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?

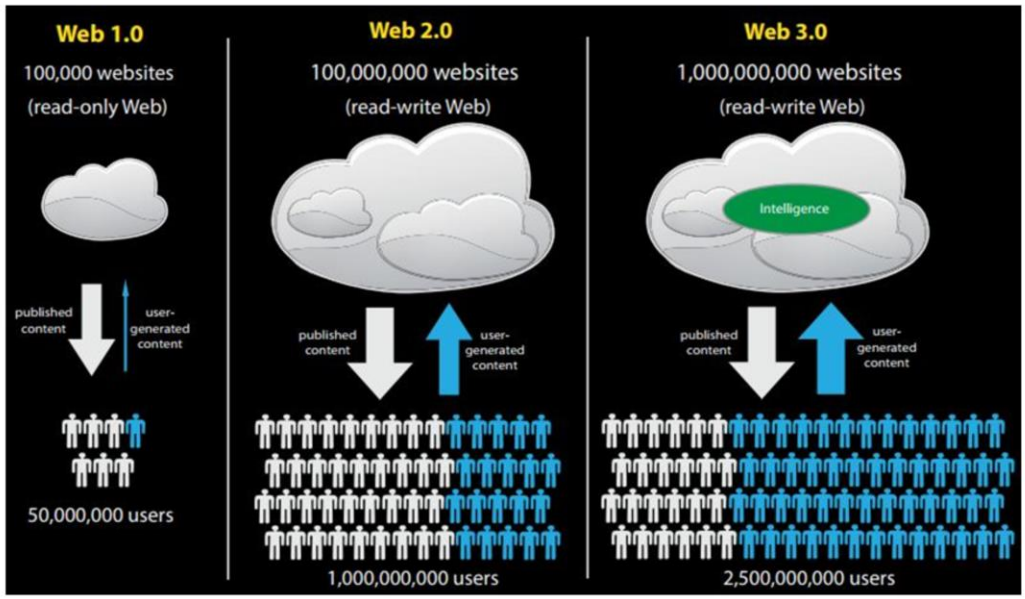
Příklad SW architektury webové aplikace využívající MVC.

Key outcomes

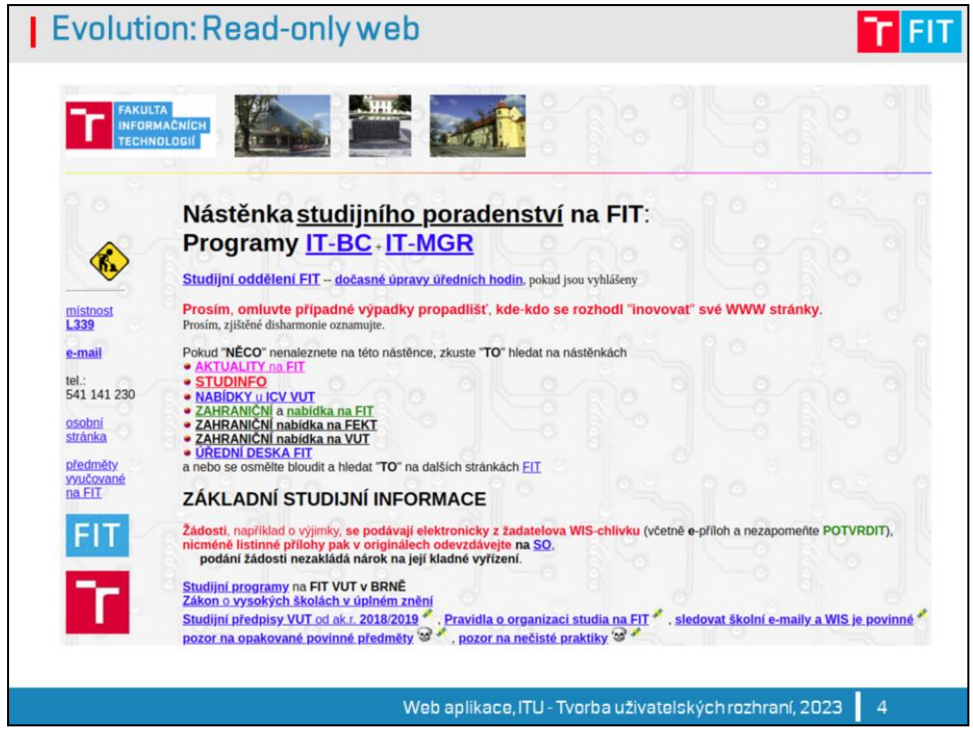
- Co je to AJAX?
  - Jaké jsou důvody jeho zásadního přínosu pro webové aplikace?
  - A fetch, Promises, WebSockets atd.
- Co je to responzivní design?
- Co je to DOM? Jak se s ním pracuje?
- Co je to Javascript?
  - Jaké znáte knihovny zvyšující efektivitu vývoje klientských webových aplikací?
- Co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

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 2

Evaluation of the web



Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 3

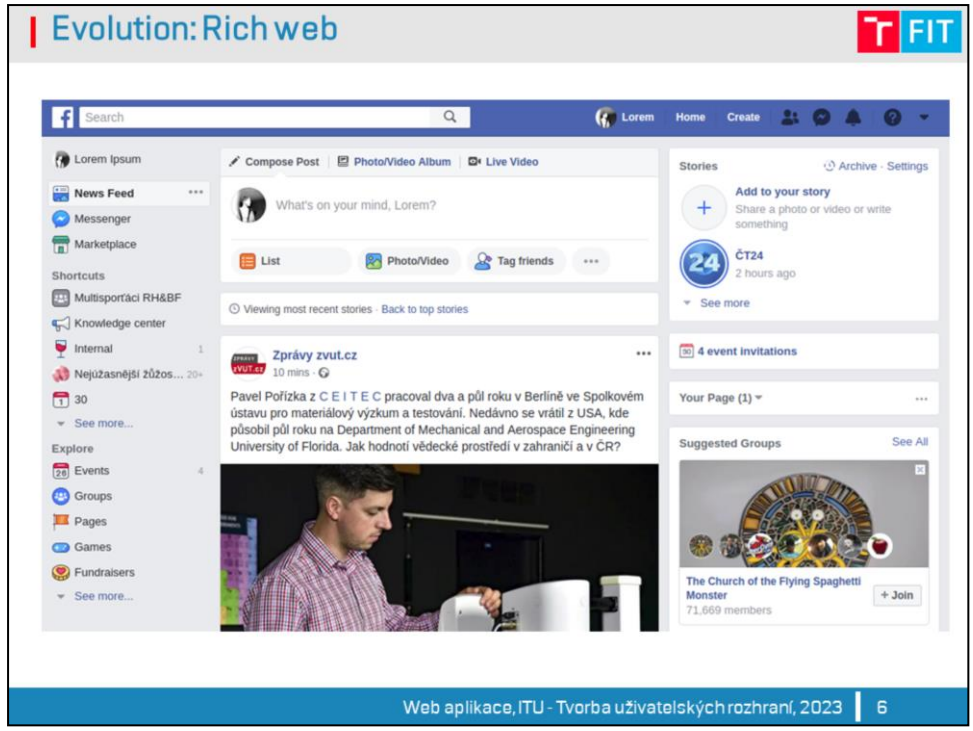


Na začátku webu bylo vše statické
Stránky jsou read-only a o uživatelském rozhraní jak jej chápeme dnes se moc nedá mluvit
Typicky šlo o osobní stránky

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

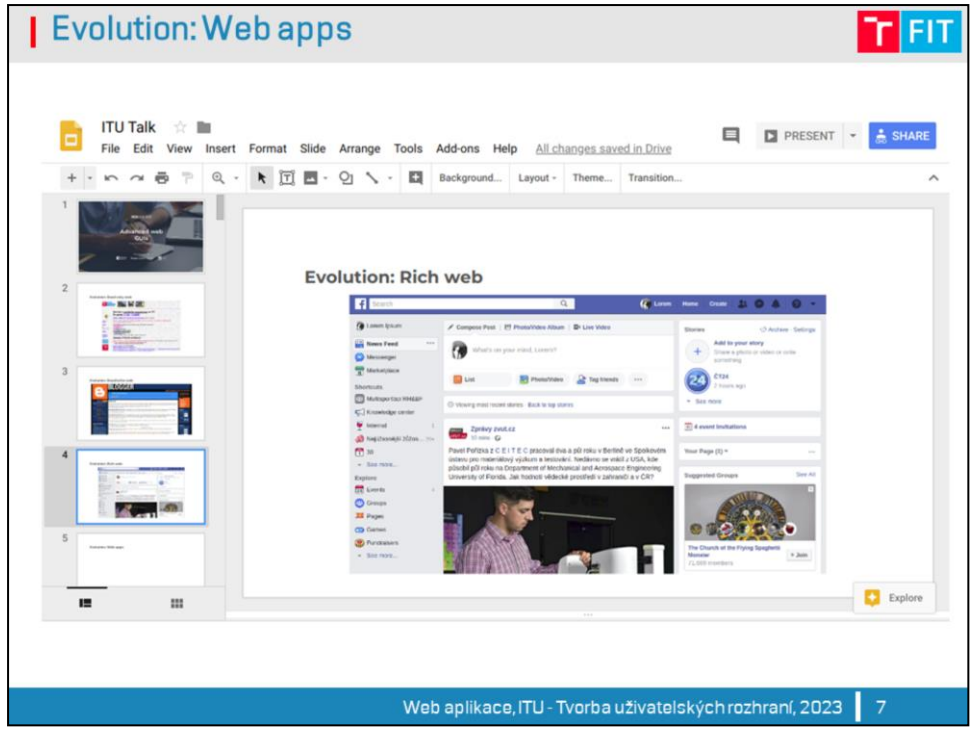


Web jak jej nyní známe
Rozhraní se skládá z množství dynamických prvků
Technicky vzato jde stále o formuláře - něco se vyplní/nastaví a nakonec odešle - vizuálně už ale vypadají značně jinak
Hlavní zlom nastal s možností spouštět requesty bez opětovného načítání stránky

- 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

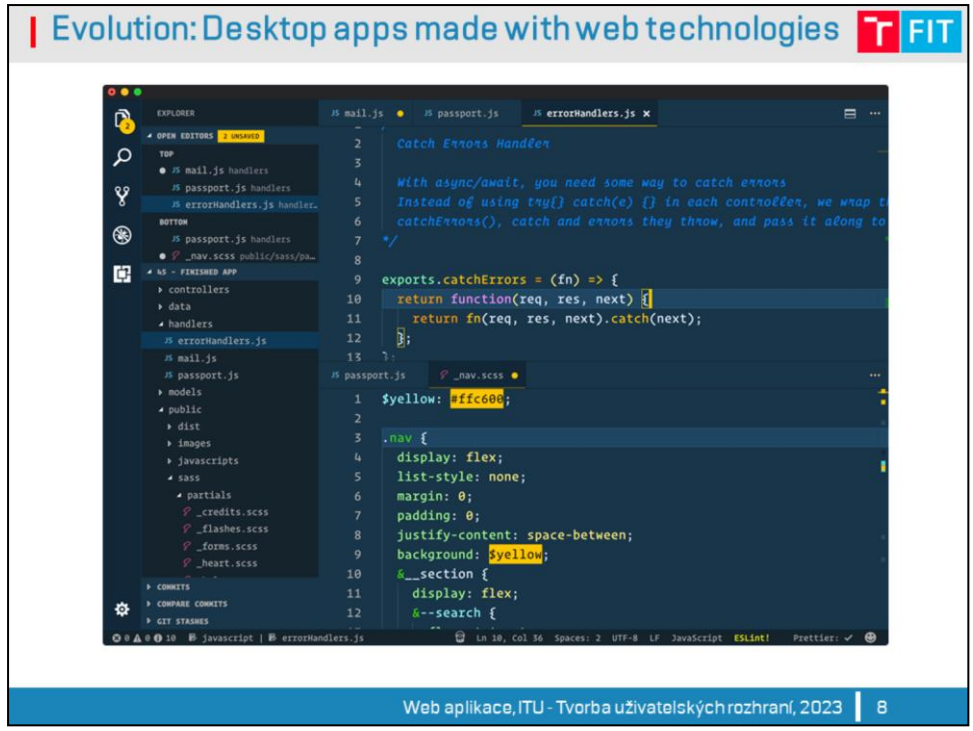
Web as we know it today
UI consists of many dynamic elements
Technically, they're still forms, but they often don't look that way
The deal breaker is the ability to do server requests without loading the whole page

- 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



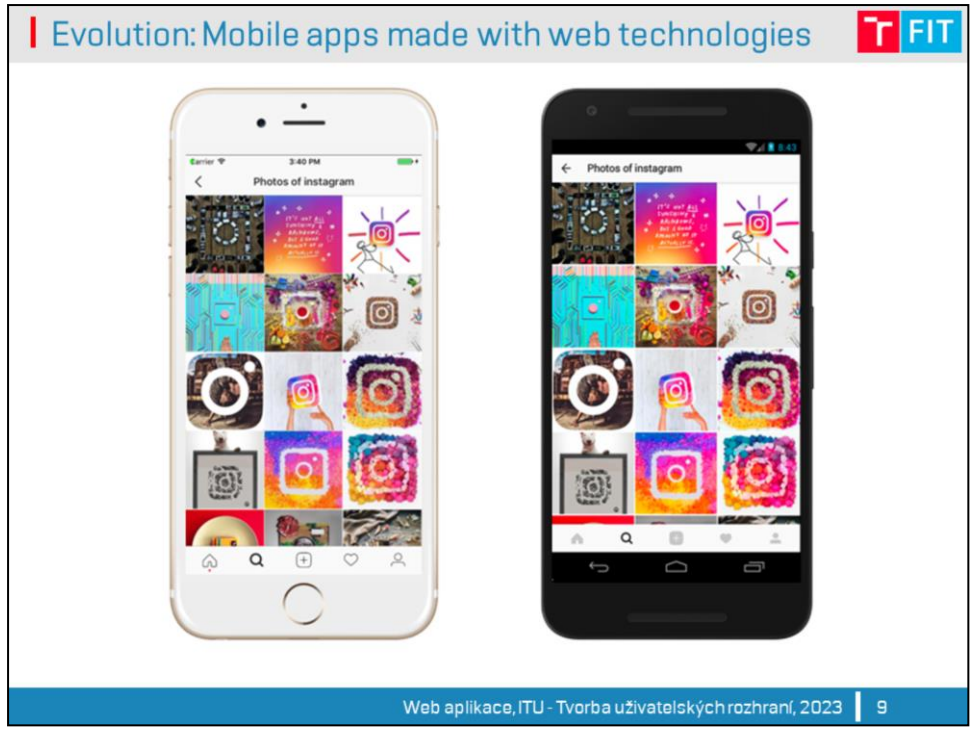
- Extrémním příkladem rozvoje webu jsou webové aplikace
  Jejich rozhraní je často k nerozeznání od desktopových verzí
  Formuláře už se vytrácejí a nahrazují je klasické prvky UI - lišty s ikonami, menu, kontextové menu
  Možnost interakce přímo se systémem skrze prohlížeč. Například drag & drop souborů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



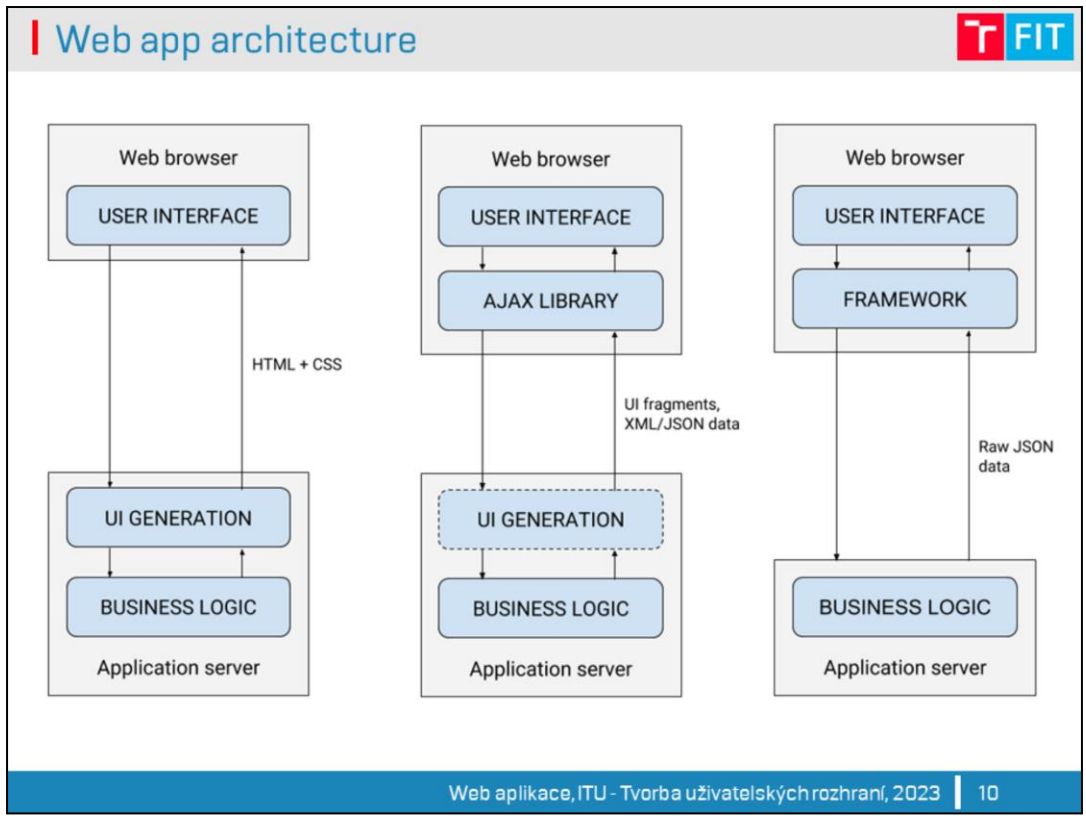
Díky rozvoji webových technologií, jejich přenositelnosti a popularitě se začaly objevovat desktopové aplikace na nich založené
Electron - JS knihovna obalující aplikaci do customizovaného Chromium okna. Výsledkem je dojem desktopové aplikace
Příklady: Visual Studio Code, Atom, Slack, Skype, GitHub

Thanks to popularity of web technologies, they started getting used for desktop apps too
Electron - JS library wrapping web app inside a customized Chromium window. The result looks like native desktop app
Examples: Visual Studio Code, Atom, Slack, Skype, GitHub



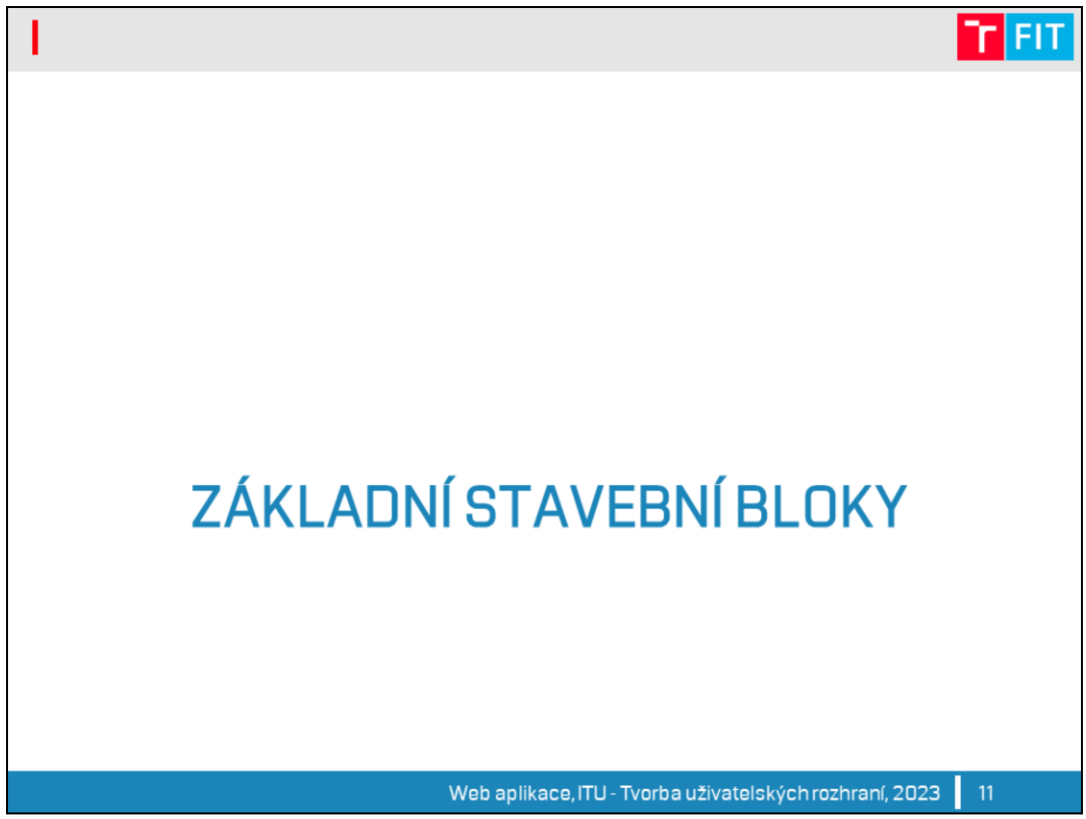
Webové technologie se prosazují také na mobilech a podobných zařízeních
Nativní aplikace může být i webová stránka běžící v nějakém wrapperu
Díky React Native lze dokonce použít pro mobilní aplikaci stejný kód jako pro její webovou verzi
Mobilní web přináší nové možnosti uživatelského vstupu (touch, 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,...

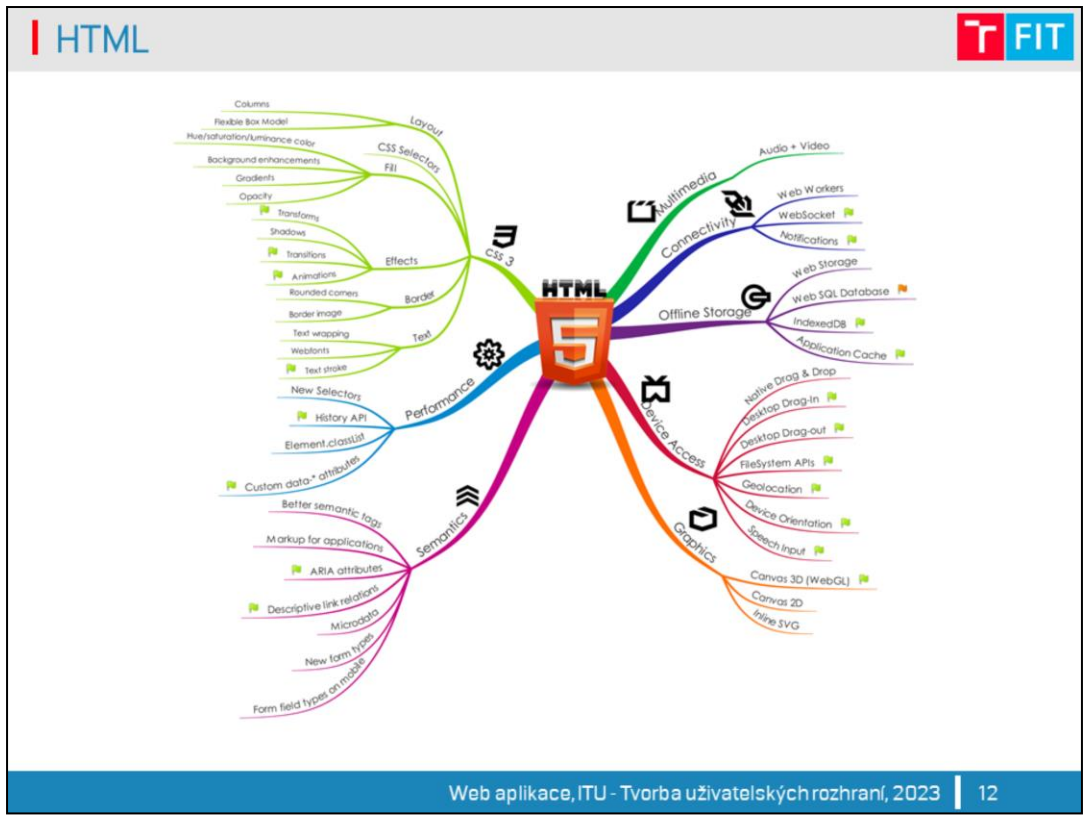
Web technologies are used on mobiles and mobile devices too
Native app can be just a website running inside a native wrapper
React Native lets developer use the same code for mobile app as for web
Mobile web brings new input methods (touch, multi-touch, photo, sound, location,...) which affects web technologies 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 snippety 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


FIT
ZÁKLADNÍ STAVEBNÍ BLOKY
Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 11



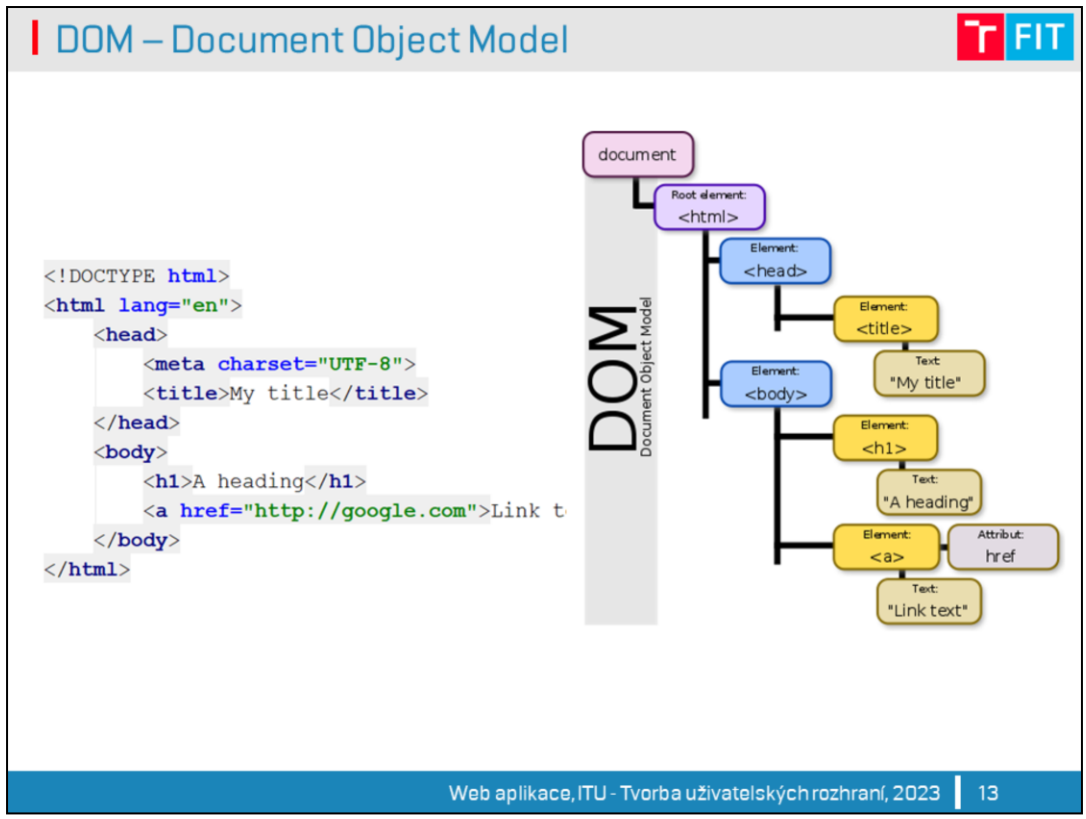
HTML – HyperText Markup Language

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
- Prezentace (X)HTML, XML a podobných formátů
- Jeden uzel je instance DOMHTMLElement
- Přístup k elementům dokumentu
- Změna obsahu nebo atributů elementů
- Změna struktury nebo stylu dokumentu

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

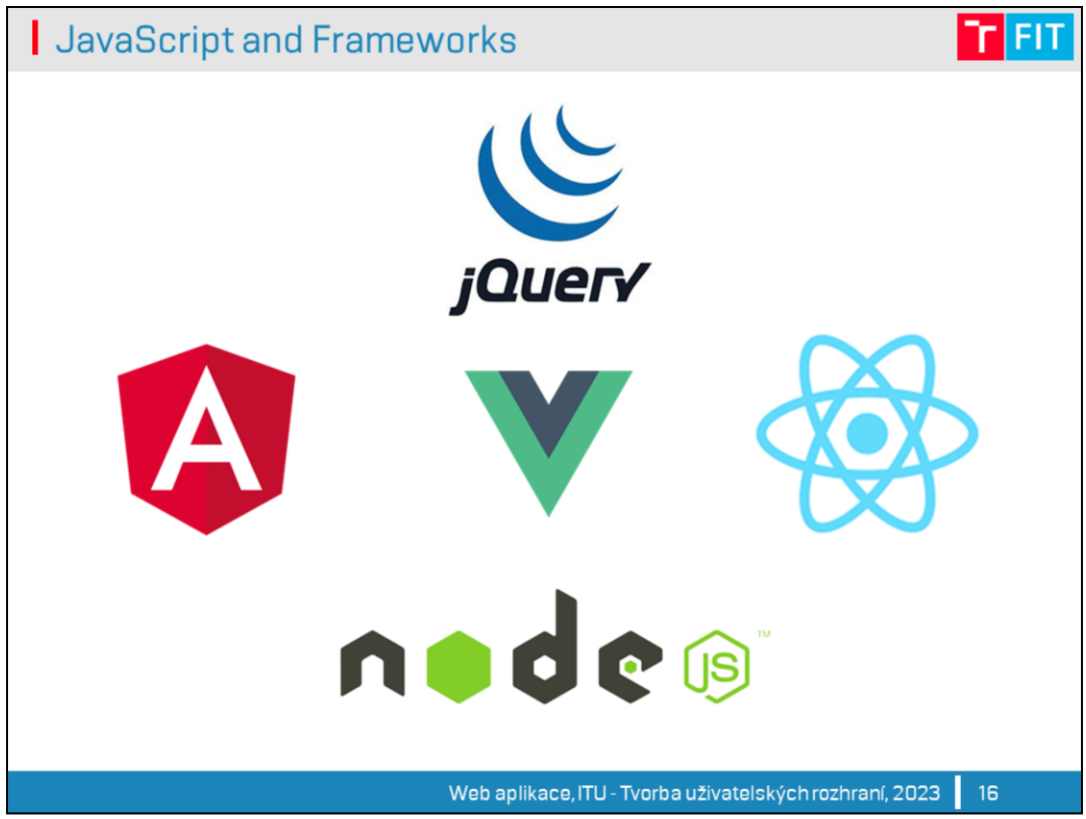
https://www.sassmeister.com/
http://lesscss.org/

CSS Frameworks

Komponenty uživatelského rozhraní

Rychlý vývoj, hezký design, responzivnost

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



JavaScript

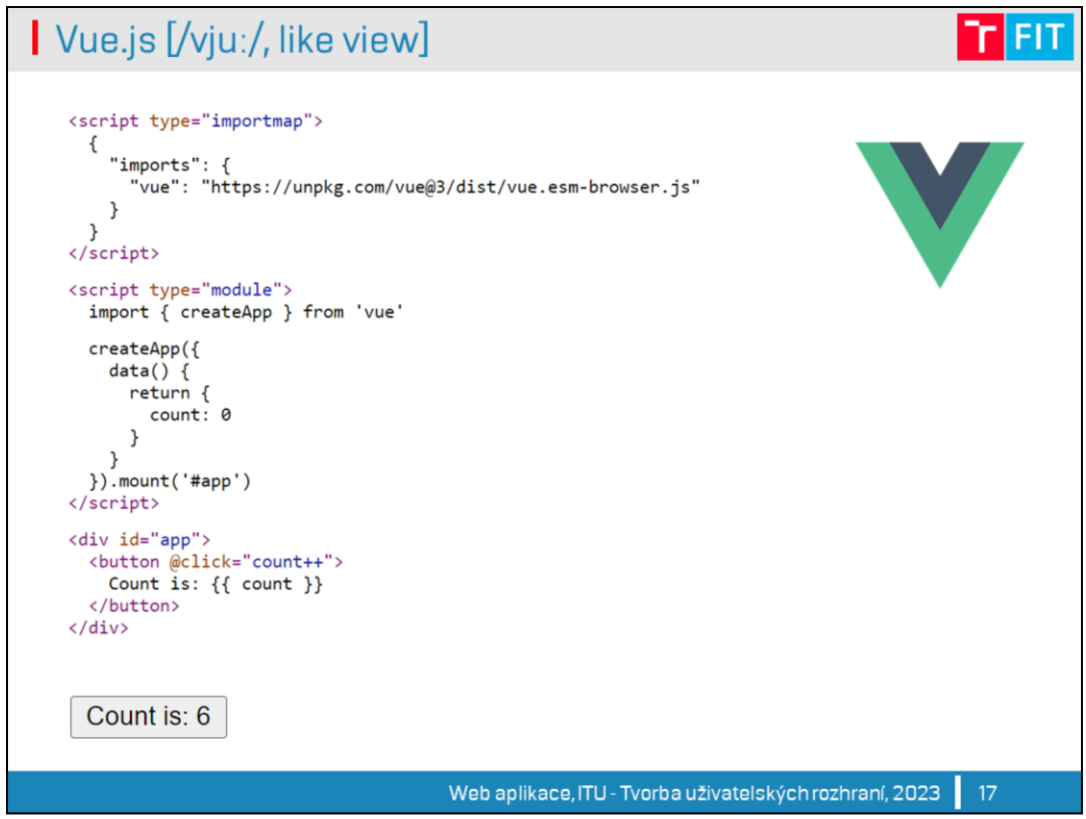
- **standard pro programování aplikací v HTML (interakce s DOM)**
- nemá nic společného s Javou, není nezávislý na klientovi

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

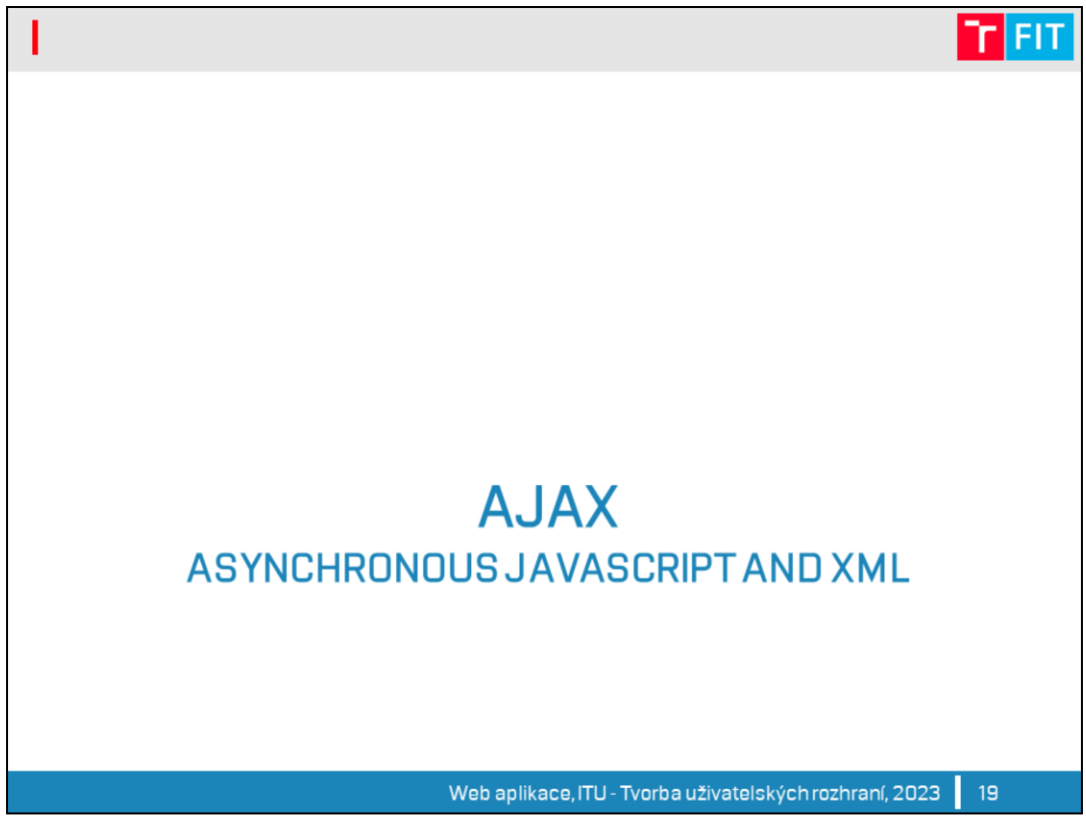
- progressive framework for building user interfaces
- designed from the ground up to be incrementally adoptable
- core library is focused on the view layer only
- easy to pick up and integrate with other libraries or existing projects

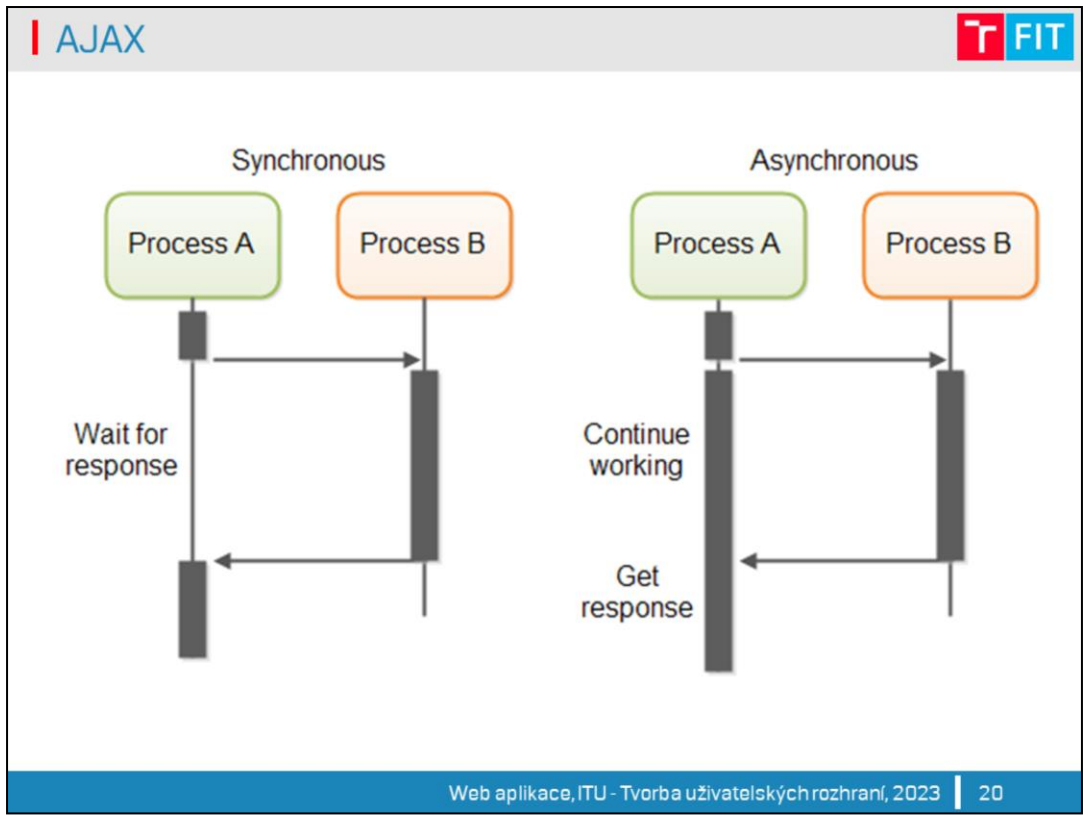
https://vuejs.org/v2/guide/

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) {...}


FIT
AJAX
ASYNCHRONOUS JAVASCRIPT AND XML
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

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 ramci stranky unikatni</div>
document.getElementById('mojeId').innerHTML = '<b>novy obsah
elementu</b>';
```

- Získání hodnoty z formulářového prvku

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

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 22

XMLHttpRequest – vytvoření objektu

- Ošetření různých prohlížečů

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

Vytvoření objektu → Nastavení parametrů → Odeslání → Obsluha událostí → Zobrazení výsledku

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 23

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;
```

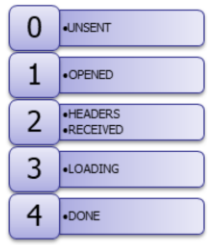
- Odeslání požadavku na server

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

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 24

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




```
function processData() {
    if (xmlHttp.readyState == 4) {
        if (xmlHttp.status == 200) {
            // ..
            // zpracování a zobrazení
            // odpovědi ze serveru
            // ..
        }
    }
}
```

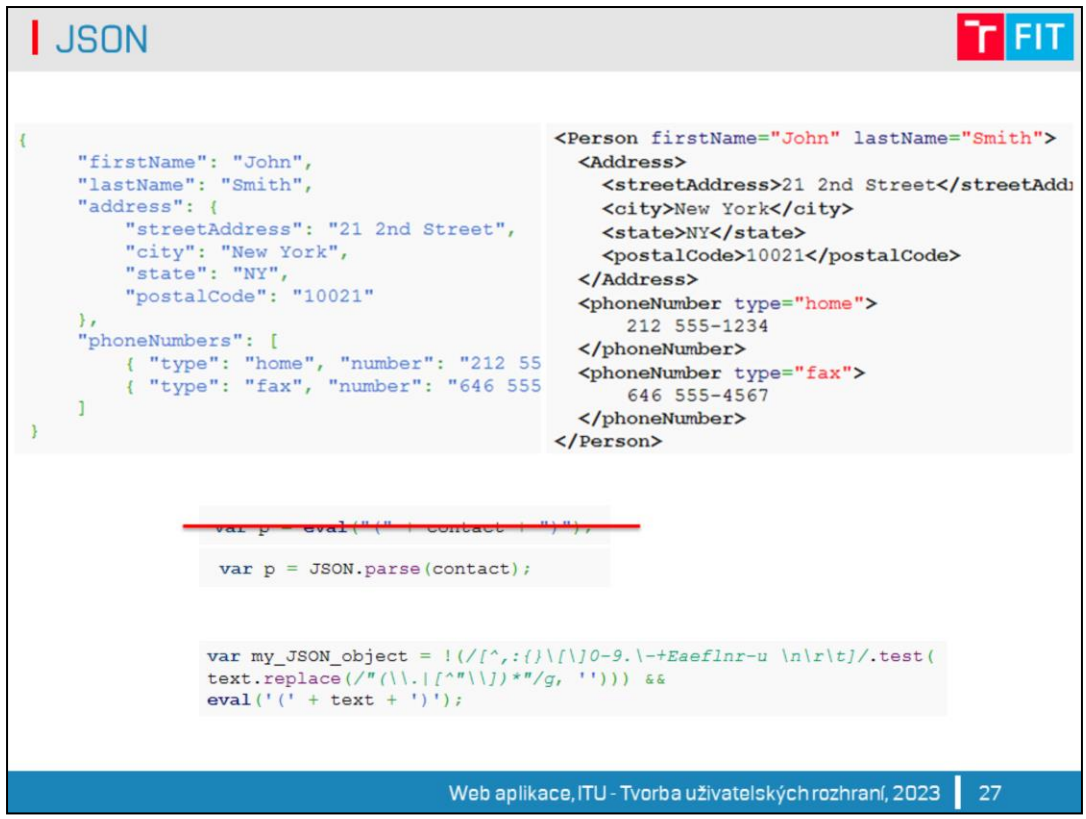
Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 25

XMLHttpRequest – zobrazení výsledku

- Formát dat odpovědi serveru – na cvičení: JSON

```
xmlHttp.onreadystatechange = function() {
    if ((xmlHttp.readyState == 4) && (xmlHttp.status == 200)) {
        console.log(xmlHttp.responseText);

        var pole = JSON.parse(xmlHttp.responseText);
        for (var i in pole) {
            console.log(pole[i]);
        }
    }
}
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 26



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

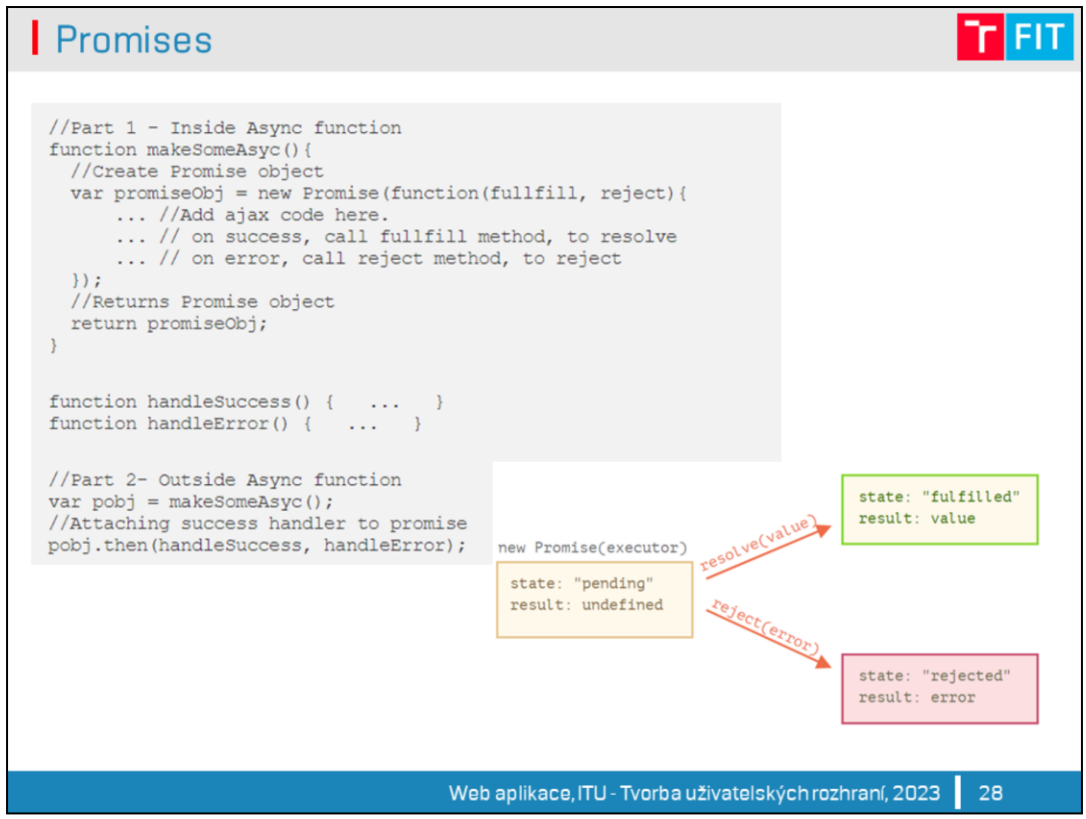
konstrukce JavaScript objektu

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



Promise – tool for managing asyc. 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

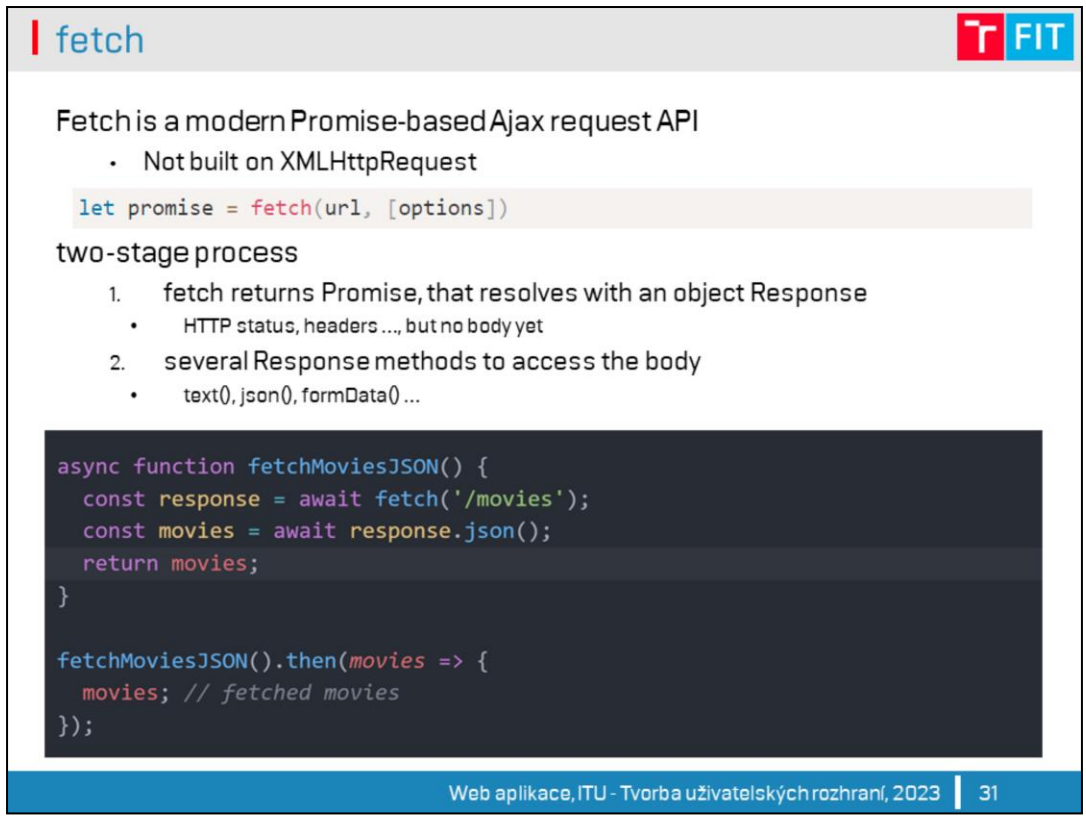
```
function makeAjaxCall(url, methodType){
   var promiseObj = new Promise(function(resolve, reject){
      var xhr = new XMLHttpRequest();
      xhr.open(methodType, url, true);
      xhr.send();
      xhr.onreadystatechange = function(){
      if (xhr.readyState === 4){
         if (xhr.status === 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;
}
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 29

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);
}
```

Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 | 30



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

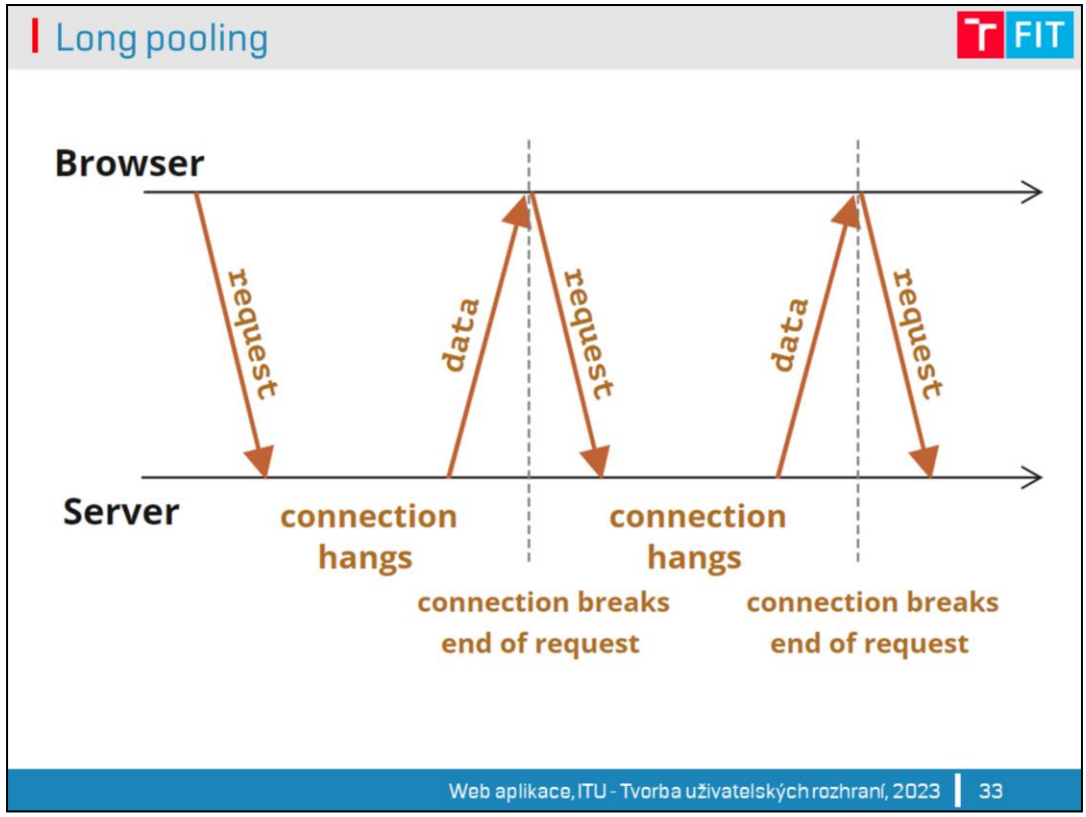
await – operator is used to wait for a Promise

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 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

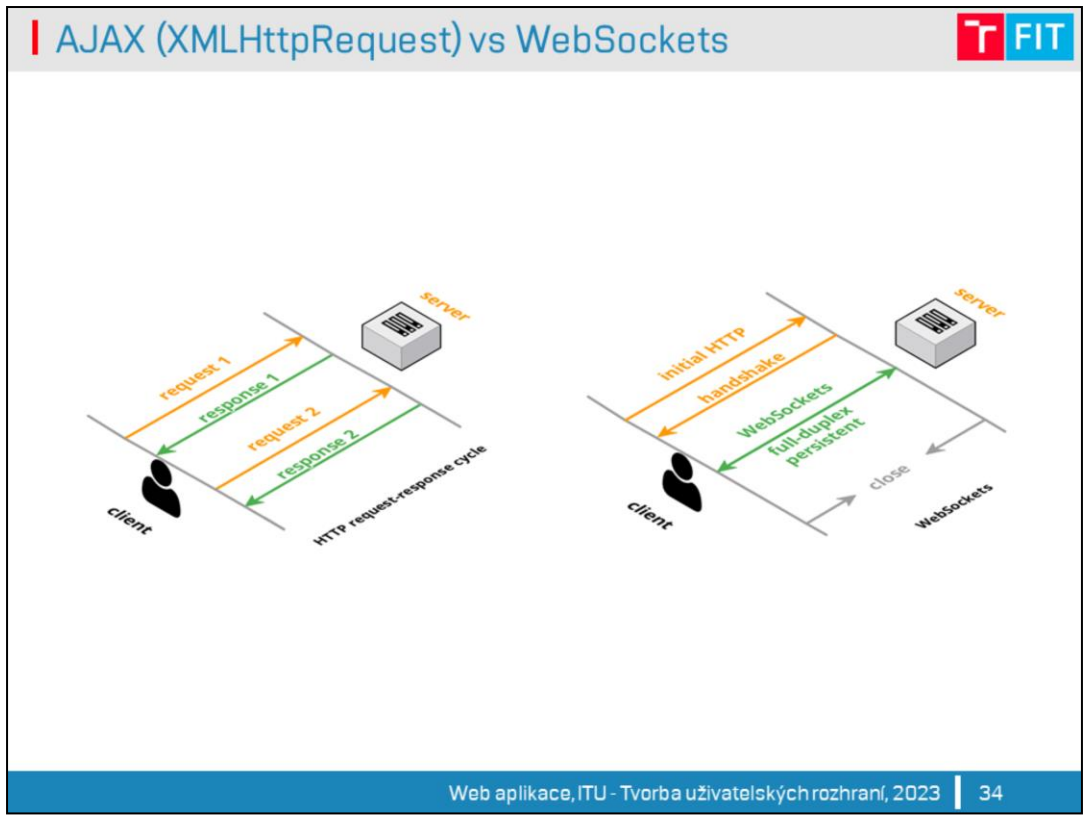
Periodic pooling, periodic requests – each 10s. High overheads – „large" headers, establishing/closing connection etc.

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

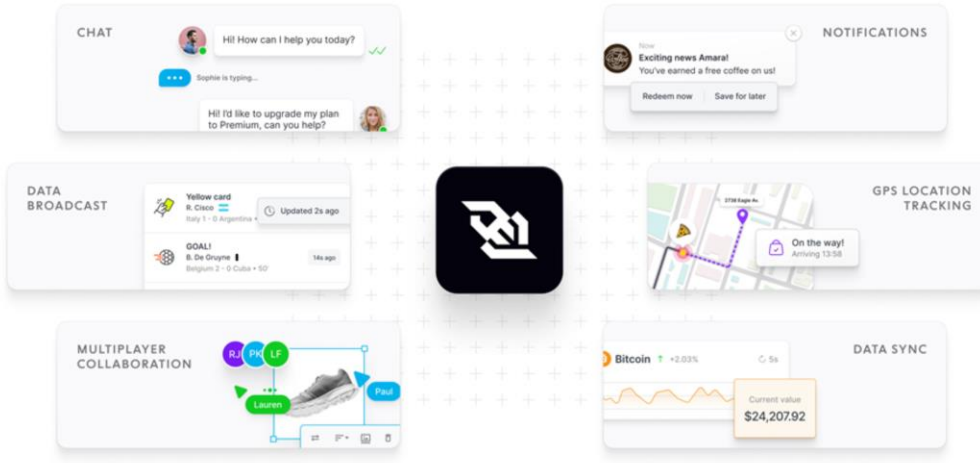
- Part of the new APIs in HTML 5
- Real-time two-way communication
- New protocol - requires both client and server support
- E.g. removes the need for periodic polling
- Variant with SSL (wss protocol)

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.

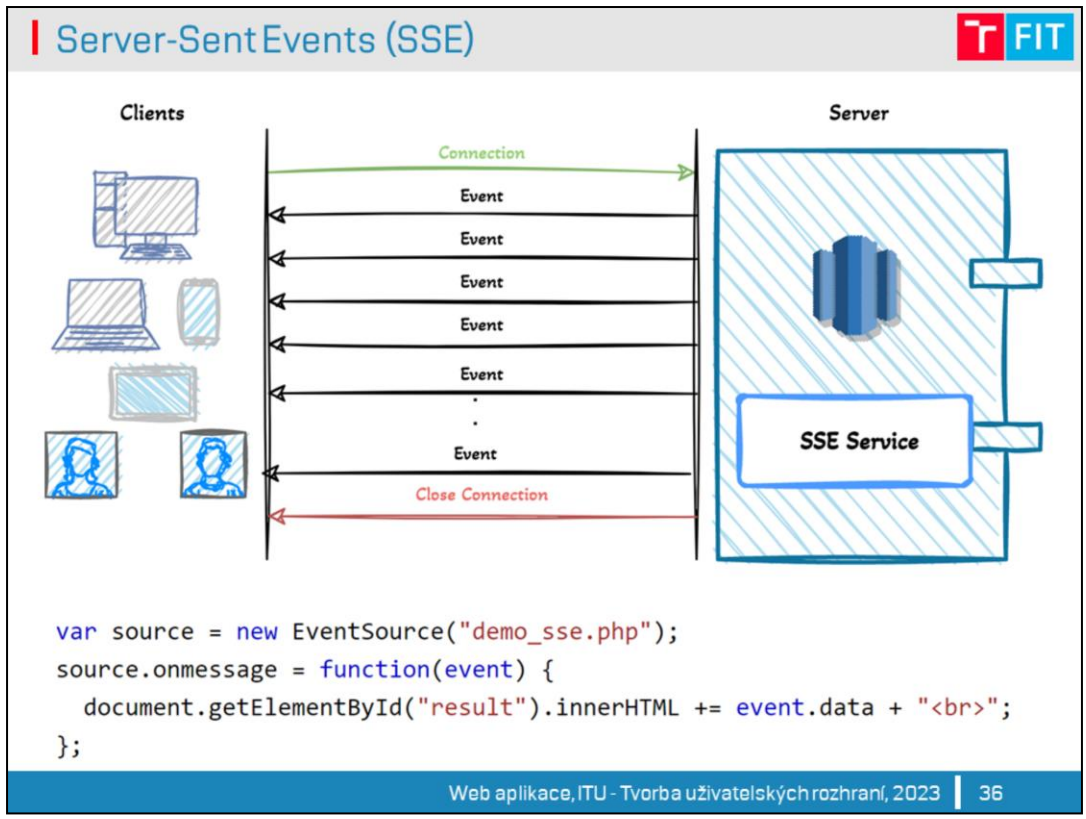
Websockets

good for realtime or near-realtime need for data consumption



not good always

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

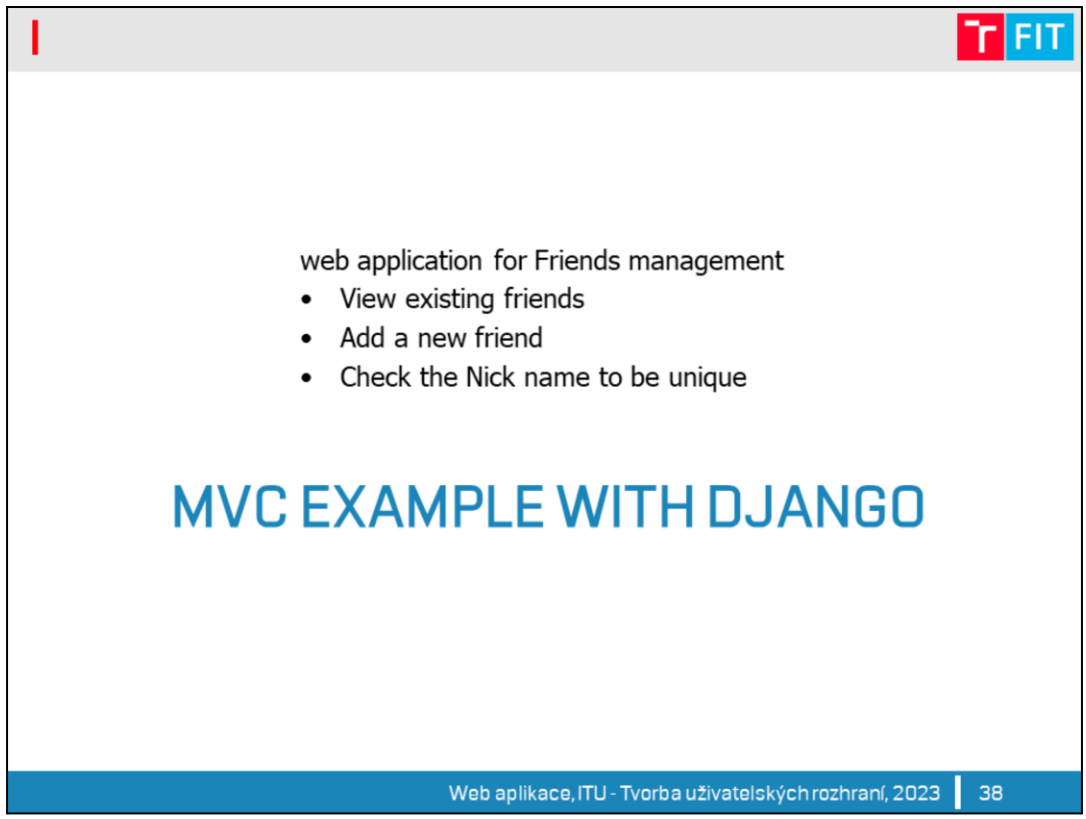


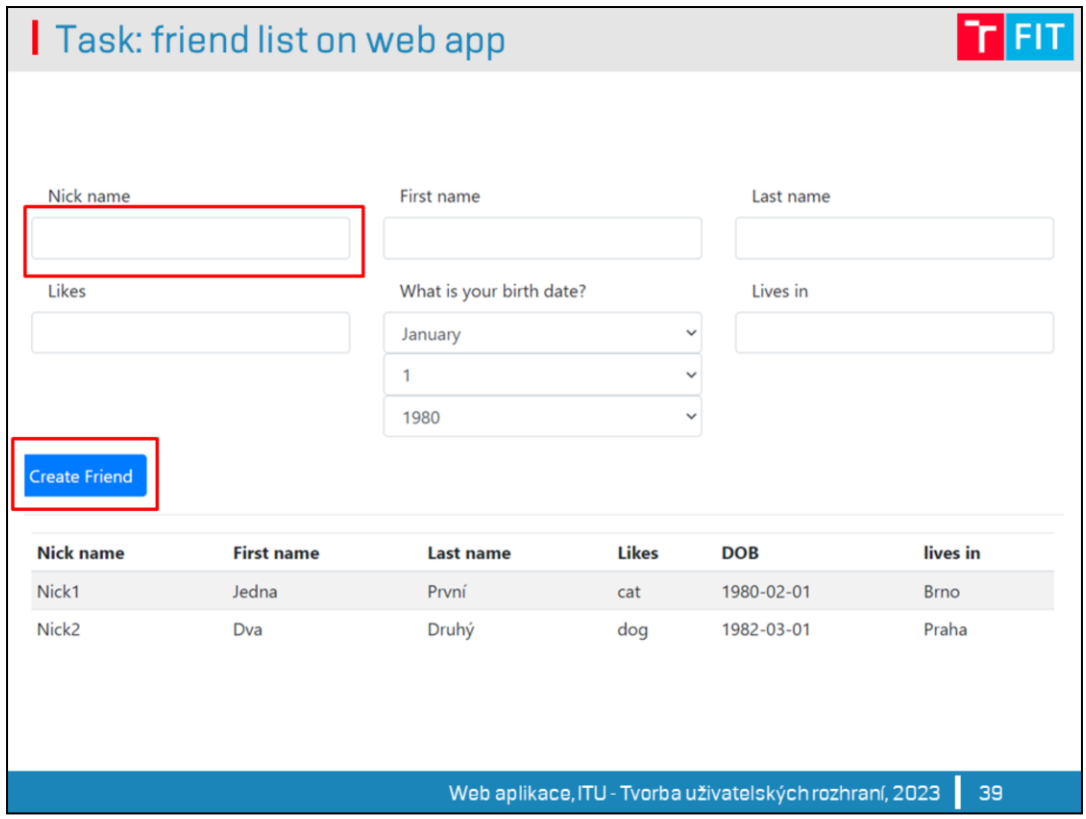
SSE is a technology that provides asynchronous communication with event stream from server to the client over HTTP for web applications. The server can send un-directional messages/events to the client and can update the client asynchronously.

CRUD

- 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


FIT
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

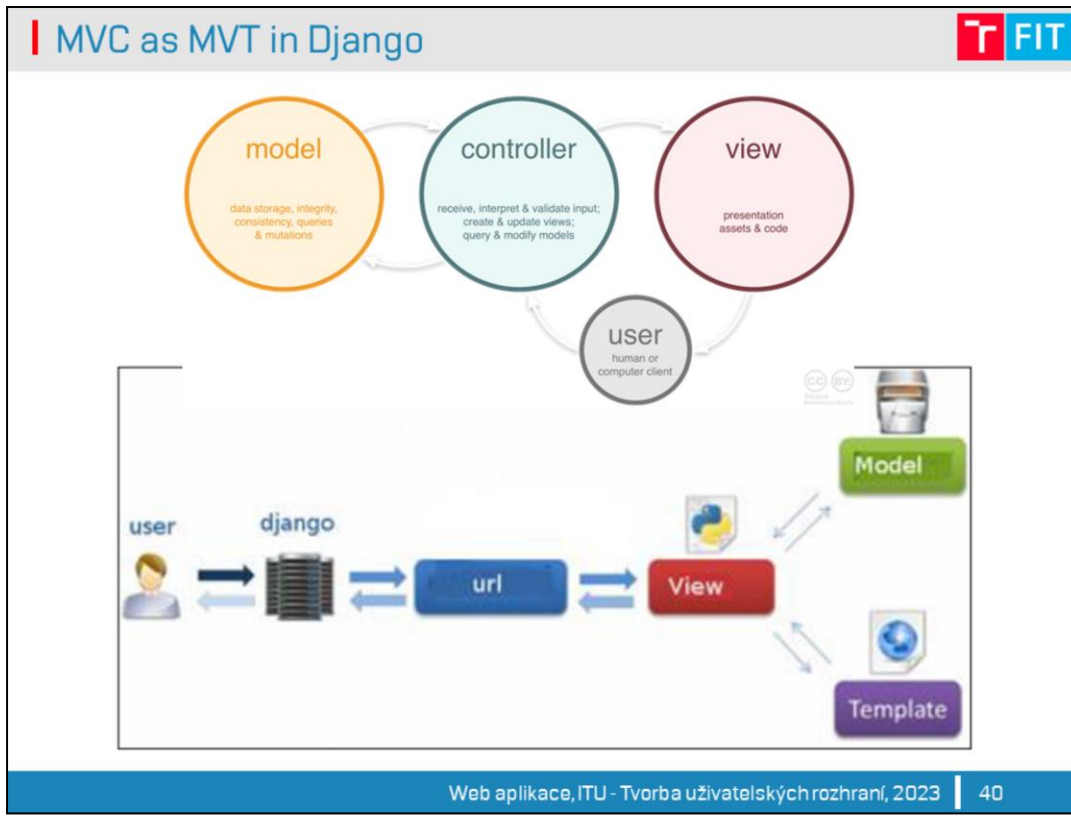


Task: web application for Friends management

- 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/



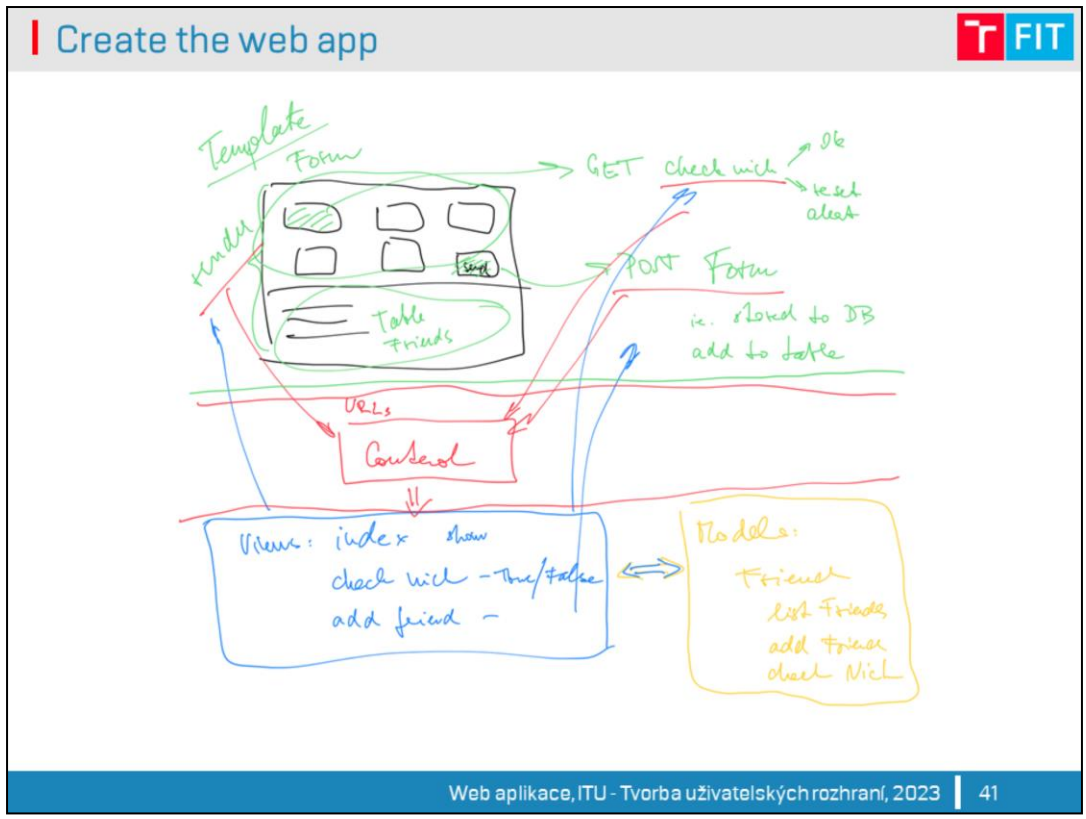
Django

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

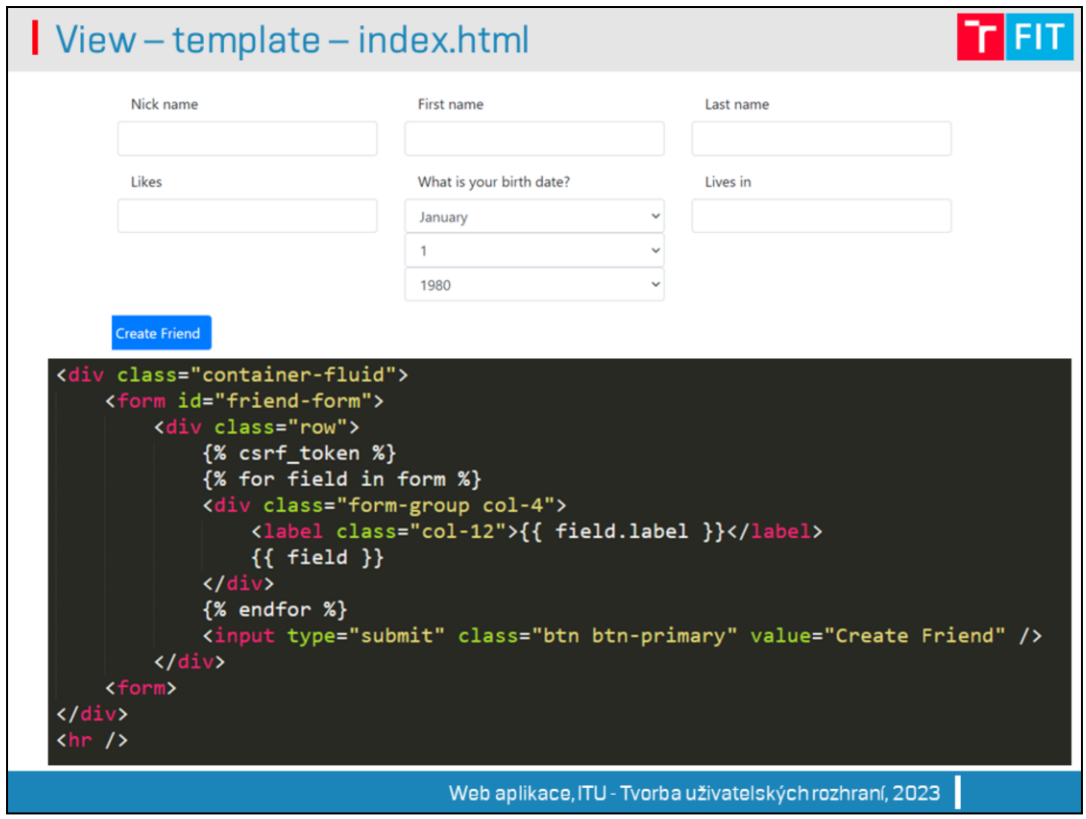
Django implementuje MVC jako MVT (Model-View-Template)

- Controller
  - je víceméně vestavěný jakou 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 date 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 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 fileds of the Form
- Form is generated from particular attributes of the Model (see slide Model and Form)

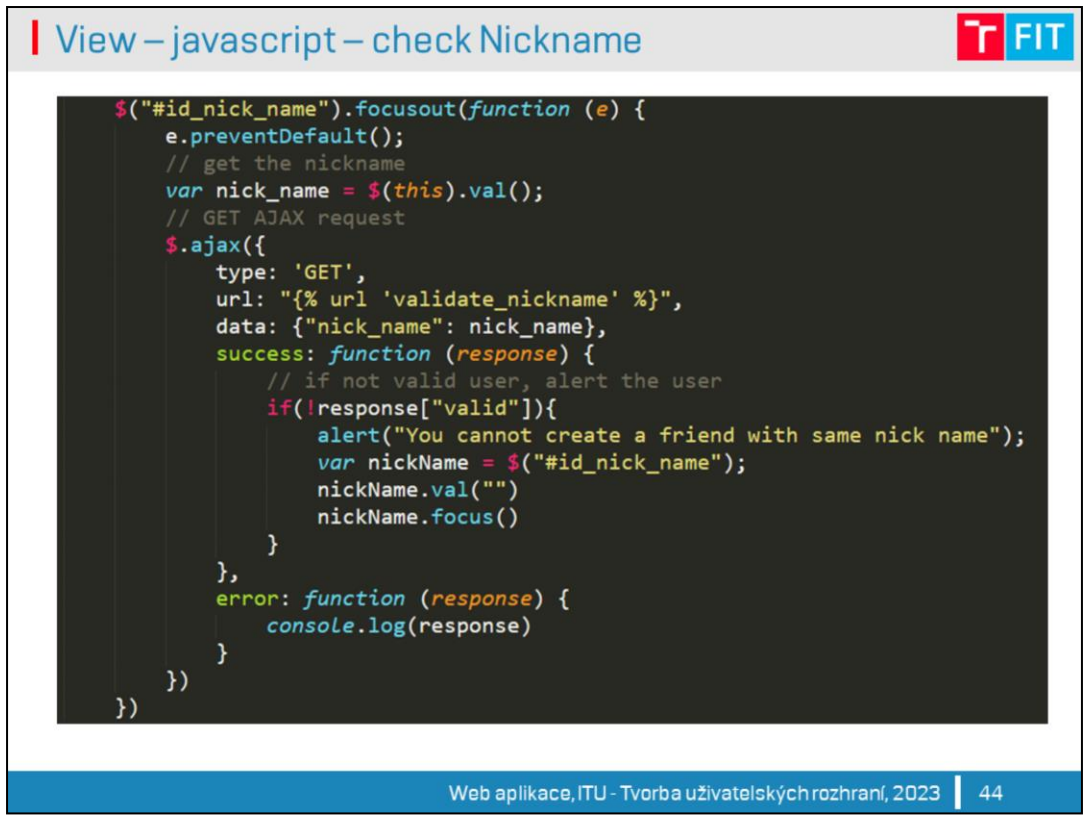
Cross Site Request Forgery - prevence

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)

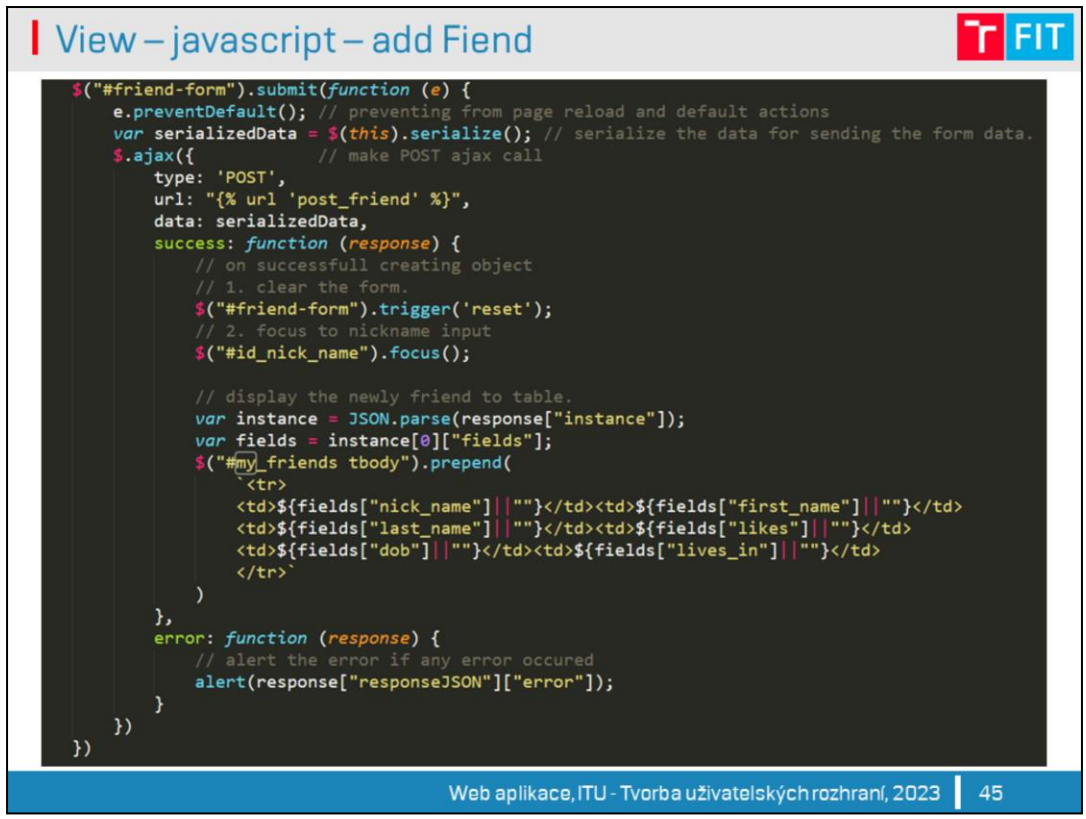


View and user interaction – html template

- 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

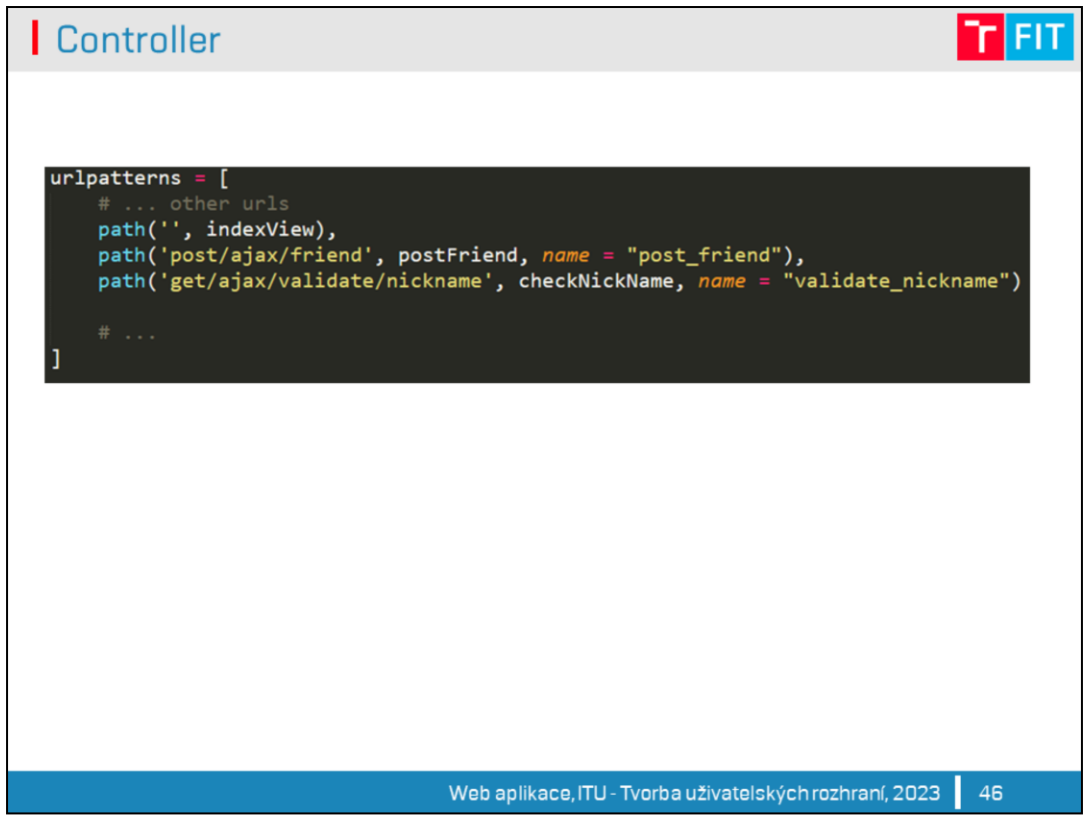


View and user interaction – html template

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

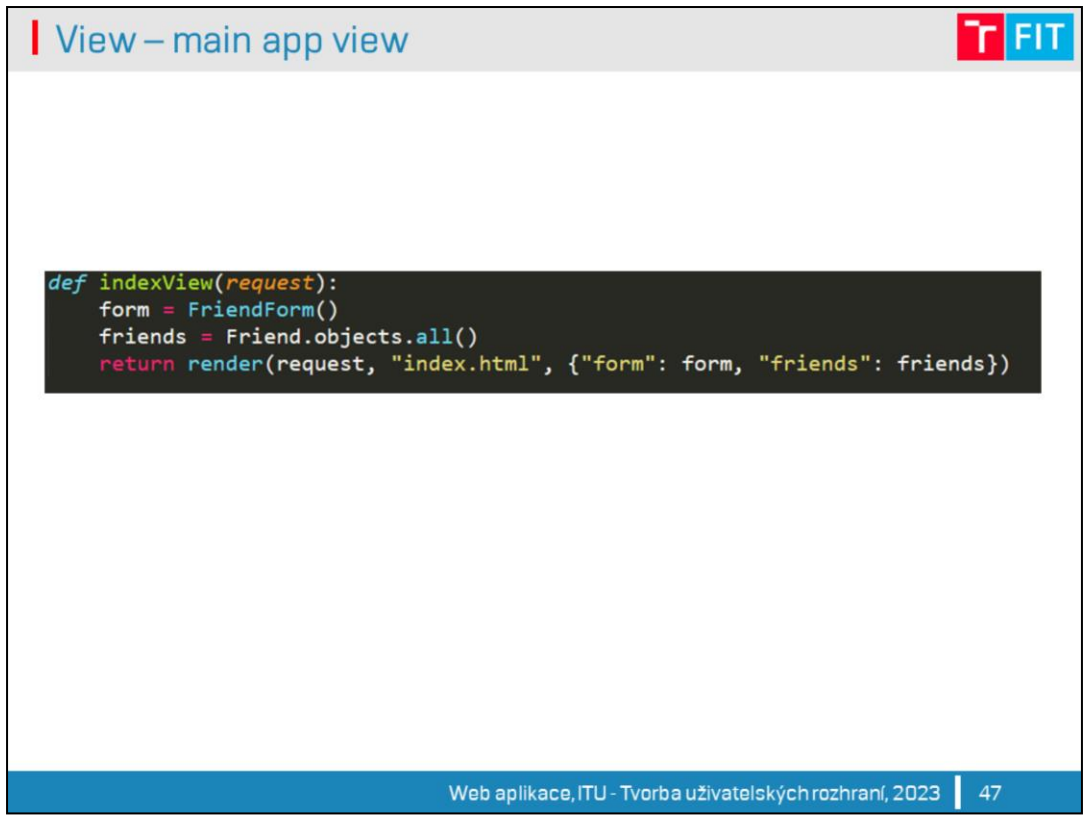
Send the form values to add new Friend

- 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

- Maps user requests to view functions



- 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 uniquness of the given Nickname using Model API
- confirm by sending the result back in responce
- will be catched 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 Frieds

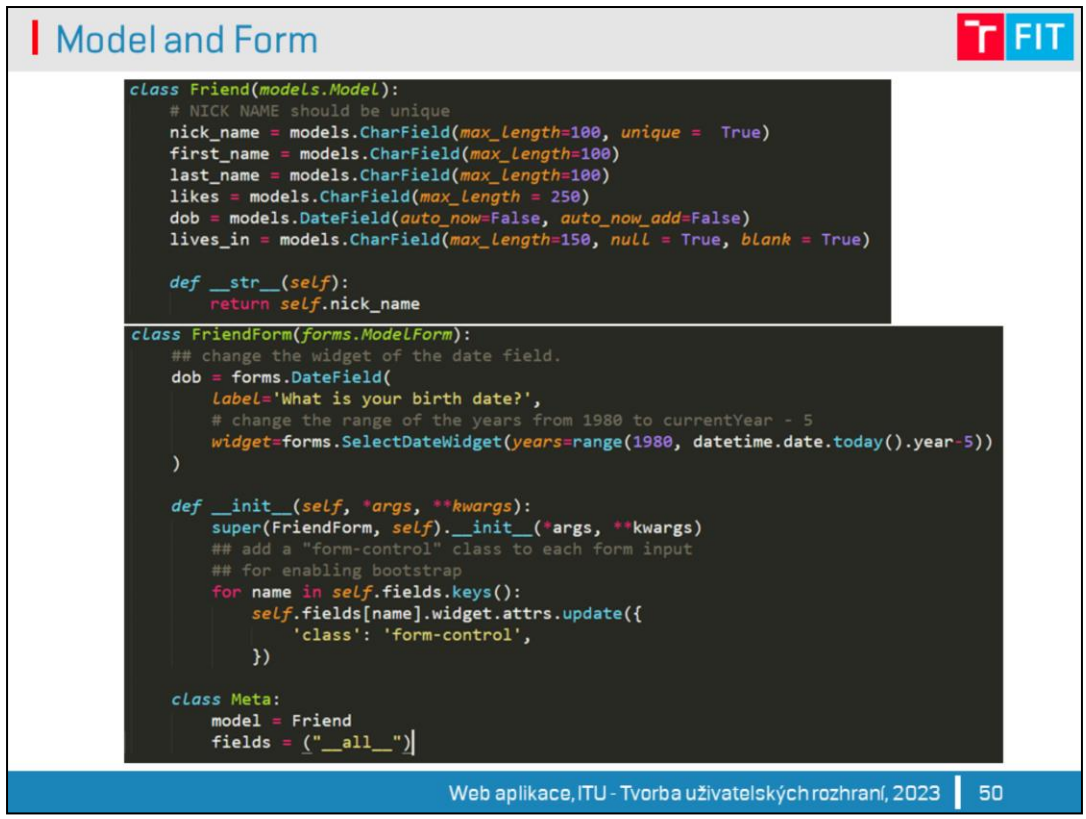
```python
def postFriend(request):
    # request should be ajax and method should be POST.
    if request.is_ajax and request.method == "POST":
        # get the form data
        form = FriendForm(request.POST)
        # save the data and after fetch the object in instance
        if form.is_valid():
            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 occured.
            return JsonResponse({"error": form.errors}, status=400)

    # some error occured
    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 responce
- will be catched by function sending the request (AJAX)
- the funciton processes the response - show data in the table/list of Friends (see slide Add Friend)



Model

- Defines attributes/fields

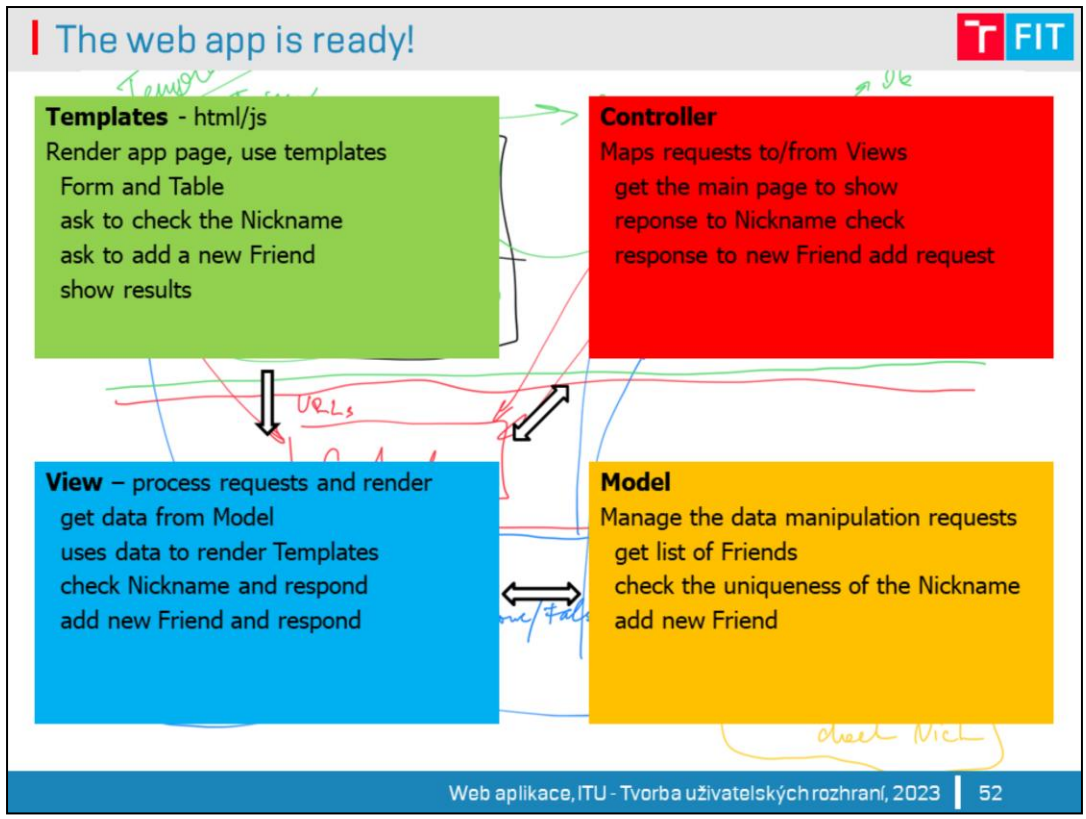
Form

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

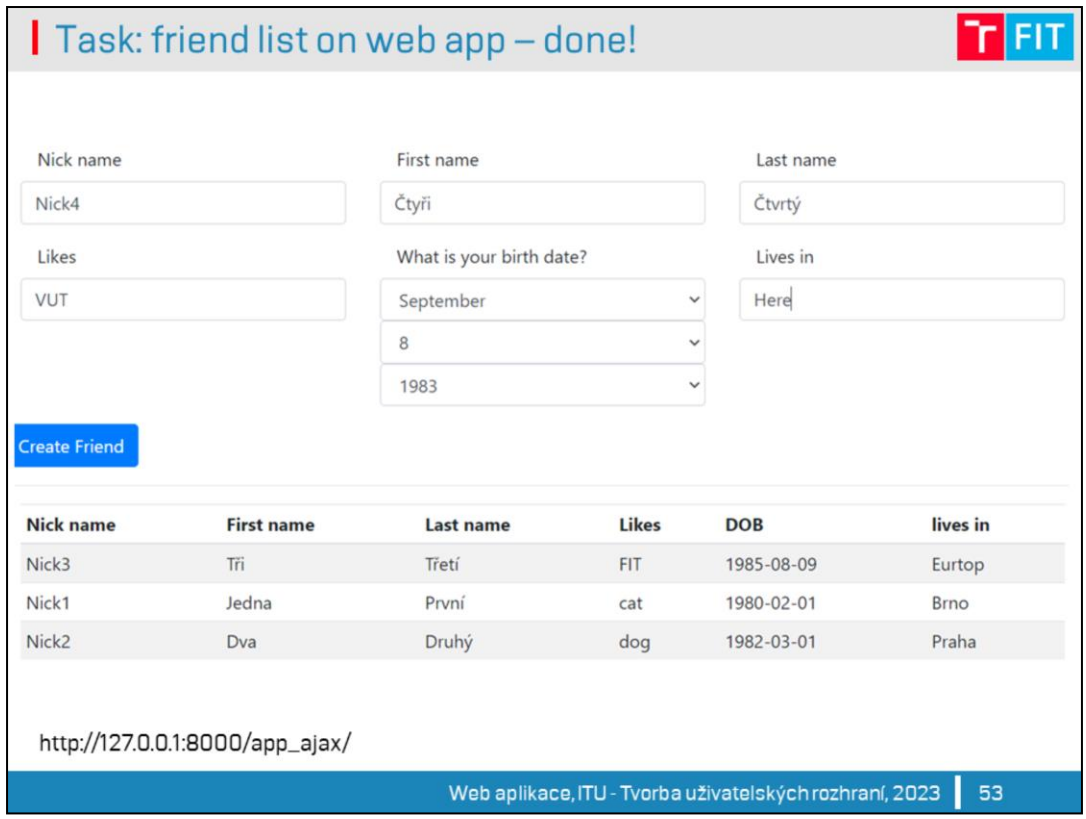
View and user interaction – html template

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

The main teplate, defines the web app structure



- