TDP013 – Web Programming and Interactivity

Lecture 1: JavaScript, Node.js, Express, MongoDB

Huanyu Li

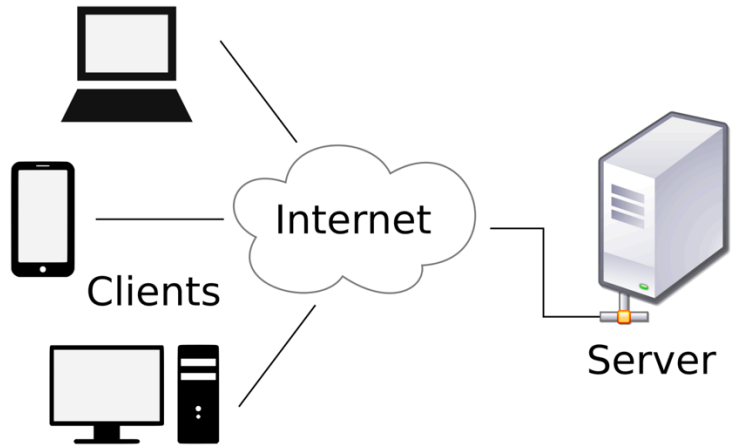
Human-Centered Systems, Department of Computer and Information Science

LINKÖPING UNIVERSITY

The Web

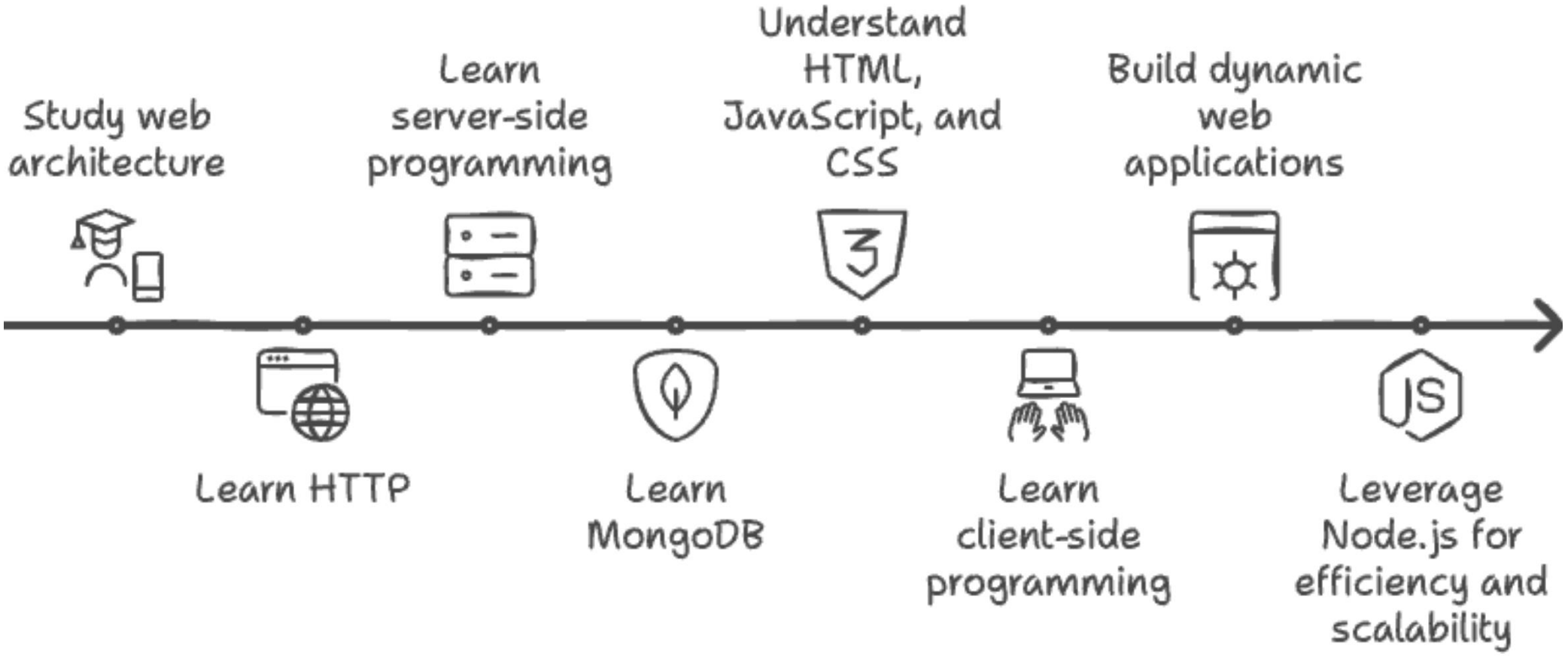
Web applications

- No installations required
  - One version for all operating systems
  - Easy to “try it out”
- Web applications can run on all devices with a browser
- Updates can be made available immediately
