VII

Making Node.js Fast

Optimization and Data Structures

Mattias Hansson, 2022

Story: Cargo Cults

NODE.JS WARSAW

Story: Cargo Cults

Lesson

Don’t do Cargo Cult Programming!

VII

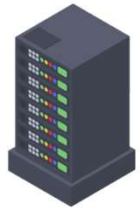
Making Node.js Fast

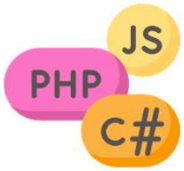
Optimization and Data Structures

Mattias Hansson, 2022

What can we do to speed up a program?

Increase hardware performance

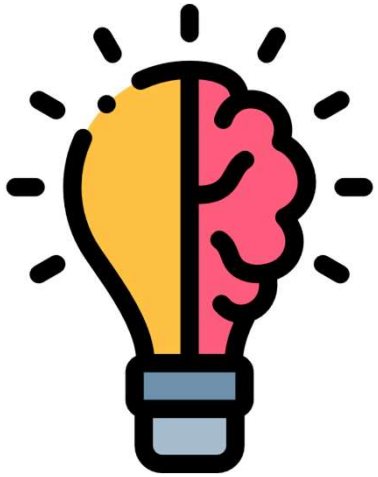




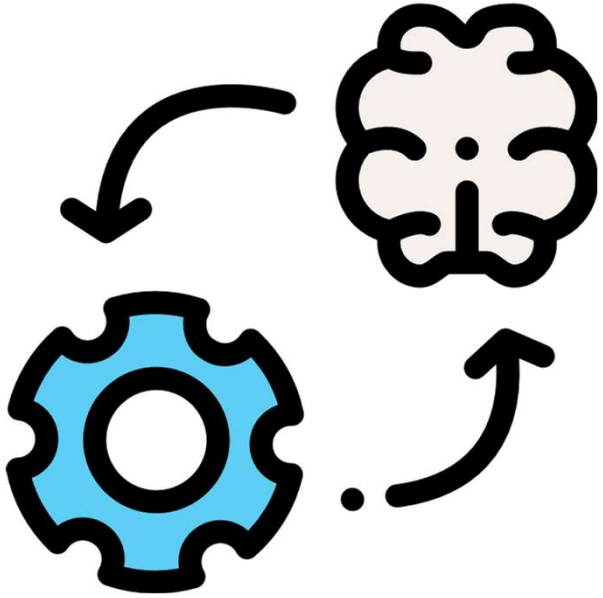
Change programming language

Rewrite the program to do the same with less work or in a better way

Optimization

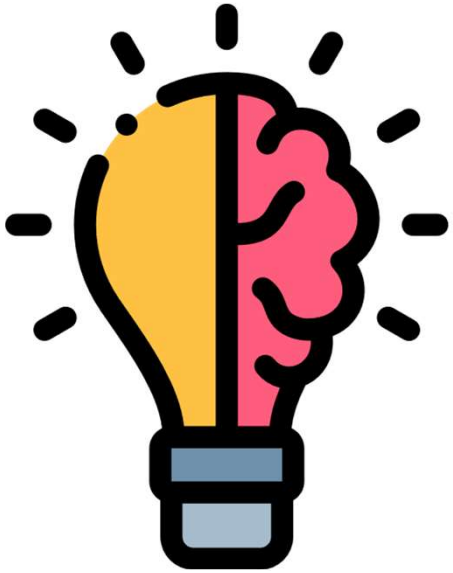


Way of Thinking

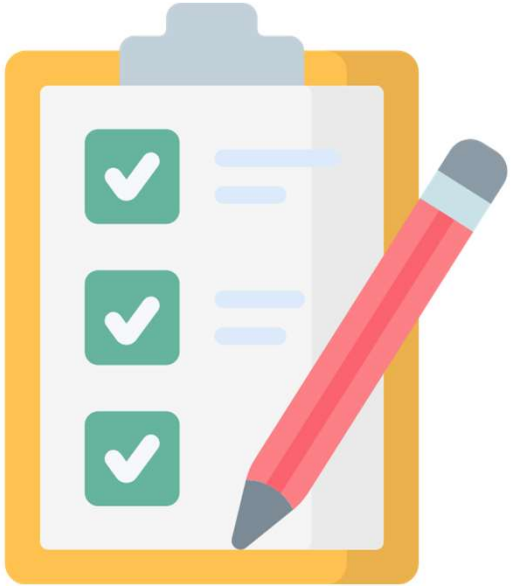


Practice

Way of Thinking



VS

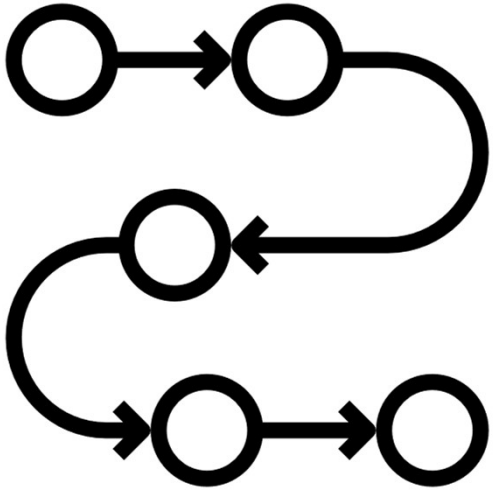


‘Recipe’

Don’t optimize!

