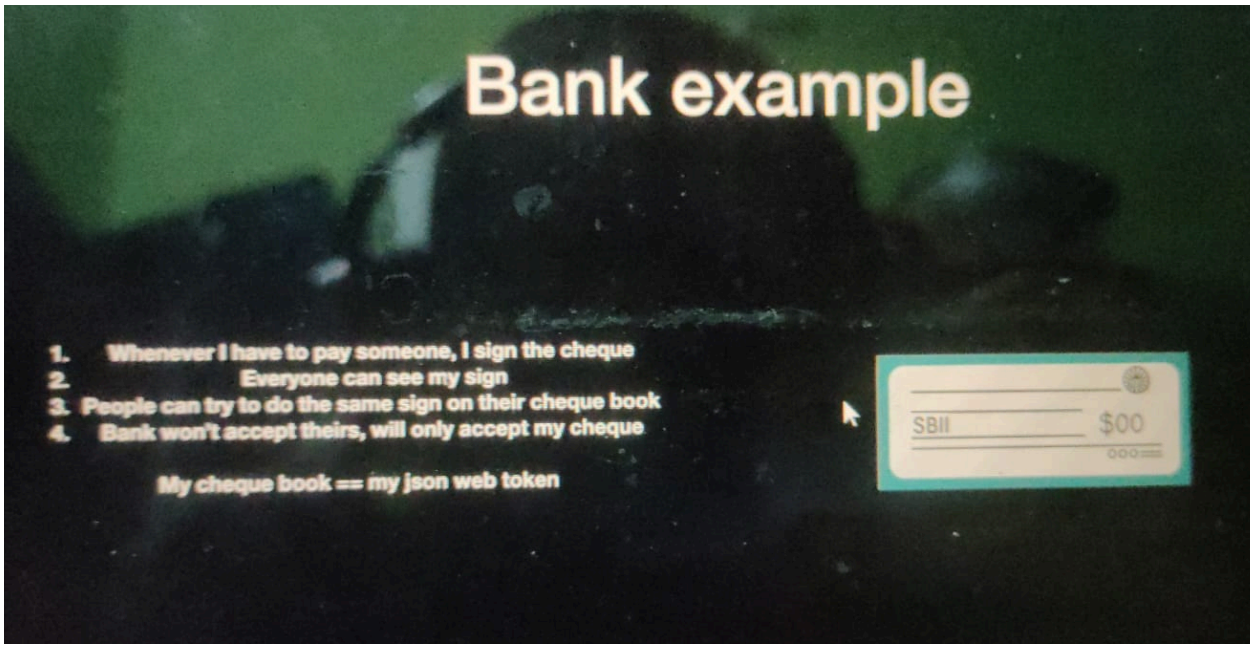
Authentication,JWT and try/catch

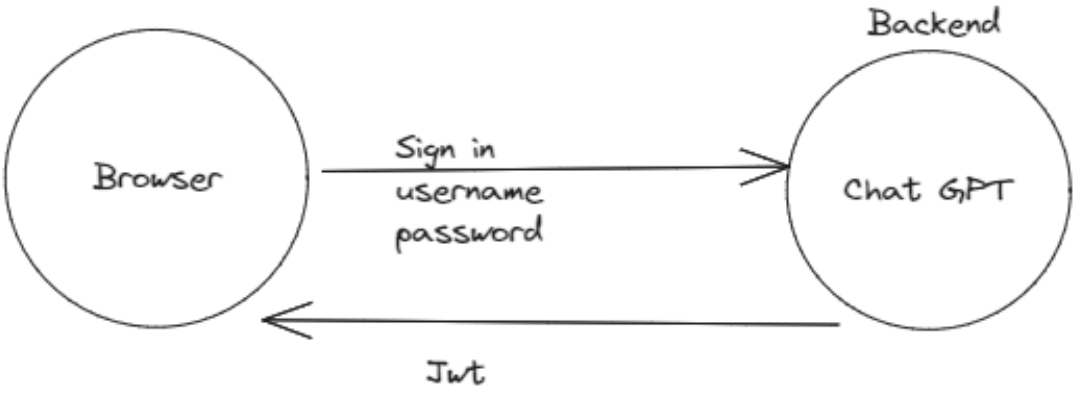
Lets take example of Nishu who want to open a bank account.