  - Without any user actions required
- Lower maintenance costs
  - Compared with desktop applications



picture from https://en.wikipedia.org/wiki/Web_server

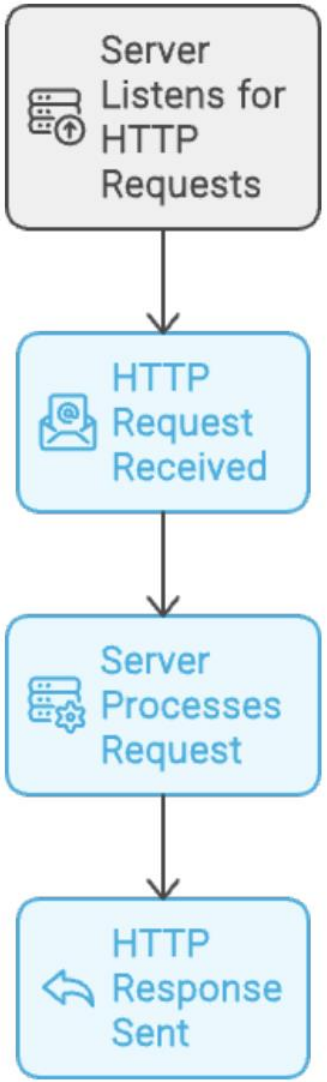
Develop a Web application



The nature of a web server

- Hardware, a computer storing software and component files
- Software, controlling how web users access those component files

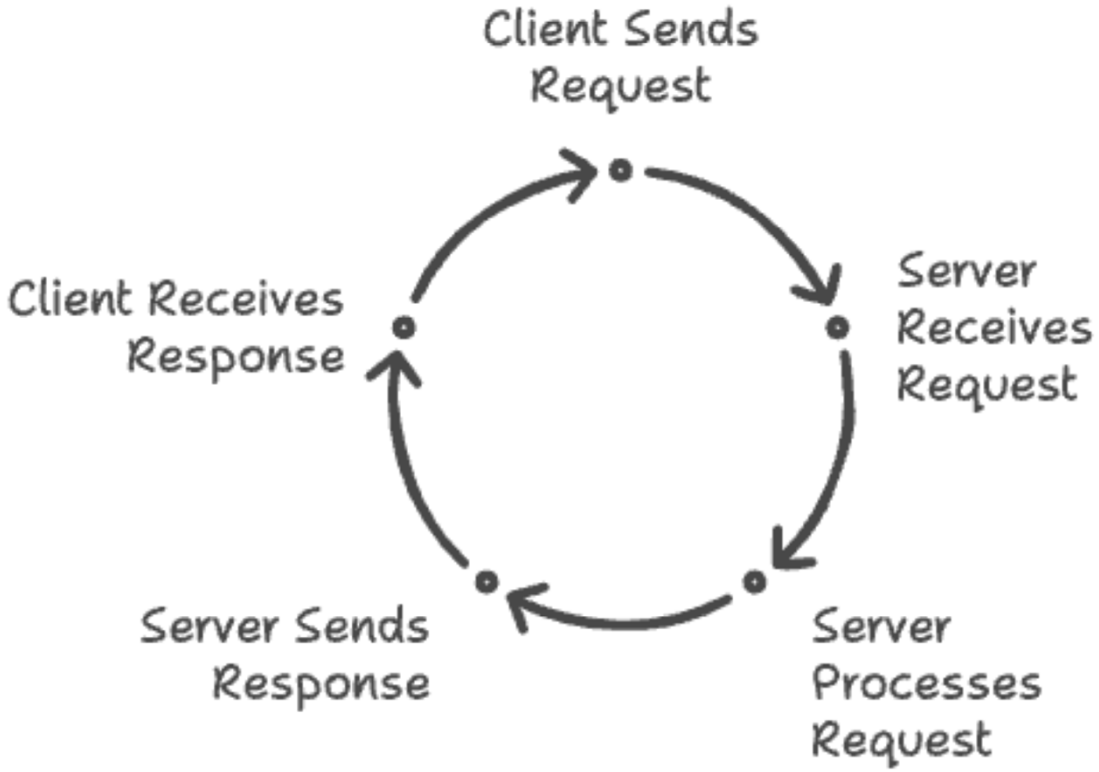
- Static Web pages
- Dynamic Web applications



