de salt dans le processus.
- Un salt est simplement une donnée aléatoire utilisée comme entrée supplémentaire dans la fonction de hachage pour protéger votre mot de passe.
- La chaîne aléatoire du salt rend le hachage imprévisible.
- Pour générer un salt, vous pouvez utiliser la méthode genSalt() du module.

EXEMPLE D'UTILISATION DE LA FONCTION GENSALT

- Exemple de génération d'un hash en ajoutant un salt :

```
const bcrypt = require('bcrypt');

bcrypt.genSalt(10, (err, salt) => {
    bcrypt.hash('password', salt, (err, hash) => {
        console.log(hash);
    })
})
```

VÉRIFIER UN MOT DE PASSE

- Une fois que vous avez enregistré le hachage dans la base de données, vous pouvez comparer l'entrée en texte brut de l'utilisateur avec le hachage enregistré en utilisant la méthode compare().
- La function compare accepte trois parametres :
  - Le mot de passe en clair pour la comparaison
  - La chaîne de hachage créée précédemment
  - Et la fonction callback une fois le processus de comparaison terminé.

EXEMPLE D'UTILISATION DE LA FONCTION COMPARE

- Exemple 1 :

```
const bcrypt = require('bcrypt');

bcrypt.compare('Password',
'$2b$10$ZWvig8eCezwa4sRJ0lg2yujrgX767pKRVJ2it7hcgO1E.OQJ1MWZu’,
(err, result)=>{
    console.log(result)
})
```

- Exemple 2 :

```
const bcrypt = require('bcrypt');
const result = bcrypt.compareSync('password',
'$2b$10$ZWvig8eCezwa4sRJ0lg2yujrgX767pKRVJ2it7hcgO1E.OQJ1MWZu');
console.log(result)
```