1. Nishu goes into bank and open a bank account
2. Nishu deposits some money in
3. Nishu gets back a cheque book from the bank

We get a chequebook is like a authenticated way , make sure to keep it safe , it is not kept safe money is gone

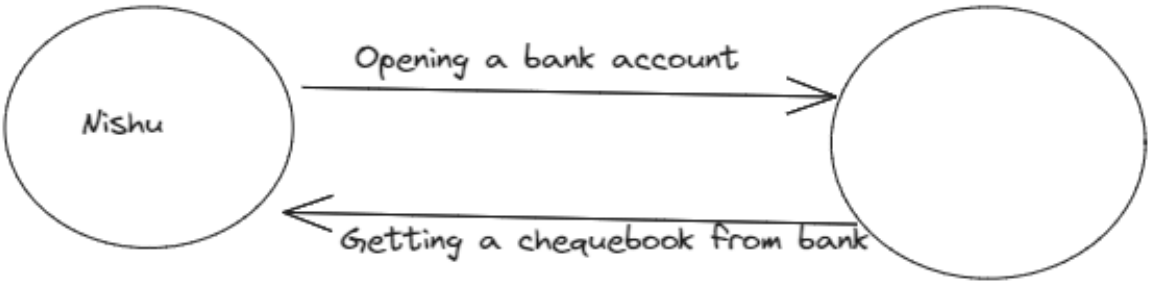


Our chequebook == ours json web token



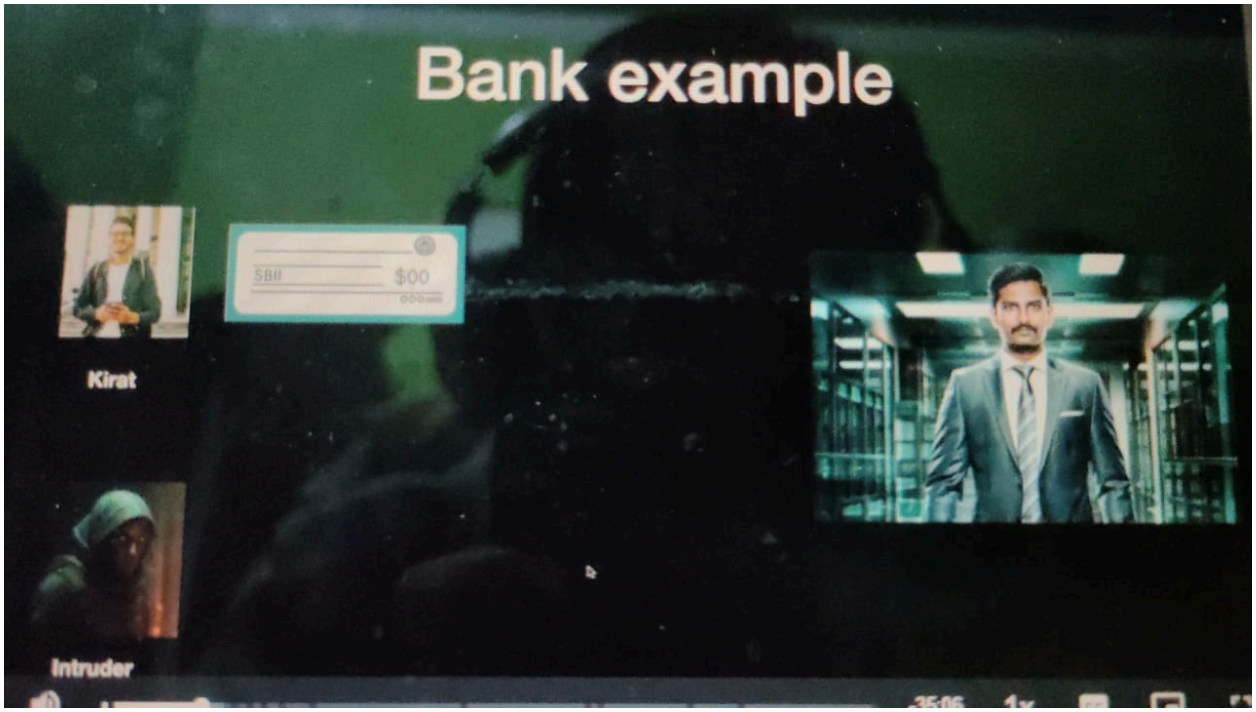
whoever has access to the jwt has access to your chat gpt

These both are kind of similar

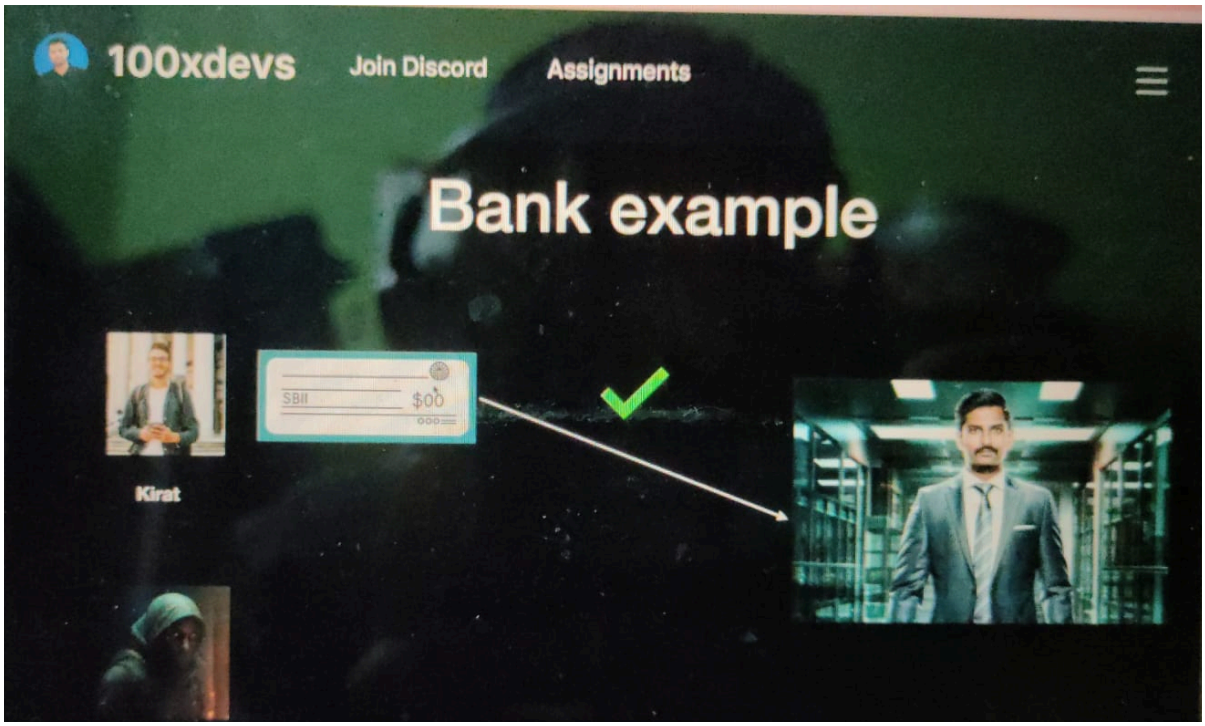


Let's understand in more detail

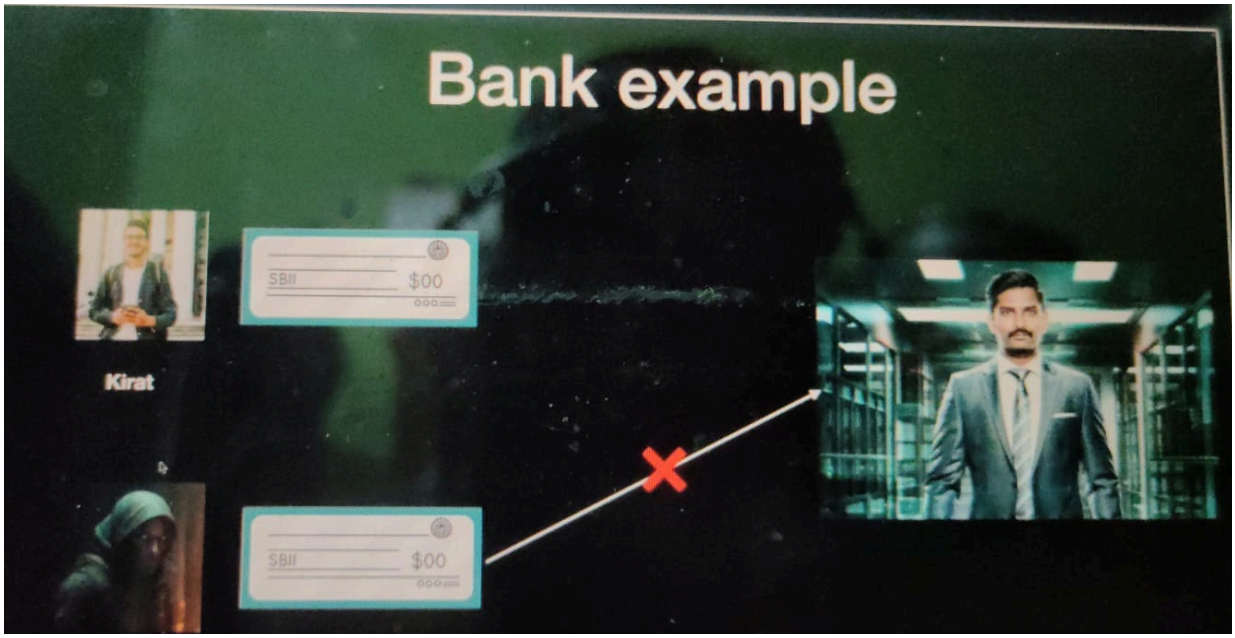
1. Something bank give me the first time I visit it (sign in)
2. Something I need to keep safe(Jwt), If I lose a cheque, someone can debit my account
3. Even though other people can see my signature/ how the cheque looks, the bank will catch them if they try to re-create it.