HTTP communication

- HTTP, as a request-response protocol in client-server model

- Client sends request
- Server receives request
- Server processes request
- Server sends response
- Client receives response



HTTP overview

- Terms
  - HTTP methods

HTTP overview

- Terms
  - HTTP methods
    - GET: request data from a server
    - POST: send data to a server to create/update a resource
    - PUT
    - HEAD
    - DELETE
    - etc.

HTTP overview

- Terms
  - HTTP methods
  - Response Codes/HTTP status messages

HTTP overview

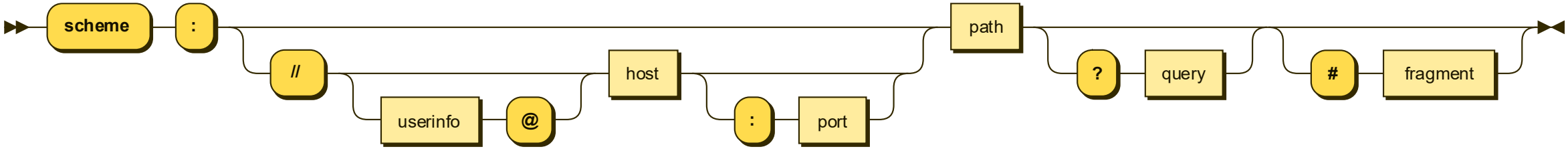
- Terms
  - HTTP methods
  - Response Codes/HTTP status messages
    - 1xx: Information
    - 2xx: Successful
    - 3xx: Redirection
    - 4xx: Client Error
    - 5xx: Server Error

HTTP overview

- Terms
  - HTTP methods
  - Response Codes/HTTP status messages
  - Server Ports
    - 80: HTTP
    - 443: HTTPS

How do we call a server

- URL (Uniform Resource Locator)
  - schema:
  - host:
  - port
  - path
  - query string
  - fragment



picture from https://en.wikipedia.org/wiki/URL

JavaScript

LINKÖPING UNIVERSITY

JavaScript (ES6)

- More or less complete support in all modern browsers
- JavaScript runs only in one thread (generally speaking)
- Object-oriented programming is supported but costs more memory as each object defines its own functions

Most important additions in ES6

- arrow functions
- let + const
- iterators + for .. of
- promises
- template strings
- classes
- modules

JavaScript Syntax

```
// Declare a global variable
let foo = "Hello";

// Define a function
function helloWorld(repeat) {
   // Declare a local variable
   let bar = "World!";

        // Output log to console
        console.log(foo);

        let greeting = "";
        // for-loop
        for (let i = 0; i < repeat; i++) {
                // Merge the strings and add to greeting
                greeting += foo + " " + bar;
        }
        return greeting;
}

// function call
helloWorld(12);
```

Callback functions

- An important concept in JavaScript
- A function passed as an argument to other functions
- Hands over responsibility for capturing data and events to the called function
- Makes up a large part of JavaScript and the interaction with third-party libraries

- “Once you have downloaded the image, do this ...”

