Software Development Concepts

Fundamental Concepts and Paradigms
in the Software Engineering Profession

SoftUni Team

Technical Trainers

Software University

Software University
https://softuni.bg

SoftUni

Table of Contents

Table of Contents

Table of Contents

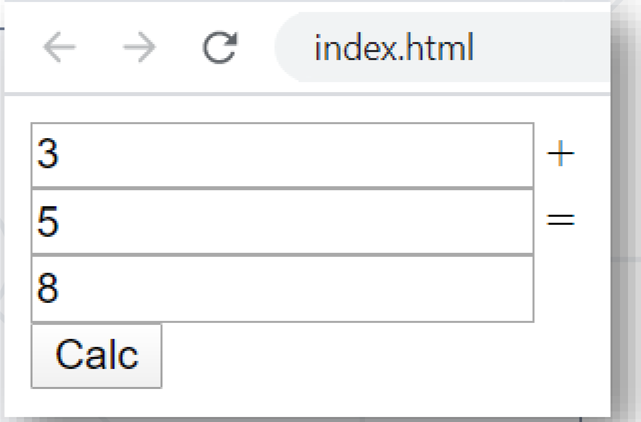
Front-End

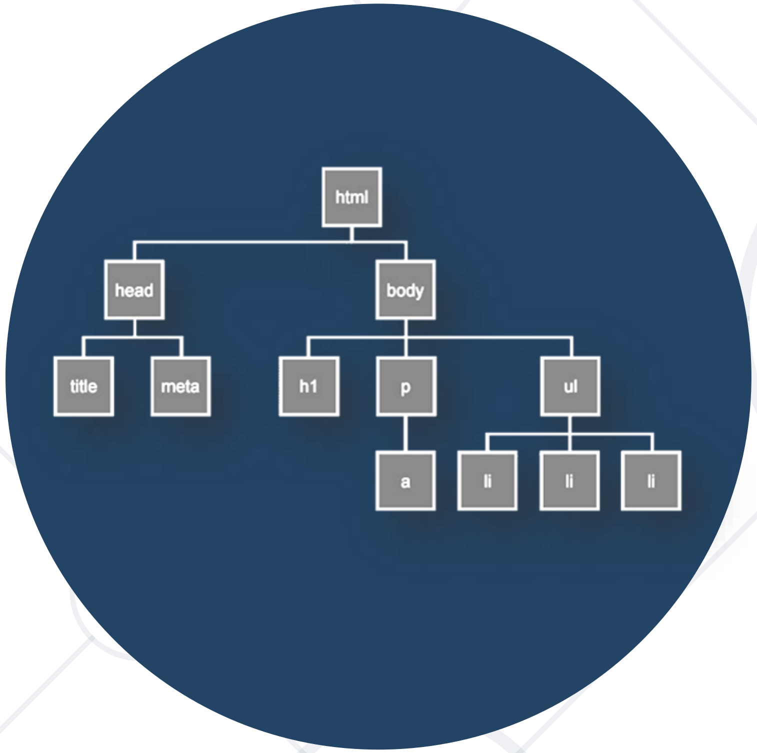
Web Front-End and DOM

- **Web front-end technologies** (see *https://platform.html5.org*)
  - HTML, CSS, JavaScript, DOM, AJAX
  - JS front-end frameworks (e.g. React, Angular, Vue)
- **DOM** (the Document Object Model)
  - DOM == a tree of UI and other elements
  - Documents in the Web browser are represented by a **DOM tree**
  - The **DOM API** allows changing the DOM from JS

Using the DOM API – Example

```
<input type="text" id="firstNum" /> +
<input type="text" id="secondNum" /> =
<input type="text" id="sum" />
<button id="calc">Calc</button>
<script>
  document.getElementById("calc").onclick = function() {
    document.getElementById("sum").value =
      Number(document.getElementById("firstNum").value) +
      Number(document.getElementById("secondNum").value);
  }
</script>
```





DOM Interaction

Live Demo

https://repl.it/@nakov/summator-js-dom

AJAX and RESTful APIs

- **AJAX** is a technology for asynchronous execution of HTTP requests from client-side JavaScript

```
let httpRequest = fetch('https://some-url…');
httpRequest.then(function(httpResponse) {
  // Process the HTTP response here and update the DOM tree …
});
```

- **RESTful APIs** are HTTP-based Web services
  - The HTTP methods **GET**, **POST**, **PUT** and **DELETE** retrieve, create, modify and delete data