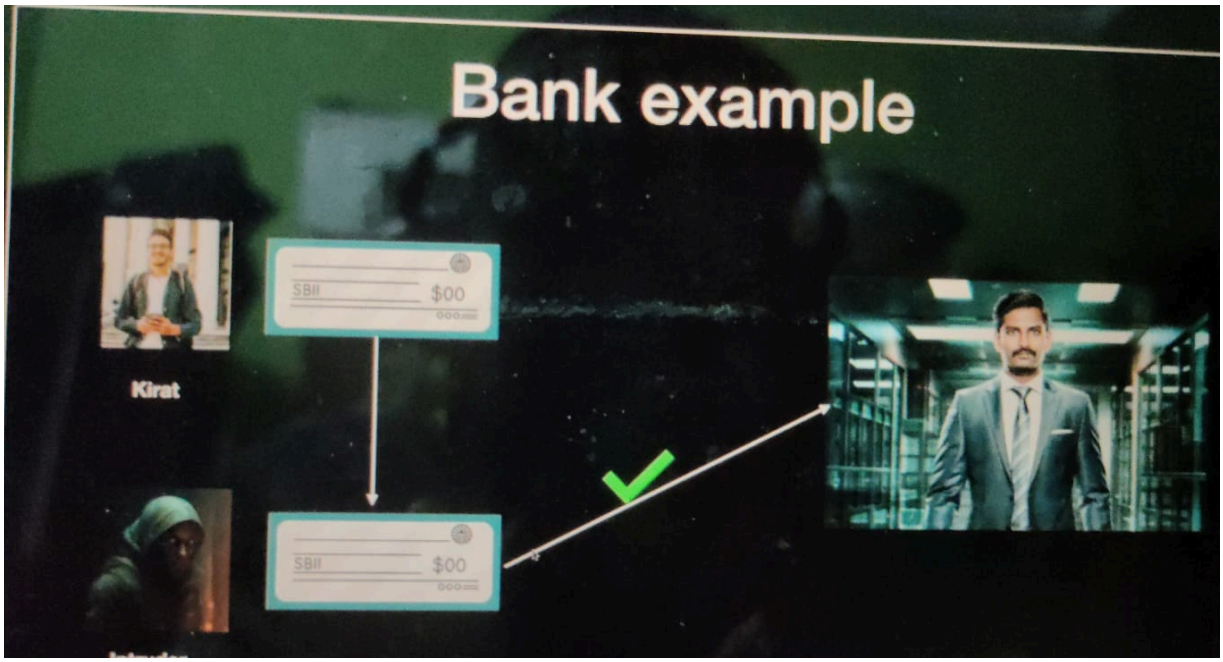
You can do anything with your chequebook



But if someone impersonates you and your chequebook, the bank will reject it .



