



Co je to AJAX?

Jaké jsou důvody jeho zásadního přínosu pro webové aplikace?

A fetch, Promises, WebSockets atd.

Co je to responzivní design?

Co je to DOM? Jak se s ním pracuje?

Co je to Javascript?

Jaké znáte knihovny zvyšující efektivitu vývoje klientských webových aplikací?

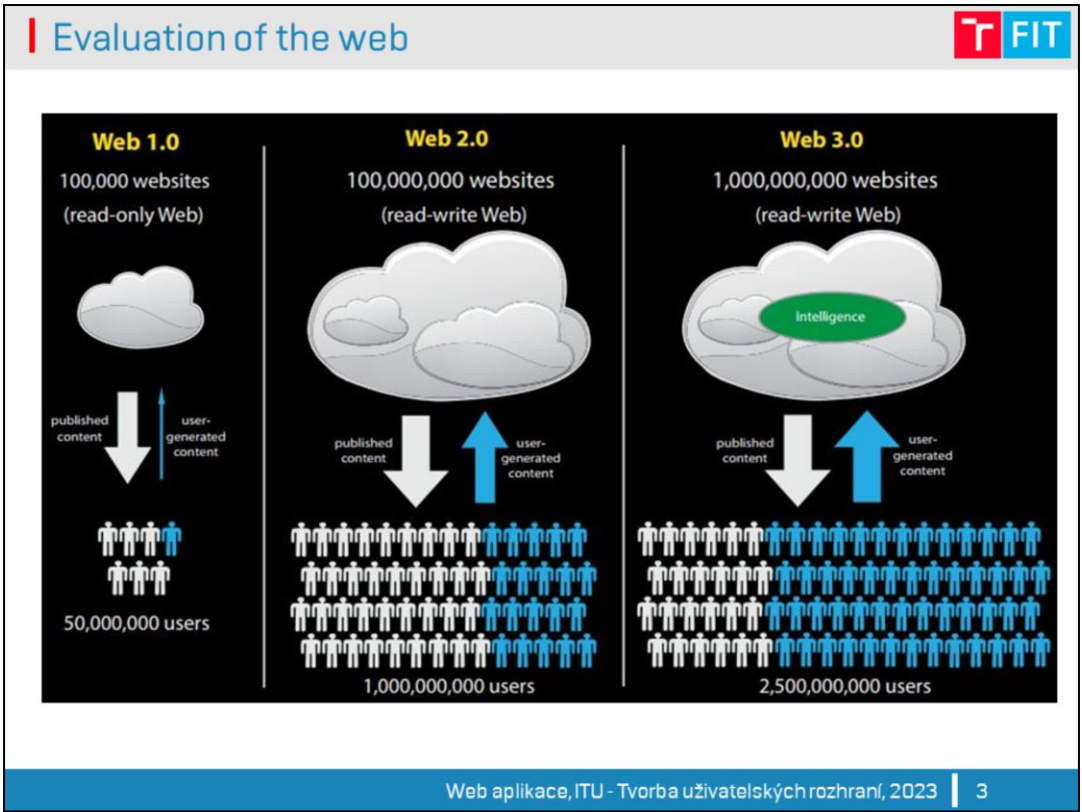
Co je to JSON a proč je pro vývoj webových aplikací užitečný?

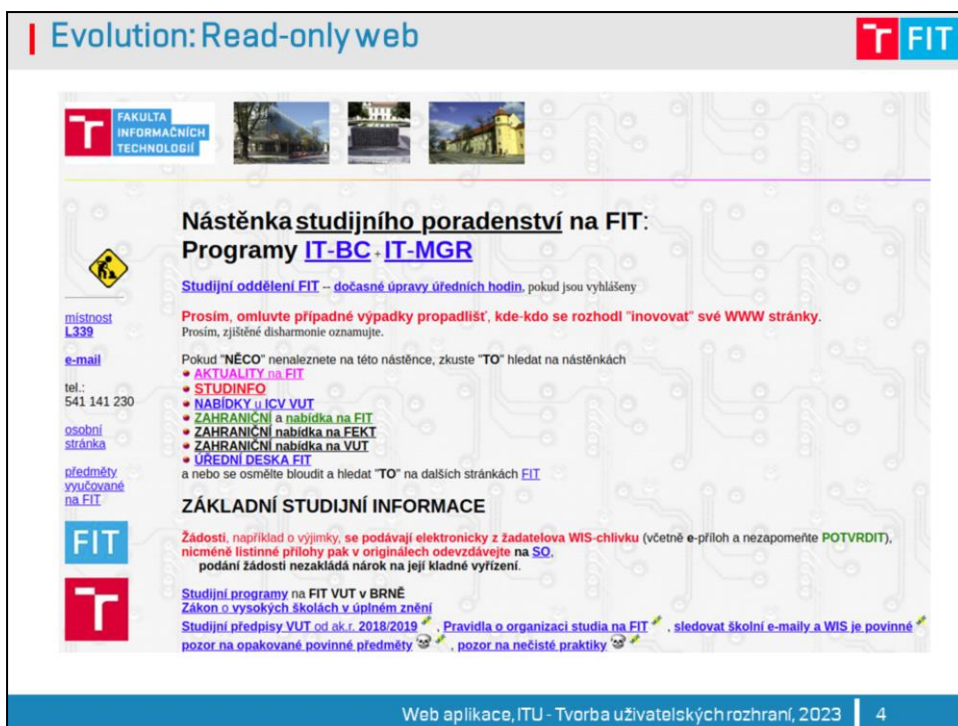
Příklad SW architektury webové aplikace využívající MVC.

Key outcomes



- Co je to AJAX?
 - Jaké jsou důvody jeho zásadního přínosu pro webové aplikace?
 - A fetch, Promises, WebSockets atd.
- Co je to responzivní design?
- Co je to DOM? Jak se s ním pracuje?
- Co je to Javascript?
 - Jaké znáte knihovny zvyšující efektivitu vývoje klientských webových aplikací?
- Co je to JSON a proč je pro vývoj webových aplikací užitečný?
- Příklad SW architektury webové aplikace využívající MVC.





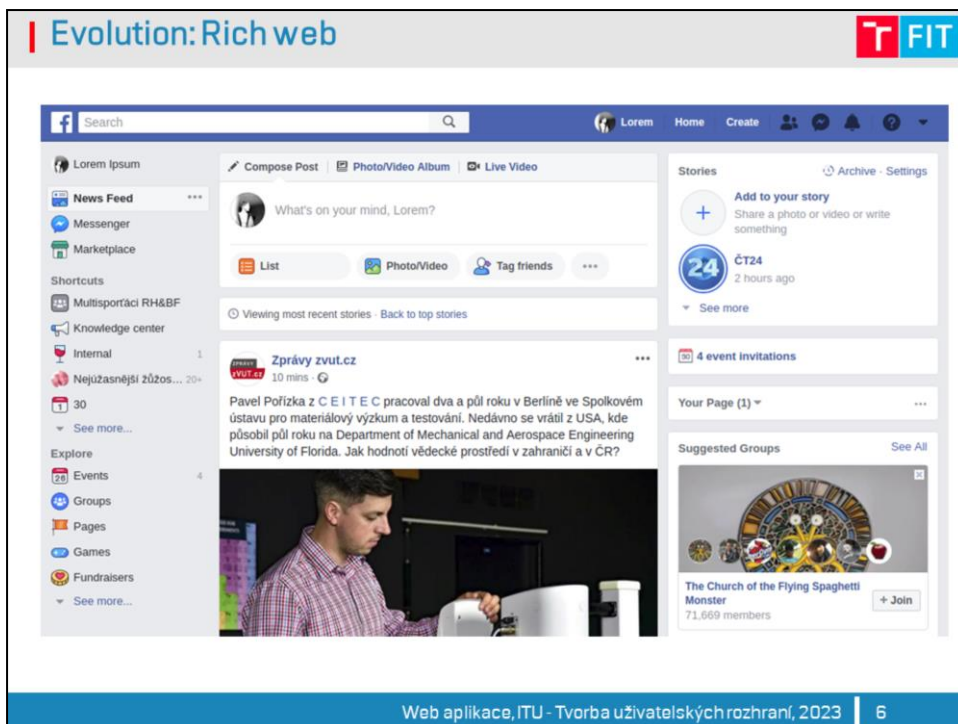
Na začátku webu bylo vše statické
Stránky jsou read-only a o uživatelském rozhraní jak jej chápeme dnes se
moc nedá mluvit
Typicky šlo o osobní stránky

Everything was static in the beginning
Pages are read-only and we cannot talk about UI as we know it today
Typically personal sites



Prvním dynamickým uživatelským rozhraním byly formuláře
 Přes formulář se vyhledává, komentuje, přihlašuje, nakupuje,... Každé
 odeslání znamená načtení nové stránky
 Návštěvníci webu už nejsou čistými příjemci, ale mají možnost reagovat
 nebo obsah sami vytvářet bez znalosti programování
 Roste potřeba mít příjemné rozhraní. Jeho kvalita často rozhoduje o
 aktivitě uživatelů, typicky na e-shopu

Forms were the first really dynamic user interface
 Forms allows users to search, comment, log in, shop,... But every submit
 means full reload of the page
 Visitors are no longer just readers. They can actively react on the content
 or create it themselves without programming knowledge
 The need for nice UI increases. Its quality can make difference in user
 activity, for example buying some e-shop product or not



Web jak jej nyní známe

Rozhraní se skládá z množství dynamických prvků

Technicky vzato jde stále o formuláře - něco se vyplní/nastaví a nakonec odešle - vizuálně už ale vypadají značně jinak

Hlavní zlom nastal s možností spouštět requesty bez opětovného načítání stránky

Možnost donačítat další data, např. Stránkování

Možnost odeslat komentář, nakoupit, ...

