WHAT IS NODE?

ENVIROMENT FOR EXECUTING JAVASCRIPT OUTSIDE BROWSER

EXAMPLE USE CASE

CHAT BOTS

slack


15:48
#marketing
Elena Nowak 12:50
Hi @Will, how are you getting on with the creative for the autumn campaign?
Will Rodrigues 12:55
Really well! Just putting the finishing
A1 Marketing
Will Rodrigues
All unread
All threads
Channels
accounting-costs
design-feedback 1
events
marketing
media-and-pr
urgent-issues
Direct Messages
slackbot
Will Rodrigues
Elena Nowak

COOL FEATURE OF NODE

MODULE SYSTEM

WHAT ARE MODULES? THINK OF THEM AS....

COLLECTION OF LIBRARIES WITH USEFUL FUNCTIONALITY

EXAMPLE OF NODE MODULE

FILE SYSTEM

```
1 // pull in file system library into your code
2 const fs  = require('fs');
3
4 // use it to read content from a file
5 fs.readFile('./file.txt', 'utf8', (err, data) => {});
6
```

TYPES OF NODE MODULES

1. INBUILT

2. COMMUNITY BUILT

3. CUSTOM BUILT

HOW WAS NODE DESIGNED?

ASYNCHRONOUS BY DEFAULT (NON BLOCKING)

```
1 doStep1();
2
3 console.log('Step2');
4
5 doStep3();
6
7 console.log('Step4');
```

UNLIKE SYNCHRONOUS LANGUAGES

NODE DOESN'T ALWAYS WAIT FOR PREVIOUS LINE TO FINISH EXECUTING

WHY? NODE INTERPRETS

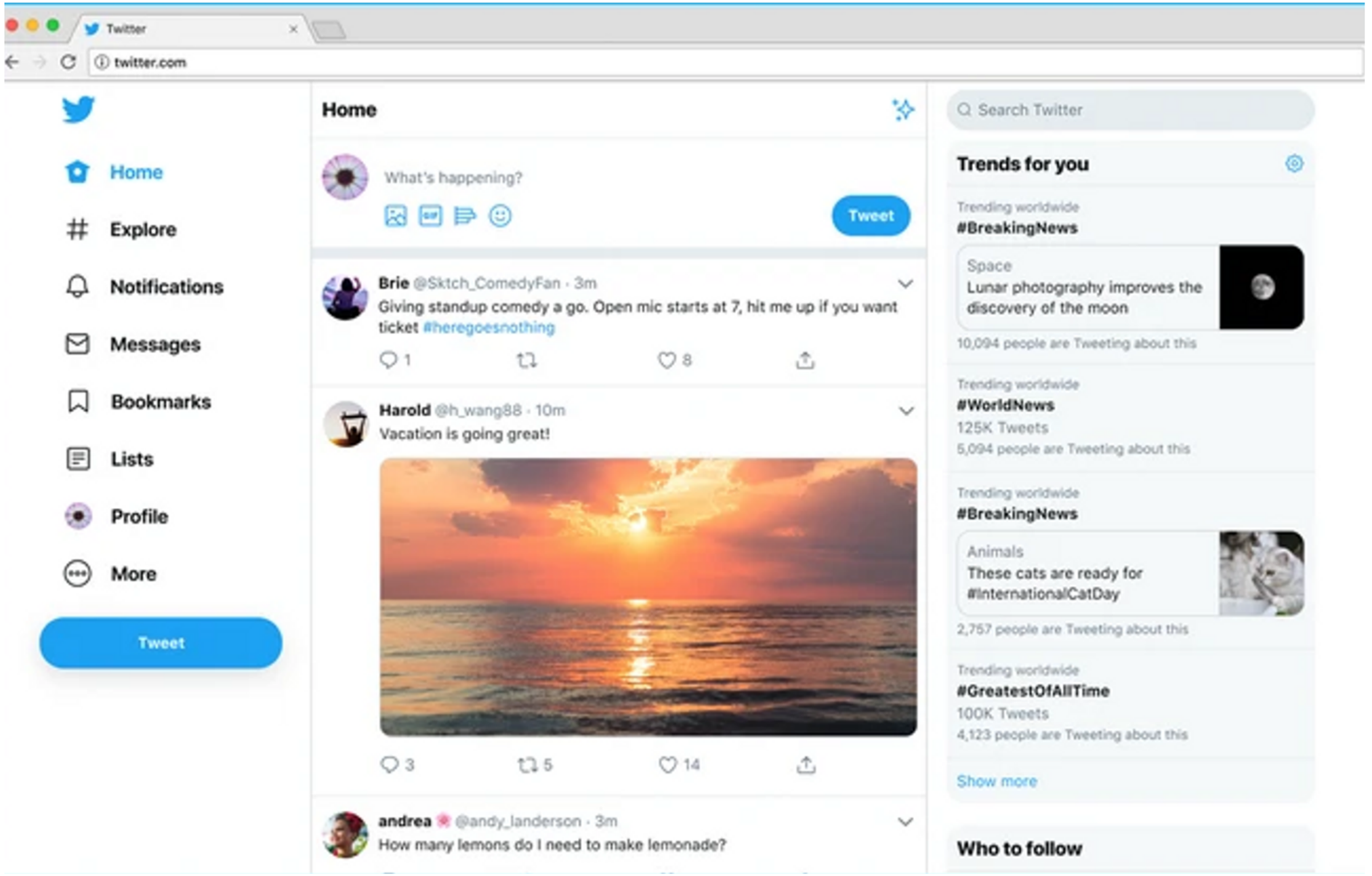
SOME OPERATIONS AS SYNCHRONOUS (BLOCKING)

AND SOME ASYNCRONOUS (NON BLOCKING)

```
1 doStep1();
2
3 console.log('Step2');
4
5 doStep3();
6
7 console.log('Step4');
```

BENEFITS

PERFORMANCE


Twitter
twitter.com
Home
Explore
Notifications
Messages
Bookmarks
Lists
Profile
More
Tweet
Home
What's happening?
Tweet
Brie @Sktch_ComedyFan · 3m
Giving standup comedy a go. Open mic starts at 7, hit me up if you want ticket #heregoesnothing
1
8
Harold @h_wang88 · 10m
Vacation is going great!
3
5
14
andrea @andy_landerson · 3m
How many lemons do I need to make lemonade?
Search Twitter
Trends for you
Trending worldwide
#BreakingNews
Space
Lunar photography improves the discovery of the moon
10,094 people are Tweeting about this
Trending worldwide
#WorldNews
125K Tweets
5,094 people are Tweeting about this
Trending worldwide
#BreakingNews
Animals
These cats are ready for #InternationalCatDay
2,757 people are Tweeting about this
Trending worldwide
#GreatestOfAllTime
100K Tweets
4,123 people are Tweeting about this
Show more
Who to follow

```
1 fetchAvatar();
2 fetchTweets();
3 fetchTrends();
4
```

ASYNCHRONOUS PROGRAMMING

COMES WITH CHALLENGES TOO

CHALLENGES

IMAGINE WE HAVE AN APP WITH A LOT OF ASYNC FUNCTIONS

CHALLENGES

HOW DO WE KNOW WHEN AN OPERATION HAS FINSIHED?

```
1 fetchAvatar();
2 fetchTweets();
3 fetchTrends();
4
```

CHALLENGES

WHAT IF AN OPERATION DEPENDS ON ANOTHER OPERATION FINISHING?

CHALLENGES

WHAT HAPPENS IF ONE OF OPERATIONS FAIL? HOW DO WE HANDLE THIS?

CHALLENGES

IF CODE DOESN'T EXECUTE IN SAME ORDER YOU WRITE IT IS HARD TO FOLLOW ...

SOLUTIONS

CALLBACKS

PROMISES

PROMISES & ASYNC AWAIT

WHAT ARE PROMISES?

PROMISE REPRESENTS RESULT OF ASYNCHRONOUS OPERATIONS

PROMISES.. WHAT TO REMEMBER!

PROMISE CAN RESULT

1) SUCCESS
2) FAILURE

TO ACCESS THE RESULT OF A SUCCESSFUL OPERATION

1) SUCCESS

USE **.THEN** KEYWORD

TO ACCESS THE RESULT OF A FAILED OPERATION

1) FAILURE

USE **.CATCH** KEYWORD

HOW TO CREATE PROMISES

1) USE LIBRARY

2) NODE HAS NATIVE SUPPORT

3) SOME FUNCTIONS AUTOMATICALLY RETURN A PROMISE

EXAMPLE USING LIBRARY: PROMISIFY

```
1  const fs = require("fs");
2
3  // use node library
4  const { promisify } = require("util");
5
6  // create promise object
7  const readFile = promisify(fs.readFile);
8
9  // look at .then keyword
10 readFile("./file.txt", "utf8").then(data => {
11   console.log("content of file: " + data);
12 });
```

TO ACCESS THE RESULT OF A FAILED OPERATION

1) FAILURE

USE **.CATCH** KEYWORD

HANDLE A FAILURE/ERROR WITH PROMISIFY

```
1  const readFile = promisify(fs.readFile);
2
3  readFile("./deletedFile.txt", "utf8")
4    .then(data => {
5      console.log(data);
6    })
7    .catch(err => {
8      console.log("Error occured:", err);
9    });
```

WHAT ARE ASYNC AND AWAIT

ASYNC /AWAIT ALLOW YOU TO WRITE ASYNCHRONOUS CODE THAT'S MORE READABLE

ASYNC AWAIT KEYWORDS IN ACTION

```
1  const asyncFunction = async () => {
2
3    const step1 = await fetchData();
4    // Wait for this
5
6    const step2 = await saveData();
7    // Then wait for that
8
9    const step3 = await copySavedData();
10
11 };
12
```

ASYNC

ASYNC IN FRONT OF FUNCTIONS MEANS IT RETURNS A PROMISE

AWAIT

AWAIT IN FRONT OF FUNCTION MAKES JAVASCRIPT WAIT UNTIL PROMISE SETTLES

LOOK AT HOW READABLE THE CODE IS NOW....

```
1  const asyncFunction = async () => {
2
3    const step1 = await fetchData();
4    // Wait for this
5
6    const step2 = await saveData();
7    // Then wait for that
8
9    const step3 = await copySavedData();
10
11 };
12
```

RULE!

YOU CAN ONLY USE AWAIT INSIDE A FUNCTION THAT HAS ASYNC AROUND IT

IF YOU MISSED ASYNC, YOUR CODE WILL ERROR

```
1  const asyncFunction = async () => {
2
3    const step1 = await fetchData();
4    // Wait for this
5
6    const step2 = await saveData();
7    // Then wait for that
8
9    const step3 = await copySavedData();
10
11 };
12
```

LET'S ATTEMPT SOME CODE

UNDERSTANDING ASYNCHRONOUS DESIGN

TASK 1

WRITE YOUR FIRST PROMISE

TASK 2

WRITE YOUR FIRST ASYNC AWAIT FUNCTION

TASK 3

ERRORS

TASK 4

SIMPLE ERROR HANDLING

TASK 5