AJAX and REST

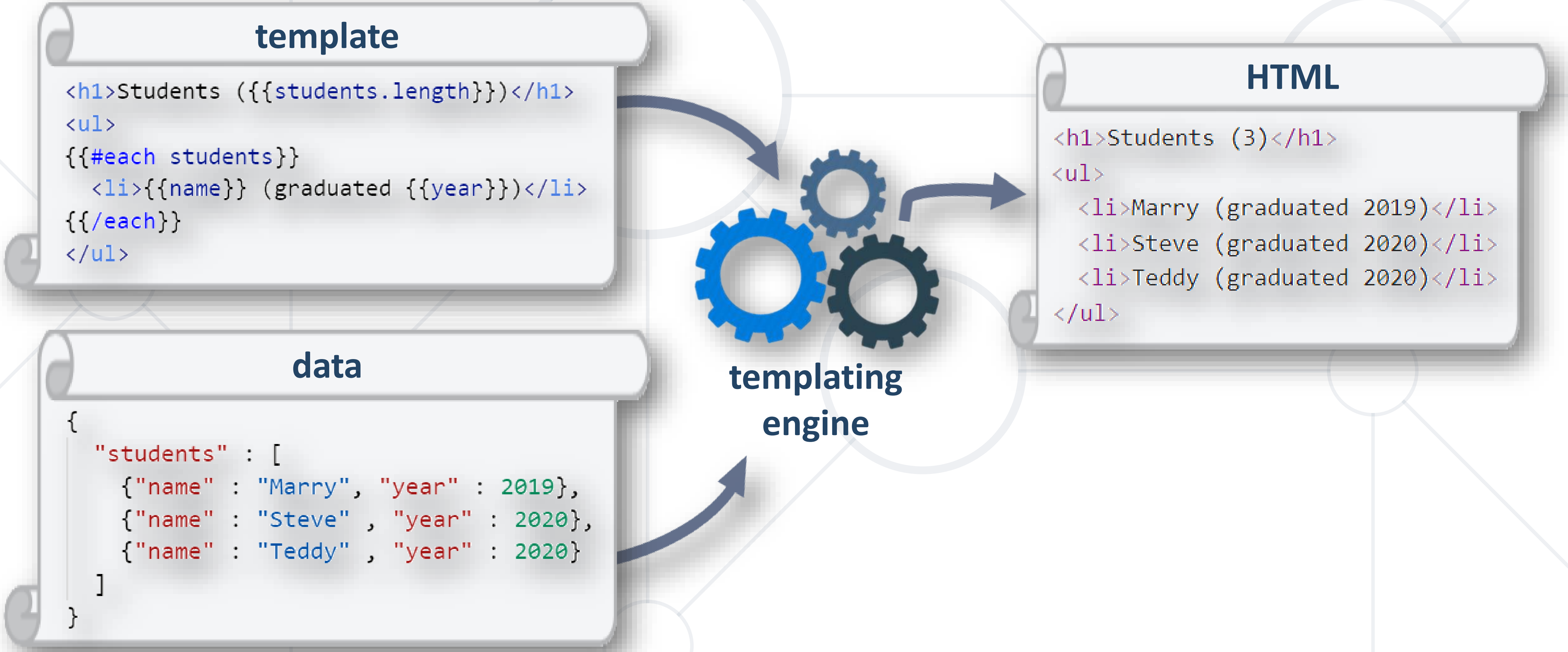
Live Demo

https://repl.it/@nakov/RESTful-API-js

https://repl.it/@nakov/RESTful-API-client-example

Templating Engines

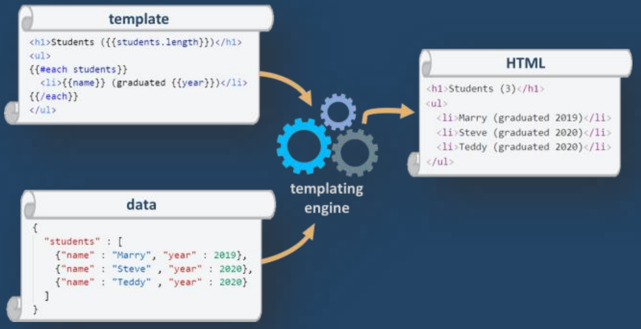
- **Templating engines** render data as HTML through a **template**



```
<h1>Students ({{students.length}})</h1>
<ul>
{{#each students}}
  <li>{{name}} (graduated {{year}})</li>
{{/each}}
</ul>
```

```
{
  "students" : [
    {"name" : "Marry", "year" : 2019},
    {"name" : "Steve" , "year" : 2020},
    {"name" : "Teddy" , "year" : 2020}
  ]
}
```

```
<h1>Students (3)</h1>
<ul>
  <li>Marry (graduated 2019)</li>
  <li>Steve (graduated 2020)</li>
  <li>Teddy (graduated 2020)</li>
</ul>
```



```
<h1>Students ({{students.length}})</h1>
<ul>
{{#each students}}
  <li>{{name}} (graduated {{year}})</li>
{{/each}}
</ul>
```

```
{
  "students" : [
    {"name" : "Marry", "year" : 2019},
    {"name" : "Steve" , "year" : 2020},
    {"name" : "Teddy" , "year" : 2020}
  ]
}
```

```
<h1>Students (3)</h1>
<ul>
  <li>Marry (graduated 2019)</li>
  <li>Steve (graduated 2020)</li>
  <li>Teddy (graduated 2020)</li>
</ul>
```

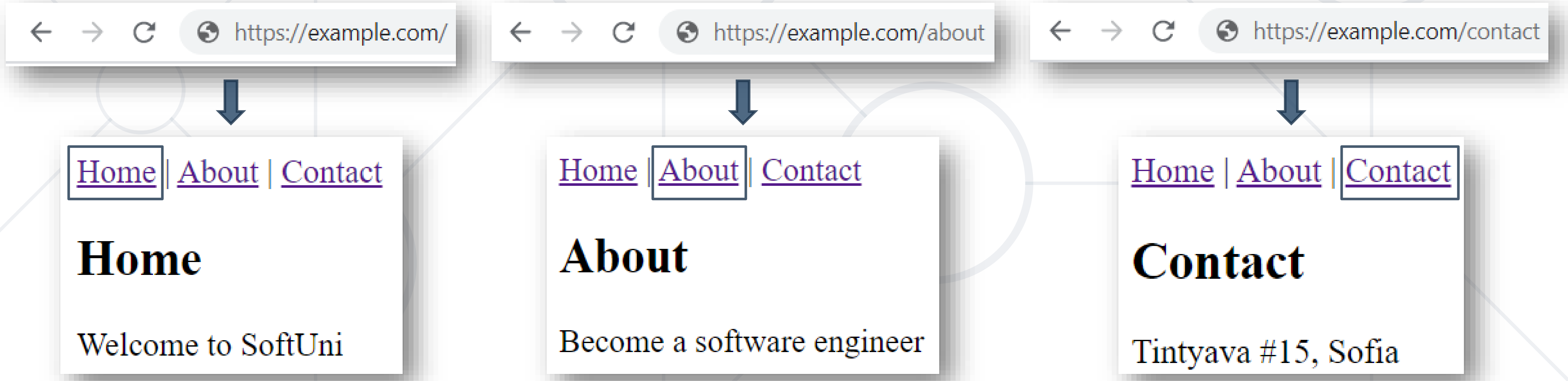
Rendering UI with a Templating Engine

Live Demo

https://repl.it/@nakov/Handlebars-example-JS

Routing and Routing Libraries

- **Routing** is about switching between different **UI views**, based on the changes of the current **URL** (holding the route)
- **Routing libraries** switch the view by URL like this:



Home

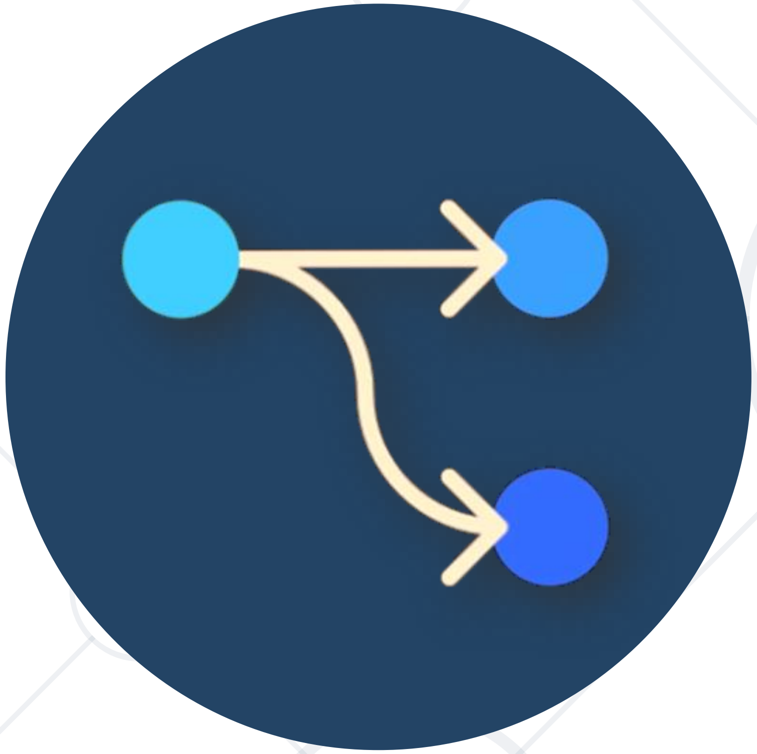
Welcome to SoftUni

About

Become a software engineer

Contact

Tintyava #15, Sofia



Navigation with Routing Library

Live Demo

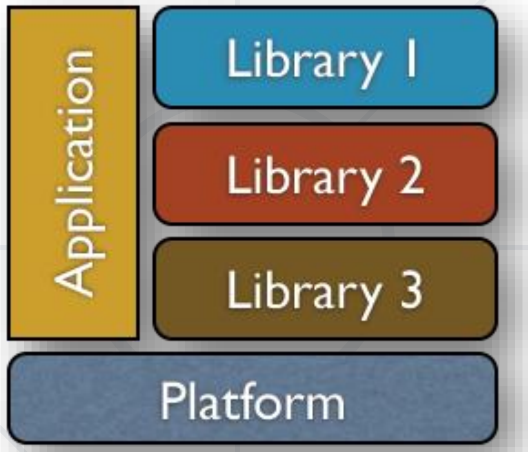
https://repl.it/@nakov/routing-with-sammy-js

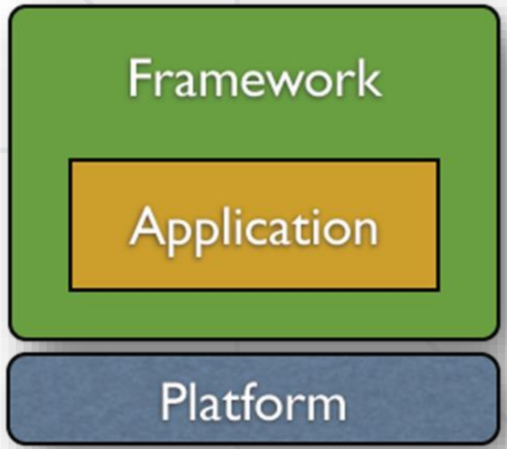
User Interface and Front-End Frameworks

- **Graphical User Interface** (GUI) systems provide forms, dialogs and UI controls for desktop and mobile apps
  - Examples: Windows Forms, XAML, WPF, Qt
- **Mobile UI** toolkits / frameworks provide UI controls and structure for mobile apps
  - Examples: Apple UIKit, Android UI, Flutter
- **Web front-end frameworks** and **UI libraries** provide user interface elements and structure for **Web apps**
  - Examples of **UI frameworks**: Angular, React, Vue.js, Meteor
  - Examples of **UI libraries**: Kendo UI, Sencha Ext JS, Onsen UI

Libraries vs. Frameworks

- **Libraries** provide **components** / **functionality** / **UI controls** for integration into existing apps
  - The **app controls the library** components
  - Examples: UI control library, Excel reader
- Development **frameworks** are foundations, which developers extend to build an app
  - The framework **controls the app lifecycle** and your code plugs in it (**inversion of control** – IoC)
  - Examples: MVC framework, ORM framework

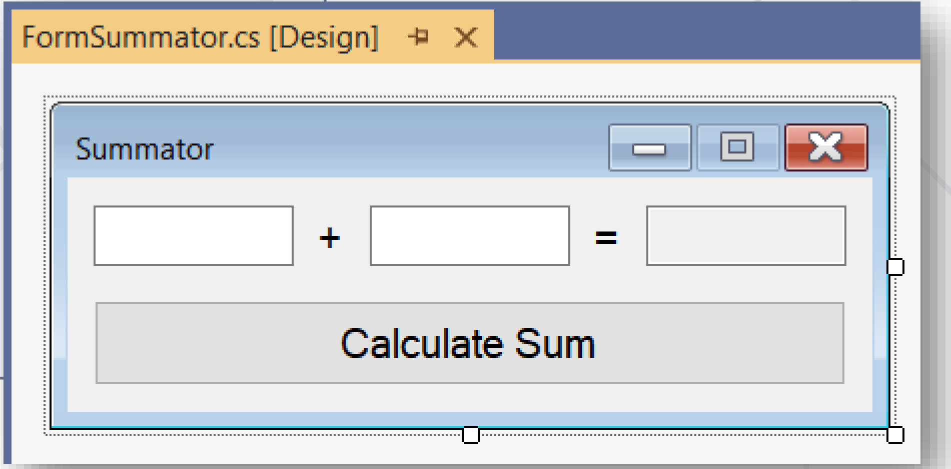




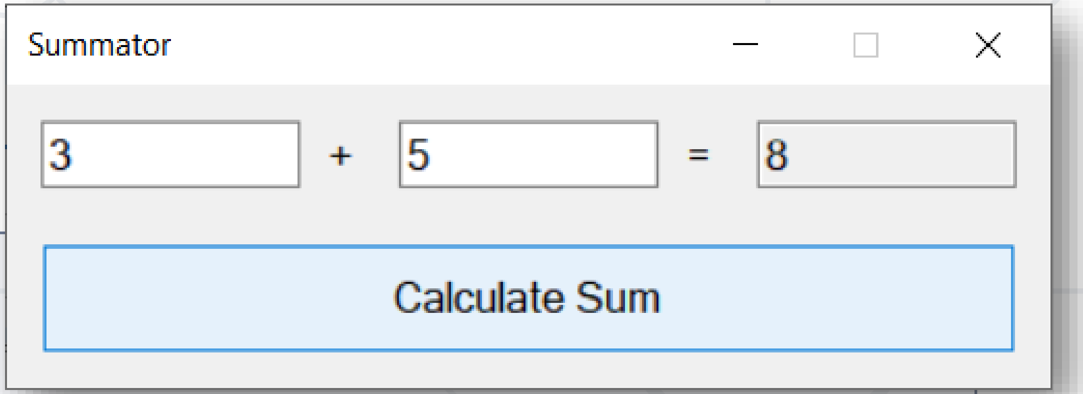
Windows Forms – Example

- **Windows Forms** is GUI framework for .NET developers
  - Provides programming model and rich UI control library

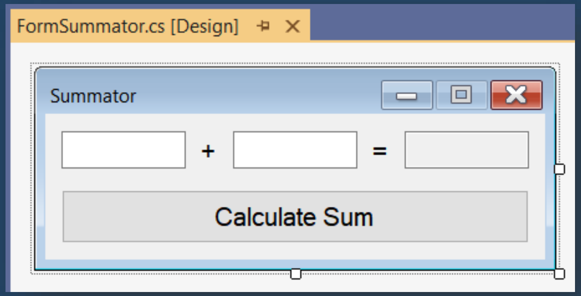
```
public partial class FormSummator : Form
{
  private TextBox textBox1;
  private Label labelPlus;
  private Label labelEqual;
  private TextBox textBox2;
  private TextBox textBoxSum;
  private Button buttonCalc;
}
```



Windows Forms – Example



```
public partial class FormSummator
{
  private void buttonCalc_Click(object sender, EventArgs e)
  {
    decimal firstNum = decimal.Parse(this.textBox1.Text);
    decimal secondNum = decimal.Parse(this.textBox2.Text);
    decimal sum = firstNum + secondNum;
    this.textBoxSum.Text = sum.ToString();
  }
}
```



Windows Forms

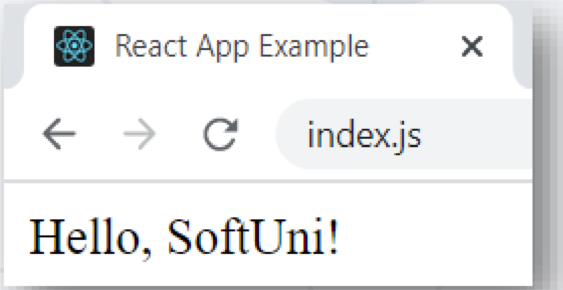
Live Demo

React

- **React** is a powerful **JavaScript library** from Facebook for building Web UI using HTML, CSS and JS
  - The UI is built from **JSX components**, which combine HTML + JS

```
class HelloMessage extends React.Component {
  render() {
    return (<div>Hello, {this.props.name}!</div>);
  }
}

ReactDOM.render(<HelloMessage name="SoftUni" />,
  document.getElementById('root'));
```

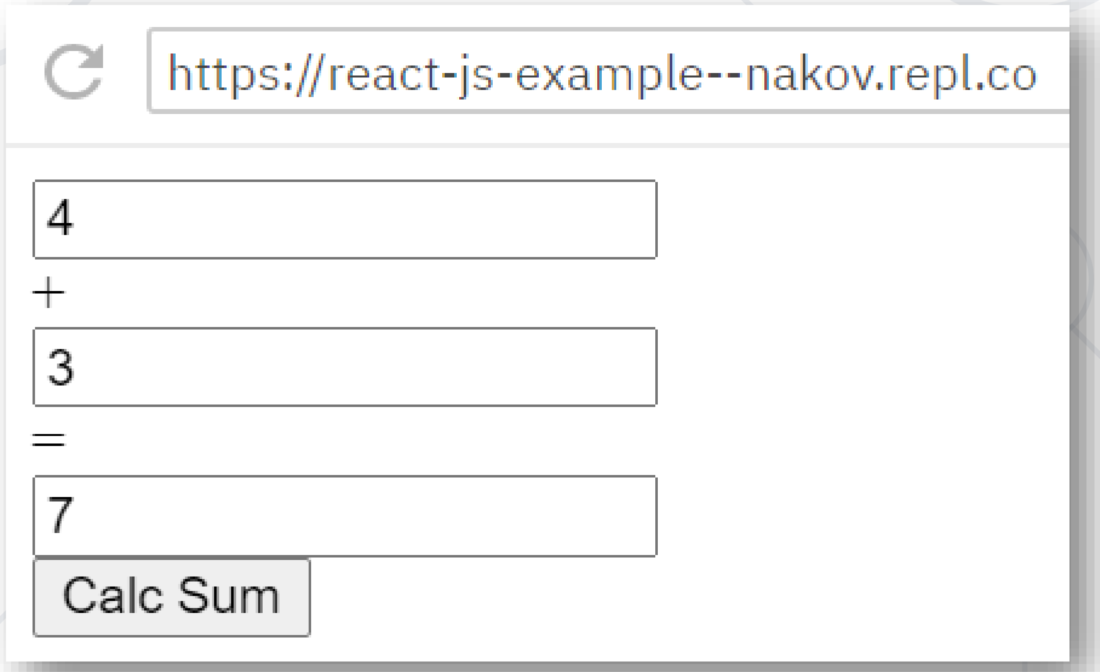


Hello, SoftUni!