V delším horizontu toto vedlo k single page apps (SPA), tj. stránek které se načtou pouze jednou

Web as we know it today

UI consists of many dynamic elements

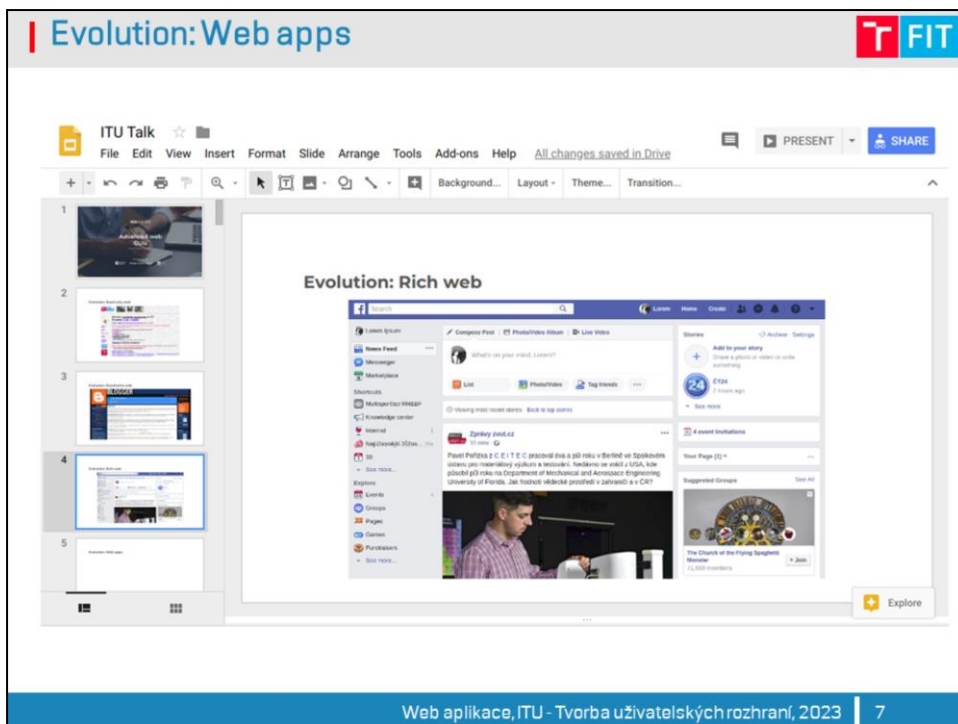
Technically, they're still forms, but they often don't look that way

The deal breaker is the ability to do server requests without loading the whole page

Loading delayed or additional data, e.g. paging

Submitting a comment, adding goods to cart,...

In longer horizon this lead to single page apps (SPA) = pages that only load once



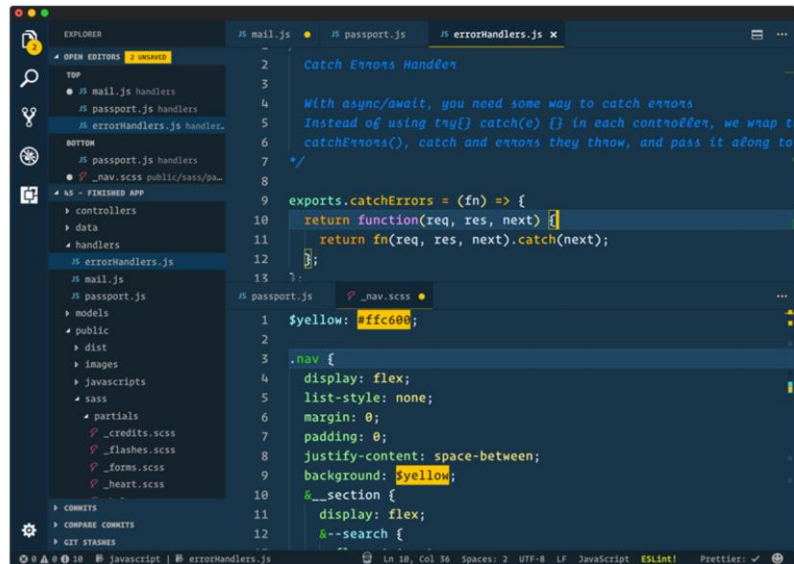
- Extrémním příkladem rozvoje webu jsou webové aplikace. Jejich rozhraní je často k nerozeznání od desktopových verzí. Formuláře už se vytrácejí a nahrazují je klasické prvky UI - lišty s ikonami, menu, kontextové menu. Možnost interakce přímo se systémem skrze prohlížeč. Například drag & drop souborů přímo do editovaného dokumentu nebo přístup stránky ke stavu internetového připojení.

Extreme case of web technologies usage are web apps

The UI is typically the same as it would be in native app

We can no longer see any forms in UI and everything looks native - toolbars, menus, context menus

Websites are able to interact with the host system. For example dropping files into the edited document, checking battery or connection status

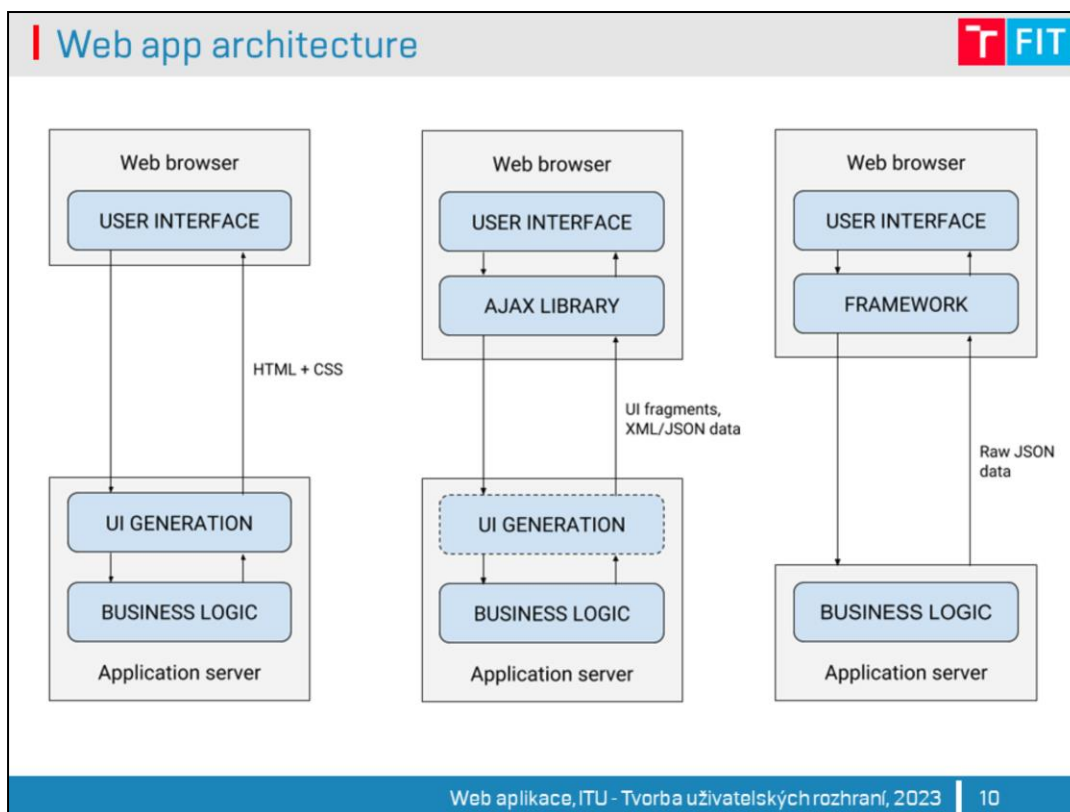




Webové technologie se prosazují také na mobilech a podobných zařízeních
Nativní aplikace může být i webová stránka běžící v nějakém wrapperu
Díky React Native lze dokonce použít pro mobilní aplikaci stejný kód jako pro její webovou verzi

Mobilní web přináší nové možnosti uživatelského vstupu (touch, multi-touch, foto, zvuk, poloha,...) které zpětně ovlivňují webové technologie
Přináší i nové výzvy - různé rozměry obrazovky, pomalé/nestabilní/neexistující připojení k internetu, úspora baterie,...

Web technologies are used on mobiles and mobile devices too
Native app can be just a website running inside a native wrapper
React Native lets developer use the same code for mobile app as for web
Mobile web brings new input methods (touch, multi-touch, photo, sound, location,...) which affects web technologies back
New challenges too - different screen sizes, slow/unstable/non-existing connection, battery drain,...

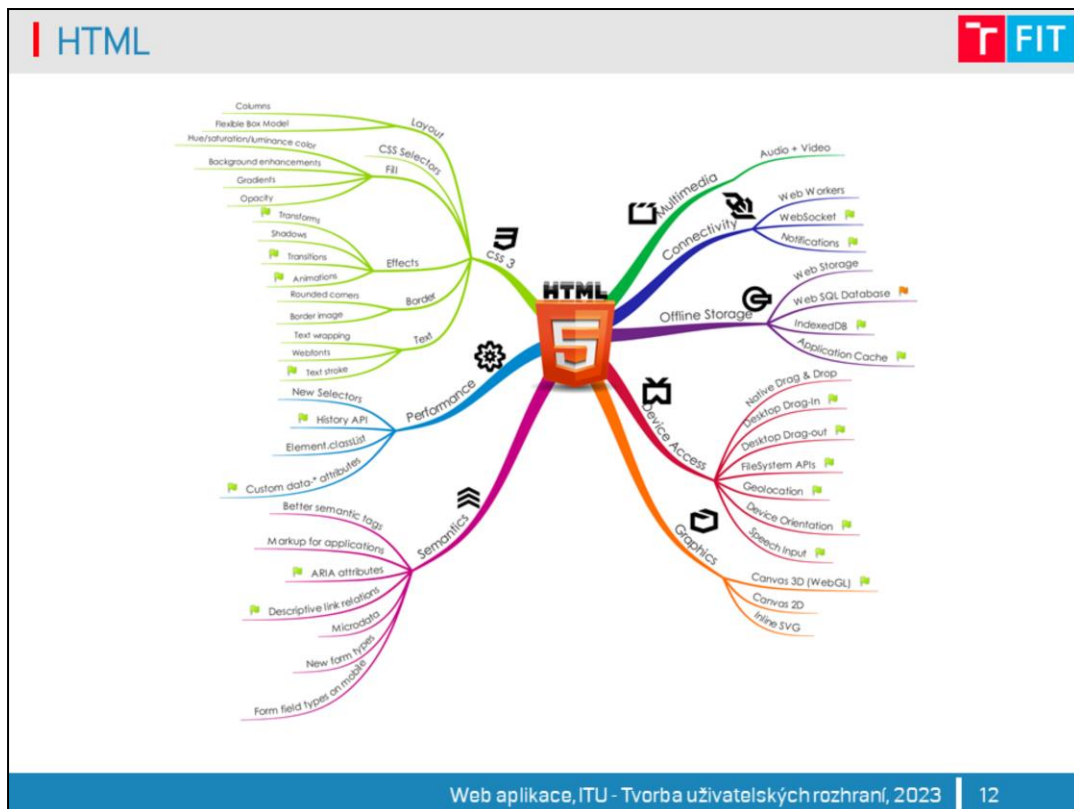


- Statický web: backend generuje celé HTML, žádná další komunikace neprobíhá
- Ajax web: backend stále generuje HTML, ale dodatečná data/UI snippets se dotahují přes asynchronní requesty bez reloadu
- Standalone frontend: backend poskytuje pouze API vracející data. Frontend je samostatná aplikace komunikující plně přes asynchronní requesty. Decoupling - více frontendů, backend neřeší UI

- Static web: backend generates full HTML, there's no other communication
- Ajax web: backend still generates HTML, but there are additional requests for data/UI fragments done asynchronously
- Standalone frontend: backend is only API returning raw data. Frontend is a standalone app which communicates fully through async requests. Decoupling - more frontends, backend has no knowledge of UI



A mockup of a web application interface. It features a light gray header bar at the top with a red vertical bar on the left and a logo on the right consisting of a red square with a white 'T' and a blue square with the text 'FIT' in white. The main content area is white and contains the text 'ZÁKLADNÍ STAVEBNÍ BLOKY' in a large, blue, sans-serif font. At the bottom, there is a blue footer bar with the text 'Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023' and a small white vertical bar followed by the number '11'.



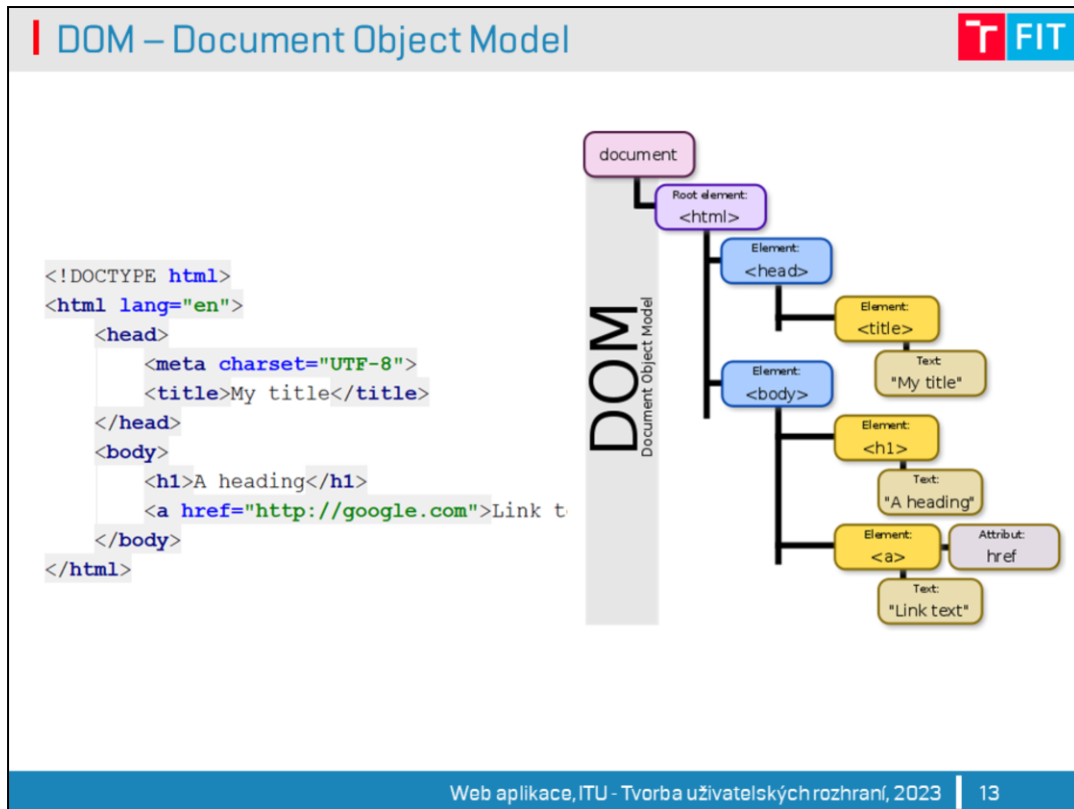
HTML – HyperText Markup Language

umožňuje strukturovat dokument: statický značkovací jazyk

dává částem dokumentu různý význam: množina značek a jejich atributů, element dokumentu, atributy elementu

struktura dokumentu:

- deklarace DTD – !DOCTYPE (Definice pravidel, verze HTML, není součástí HTML dokumentu)
- kořenový element – html
- hlavička elementu – head (titulek, jazyk, kódování)
- tělo dokumentu – body



Objektový Model Dokumentu

- Platformově a jazykově nezávislý objektový model
- Prezence (X)HTML, XML a podobných formátů
- Jeden uzel je instance DOMHTML Element
- Přístup k elementům dokumentu
- Změna obsahu nebo atributů elementů
- Změna struktury nebo stylu dokumentu

CSS – Cascading Style Sheets

Selector

h1

Declaration

{ color:blue; font-size:12px; }

Declaration

Property

Value

Property

Value

Welcome to My Homepage

Use the menu to select different Stylesheets

Stylesheet 1

Stylesheet 2

Stylesheet 3

Stylesheet 4

No Stylesheet

Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting one of the following links: [Stylesheet1](#), [Stylesheet2](#), [Stylesheet3](#), [Stylesheet4](#).

No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet: [No Stylesheet](#).

Side-Bar

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Welcome to My Homepage

Use the menu to select different Stylesheets

Stylesheet 1

Stylesheet 2

Stylesheet 3

Stylesheet 4

No Stylesheet

Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting one of the following links: [Stylesheet1](#), [Stylesheet2](#), [Stylesheet3](#), [Stylesheet4](#).

No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet: [No Stylesheet](#).

Side-Bar

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Welcome to My Homepage

Use the menu to select different Stylesheets

Stylesheet 1

Stylesheet 2

Stylesheet 3

Stylesheet 4

No Stylesheet

Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting one of the following links: [Stylesheet1](#), [Stylesheet2](#), [Stylesheet3](#), [Stylesheet4](#).

No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet: [No Stylesheet](#).

Side-Bar

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 14

oddělení obsahu dokumentu od definice jeho vzhledu

jazyk pro popis grafického vzhledu dokumentů napsaných ve značkovacích jazycích
definice vlastností pro elementy

More than one stylesheet declaration could apply to a particular piece of HTML, there has to be a known way of determining which specific stylesheet rule applies to which piece of HTML.

"Cascading" in this context means that the rule used is chosen by cascading down from the more general declarations to the specific rule required. The most specific declaration is chosen.

Definition to `<p>` tag in `<body>` overwrites the `<body>` definition, otherwise `<p>` would inherited the `<body>` definition.

posloupnost pravidel

- selektor: adresuje skupinu elementů
- seznam deklarací: určují vzhled vybraných elementů

CSS Preprocessors

- Rozšíření jazyka CSS o nové konstrukce
- Překlad do CSS, aby mu klienti rozuměli
- Zvýšení efektivity, udržovatelnosti, jeden CSS soubor, atd.

Hlavní rozšíření: Vnořené definice, Proměnné, konstanty, Matematické výrazy, *Mixins* – propojování definic

<https://www.sassmeister.com/>

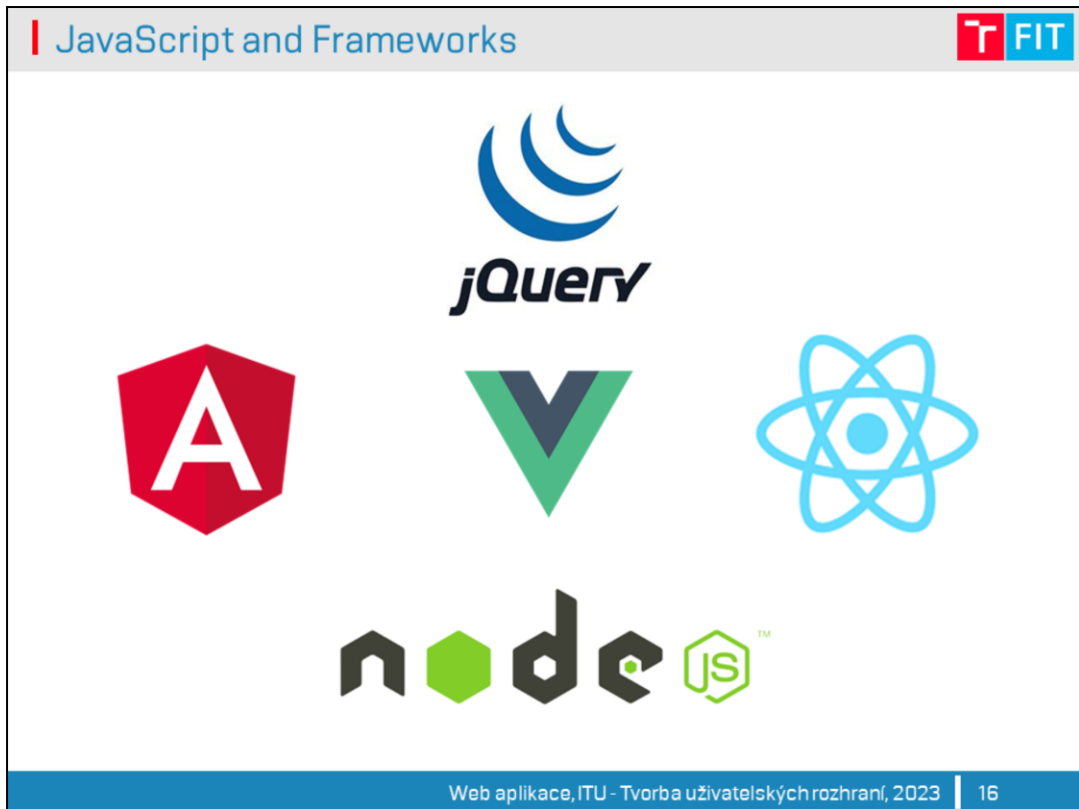
<http://lesscss.org/>

CSS Frameworks

Komponenty uživatelského rozhraní

Rychlý vývoj, hezký design, responzivnost

Např.: Bootstrap, Foundation, Gumby, Pure, ...



JavaScript

- **standard pro programování aplikací v HTML (interakce s DOM)**
- nemá nic společného s Javou, není nezávislý na klientovi

jQuery – usnadnění spolupráce JavaScriptu a HTML

- průchod, vyhledávání a manipulace DOM
- manipulace CSS, správa událostí, efekty a animace

AngularJS, Vue, ReactJS – frameworks for application with GUI development on client side.

Node.js is a JavaScript framework for servers.

Vue.js [/vju:/, like view]



```
<script type="importmap">
  {
    "imports": {
      "vue": "https://unpkg.com/vue@3/dist/vue.esm-browser.js"
    }
  }
</script>

<script type="module">
  import { createApp } from 'vue'

  createApp({
    data() {
      return {
        count: 0
      }
    }
  }).mount('#app')
</script>

<div id="app">
  <button @click="count++">
    Count is: {{ count }}
  </button>
</div>
```

Count is: 6

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 17

Vue.js

- progressive framework for building user interfaces
- designed from the ground up to be incrementally adoptable
- core library is focused on the view layer only
- easy to pick up and integrate with other libraries or existing projects

<https://vuejs.org/v2/guide/>



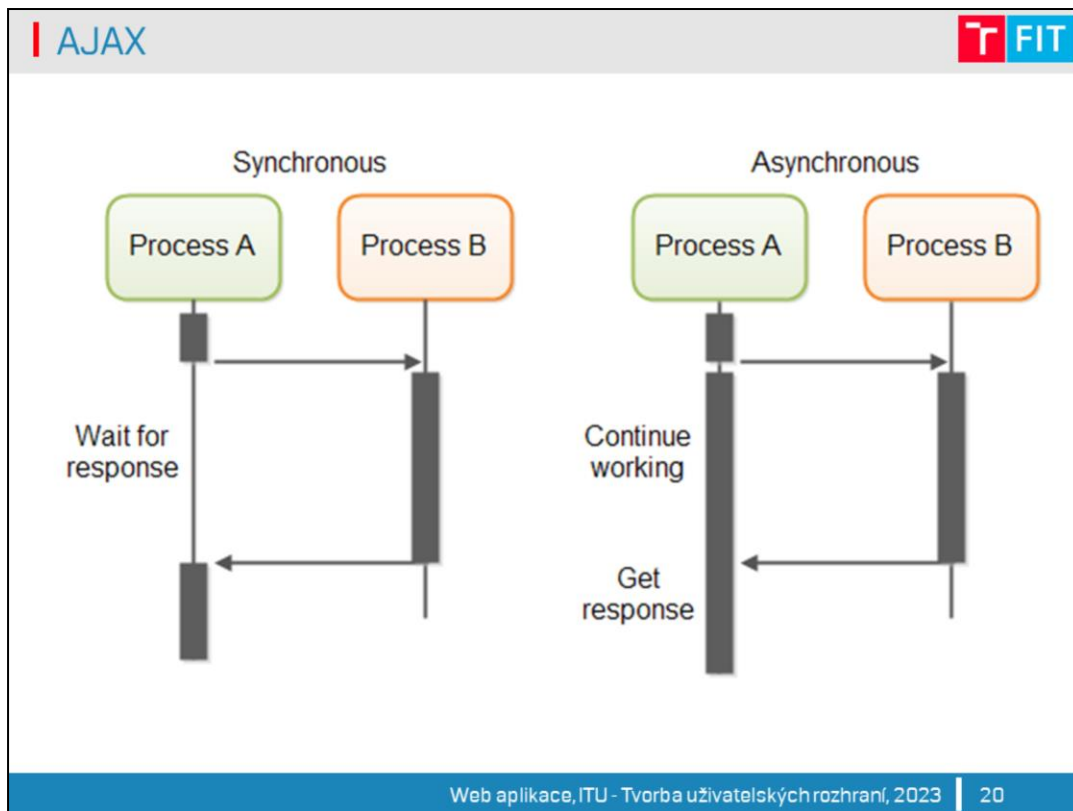
Uživatelské rozhraní plynule reagující na aktuální způsob zobrazení

- Typ zařízení, rozlišení, DPI, ...
- Není nová technologie, jen myšlenka
- Zakázání zmenšování
 - viewport – uživatelem viditelná oblast webové stránky
 - `<meta name="viewport" content="width=device-width, initial-scale=1.0">`
- Relativní jednotky: vw, vh, vmin, vmax, %, em, rem, ex, ch
- Layout – flexible grid
 - Fixes – nemění se, není vidět kraje
 - Fluid – obsah se zmenšuje
 - Adaptive – několik Fixed verzí
 - Responsive – Fluid + Adaptive
- Media queries – pravidla pro vkládání mediálního obsahu
 - `@media all and (min-width: 800px) and (max-width: 1024px) {...}`
 - `@media only screen and (orientation: portrait) {...}`



A presentation slide with a light gray header bar on the left containing a red vertical bar and on the right containing the FIT logo. The main content area is white and contains the text "AJAX" in large blue letters, followed by "ASYNCHRONOUS JAVASCRIPT AND XML" in smaller blue letters. The footer is a solid blue bar with white text.

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 19



Asynchronous Javascript And XML

- množina vzájemně souvisejících webových technologií
- asynchronní získávání dat ze serveru
- neobtěžovat aktualizováním obsahu stránek (reload)
- asynchronní přenos dat
 - XMLHttpRequest – zasílání HTTP(S) požadavků na web server a čtení odezvy serveru
- data ze serveru
 - XML – příliš obecné použití, syntakticky složitý, paměťově náročný
 - JSON – „odlehčený“ textový formát v notaci JavaScriptu
 - HTML, text
- zpracování dat na klientovi
 - JavaScript
 - VBScript – méně rozšířené
- DOM – přístup k datům v dokumentu

I Minimum z klientského JavaScriptu 1/2



- Spuštění kódu v JavaScriptu

```
<script>alert('ahoj!');</script>  
<div onclick="alert('ahoj!');"></div>  
<a href="javascript:alert('ahoj!');">odkaz</a>
```

- Periodické spuštění

- Vykonání jednou po zadaném čase:
 `setTimeout(ukazatel_funkce, 3000);`
- Periodické opakování se zadaným intervalem:
 `setInterval(ukazatel_funkce, 3000);`

Minimum z klientského JavaScriptu 2/2



- Přístup k prvkům dokumentu

```
<div id="mojeId">v rámci stránky unikátní</div>  
document.getElementById('mojeId').innerHTML = '<b>nový obsah  
elementu</b>';
```

- Získání hodnoty z formulářového prvku

```
<input type="text" id="mojeId2"/>  
alert(document.getElementById('mojeId2').value);
```

| XMLHttpRequest – vytvoření objektu



- Ošetření různých prohlížečů

```
function createXmlHttpRequestObject() {  
    var xmlhttp;  
    try {  
        xmlhttp = new XMLHttpRequest(); // all browsers except IE6  
    } catch (e) {  
        try {  
            xmlhttp = new ActiveXObject("Microsoft.XMLHttp");  
        } catch (e) {  
            // ignore error  
        }  
    }  
    if (!xmlhttp) {  
        alert ("Error creating the XMLHttpRequest object.");  
    } else {  
        return xmlhttp;  
    }  
}
```

Vytvoření
objektu

Nastavení
parametrů

Odeslání

Obsluha
událostí

Zobrazení
výsledku

| XMLHttpRequest – nastavení a odeslání



- Inicializace a nastavení hlavičky požadavku

```
var params = "firstNumber=" + first + "&secondNumber=" + second;  
  
xmlHttp.open("POST", "...API URL...", true);  
xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded;");
```

- Nastavení funkce pro obsluhu událostí

```
xmlHttp.onreadystatechange = processData;
```

- Odeslání požadavku na server

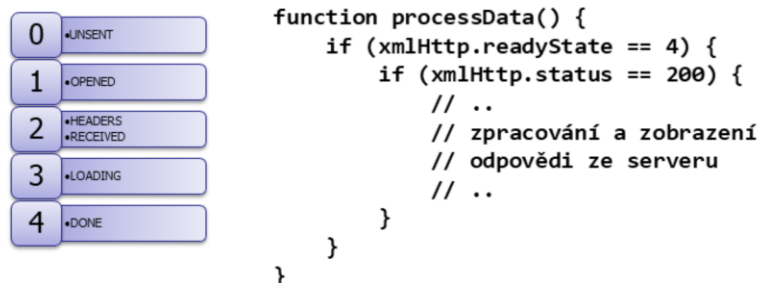
```
xmlHttp.send(params);
```



XMLHttpRequest – obsluha událostí



- Při každé změně stavu objektu se zavolá nastavená callback funkce (několikrát v průběhu požadavku!)
- Důležitá je hodnota `xmlHttp.readyState` a `xmlHttp.status`