“Premature optimization”:

- During PoC, before v 1.0
- The ‘skill trap’
- Not if it’s not needed. Ever!



NODE.JS WARSAW

When do we optimize?

Only when it’s the best thing we can do right now.

- Usability
- Infrastructure
- As a hobby

NODE.JS WARSAW

Measure, measure, measure!

Always measure (profile) first!

NODE.JS WARSAW

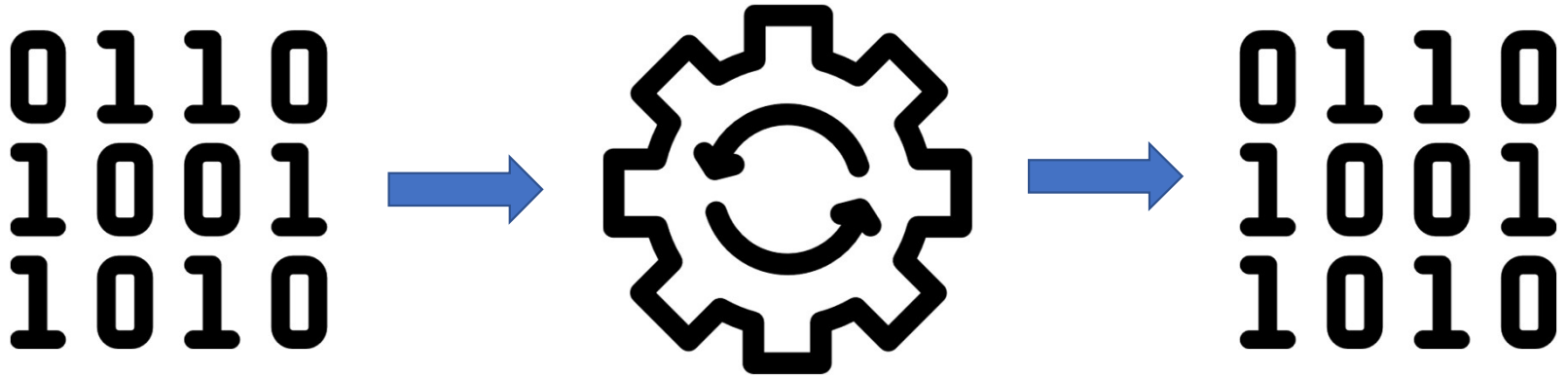
Way of thinking

'What am I asking the computer to do?'

- With the program
- In this function
- In this statement (line)

NODE.JS WARSAW

“The diagram of every program”