React

Live Demo

https://repl.it/@nakov/react-js-example



https://react-js-example--nakov.repl.co

4

+

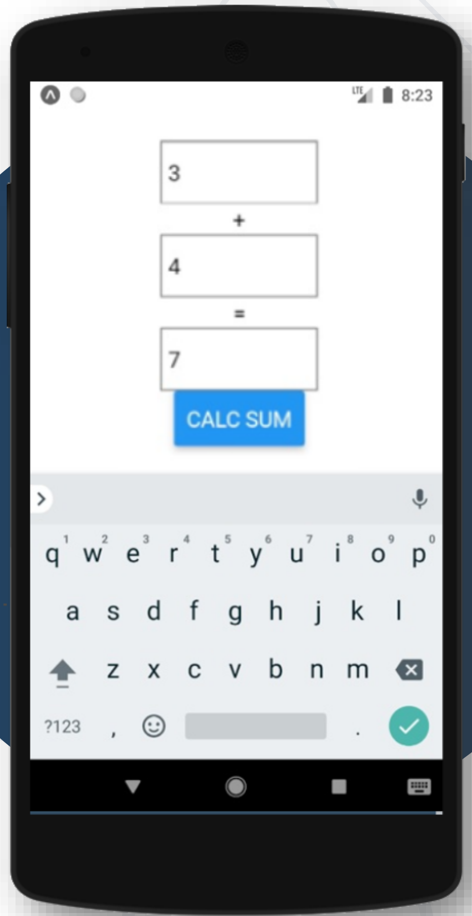
3

=

7

Mobile Apps – Technologies

- Two major mobile app platforms: **Android** and **iOS**
- **Mobile app** development technologies
  - **Android**: Java / Kotlin + Android SDK + Android Studio
  - **iOS**: Swift (or Objective-C) + iOS SDK + Xcode + Mac
  - **Hybrid mobile apps**: JS + HTML5 + WebView (e.g. Cordova)
  - **Native JS mobile apps**: JavaScript + native UI
    - Examples: React Native, NativeScript
  - **Others**: Xamarin (C#), Flutter (Dart)



React Native App

Live Demo

https://snack.expo.io/@nakov/summator-react-native

Back-End

Back-End Technologies

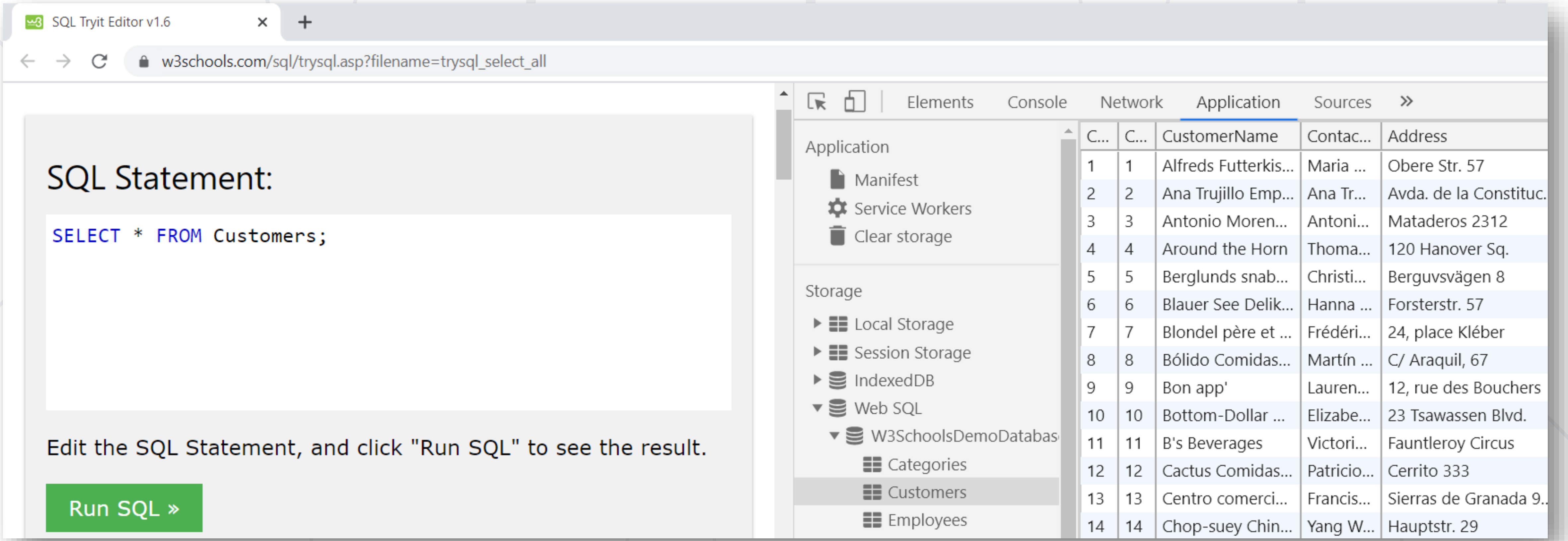
- **Back-end technologies** are about server-side programming
  - **Data management** technologies and **ORM frameworks**
  - Backend **Web frameworks** and **MVC** frameworks
  - **REST API** frameworks, **reactive** APIs, other services and APIs
  - **Microservices**, **containers** and **cloud**
- **Back-end developers** work on the server-side
  - They deal with the business logic,
    data processing, data storage, APIs

Databases

- **Databases** hold and manage data in the back-end systems
- **Relational databases** (RDBMS)
  - Hold data in **tables + relationships**
  - Use the **SQL** language to query / modify data
  - Examples: MySQL, PostgreSQL, Web SQL in HTML5
- **NoSQL databases**
  - Hold collections of documents or key-value pairs
  - Examples: MongoDB, IndexedDB in HTML5

Web SQL – Example

- **Web SQL** is a relational database, embedded the Web browsers
  - It is fully functional **RDBMS system**, runs at the **client-side**



SQL Statement:

```
SELECT * FROM Customers;
```

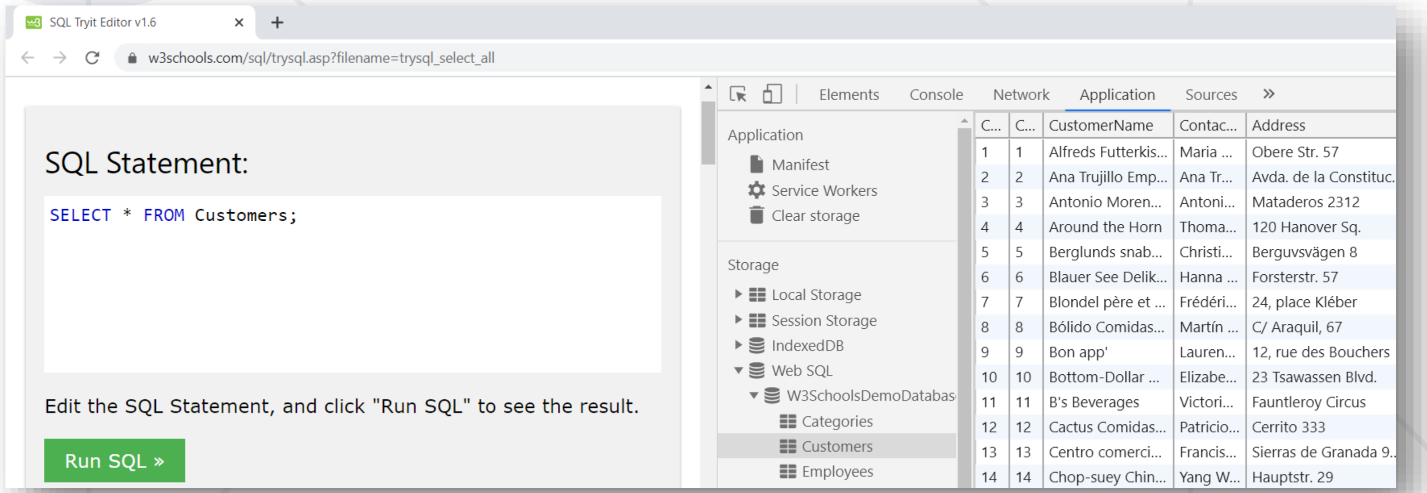
Edit the SQL Statement, and click "Run SQL" to see the result.

C...	C...	CustomerName	Contac...	Address
1	1	Alfreds Futterkis...	Maria ...	Obere Str. 57
2	2	Ana Trujillo Emp...	Ana Tr...	Avda. de la Constituc.
3	3	Antonio Moren...	Antoni...	Mataderos 2312
4	4	Around the Horn	Thoma...	120 Hanover Sq.
5	5	Berglunds snab...	Christi...	Berguvsvägen 8
6	6	Blauer See Delik...	Hanna ...	Forsterstr. 57
7	7	Blondel père et ...	Frédéri...	24, place Kléber
8	8	Bólido Comidas...	Martín ...	C/ Araquil, 67
9	9	Bon app'	Lauren...	12, rue des Bouchers
10	10	Bottom-Dollar ...	Elizabe...	23 Tsawassen Blvd.
11	11	B's Beverages	Victori...	Fauntleroy Circus
12	12	Cactus Comidas...	Patricio...	Cerrito 333
13	13	Centro comerci...	Francis...	Sierras de Granada 9..
14	14	Chop-suey Chin...	Yang W...	Hauptstr. 29

Web SQL

Live Demo

https://w3schools.com/sql/



SQL Statement:

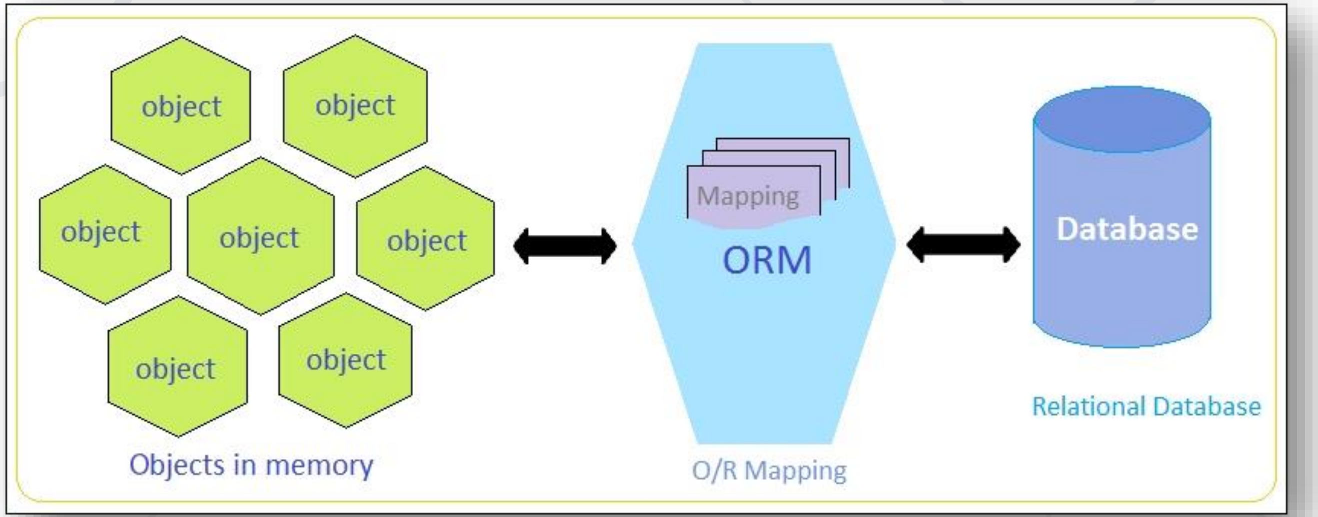
```
SELECT * FROM Customers;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

C...	C...	CustomerName	Contac...	Address
1	1	Alfreds Futterkis...	Maria ...	Obere Str. 57
2	2	Ana Trujillo Emp...	Ana Tr...	Avda. de la Constituc.
3	3	Antonio Moren...	Antoni...	Mataderos 2312
4	4	Around the Horn	Thoma...	120 Hanover Sq.
5	5	Berglunds snab...	Christi...	Berguvsvägen 8
6	6	Blauer See Delik...	Hanna ...	Forsterstr. 57
7	7	Blondel père et ...	Frédéri...	24, place Kléber
8	8	Bólido Comidas...	Martín ...	C/ Araquil, 67
9	9	Bon app'	Lauren...	12, rue des Bouchers
10	10	Bottom-Dollar ...	Elizabe...	23 Tsawassen Blvd.
11	11	B's Beverages	Victori...	Fauntleroy Circus
12	12	Cactus Comidas...	Patricio...	Cerrito 333
13	13	Centro comerci...	Francis...	Sierras de Granada 9..
14	14	Chop-suey Chin...	Yang W...	Hauptstr. 29

ORM Frameworks

- **ORM frameworks** (object-relational mapping) allow persisting objects in relational database (by mapping classes to tables)
  - e.g., store JS objects in MySQL database
- Popular ORM frameworks:
  - **Entity Framework** (C#)
  - **Hibernate** (Java)
  - **Sequelize** (JavaScript)
  - **SQLAlchemy** (Python)



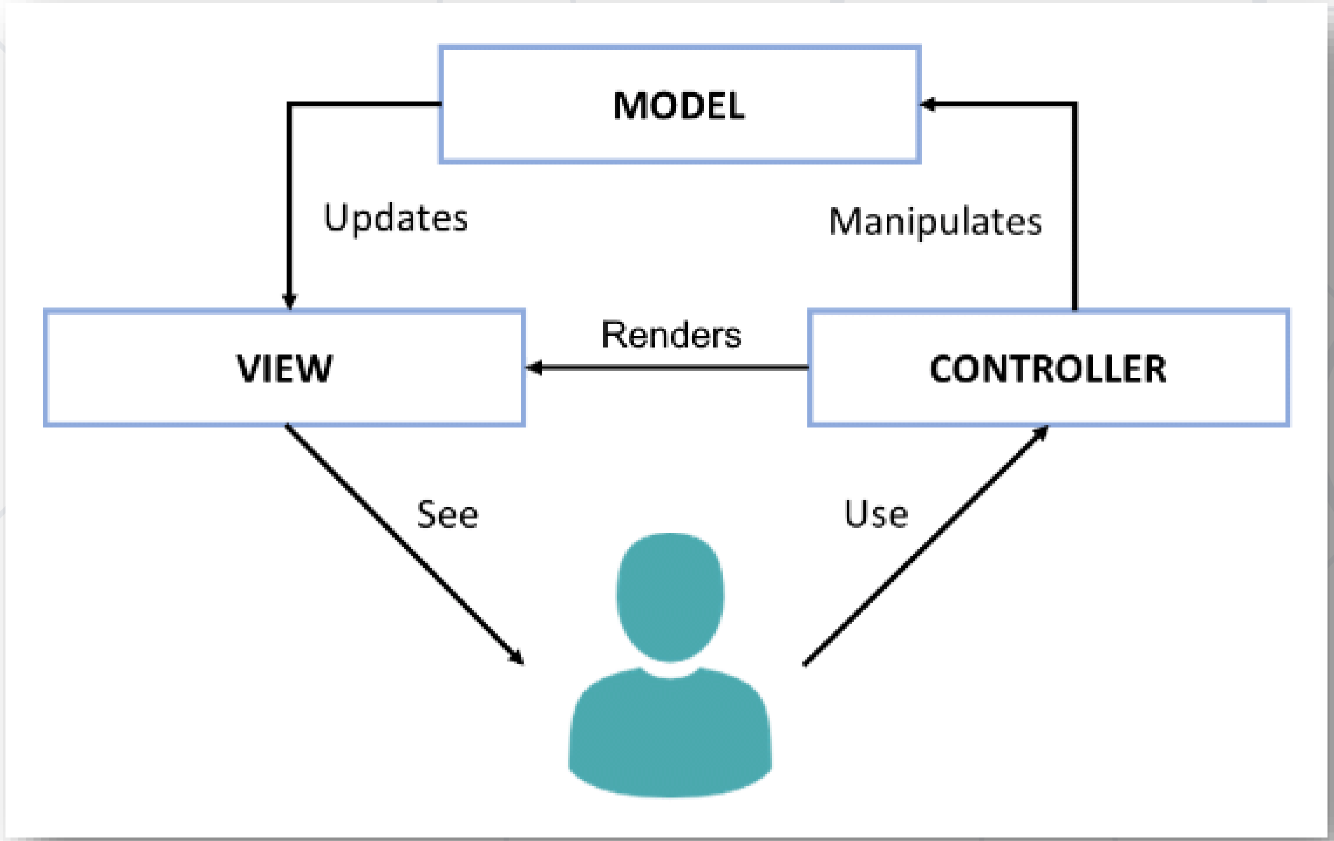
JayData ORM for Web SQL

Live Demo

https://repl.it/@nakov/jaydata-orm-example

The Model-View-Controller (MVC) Pattern

- The **Model-View-Controller** (MVC) pattern



- **Controller**
  - Handles user actions
  - Updates the model
  - Renders the view (UI)
- **Model**
  - Holds app data
- **View**
  - Displays the UI, based on the model data

Web MVC Frameworks

- **Web MVC frameworks** are used build Web applications
  - **Controllers** handle HTTP GET / POST and render a view
  - **Views** display HTML + CSS, based on the models
  - **Models** hold app data for views, prepared by controllers
- Examples of Web MVC frameworks
  - ASP.NET MVC (C#), Spring MVC (Java), Express (JS), Django (Python), Laravel (PHP), Ruby on Rails (Ruby), Revel (Go), …

MVC Frameworks

Live Demo

https://repl.it/@nakov/MVC-express-pug-example

Virtualization and Cloud

- **Virtualization** == running a **virtual machine** (VM) / virtual environment inside a physical hardware system
  - e.g., run Android VM or Linux inside a Windows host
  - Storage, memory, networking, desktops can also be virtual
- **Cloud** == computing resources, virtual machines, storage, platforms and software instances, available on demand
  - **IaaS** (infrastructure as a service) – virtual machines on demand
  - **PaaS** (platform as a service) – app deployment environments
  - **SaaS** (software as a service) – software instances, e.g. Office 365

Containers and Docker

- **Container image** == software, packaged with its dependencies, designed to run in a virtual environment (like Docker)
  - e.g., WordPress instance (Linux + PHP + Apache + WordPress)
  - Simplified installation, configuration and deployment
- **Docker** is the most popular containerization platform
  - Runs **containers** from local **image** or downloaded from the **Docker Hub** online repository
  - Open-source, runs on Linux, Windows, Mac

Docker – Example

- Install **Docker** on your local computer
  - Or use the Docker online playground: *https://labs.play-with-docker.com* (with a free Docker Hub registration)
- Download and **run a Docker image** in a new container:

```
docker run -d -p:8080:80 dockersamples/static-site
```

- Open the exposed URL: *http://localhost:8080*
- View currently running Docker containers

```
docker ps
```

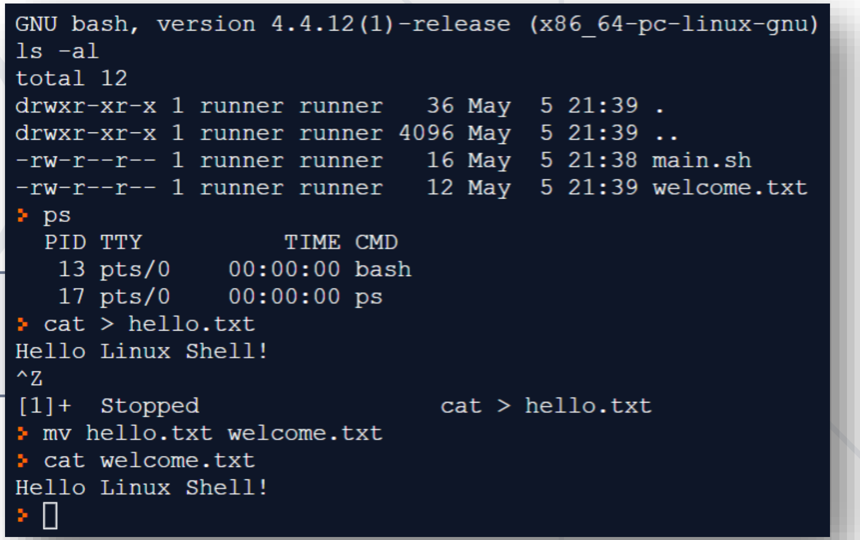
Play with Docker

Live Demo

https://labs.play-with-docker.com

Operating Systems

- Working with **operating systems** (Linux, Windows, others) is an important skill for software engineers
  - Installation, configuration and basic system administration
  - Process management, file system, users and permissions
- Sample **Linux shell commands**
  - Create a file: `cat > hello.txt`
  - Rename a file: `mv hello.txt welcome.txt`
  - View file contents: `cat welcome.txt`



```
GNU bash, version 4.4.12(1)-release (x86_64-pc-linux-gnu)
ls -al
total 12
drwxr-xr-x 1 runner runner   36 May  5 21:39 .
drwxr-xr-x 1 runner runner 4096 May  5 21:39 ..
-rw-r--r-- 1 runner runner   16 May  5 21:38 main.sh
-rw-r--r-- 1 runner runner   12 May  5 21:39 welcome.txt
› ps
  PID TTY          TIME CMD
   13 pts/0    00:00:00 bash
   17 pts/0    00:00:00 ps
› cat > hello.txt
Hello Linux Shell!
^Z
[1]+  Stopped                 cat > hello.txt
› mv hello.txt welcome.txt
› cat welcome.txt
Hello Linux Shell!
› 
```

Linux Shell Commands

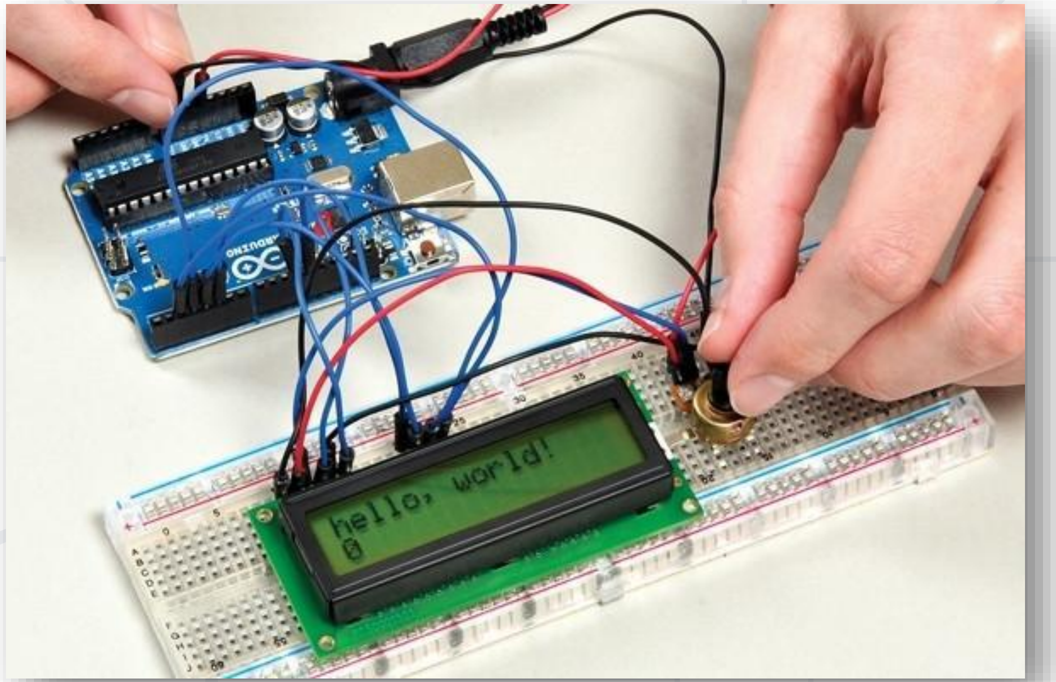
Live Demo

https://repl.it/@nakov/bash-shell-example

Internet of Things (IoT)

Embedded Systems and IoT

- **Embedded systems**
  - **Hardware + software**, dedicated to certain task, e.g. control the lights or the heating at home
  - The hardware has **limited resources** (CPU, RAM, battery, …)
- **Internet-connected** embedded systems are known as "Internet of Things" devices (IoT devices)



IoT Microcontrollers

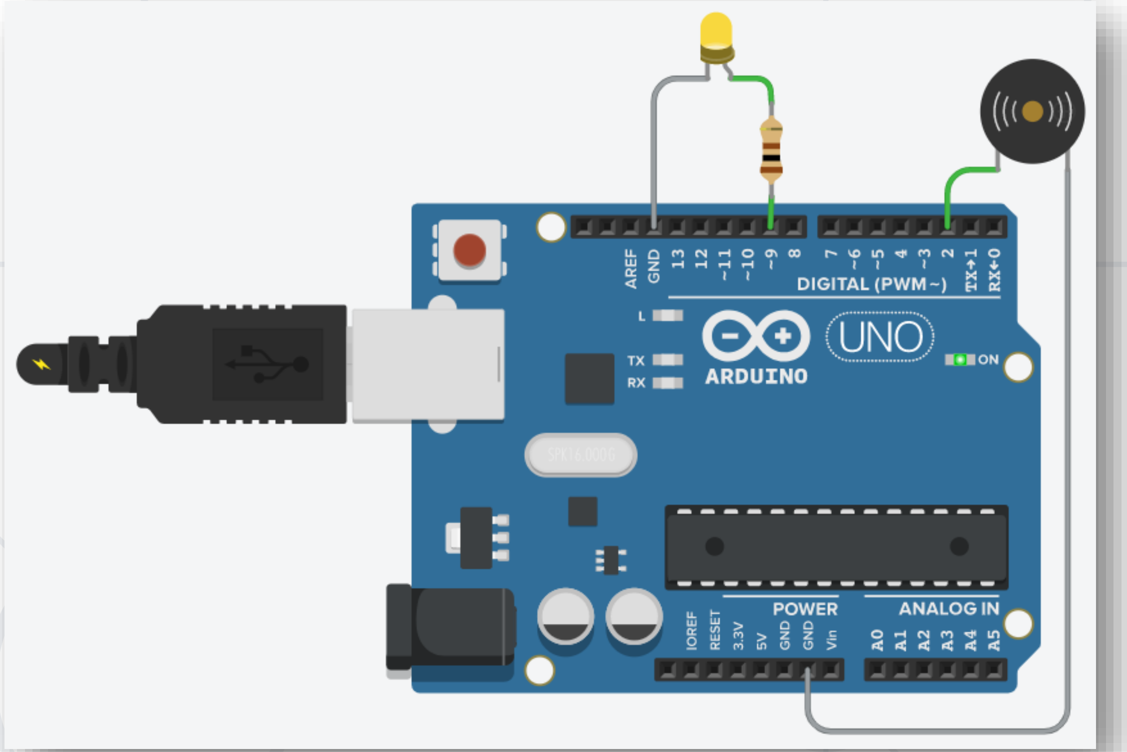
- **Microcontrollers** == microchip (CPU + RAM + GPIO) on a board
  - Examples: Arduino, ESP8266, ESP32, Micro:bit, ATmega328
- **IoT systems** consist of **microcontroller** (or mini-computer) + peripheries + software + Internet connectivity + back-end
  - **Peripherals**: LED lights, buttons, sensors, buzzers, relays, displays
  - **Back-end**: cloud-based (e.g. Blynk, Thinger) or local (home computer)
  - **Connectivity**: WiFi, Bluetooth, LoRa, 4G LTE (with SIM card), 5G
- **Programming languages** for IoT devices:
  - C, C++, JS / Python / C# (some devices)

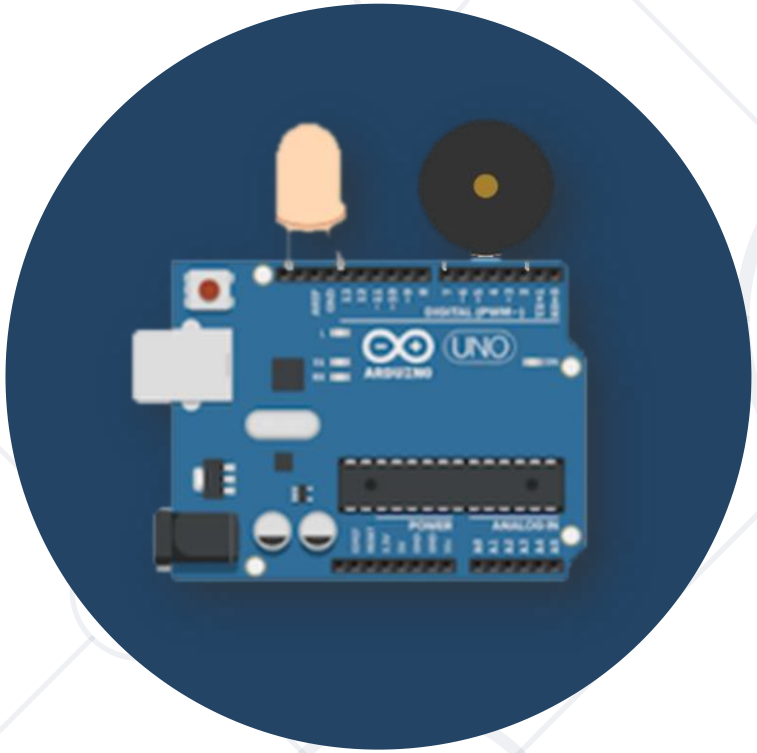
Microcontroller Arduino – Example

```
#define LED_PIN 9
#define BUZZER_PIN 2

void setup() {
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  int brightness = 0;
  while (brightness <= 255) {
    analogWrite(LED_PIN, brightness);
    delay(15);
    brightness += 3;
  }
  tone(BUZZER_PIN, 300, 100);
}
```





Arduino @ Tinkercad

Live Demo

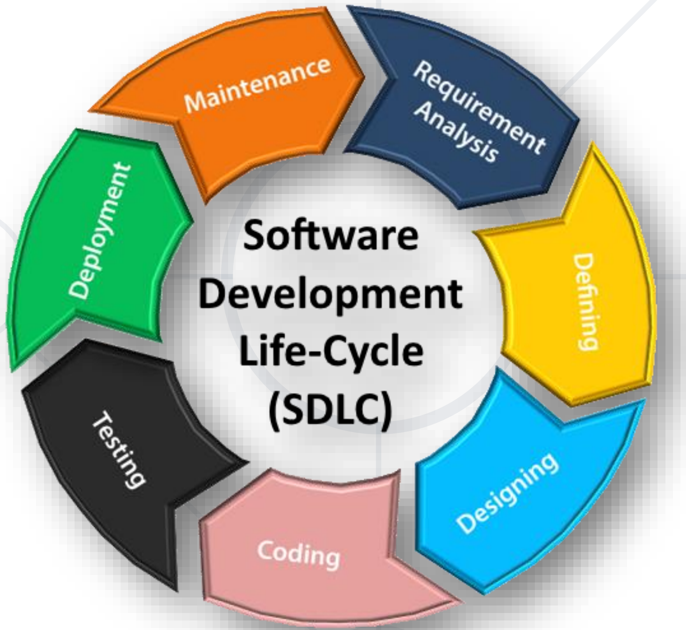
https://www.tinkercad.com/things/hjgbxEoS5TX

Software Engineering

Software Development Lifecycle (SLDC)

- **Software engineering** is not just coding!
- The **SDLC** includes the following activities:
  - **Requirements** analysis
  - Software **design**
  - **Construction**
  - **Testing**
  - **Release**
  - **Maintenance**

Software project **management**



- **Development processes** (Waterfall / Scrum / Kanban) define workflow and key practices

Software Quality Assurance (QA)

- What is **software quality assurance** (QA)?
  - Ensures the **software quality**
  - Performed by the **QA engineers**
- Two approaches
  - **Testing** (manual and automated)
  - **Code reviews** and quality inspections
- Goal: to **find** and report the **defects** (bugs)
  - Defect are tracked in an **issue tracker**

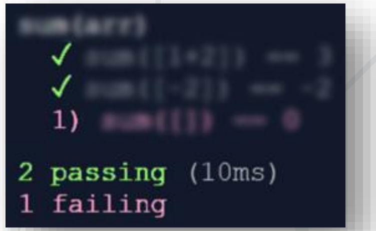
Issue Tracker

Live Demo

https://github.com/twbs/bootstrap/issues

Unit Testing

- **Unit test** == a piece of code that tests specific functionality in certain software component (unit)