XMLHttpRequest – zobrazení výsledku



- Formát dat odpovědi serveru – na cvičení: JSON

```
xmlHttpRequest.onreadystatechange = function() {  
    if ((xmlHttpRequest.readyState == 4) && (xmlHttpRequest.status == 200)) {  
        console.log(xmlHttpRequest.responseText);  
  
        var pole = JSON.parse(xmlHttpRequest.responseText);  
        for (var i in pole) {  
            console.log(pole[i]);  
        }  
    }  
}
```



JSON
T FIT

```

{
  "firstName": "John",
  "lastName": "Smith",
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumbers": [
    { "type": "home", "number": "212 555 1234" },
    { "type": "fax", "number": "646 555 4567" }
  ]
}

```

```

<Person firstName="John" lastName="Smith">
  <Address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </Address>
  <phoneNumber type="home">
    212 555-1234
  </phoneNumber>
  <phoneNumber type="fax">
    646 555-4567
  </phoneNumber>
</Person>

```

```

var p = eval("(" + contact + ")");
var p = JSON.parse(contact);

var my_JSON_object = !(/^[^,:{}\[\]0-9.\-+Eaeflnr-u \n\r\t]/.test(
text.replace(/"/g, '\\"')
)) &&
eval('(' + text + ')');
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023
27

JavaScript Object Notation

čitelný textový formát pro výměnu dat

reprezentace základních datových struktur, asociativních polí

datový formát nezávislý na programovacím jazyku

JSON je podmnožinou JavaScriptové objektové notace

konstrukce JavaScript objektu

vyhodnocením JSON řetězce

použití parseru

data v JSON formátu jsou zneužitelné k útoku na aplikaci

před vyhodnocením lepší použít parser, který je bezpečnější a rychlejší ve zpracování řetězce

Promises
T FIT

```

//Part 1 - Inside Async function
function makeSomeAsync(){
  //Create Promise object
  var promiseObj = new Promise(function(fullfill, reject){
    ... //Add ajax code here.
    ... // on success, call fullfill method, to resolve
    ... // on error, call reject method, to reject
  });
  //Returns Promise object
  return promiseObj;
}

function handleSuccess() { ... }
function handleError() { ... }

//Part 2- Outside Async function
var pobj = makeSomeAsync();
//Attaching success handler to promise
pobj.then(handleSuccess, handleError);
  
```

```

new Promise(executor)
state: "pending"
result: undefined
  
```

```

state: "fulfilled"
result: value
  
```

```

state: "rejected"
result: error
  
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023
28

Promise – tool for managing async. operations.

- object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.
 - *pending*: initial state, neither fulfilled nor rejected.
 - *fulfilled*: meaning that the operation was completed successfully.
 - *rejected*: meaning that the operation failed.
- resolve or reject function is called, cannot happen both of them
- does not transfer data itself

<https://javascript.info/promise-basics>

<https://medium.com/front-end-weekly/ajax-async-callback-promise-e98f8074ebd7>

| Promises



```
function makeAjaxCall(url, methodType){
    var promiseObj = new Promise(function(resolve, reject){
        var xhr = new XMLHttpRequest();
        xhr.open(methodType, url, true);
        xhr.send();
        xhr.onreadystatechange = function(){
            if (xhr.readyState === 4){
                if (xhr.status === 200){
                    console.log("xhr done successfully");
                    var resp = xhr.responseText;
                    var respJson = JSON.parse(resp);
                    resolve(respJson);
                } else {
                    reject(xhr.status);
                    console.log("xhr failed");
                }
            } else {
                console.log("xhr processing going on");
            }
        }
        console.log("request sent succesfully");
    });
    return promiseObj;
}
```

| Promises



```
document.getElementById("userDetails").addEventListener("click",
function(){
    // git hub url to get btford details
    var userId = document.getElementById("userId").value;
    var URL = "https://api.github.com/users/"+userId;
    makeAjaxCall(URL, "GET").then(processUserDetailsResponse,
    errorHandler);
});

document.getElementById("repoList").addEventListener("click",
function(){
    // git hub url to get btford details
    var userId = document.getElementById("userId").value;
    var URL = "https://api.github.com/users/"+userId+"/repos";
    makeAjaxCall(URL, "GET").then(processRepoListResponse,
    errorHandler);
});

function processUserDetailsResponse(userData){
    console.log("render user details", userData);
}

function processRepoListResponse(repoList){
    console.log("render repo list", repoList);
}

function errorHandler(statusCode){
    console.log("failed with status", status);
}
```

