```
tt()
```

Schedule

1. Sync vs Async
2. Async: history
3. Date Objects

Schedule

1. **Sync vs Async**
2. Async: history
3. Date Objects

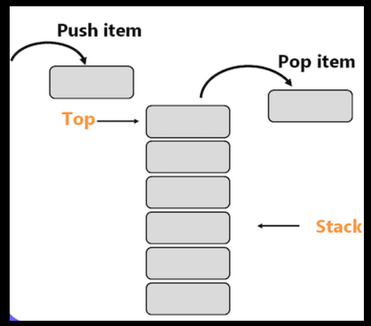
Sync/Async in JS - the problem

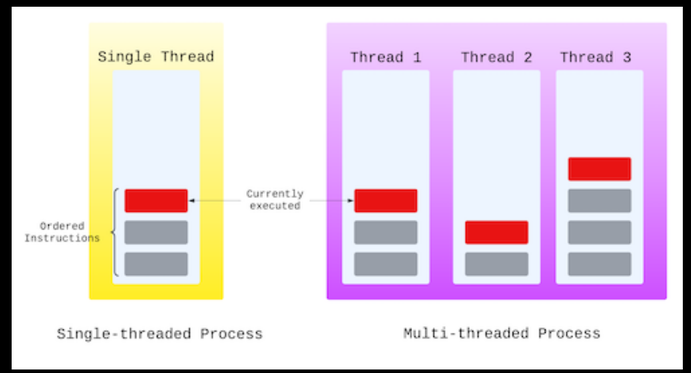
In the browser:

1. The Call Stack - (known from nested functions)

   Functions wait for other calls to be completed.

2. Threads?





3. Problem: JavaScript is single-threaded.

   If multi-threading is required, consider using C++ or Java (Python?)

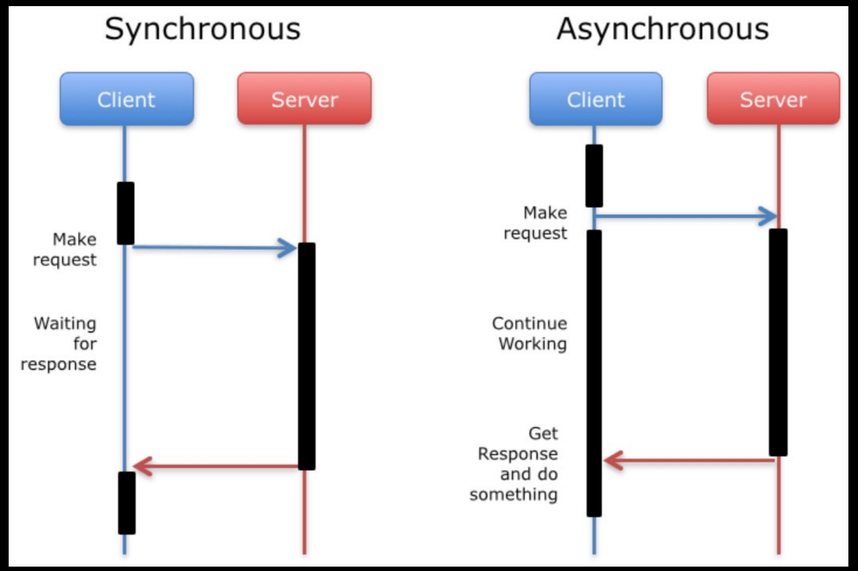
https://www.freecodecamp.org/news/javascript-asynchronous-operations-in-the-browser/

Sync/Async in JS - the solution

Javascript' solution: use client and server

Make a request, continue working

(Callbacks / Promises / Async-await)



It safes time!

(What if…. Sth goes wrong?)

https://aaryanadil.com/synchronous-and-asynchronous-javascript-examples/

Schedule

1. Sync vs Async
2. **Async: history**
3. Date Objects

Sync/Async: error handling

Callbacks: HTTPrequest / response - If-<error> etc.

Promises: .then{….} .catch{<error>}

Async/Await: try {…} catch {<error>}

Schedule

1. Sync vs Async
2. Async: history
3. **Date Objects**

Date Objects: the problem

Different cultures adopt different data formats

Sun calendars / Moon calendars

Political aspects: UTC, summer time / winter time….

Zero state:

- King’s reign (11)
- Creation (5784)
- Birth of Jesus (2023)
- Hidjira (1433)
- ….

Date Objects: JS’s solution

JavaScript **Date objects** represent a single moment in time in a platform-independent format <milliseconds since…>

Zero state: 1.1.1970, 00:00:00

Differences in time are calculated in milliseconds

Date objects translate from milliseconds to understandable date formats

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

Date Objects with JS: hands on

```
Date.now()                                  // class method

const today = new Date();                   // create a Date object

today.toString();                           // make it readable

const tomorrow = new Date("2023-11-21");    // standardized Date format

tomorrow.toString();

const aDate = new Date(2023, 11, 21);

aDate.toString();                           // why?

tomorrow.getTime() - today.getTime();       // #milliseconds in one day
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/Date

https://stackabuse.com/javascript-get-number-of-days-between-dates/

Date Objects with d3

d3: JS compatible with (far) more options

```
const today = new Date(2023, 11, 21);

const tomorrow = new Date(2023, 11, 22);

const days = d3.timeDay.count(today, tomorrow);

d3.timeWeek, d3.timeMonth etc.
```

Time scales!

https://d3js.org/d3-time

https://d3js.org/d3-scale/time

Vergeet de gids niet….

2021/2022: experiment met “de pistes”

EAPRIL23 in Belfast

Vervolg: onderzoek over motivatie en terugblik (interviews met ca 5 deelnemers)

Ik weet niet, welke pistes jullie hebben gevolgd. Jullie wel.

Verzoek: als je mee wil doen met een interview, stuur mij dan bericht

Uncaught SyntaxError
Unexpected end of input