Input data

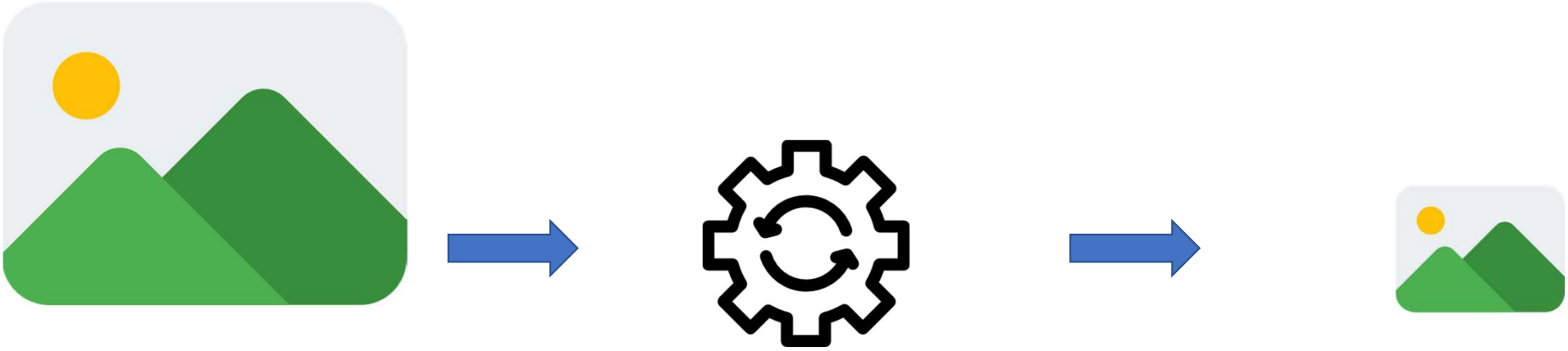
Processing

Output data

NODE.JS WARSAW

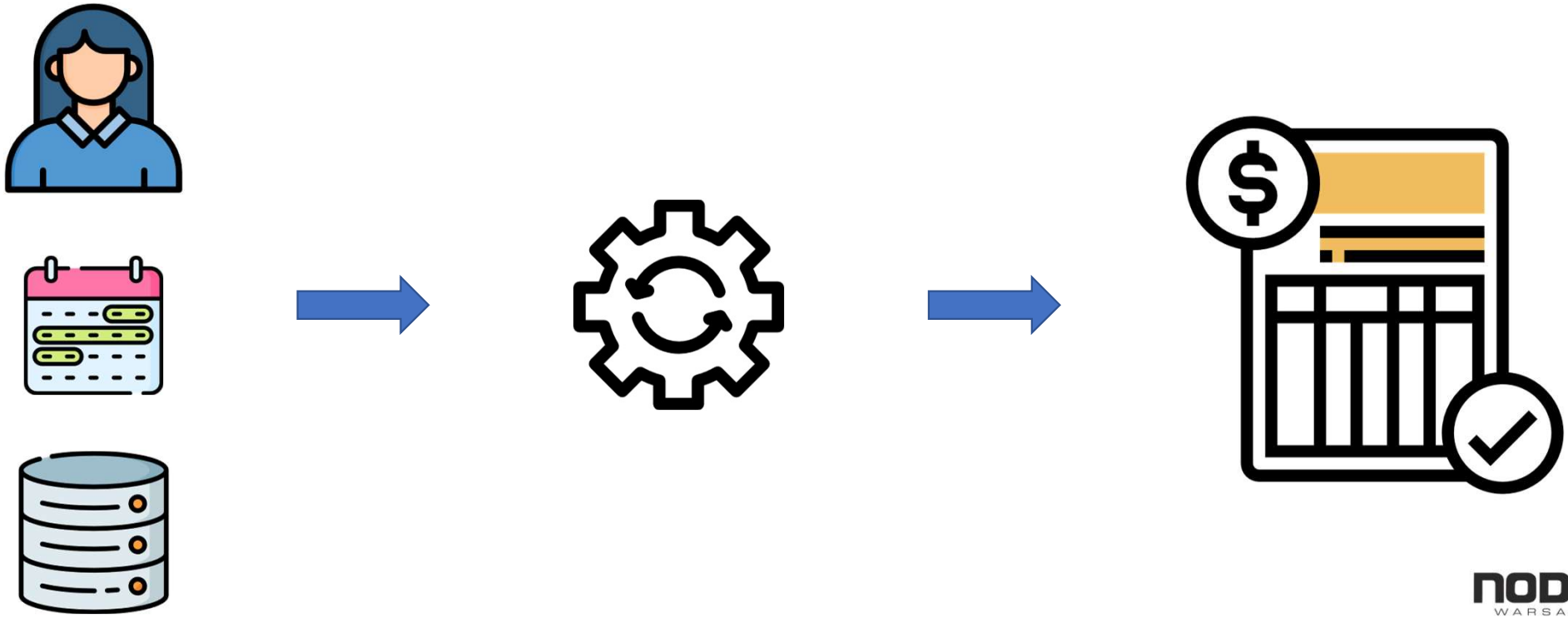
Photoshop

- Example: Resize image

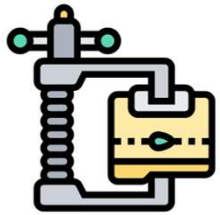


Online banking

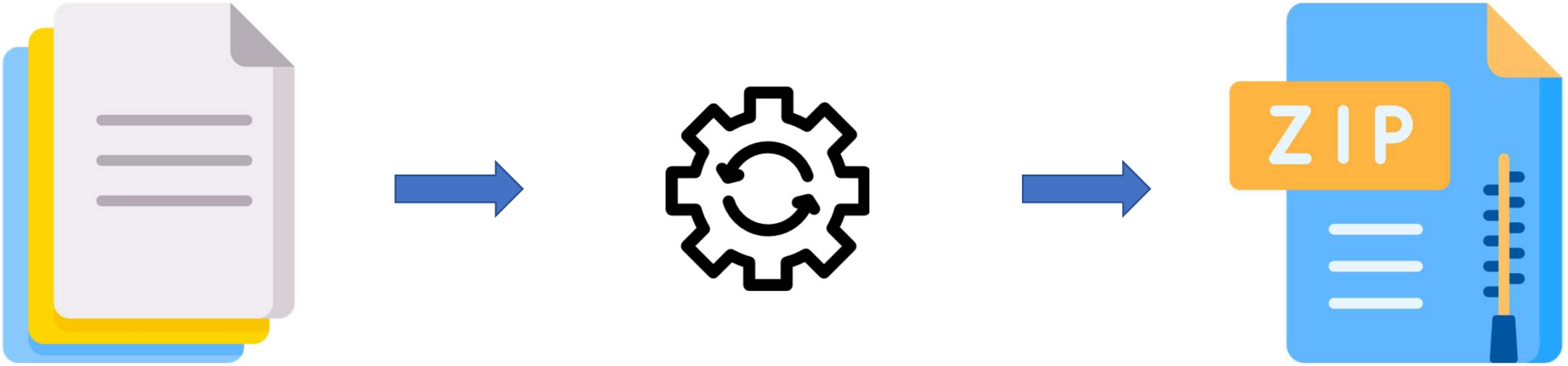
- Example: Bank statement