fetch

T FIT

Fetch is a modern Promise-based Ajax request API

- Not built on XMLHttpRequest

```
let promise = fetch(url, [options])
```

two-stage process

1. fetch returns Promise, that resolves with an object Response
 - HTTP status, headers ..., but no body yet
2. several Response methods to access the body
 - text(), json(), formData() ...

```
async function fetchMoviesJSON() {  
  const response = await fetch('/movies');  
  const movies = await response.json();  
  return movies;  
}  
  
fetchMoviesJSON().then(movies => {  
  movies; // fetched movies  
});
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 31

Fetch

- API provides an interface for fetching resources (including across the network)
- „similar“ to XMLHttpRequest, but the new API provides a more powerful and flexible feature set.

async – asynchronous function

await – operator is used to wait for a Promise

Fetch is a modern Promise-based Ajax request API that first appeared in 2015 and is supported in most browsers. It is not built on XMLHttpRequest and offers better consistency with a more concise syntax.

<https://javascript.info/fetch>

AJAX vs. fetch

T FIT



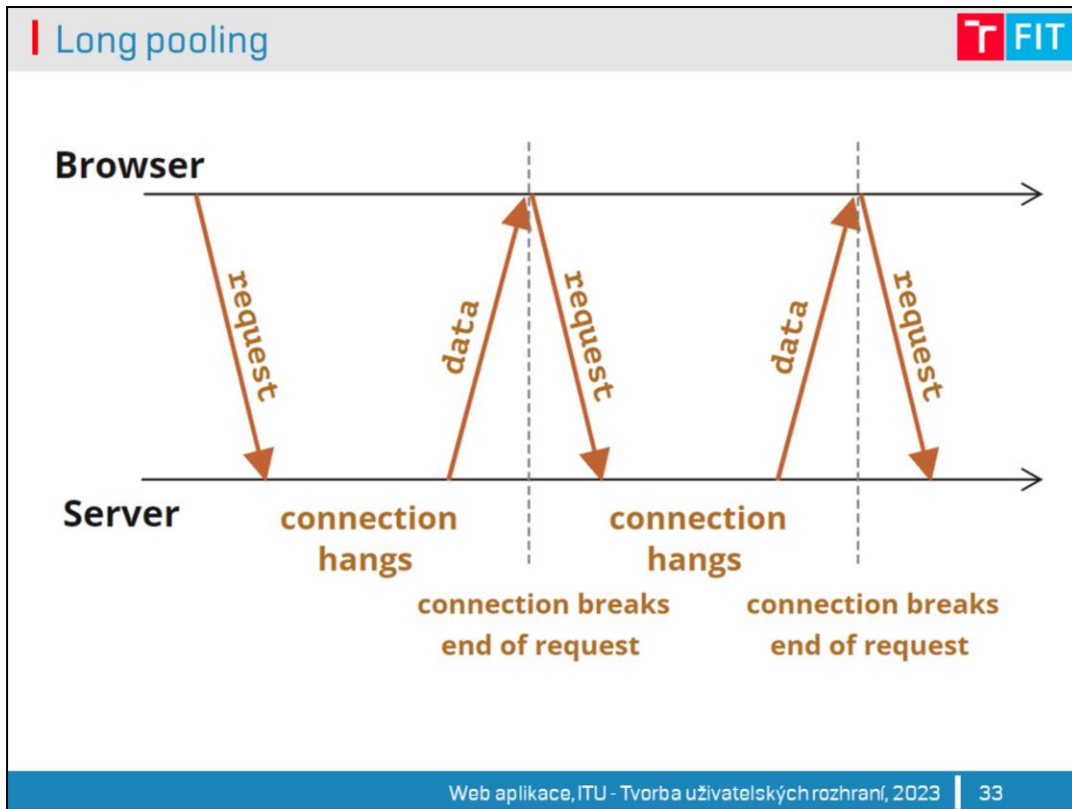
FETCH
VS
AJAX

XMLHttpRequest vs. fetch

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 32

Ajax is not a technology; instead, it refers to techniques which can load server data from a client-side script. Several options have been introduced over the years.

The right question is: **XMLHttpRequest vs. fetch**



Regular pooling

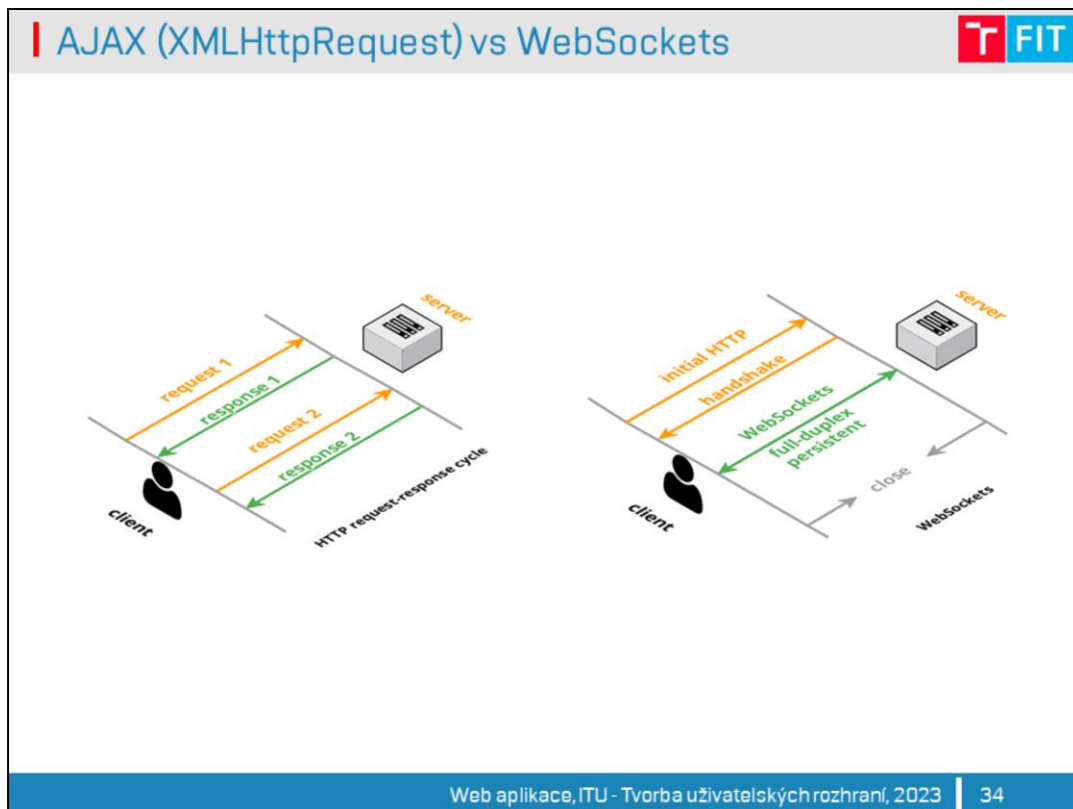
Periodic pooling, periodic requests – each 10s. High overheads – „large” headers, establishing/closing connection etc.

Long pooling

Simplest way of permanent connection with the server.

1. A request is sent to the server.
2. The server doesn't close the connection until it has a message to send.
3. When a message appears – the server responds to the request with it.
4. The browser makes a new request immediately.

For permanent communication better e.g. websockets.



AJAX

We now use "Ajax" as a generic term for any client-side process which fetches data from a server and updates the DOM dynamically without a full-page refresh.

Ajax is a core technique for most web applications and Single-Page Apps (SPAs).

XMLHttpRequest (XHR) is an API in the form of an object whose methods transfer data between a web browser and a web server.

WebSockets

WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection.

The WebSocket protocol enables interaction between a web browser (or other client application) and a web server with lower overhead than half-duplex alternatives.

<https://www.ably.io/topic/websockets>

- Part of the new APIs in HTML 5
- Real-time two-way communication
- New protocol - requires both client and server support
- E.g. removes the need for periodic polling
- Variant with SSL (wss protocol)

Cons of Websocket

- A fully HTML5-compliant web browser is required.
- AJAX-like success mechanisms are not available in Websockets.
- Websockets, unlike HTTP, do not provide intermediary/edge caching.
- HTTP is significantly easier to develop if your application doesn't take a lot of dynamic interaction.

Websockets

T FIT

good for realtime or near-realtime need for data consumption

CHAT

Hi! How can I help you today?

Sophie is typing...

Hi! I'd like to upgrade my plan to Premium, can you help?

NOTIFICATIONS

Exciting news Amaral!

You've earned a free coffee on us!

Redeem now Save for later

DATA BROADCAST

Yellow card

B. Clever

Italy 1 - 0 Argentina

Updated 2s ago

GOALS!

B. De Groot

Belgium 2 - 0 Cuba

14s ago

MULTIPLAYER COLLABORATION

R. P. L. F.

Lauren

Pajul

GPS LOCATION TRACKING

On the way!

Arriving 13:58

DATA SYNC

Bitcoin

+2.03%

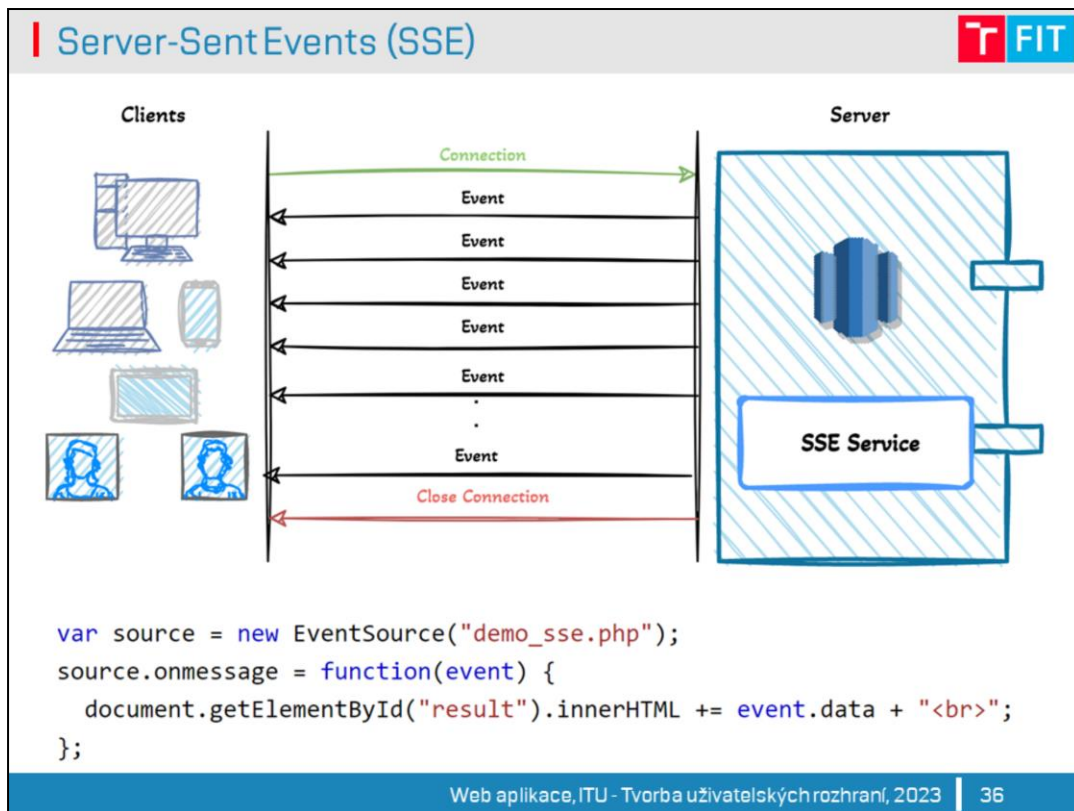
C 5s

Current value

\$24,207.92

not good always

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 35



SSE is a technology that provides asynchronous communication with event stream from server to the client over HTTP for web applications. The server can send un-directional messages/events to the client and can update the client asynchronously.

I

CRUD

T

FIT

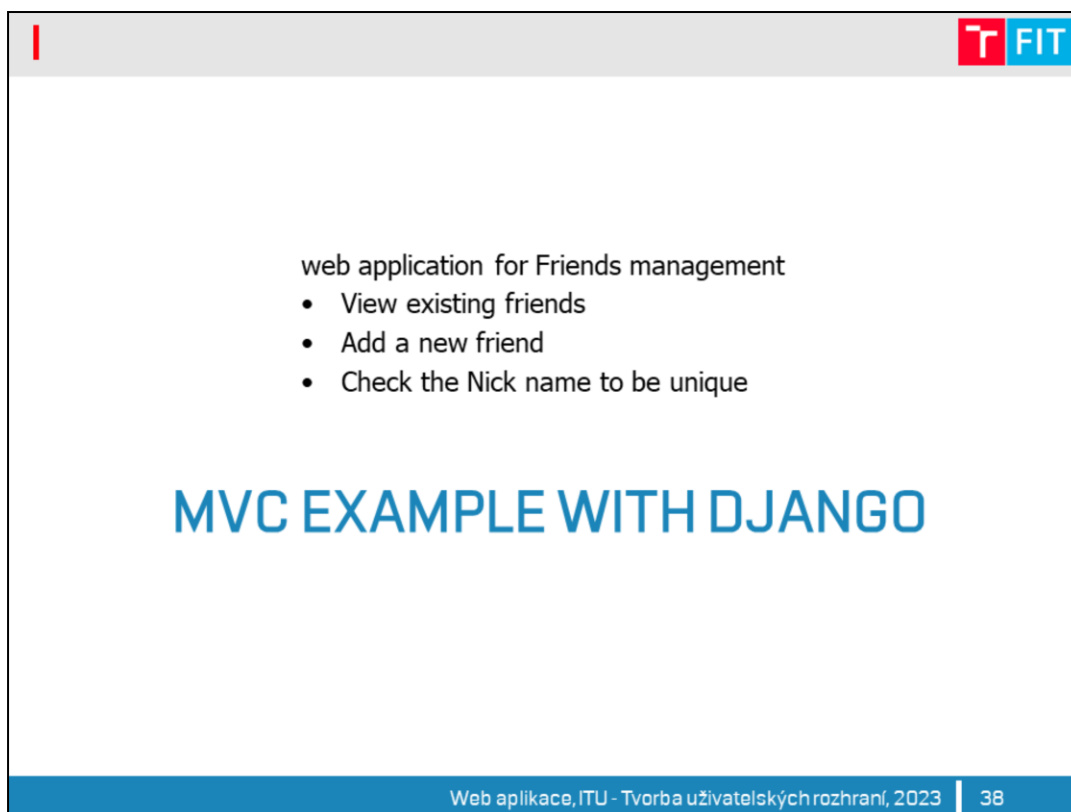


CREATE READ UPDATE DELETE

C R U D

- the major functions that are inherent to relational databases and the applications used to manage them
- too low-level for interactive web apps (e.g. Discord, web PowerPoint, YT ...)

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 37



The slide features a light gray header bar with a red vertical line on the left and the 'T FIT' logo on the right. The main content area is white and contains a list of features for a web application. Below the list, the title 'MVC EXAMPLE WITH DJANGO' is displayed in large blue letters. The footer is a solid blue bar with white text.

web application for Friends management

- View existing friends
- Add a new friend
- Check the Nick name to be unique

MVC EXAMPLE WITH DJANGO

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 38

Task: friend list on web app

T

FIT

Nick name

First name

Last name

Likes

What is your birth date?

January

▼

1

▼

1980

▼

Lives in

Create Friend

Nick name	First name	Last name	Likes	DOB	lives in
Nick1	Jedna	První	cat	1980-02-01	Brno
Nick2	Dva	Druhý	dog	1982-03-01	Praha

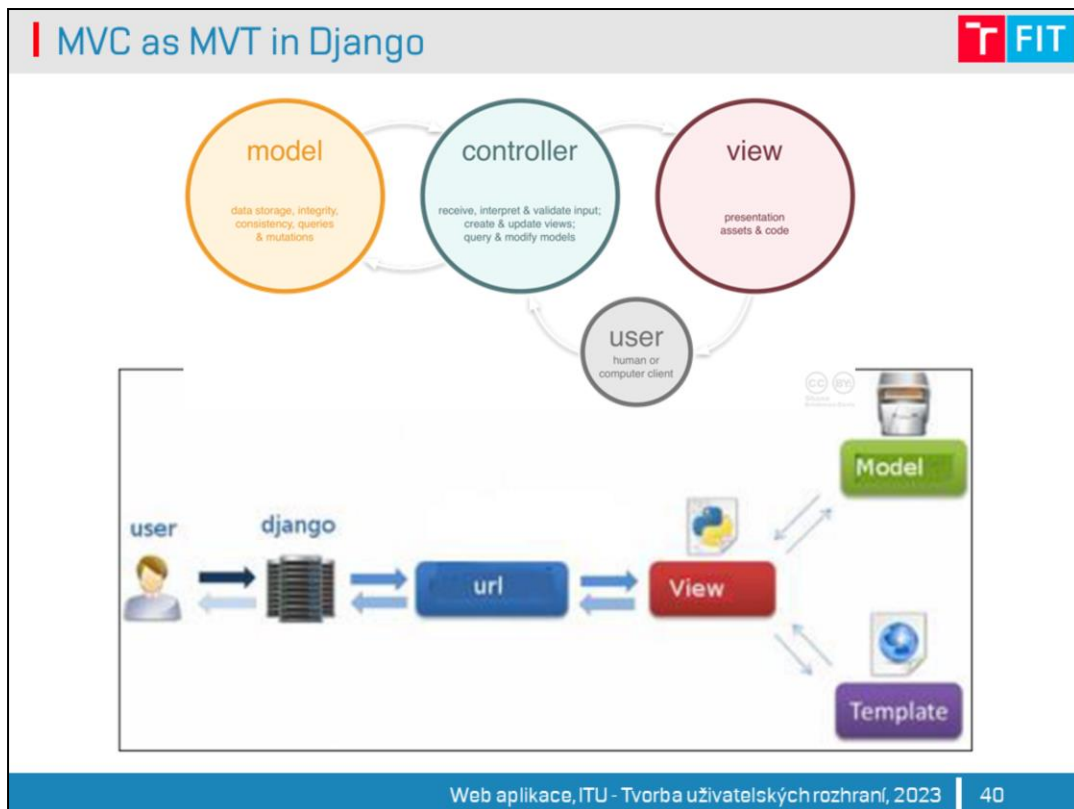
Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 39

Task: web application for Friends management

- View existing friends
- Add a new friend
- Check the Nick name to be unique
- Work modern – no reload, AJAX needed

Příklad převzat z: <https://www.pluralsight.com/guides/work-with-ajax-django>

Další inspirace a základ MVC/MVT např. zde: <https://docs.djangoproject.com/en/3.1/>



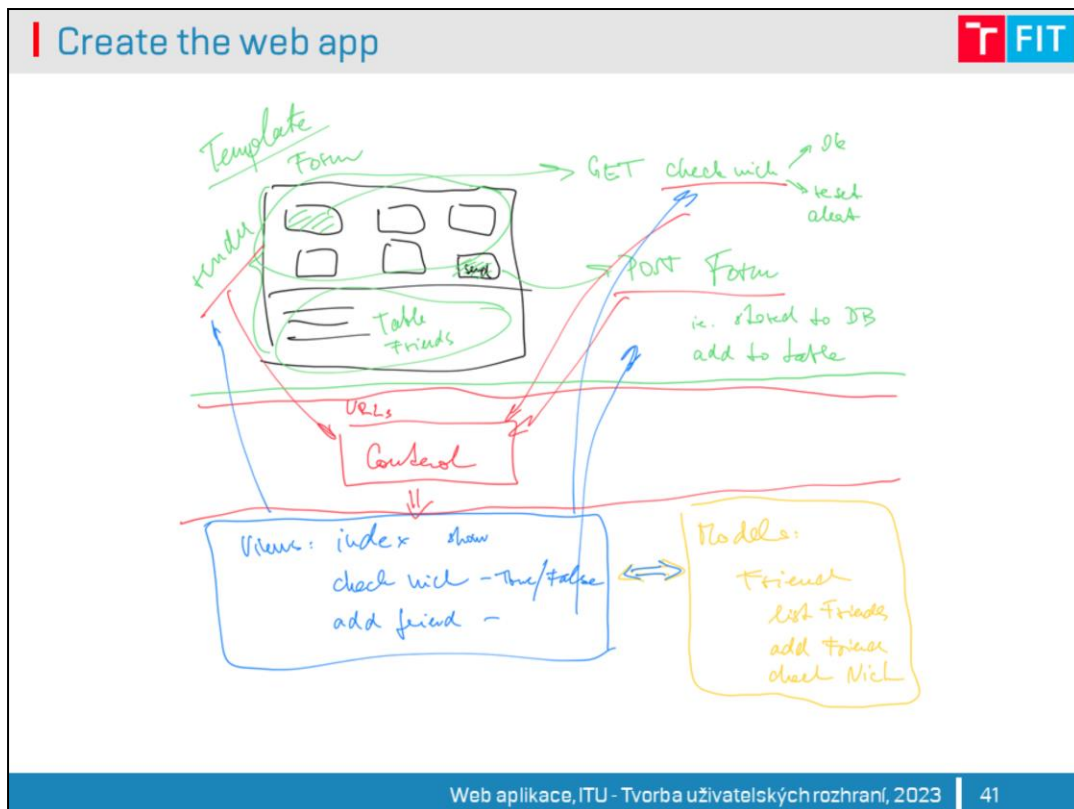
Django

- Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design

Django implementuje MVC jako MVT (Model-View-Template)

- Controller
 - je víceméně vestavěný jako router
 - mapuje uživatelské požadavky na funkce View
- View
 - kombinuje funkce a zobrazení (k zobrazení využívá Templates)
 - provede akci a zobrazí výsledek
- Model
 - udržuje data, provádí jejich správu, filtrace
 - aplikační logika je částečně přenesena na View

Pořád jsou ale **data, logika a zobrazení odděleny!**



- Create data models and associated form
 - Model, Forms
- Create view and user interaction
 - Html template, javascript
- Create controller
 - Maps user requests to view functions
- Create functions generating views and process user requests
 - Views

View – template – index.html

T FIT

Nick name

First name

Last name

Likes

What is your birth date?

January

▼

1

▼

1980

▼

Lives in

Create Friend

```

<div class="container-fluid">
  <form id="friend-form">
    <div class="row">
      {% csrf_token %}
      {% for field in form %}
        <div class="form-group col-4">
          <label class="col-12">{{ field.label }}</label>
          {{ field }}
        </div>
      {% endfor %}
      <input type="submit" class="btn btn-primary" value="Create Friend" />
    </div>
  </form>
</div>
<hr />

```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023

View and user interaction – html template

- Defines the template to render/show the application GUI
- Renders the given data to defined structures in the template

The form view

- Generated from fields of the Form
- Form is generated from particular attributes of the Model (see slide Model and Form)

Cross Site Request Forgery - prevention

View – template – index.html



```
<div class="container-fluid">
  <table class="table table-striped table-sm" id="my_friends">
    <thead>
      <tr>
        <th>Nick name</th>
        <th>First name</th>
        <th>Last name</th>
        <th>Likes</th>
        <th>DOB</th>
        <th>lives in</th>
      </tr>
    </thead>
    <tbody>
      {% for friend in friends %}
      <tr>
        <td>{{friend.nick_name}}</td>
        <td>{{friend.first_name}}</td>
        <td>{{friend.last_name}}</td>
        <td>{{friend.likes}}</td>
        <td>{{friend.dob | date:"Y-m-d"}}</td>
        <td>{{friend.lives_in}}</td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023

View and user interaction – html template

- Defines the template to render/show the application GUI
- Renders the given data to defined structures in the template

Table rendering the list of friends

- The object „friends“ contains the array of all objects of Friends (passed by View from the Model API, see slides Views)

View – javascript – check Nickname



```
$("#id_nick_name").focusout(function (e) {  
    e.preventDefault();  
    // get the nickname  
    var nick_name = $(this).val();  
    // GET AJAX request  
    $.ajax({  
        type: 'GET',  
        url: "{% url 'validate_nickname' %}",  
        data: {"nick_name": nick_name},  
        success: function (response) {  
            // if not valid user, alert the user  
            if(!response["valid"]){  
                alert("You cannot create a friend with same nick name");  
                var nickName = $("#id_nick_name");  
                nickName.val("")  
                nickName.focus()  
            }  
        },  
        error: function (response) {  
            console.log(response)  
        }  
    })  
})
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 44

View and user interaction – html template

- Defines the template to render/show the application GUI
- Renders the given data to defined structures in the template

Process the input value in the Form to check the uniqueness of the Nickname

- check the new Nickname using GET request
- calls the Controller to respond for the required response („validate_nnickname“)
- process the communication in async. way
- when nickname is valid, then continue
- otherwise execute alert and ask for another Nickname

View – javascript – add Friend