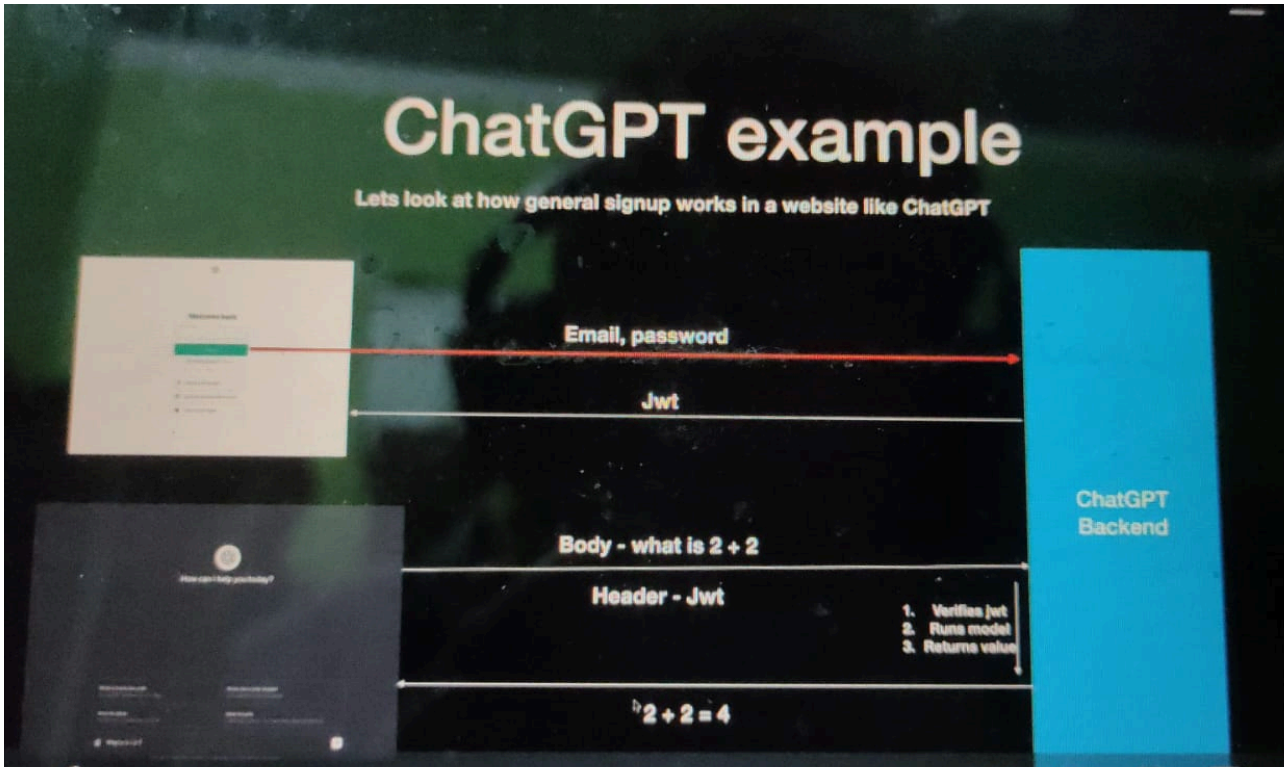
But if you give them your exact signed chequebook, then they can steal your funds.



Hence we must keep our chequebook/ jwt sent by the bank/ backend safe.

JSON web tokens are just like these cheques They are issued by the backend when you sign in Anyone can create something very similar, but the backend would reject it If you lose the original JWT, then it is a problem.

ChatGPT examples



Three things to understand about JWT

1. Generate a jwt
2. Decoding a jwt
3. Verifying a jwt

Generate a jwt

Bank creating a chequebook

Backend send a very big string (jwt) which it can **verify** it

```
const jwt = require('jsonwebtoken');
```

```
// decode, verify, generate

// the function to generate token is called sign
const value ={
    name:'john',
    accountNumber:1234567890
}

// jwt
const token = jwt.sign(value,'secret');
// token generated using this secret, and hence this token can only be
verified using this secret
console.log(token);
// encoded it
//
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiam9obiIsImFjY291bnROdW1iZ
XIiOjEyMzQ1Njc4OTAsImlhdCI6MTcwNTE3MDIzNH0.NV7Dcliks4vVVpJd0undbBzloB7q6Jx
QIsT35YHuK4U
// This token will change if we run this program again
// Seeing this token they can decode it. without giving the secret
// But they can't verify it without the secret
```