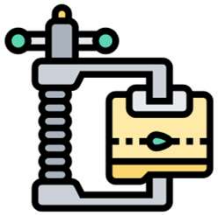
ZIP

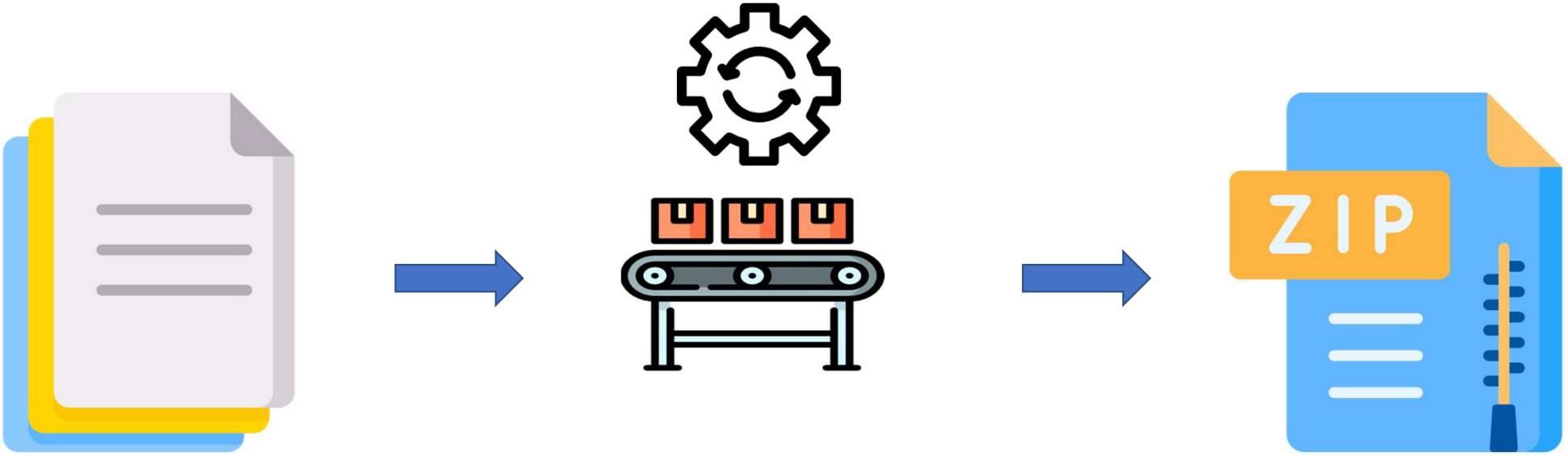


- Example: compress files

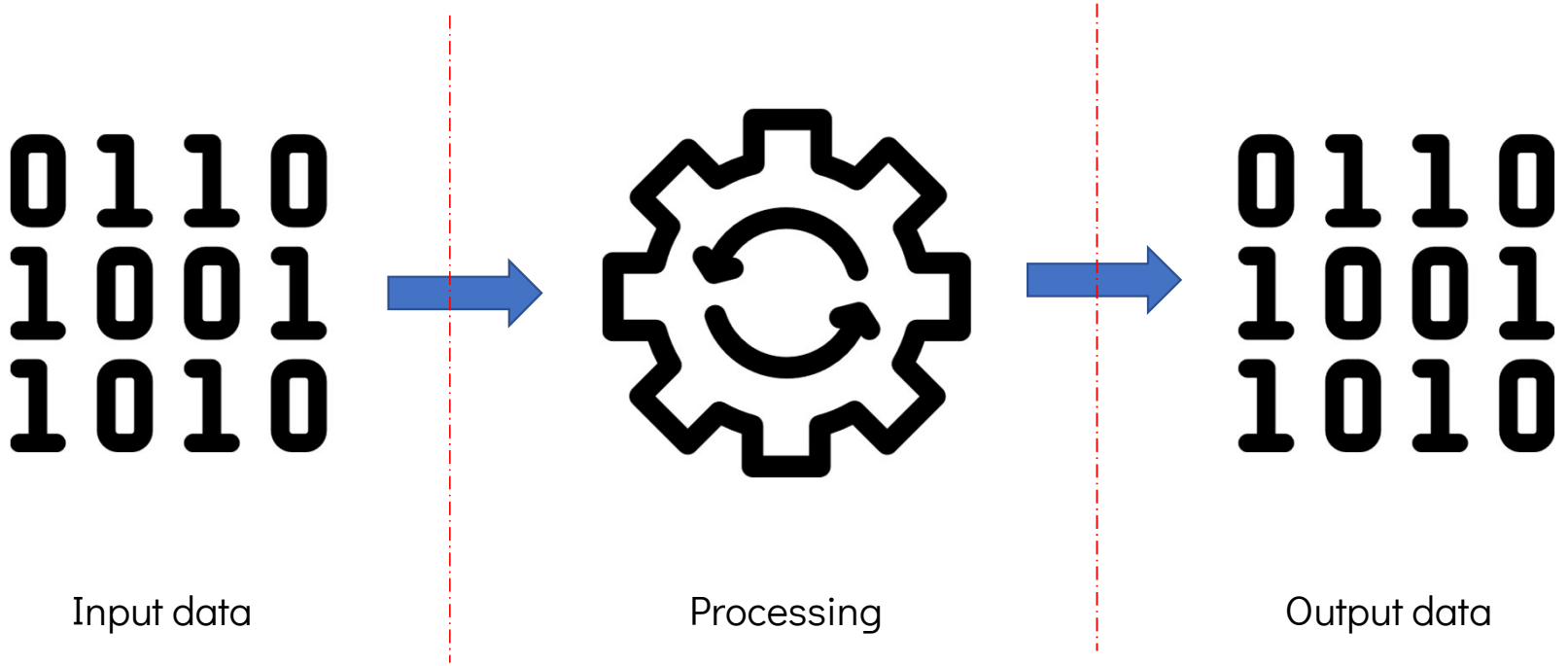


Concept: Data streaming

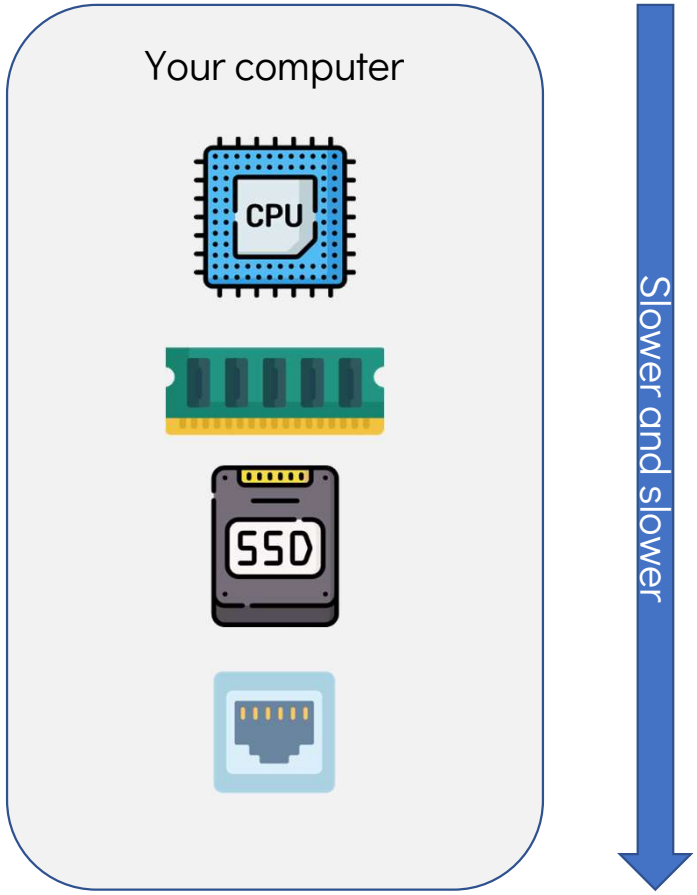




Speed = total time

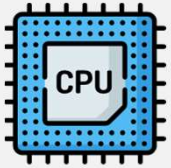


Computers and Networks



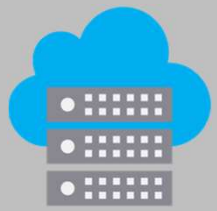
Computers and Networks

Your computer



Your network

The internet



Slower and slower

Latency in computers / networks

Lacency

“The amount of waiting time for something to finish.”

Usually with computers:

The seek time “the time to ask for and locate some data”