```

$("#friend-form").submit(function (e) {
    e.preventDefault(); // preventing from page reload and default actions
    var serializedData = $(this).serialize(); // serialize the data for sending the form data.
    $.ajax({
        // make POST ajax call
        type: 'POST',
        url: "{% url 'post_friend' %}",
        data: serializedData,
        success: function (response) {
            // on successfull creating object
            // 1. clear the form.
            $("#friend-form").trigger('reset');
            // 2. focus to nickname input
            $("#id_nick_name").focus();

            // display the newly friend to table.
            var instance = JSON.parse(response["instance"]);
            var fields = instance[0]["fields"];
            $("#my_friends tbody").prepend(
                `
                <tr>
                <td>${fields["nick_name"]}||""</td><td>${fields["first_name"]}||""</td>
                <td>${fields["last_name"]}||""</td><td>${fields["likes"]}||""</td>
                <td>${fields["dob"]}||""</td><td>${fields["lives_in"]}||""</td>
                </tr>`
            );
        },
        error: function (response) {
            // alert the error if any error occurred
            alert(response["responseJSON"]["error"]);
        }
    })
})

```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023
45

View and user interaction – html template

- Defines the template to render/show the application GUI
- Renders the given data to defined structures in the template

Send the form values to add new Friend

- serialize the form data and send using POST
- calls the Controller to respond for the required response („post_friend“)
- process the communication in async. way
- when OK, add the new Friend to the list/table of Friends
- otherwise report error

Controller



```
urlpatterns = [  
    # ... other urls  
    path('', indexView),  
    path('post/ajax/friend', postFriend, name = "post_friend"),  
    path('get/ajax/validate/nickname', checkNickName, name = "validate_nickname")  
    # ...  
]
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 46

Controller

- Maps user requests to view functions

View – main app view



```
def indexView(request):  
    form = FriendForm()  
    friends = Friend.objects.all()  
    return render(request, "index.html", {"form": form, "friends": friends})
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 47

- Create functions generating views and process user requests
 - Views

| View – use Model to check Nickname



```
def checkNickName(request):  
    # request should be ajax and method should be GET.  
    if request.is_ajax and request.method == "GET":  
        # get the nick name from the client side.  
        nick_name = request.GET.get("nick_name", None)  
        # check for the nick name in the database.  
        if Friend.objects.filter(nick_name = nick_name).exists():  
            # if nick_name found return not valid new friend  
            return JsonResponse({"valid":False}, status = 200)  
        else:  
            # if nick_name not found, then user can create a new friend.  
            return JsonResponse({"valid":True}, status = 200)  
  
    return JsonResponse({}, status = 400)
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 48

Functions generating views and process user requests

Check the Nickname

- check the uniqueness of the given Nickname using Model API
- confirm by sending the result back in response
- will be caught by function sending the request (AJAX)
- when not true, the Nickname in the Form is reset and alert is executed (see slide Check Nickname)
- otherwise continue filling the Form

| View – add Friend and update list of Friends



```
def postFriend(request):
    # request should be ajax and method should be POST.
    if request.is_ajax and request.method == "POST":
        # get the form data
        form = FriendForm(request.POST)
        # save the data and after fetch the object in instance
        if form.is_valid():
            instance = form.save()
            # serialize in new friend object in json
            ser_instance = serializers.serialize('json', [ instance, ])
            # send to client side.
            return JsonResponse({"instance": ser_instance}, status=200)
        else:
            # some form errors occurred.
            return JsonResponse({"error": form.errors}, status=400)

    # some error occurred
    return JsonResponse({"error": ""}, status=400)
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 49

Functions generating views and process user requests

Add new Friend

- store the data using Model API
- confirm by sending the stored data (Form) back in response
- will be caught by function sending the request (AJAX)
- the function processes the response - show data in the table/list of Friends (see slide Add Friend)

| Model and Form



```
class Friend(models.Model):
    # NICK NAME should be unique
    nick_name = models.CharField(max_length=100, unique = True)
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    likes = models.CharField(max_length = 250)
    dob = models.DateField(auto_now=False, auto_now_add=False)
    lives_in = models.CharField(max_length=150, null = True, blank = True)

    def __str__(self):
        return self.nick_name

class FriendForm(forms.ModelForm):
    ## change the widget of the date field.
    dob = forms.DateField(
        label='What is your birth date?',
        # change the range of the years from 1980 to currentYear - 5
        widget=forms.SelectDateWidget(years=range(1980, datetime.date.today().year-5))
    )

    def __init__(self, *args, **kwargs):
        super(FriendForm, self).__init__(*args, **kwargs)
        ## add a "form-control" class to each form input
        ## for enabling bootstrap
        for name in self.fields.keys():
            self.fields[name].widget.attrs.update({
                'class': 'form-control',
            })

    class Meta:
        model = Friend
        fields = ("__all__")
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 50

Model

- Defines attributes/fields

Form

- Inherits built-in form
- Modifies the widgets (date) and form input class

View – template – body.html



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <title>{% block title %}{% endblock title %}</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
  {% block style %}
  {% endblock style %}
</head>

<body>
  {% block content %}
  {% endblock content %}

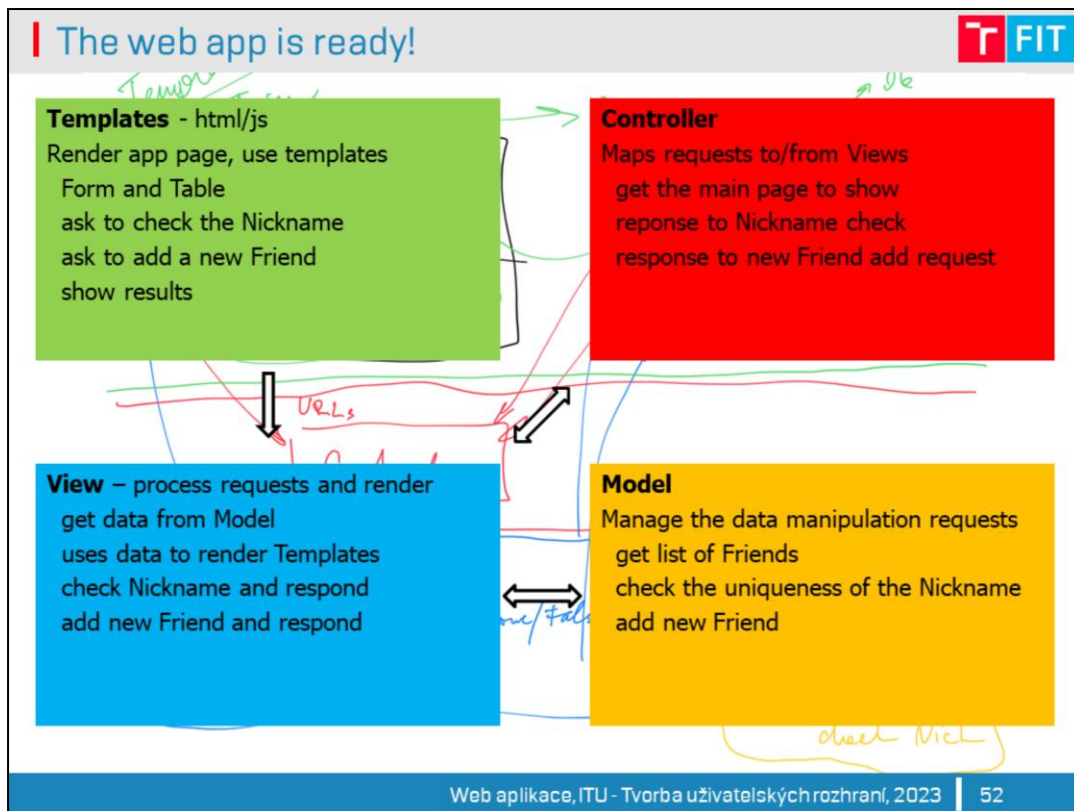
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
  {% block javascript %}
  {% endblock javascript %}
</body>
</html>
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 51

View and user interaction – html template

- Defines the template to render/show the application GUI
- Renders the given data to defined structures in the template

The main template, defines the web app structure



- Create data models and associated form
 - Model, Forms
- Create view and user interaction
 - Html template, javascript
- Create controller
 - Maps user requests to view functions
- Create functions generating views and process user requests
 - Views

Task: friend list on web app – done!

Nick name

Nick4

First name

Čtyři

Last name

Čtvrtý

Likes

VUT

What is your birth date?

September

8

1983

Lives in

Here

Create Friend

Nick name	First name	Last name	Likes	DOB	lives in
Nick3	Tři	Třetí	FIT	1985-08-09	Eurtop
Nick1	Jedna	První	cat	1980-02-01	Brno
Nick2	Dva	Druhý	dog	1982-03-01	Praha

http://127.0.0.1:8000/app_ajax/

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 53

Task: web application for Friends management

Solved:

- main app page shows the data from Model and prepare the form
- when adding new Friend, check the new Nick name and if it does not exist, continue filling the form
- „Create Friend” button adds the new Friend to the Model (database) and shows it in the list of Friends
- everything is happening on „background” - no reload of the page happens – we are modern 😊

C:\sendbox





- TicTacToe in React
 - <https://reactjs.org/tutorial/tutorial.html#setup-option-1-write-code-in-the-browser>
- React CRUD Web API
 - <https://bezkode.com/react-crud-web-api/>

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 54