Decoding a jwt

Someone can look at your signature and try to copy it. Only the backend matching which created the jwt token can verify it.

```
const jwt = require('jsonwebtoken');

const contents = {
    name:'john',
    accountNumber:1234567890,
    iat: 1234567890
}
const newToken = jwt.sign(contents,"asfasvad");
// they dont know the secret even if they know the content of the jwt
console.log(newToken);
//
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiam9obiIsImFjY291bnROdW1iZ
```

```
XIiOjEyMzQ1Njc4OTAsImlhdCI6MTIzNDU2Nzg5MH0.DiEAtaYc6daeh_GcvRthQKXytydSK9p
iwrmG3nqfbOw
```

Verifying

We are using jwtwebtoken created using another secret and trying to check whether it will do when we try to verify that jwt using “secret” key

```
const jwt = require('jsonwebtoken');
const value = {
    name:'john',
    accountNumber:1234567890
}

const token =
jwt.verify("eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiam9obiIsImFjY
291bnROdW1iZXIiOjEyMzQ1Njc4OTAsImlhdCI6MTIzNDU2Nzg5MH0.DiEAtaYc6daeh_GcvRt
hKXytydSK9piwrmG3nqfbOw","secret")
```

Output:

```
PS C:\Users\NTC\Desktop\Dev\Week3\JWTPractice> node "c:\Users\NTC\Deskt
op\Dev\Week3\JWTPractice\index.js"

c:\Users\NTC\Desktop\Dev\node_modules\jsonwebtoken\verify.js:40
      if (err) throw err;
               ^
JsonWebTokenError: invalid signature
    at c:\Users\NTC\Desktop\Dev\node_modules\jsonwebtoken\verify.js:171
:19
    at getSecret (c:\Users\NTC\Desktop\Dev\node_modules\jsonwebtoken\ve
rify.js:97:14)
    at module.exports [as verify] (c:\Users\NTC\Desktop\Dev\node_module
s\jsonwebtoken\verify.js:101:10)
    at Object.<anonymous> (c:\Users\NTC\Desktop\Dev\Week3\JWTPractice\i
ndex.js:7:19)
    at Module._compile (node:internal/modules/cjs/loader:1375:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1434:10
)
    at Module.load (node:internal/modules/cjs/loader:1206:32)
    at Module._load (node:internal/modules/cjs/loader:1022:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modul
es/run_main:142:12)
    at node:internal/main/run_main_module:28:49
```

Since the secret is different it will get rejected.

Anyone can see the content of jwt , but only the server which created it can verify it.

Try catch

Throwing and catching error in JS

Let see an example of simple function.

```
function getLength(name){
    return name.length;
}
const ans = getLength('johnCena');
console.log(ans);
```

Lets say we forgot to send any input in the function getLength()

What will the terminal show?

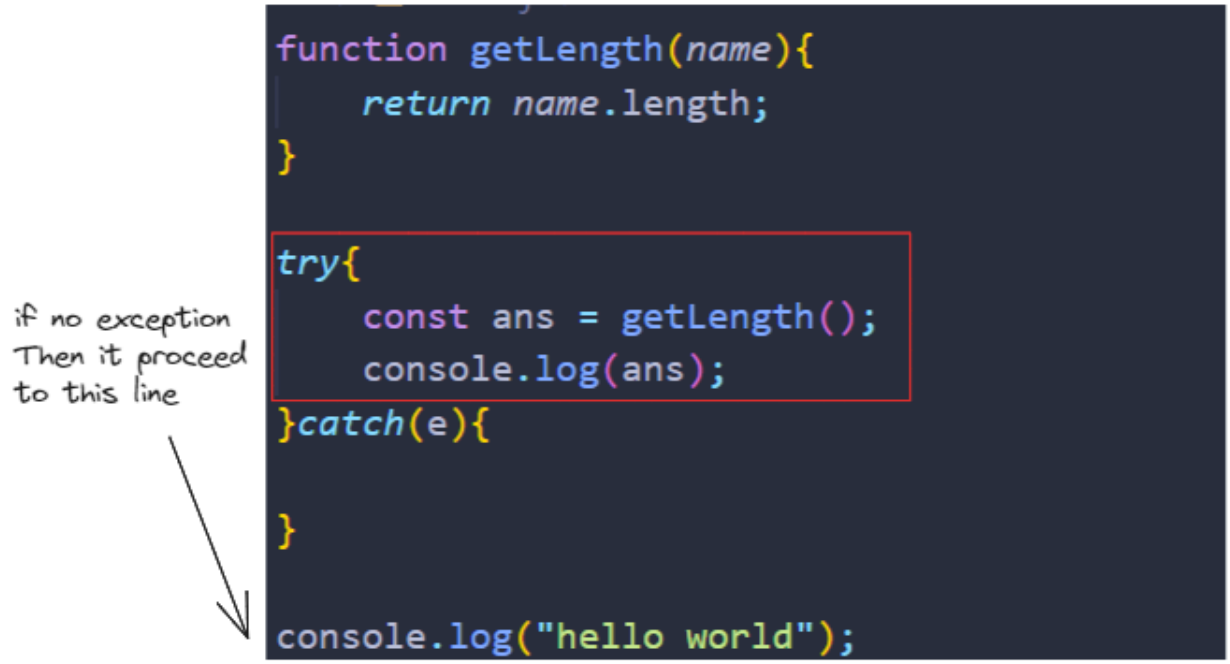
```
PS C:\Users\NTC\Desktop\Dev\Week3\JWTPractice> node "c:\Users\NTC\Deskt
op\Dev\Week3\JWTPractice\index.js"
c:\Users\NTC\Desktop\Dev\Week3\JWTPractice\index.js:2
    return name.length;
               ^

TypeError: Cannot read properties of undefined (reading 'length')
    at getLength (c:\Users\NTC\Desktop\Dev\Week3\JWTPractice\index.js:2
:17)
    at Object.<anonymous> (c:\Users\NTC\Desktop\Dev\Week3\JWTPractice\i
ndex.js:4:13)
    at Module._compile (node:internal/modules/cjs/loader:1375:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1434:10
)
    at Module.load (node:internal/modules/cjs/loader:1206:32)
    at Module._load (node:internal/modules/cjs/loader:1022:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modul
es/run_main:142:12)
    at node:internal/main/run_main_module:28:49
```

Undefined is another type in javascript

It throws error and stops execution right away.

Program panics here, this is not good, code like this should not stop, we need to catch these errors

When an exception is raised the process exits since the JS program doesn't want to proceed anymore.
But you might want the program to continue executing. That is where you can use try catches.



If certain codebase is uncertain then we will wrap it inside the try-catch block

```
// jwt.verify either returns value or throws an error
// hence wrapping it in try-catch is good

try{
    let ans ;
    console.log(ans.length());
}catch(e){
    console.log(e);
```

```
}

console.log("HELLO WORLD");
```

Terminal:

```
PS C:\Users\NTC\Desktop\Dev\Week3\JWTPractice> node "c:\Users\NTC\Deskt
op\Dev\Week3\JWTPractice\index.js"
TypeError: Cannot read properties of undefined (reading 'length')
    at Object.<anonymous> (c:\Users\NTC\Desktop\Dev\Week3\JWTPractice\i
ndex.js:6:21)
    at Module._compile (node:internal/modules/cjs/loader:1375:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1434:10
)
    at Module.load (node:internal/modules/cjs/loader:1206:32)
    at Module._load (node:internal/modules/cjs/loader:1022:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modul
es/run_main:142:12)
    at node:internal/main/run_main_module:28:49
HELLO WORLD
```

```
3
4   try{
5       let ans ;
6       console.log(ans.length());
7   }catch(e){
8       console.log("Inside catch block");
9   }
10
11  console.log("HELLO WORLD");
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

```
PS C:\Users\NTC\Desktop\Dev\Week3\JWTPractice> node "c:\Users\NTC\Deskt
op\Dev\Week3\JWTPractice\index.js"
Inside catch block
HELLO WORLD
```

Assignment

1. **Write a function that takes in a username and password and returns a JWT token with the username encoded. Should return null if the username is not a valid email or if the password is less than 6 characters. Try using the zod library here**

Initial code:

```
const jwt = require('jsonwebtoken');
const jwtPassword = 'secret';

/**
 * Generates a JWT for a given username and password.
 *
 * @param {string} username - The username to be included in the JWT
payload.
 *                            Must be a valid email address.
 * @param {string} password - The password to be included in the JWT
payload.
 *                            Should meet the defined length requirement
(e.g., 6 characters).
 * @returns {string|null} A JWT string if the username and password are
valid.
 *                        Returns null if the username is not a valid
email or
 *                        the password does not meet the length
requirement.
 */
function signJwt(username, password) {
    // Your code here
}
```

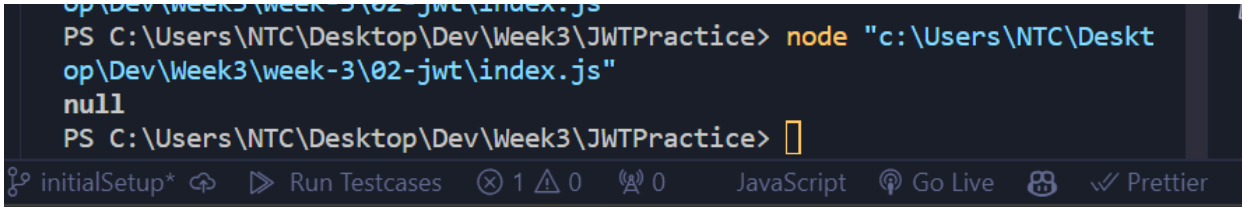
npm install zod jsonwebtoken

Solution:

```
const jwt = require('jsonwebtoken');
const jwtPassword = 'secret';
const zod = require('zod');
```

```
const emailSchema = zod.string().email();
const passwordSchema = zod.string().min(6);

function signJwt(username, password) {
   const usernameResponse = emailSchema.safeParse(username);
    const passwordResponse = passwordSchema.safeParse(password);
    if(!usernameResponse.success || !passwordResponse.success) {
        return null;
    }
   const signature = jwt.sign({
        username
    }, jwtPassword, )
    return signature;
}
// we input the wrong email
console.log(signJwt("acasda","ascxacavasd"))
```

Terminal:



2. **Write a function that takes a jwt as input and returns true if the jwt can be DECODED (not verified). Return false otherwise**

Initial code:

```
const jwt = require('jsonwebtoken');
const jwtPassword = 'secret';
const zod = require('zod');

/**
 * Decodes a JWT to reveal its payload without verifying its authenticity.
 *
 * @param {string} token - The JWT string to decode.
```

```
 * @returns {object|false} The decoded payload of the JWT if the token is
a valid JWT format.
 *                         Returns false if the token is not a valid JWT
format.
 */
function decodeJwt(token) {
    // Your code here
}
```

Solution:

```
const jwt = require('jsonwebtoken');
function decodeJwt(token) {
    // Your code here
    const decoded = jwt.decode(token);//doesn't need secret
    // same as going to jwt.io and pasting the token
    if(decoded) {
        return true;
    }else {
        return false;
    }
}
// wrong jwt
console.log(decodeJwt("sacassa"))
```

3. **Write a function that takes a jwt as input and returns true if the jwt can be VERIFIED. Return false otherwise**

Initial code:

```
const jwt = require('jsonwebtoken');
const jwtPassword = 'secret';
const zod = require('zod');
/**
 * Verifies a JWT using a secret key.
 *
 * @param {string} token - The JWT string to verify.
 * @returns {boolean} Returns true if the token is valid and verified
using the secret key.
```

```
 *                    Returns false if the token is invalid, expired, or
not verified
 *                    using the secret key.
 */
function verifyJwt(token) {
    // Your code here
}
```

Solution:

Let see this code which we could misunderstood as correct solution.

```
const jwt = require('jsonwebtoken');
const jwtPassword = 'secret';

function verifyJwt(token) {
    // Your code here
    const verified = jwt.verify(token, jwtPassword);
    if(verified) {
        return true;
    }else {
        return false;
    }
}

const ans = verifyJwt("ascas")
console.log(ans)
```

Let see the terminal

```
PS C:\Users\NTC\Desktop\Dev\Week3\JWTPractice> node "c:\Users\NTC\Deskt
op\Dev\Week3\JWTPractice\practice.js"

c:\Users\NTC\Desktop\Dev\node_modules\jsonwebtoken\verify.js:40
      if (err) throw err;
               ^
JsonWebTokenError: jwt malformed
    at module.exports [as verify] (c:\Users\NTC\Desktop\Dev\node_module
s\jsonwebtoken\verify.js:70:17)
    at verifyJwt (c:\Users\NTC\Desktop\Dev\Week3\JWTPractice\practice.j
s:6:26)
    at Object.<anonymous> (c:\Users\NTC\Desktop\Dev\Week3\JWTPractice\p
ractice.js:14:13)
    at Module._compile (node:internal/modules/cjs/loader:1375:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1434:10
)
    at Module.load (node:internal/modules/cjs/loader:1206:32)
    at Module._load (node:internal/modules/cjs/loader:1022:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modul
es/run_main:142:12)
    at node:internal/main/run_main_module:28:49

Node.js v21.5.0
PS C:\Users\NTC\Desktop\Dev\Week3\JWTPractice>
```