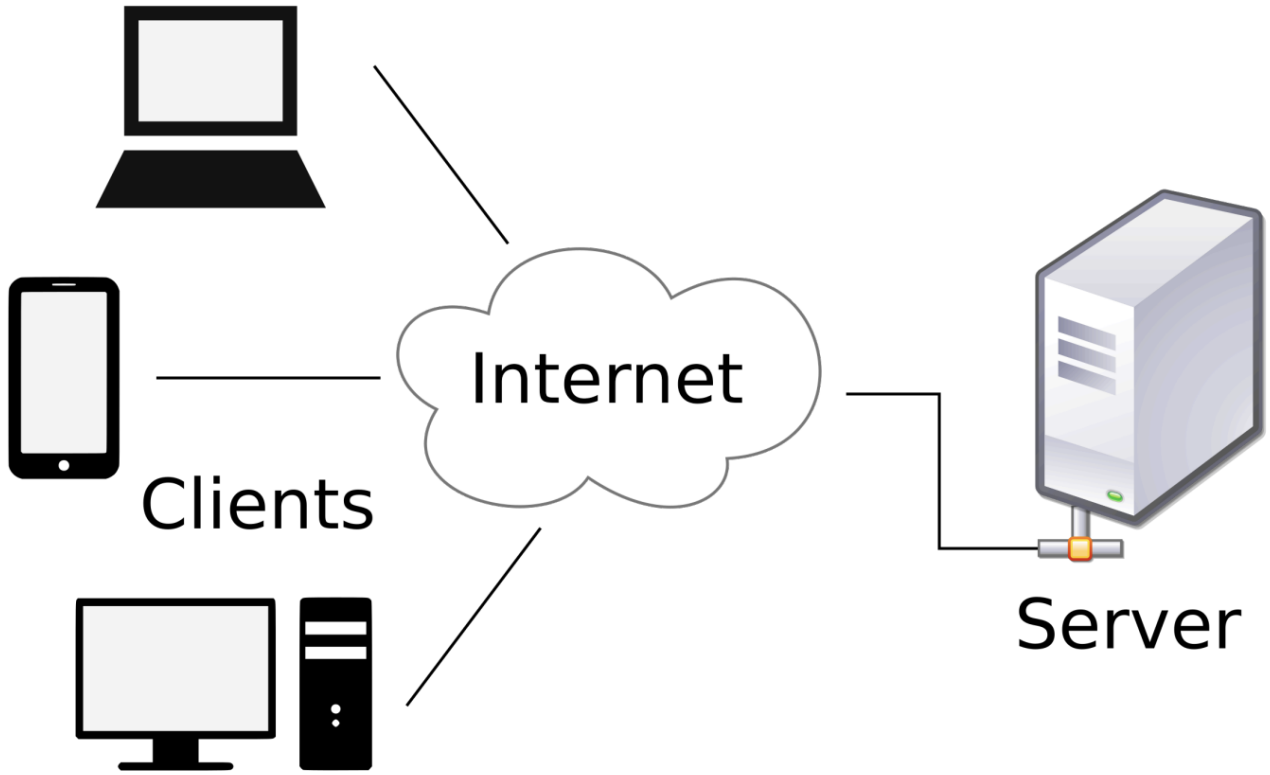
- We will go into more detail about callbacks and asynchronous calls later

Assignment and Scope

- different ways to declare variables
  - var (ES5)
  - let (ES6)
  - const (ES6)
- Differences
  - var, has a scope defined by the nearest function
  - let, const, have a scope defined by the nearest block