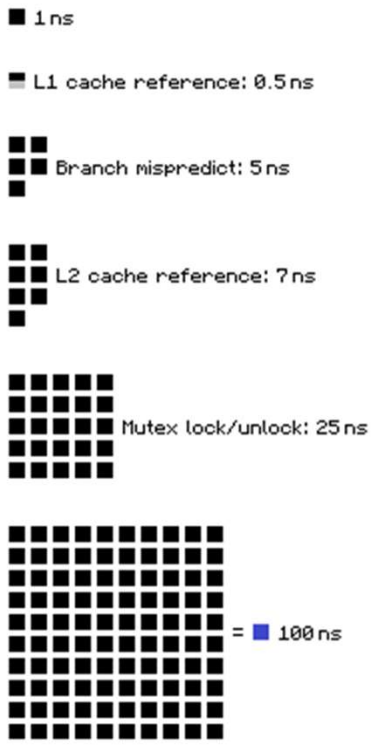
+

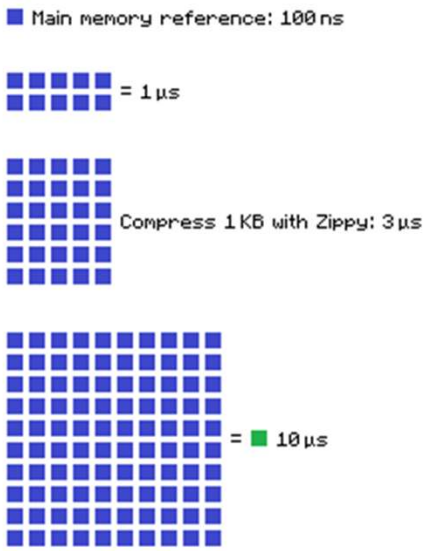
The transfer time “the time is takes to transfer the data”

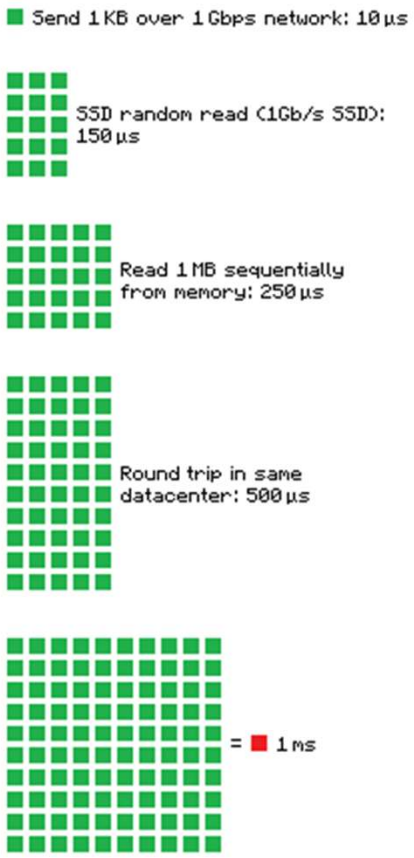
Latency: Ping

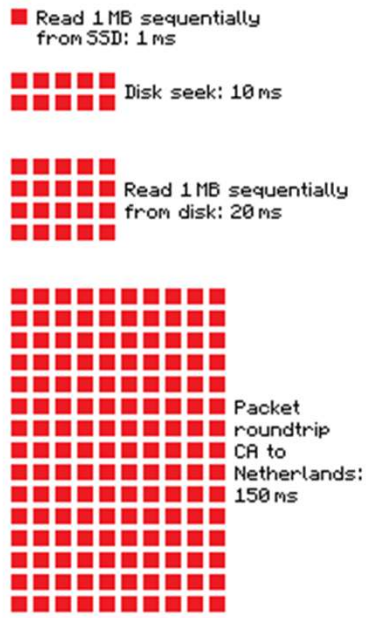
Latency: Download

Latency Numbers Every Programmer Should Know





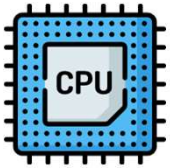




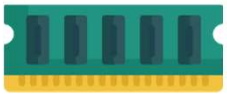
Source: https://gist.github.com/2841832

Human understandable latencies

L1 cache reference 0.5 ns One heart beat (0.5 s)
L2 cache reference 7 s Long yawn



Main memory reference 100 s Brushing your teeth



Read 1 MB sequentially from memory 3 days A long weekend
SSD random read 1 day, 17 hours A normal weekend

Round trip within same datacenter 6 days (local network) A medium vacation

Read 1 MB sequentially from SSD 12 days Waiting for almost 2 weeks for a delivery

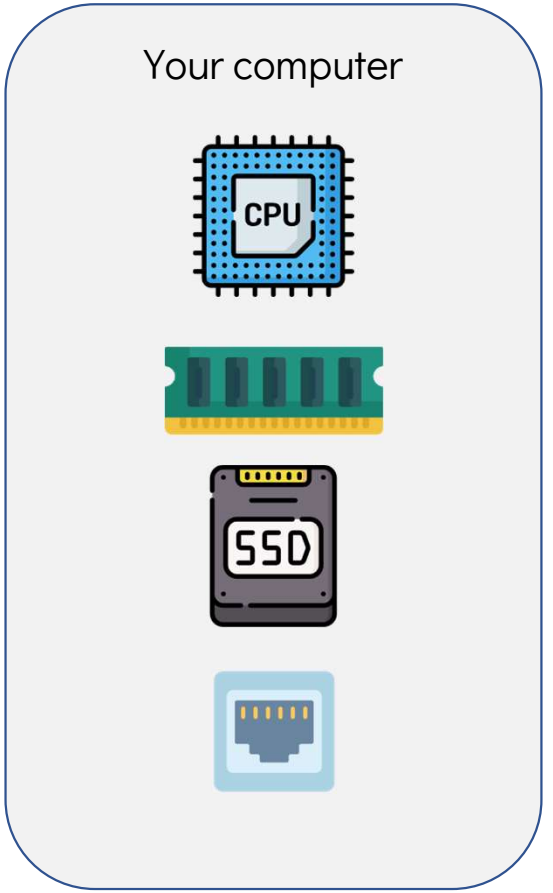
Send packet CA->Netherlands->CA 4 years 10 months Average time it takes to complete a bachelor's degree

Credit: https://stackoverflow.com/questions/44485470/understanding-o1-vs-on-time-complexity-intuitively

Concept #1

Can we bring data closer to the processing?

Example: By using (built in) cache

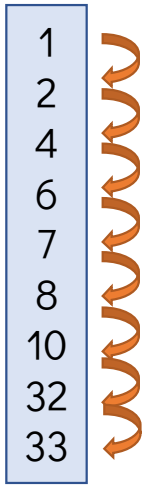


Demo 1

Concept #2 (algorithm primer)

Can we access the data smarter?

Example: look for value in list



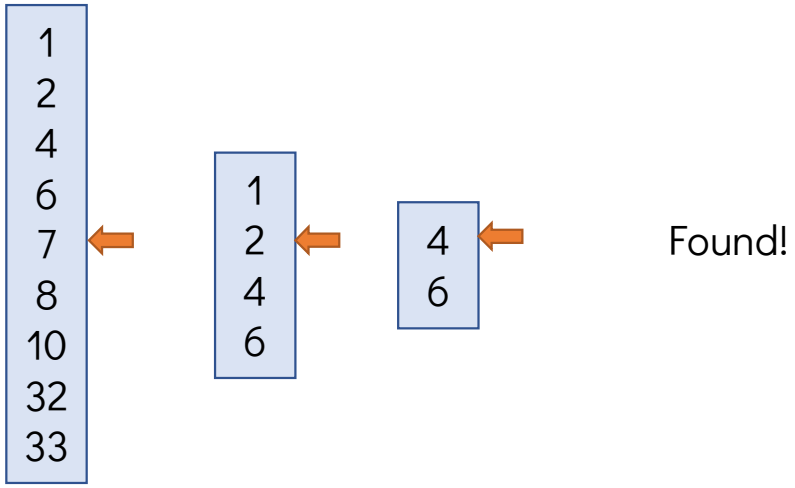
NODE.JS WARSAW

Concept #2 (algorithm primer)

Can we access the data smarter?

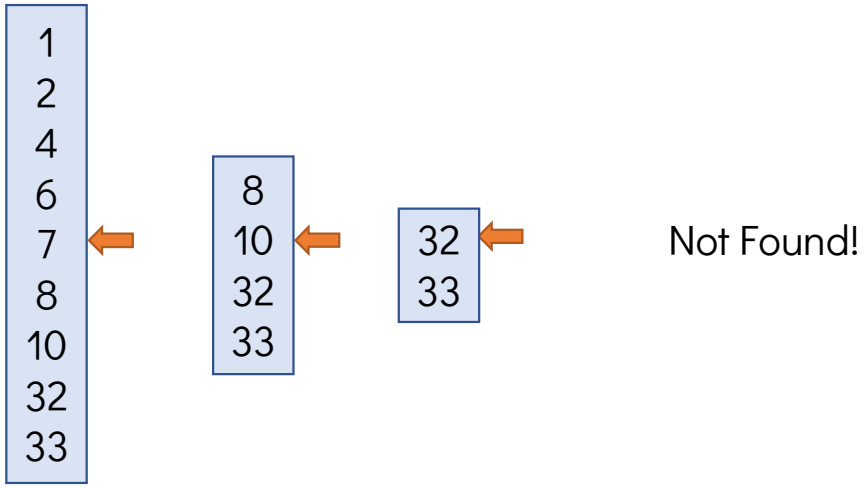
Example: binary search

Example: look for 4



Concept #2

Example: look for 24



Demo 2

Way of thinking

‘What am I asking the computer to do?’

```
r = binary_search(arr, search_value) !== -1;
```

```
r = arr.indexOf(search_value) !== -1;
```

Concept #3

Latency multiplies

1000 tasks each taking 1 ms is one second!

1000 simultaneous requests, avg. resonse time = ½ sec!