```
sum(arr)
  ✓ sum([1+2]) == 3
  ✓ sum([-2]) == -2
  1) sum([]) == 0

2 passing (10ms)
1 failing
```

```
function testSum() {
  if (sum([1, 2]) != 3)
    throw "1+2 != 3";
  if (sum([-2]) != -2)
    throw "-2 != -2";
  if (sum([]) != 0)
    throw "empty sum != 0";
}
```

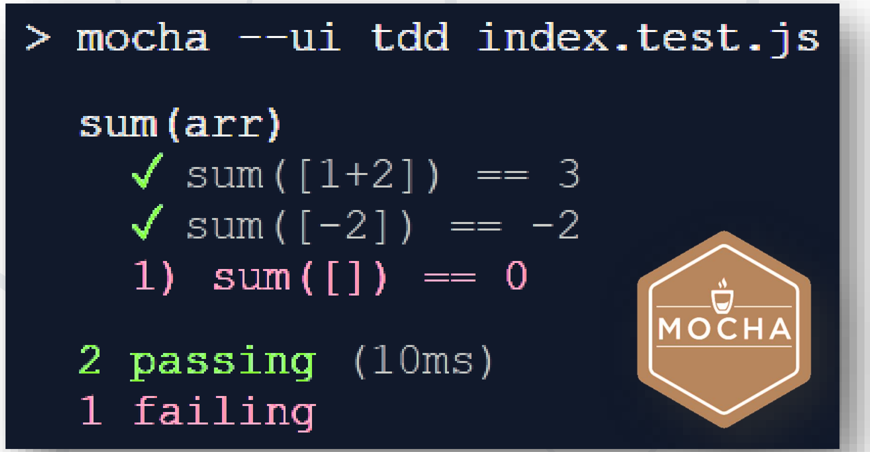
```
function sum(arr) {
  let sum = 0;
  for (let item of arr)
    sum += item;
  return sum;
}
```

Unit Testing Framework

- **Unit testing frameworks** simplify unit testing and reporting
  - Example: **Mocha** JS testing framework

```
const assert = require('assert');

suite('sum(arr)', function() {
  test('sum([1+2]) == 3', function() {
    assert.equal(sum([1, 2]), 3); });
  test('sum([-2]) == -2', function() {
    assert.equal(sum([-2]), -2); });
  test('sum([]) == 0', function() {
    assert.equal(sum([]), 0); });
});
```



```
> mocha --ui tdd index.test.js

  sum(arr)
    ✓ sum([1+2]) == 3
    ✓ sum([-2]) == -2
    1) sum([]) == 0

  2 passing (10ms)
  1 failing
```

Unit Testing with Mocha

Live Demo

https://repl.it/@nakov/mocha-unit-test-example-js

Source Control Systems

- **Source control systems** keep the source code (+ other project assets) in a shared **repository**
  - Developers can **clone** a repository, **pull** the latest version, **commit** & **push** local changes, view the change logs, etc.
- **Git** is the most popular source control system
  - Other version control systems: SVN, TFS, Perforce
- **GitHub** is the #1 site for Git project hosting
  - Git hosting + issue tracker + project tracker + build system

GitHub – Example

- Clone a repository from GitHub

```
git clone https://github.com/SoftUni/playground
```

- Modify local files

```
notepad README.md
```

- Commit changes (local)

```
git add . & git commit -m "Added something"
```

- Push the changes to GitHub

```
git push
```

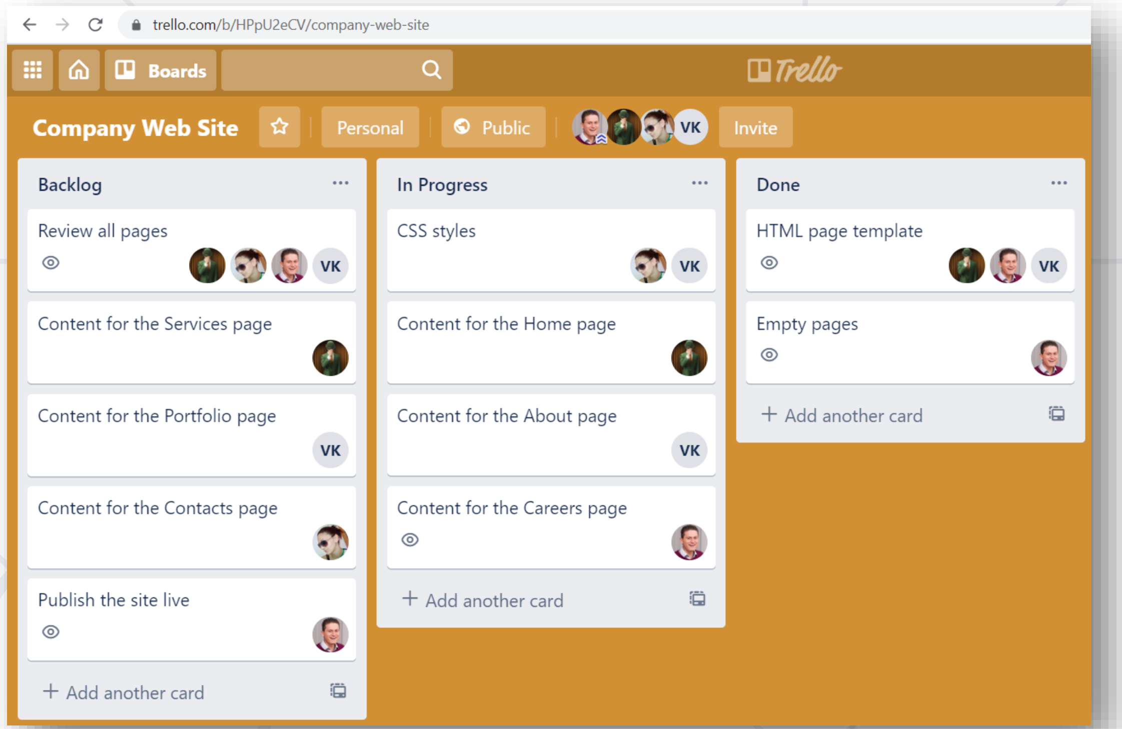
Git and GitHub

Live Demo

https://github.com/SoftUni/playground

Project Trackers and Kanban Boards

- **Project trackers** organize and track project tasks
  - **Tasks** may have description, sub-tasks, assigned people, deadline
- **Kanban boards** visualize the work on a project
  - Typical columns: Backlog, In Progress, Done
  - Examples: Trello, GitHub Projects


trello.com/b/HPpU2eCV/company-web-site

Backlog	In Progress	Done
Review all pages	CSS styles	HTML page template
Content for the Services page	Content for the Home page	Empty pages
Content for the Portfolio page	Content for the About page	+ Add another card
Content for the Contacts page	Content for the Careers page	
Publish the site live	+ Add another card	
+ Add another card		

Trello Project Board

Live Demo

https://trello.com/b/HPpU2eCV/company-web-site

Summary

- Front-end development concepts: front-end, UI concepts, DOM, AJAX, routing, templating, UI frameworks
- Back-end development concepts, RESTful services, databases, ORM frameworks, MVC architecture, cloud, containers, Docker, …
- Embedded systems and IoT, Arduino, ESP32
- Software engineering, source control systems, QA, unit testing, Kanban, …