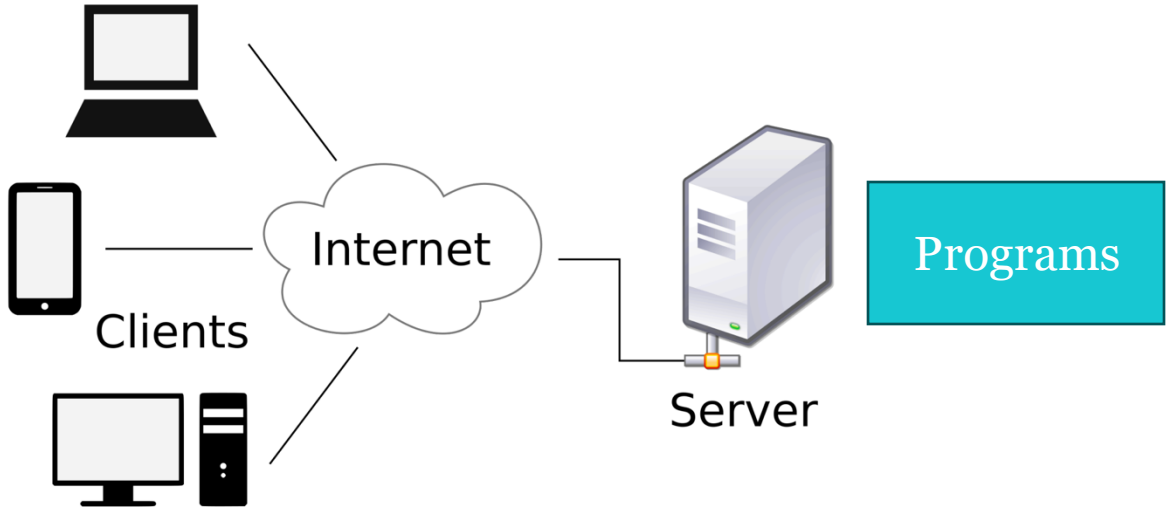
- Example
  - https://jsfiddle.net/08frseu1/43/

Node.js


Internet
Clients
Server

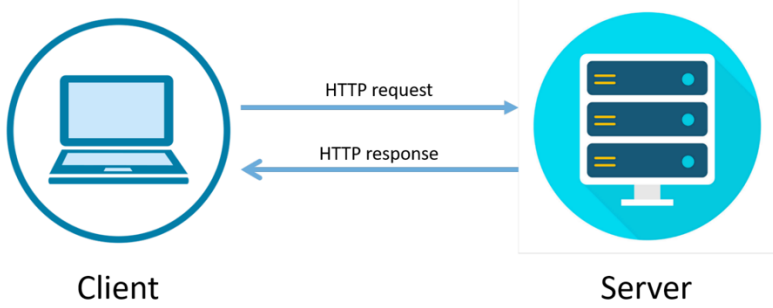
Server side

- Static file server
  - files are never updated
- more options for dynamic web server
  - PHP
  - Python
  - Java
  - Node.js

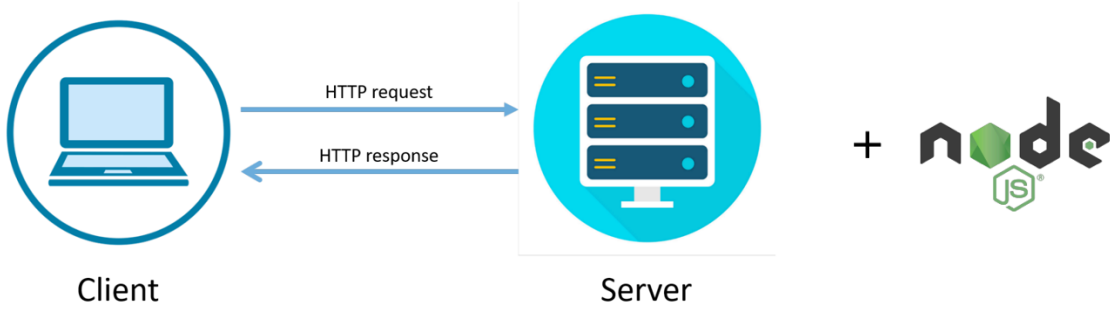


Classical model vs. Node.js

- Classical model
  - A new thread for each incoming call
  - overhead per request



- Node.js
  - only one thread where all requests end up in one event-loop



Node.js

- An environment for running JavaScript on the server
- Built on Chrome V8 JavaScript engine
- JavaScript's event structure with callbacks is widely used in Node.js

- Open source and available via github
  - https://github.com/nodejs/node

Node.js

- Can be used to set up an HTTP server
- The server can receive data sent with POST, GET, DELETE, etc. and return, e.g., JSON
- Node.js can also communicate with a database
  - write data to, or read data from a database

Node.js with JavaScript ES6

- Node.js supports almost everything from ES6
- Import or export modules requires definition of the type in package.json file

Node.js with JavaScript ES6

- Node.js only runs on the server side
- Various frameworks can be used for the front-end and back-end for the development
  - Keep it in mind when you look for resources!
- Installation in different ways (apt install), otherwise nvm (recommended)
  - curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh | bash

Start a new Node.js project

```
# create folder and index.js
mkdir my-app
cd my-app
touch index.js
# Initiate a project
npm init -y
# Install nodemon (so we don't have to restart node)
npm install nodemon
# Add start instruction to package.json under "script""
"start": “nodemon index.js”
```

Code Example

Callback and asynchronous calls

LINKÖPING UNIVERSITY

Event-loop

- Node.js only uses on thread and all requests are executed in this thread
- If Node.js waits for each line of the code to execute before continuing, it means that everyone who made calls to the server need to wait

```
// a function that needs longer running time
let data = ProcessNeedsLongerRunning()
```

If we have such a function call above, the response can be very slow

- Node.js uses Promises to handle asynchronous operations

Asynchronous calls

- Run a function without pausing
- Utilize callbacks or Promises
- Asynchronous functions are marked with async, and return promises

```
async function doSomething(){
        // e.g., time consuming processing
        return "Hello World"
}
```

- To wait on an async function use await
  - wait for a resolved promise, inside an async function
  - can be used to make asynchronous calls behave serially

```
async function main() {
        let a = await doSomething();
        console.log(a);
}
```

Asynchronous calls - Promise

- Object representing a “promise”
- Acts as a placeholder for a result to be available at some point
- 3 states
  - pending: initial state
  - fulfilled: the operation succeeded
  - rejected: the operation failed
- Such an object is created using *“new Promise()”* constructor
  - the constructor takes an argument, i.e., an executor function with 2 arguments
    - resolve: a function to call if the operation succeeds
    - reject: a function to call if the operation fails

Asynchronous calls - Promise

- *.then(...)*
  - this block handles successful resolutions
- *.catch(...)*
  - this block handles rejections that occurred in the promise or any of the .then blocks
- multiple .then(...) can be defined for the same Promise

Asynchronous calls - Promise

```
function loadData(){
        return [
                {'title': 'Gone in 60 seconds', 'year': 2000},
                {'title': 'Pulp Fiction', 'year': 1994}
        ]
}
let p = new Promise((resolve, reject) => {
        let data = loadData()
        if(data !== null){
                resolve(data)
        } else {
                reject('Failed to load data')
        }
})
p.then((x) => {
        // ‘then’ is called if we succeed
        console.log('Data loaded successfully:')
        console.log(JSON.stringify(x, null, 2))
}).catch((msg) => {
        // 'catch' is called if we fail
        console.log(`Something went wrong: ${msg}`)
})
```

What are callbacks?

- Functions as arguments to functions
- Hands over the responsibility for capturing data and events to the called function
- In JavaScript and third-party libraries
- “If I give you my passport, could you pick up the package I ordered, leave it outside my door and then call me?”

Code Example

Express.js

What is Express.js

- A web application framework for Node.js
- Facilitates and speeds up the development of Node.js back-ends
- Easy to get started

```
mkdir app
cd app
npm init
npm install express
```

```
import express from 'express'
const app = express()
app.get('/', function (req, res) {
        res.send('Hello World!')
});
let server = app.listen(3000, () => {
        let host = server.address().address
        let port = server.address().port
        console.log(`Lyssnar på http://${host}:${port}`)
})
```

Frameworks built on Express.js

- Feathers: Build prototypes in minutes and production ready real-time apps in days.
- ItemsAPI: Search backend for web and mobile applications built on Express and Elasticsearch.
- KeystoneJS: Website and API Application Framework / CMS with an auto-generated React.js Admin UI.
- Kraken: Secure and scalable layer that extends Express by providing structure and convention.
- LEAN-STACK: The Pure JavaScript Stack.
- LoopBack: Highly-extensible, open-source Node.js framework for quickly creating dynamic end-toend REST APIs.
- MEAN: Opinionated fullstack JavaScript framework that simplifies and accelerates web application development.
- Sails: MVC framework for Node.js for building practical, production-ready apps.
- Bottr: Framework that simplifies building chatbot applications.
- Hydra-Express: Hydra-Express is a light-weight library which facilitates building Node.js Microservices using ExpressJS.
- Blueprint: Highly-configurable MVC framework for composing production-ready services from reusable components
- Locomotive: Powerful MVC web framework for Node.js from the maker of Passport.js

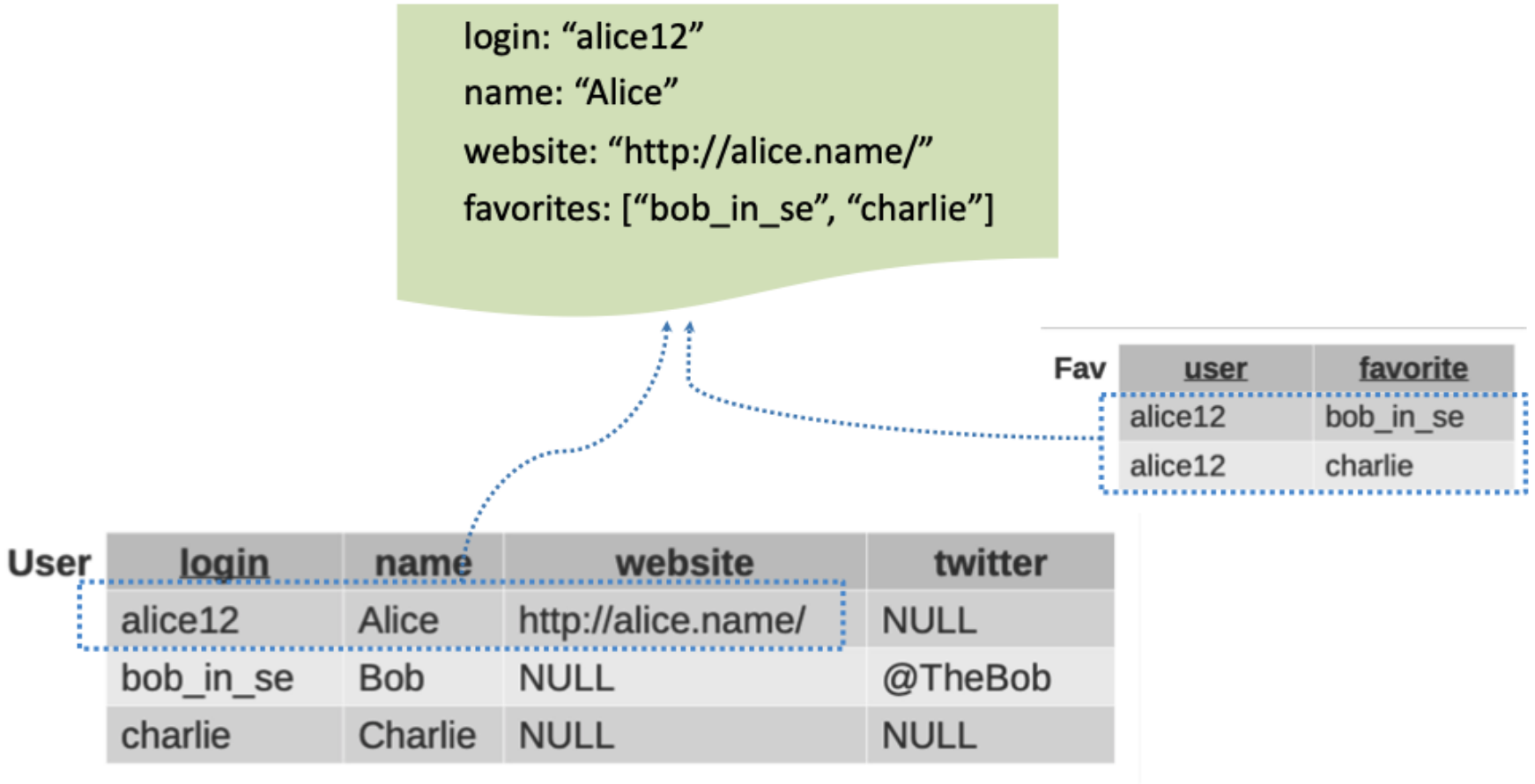
Code Example

MongoDB

MongoDB

- Data Model: Document stores
  - Store data as documents
  - a dictionary of key/value pairs
    - keys are unique
    - values are documents for instance

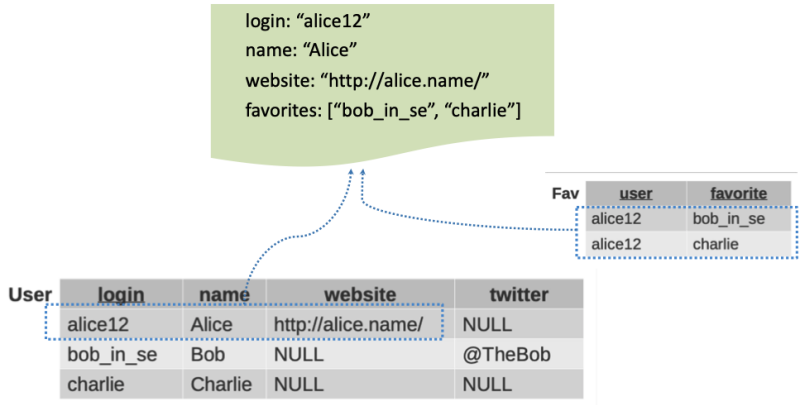
MongoDB


login: “alice12”
name: “Alice”
website: “http://alice.name/”
favorites: [“bob_in_se”, “charlie”]
Fav
user
favorite
alice12
bob_in_se
alice12
charlie
User
login
name
website
twitter
alice12
Alice
http://alice.name/
NULL
bob_in_se
Bob
NULL
@TheBob
charlie
Charlie
NULL
NULL

MongoDB

- Queries:
  - querying in terms of conditions on document content

```
const cursor = db.collection(User').find({ twitter: {$ne : null} });
```



Lab 1

LINKÖPING UNIVERSITY

Lab 1

- Build a Web server using Node.js, Express.js, MongoDB
- Test your code and code coverage with Mocha framework, Istanbul/Istanbul nyc

- Register in pairs on webreg, deadline tomorrow (September 3rd)
- Scheduled sessions on September 3rd, 5th, 10th and demonstration on 12th
- Code submission deadline is September 12th 23:59 CEST (send an email with you gitlab repo to lab assistant)

- More information on:
  - https://www.ida.liu.se/~TDP013/laborationer/lab1.25.sv.shtml

LINKÖPING UNIVERSITY