JavaScript Data Structures

- Lists []
- Objects {}

Lists and Objects work with JSON

NODE.JS WARSAW

JavaScript List [] (aka Array)

- Zero or more items
- Ordered
- Index starts from 0 (zero)
- Access items by index fast
- Items can be other Lists and Objects

NODE.JS WARSAW

JavaScript List (aka Array)

List examples:
[ ]

[1, 2, 3]

[“aa”, “bb”, “cc”]

[“a”, “b”, “c”, 342, 9]

[“a”, [“hello”, “world”], 15, “z”]

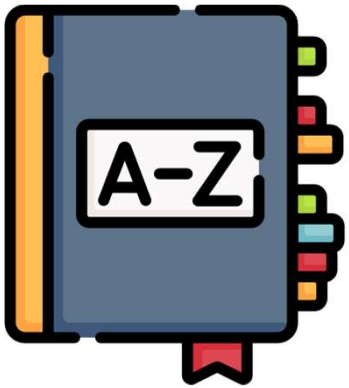
NODE.JS WARSAW

JavaScript List Characteristics

- Search for item 
- Add item 
  - Add item (push) to the end of the list 
- Delete item 
  - Delete item (pull) from the end of the list 
- Access item 

JavaScript Object {}

- Zero or more properties (Key, Value)
- Keys can be number or string
- Keys are unique
- Values can be primitives (number, string etc.) or other Lists and Objects
- Access by key fast



NODE.JS WARSAW

JavaScript Object

Object examples

{ }

```
{
  “name”: “Mattias”,
  “height”: 189
}
```

NODE.JS WARSAW

JavaScript Object

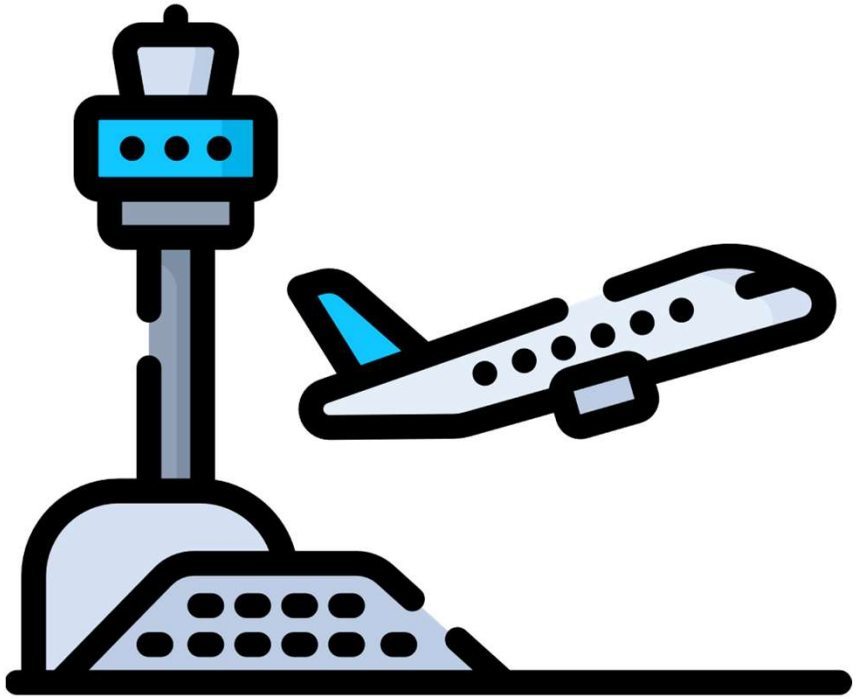
Object examples

```
{
  “bird”: [“owl”, “hawk”],
  “bear”: [“grizzly”, “polar”, “grylls”],
  “dog”: [“asian shepherd”]
}
```

JavaScript Object Characteristics

- Search for item 
- Add item 
- Delete item 
- Access item 

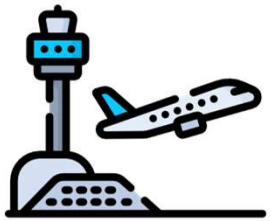
Practice: Airport database



Open data from: https://ourairports.com/

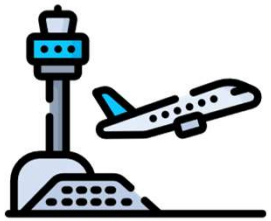
NODE.JS WARSAW

Unoptimized



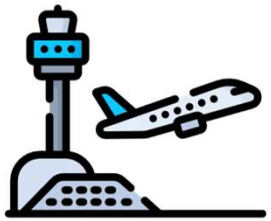
- Lookup the result (by iteration)

Memoization



- Lookup the result
- Store the calculated result for future lookups

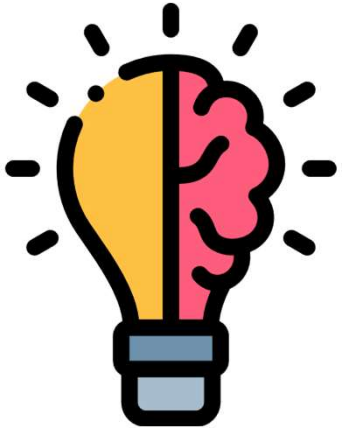
(Lazy) Precalculation



- On first lookup, build the precalculated data
- On subsequent lookups, returns precalculated value

Summary

- ‘What am I asking the computer to do?’
- Measure, measure, measure
- Learn more about algorithms and data structures



The End

Making Node.js Fast

Optimization and Data Structures

Mattias Hansson, Node.js Warsaw VII, 2022-03-16

NODE.JS WARSAW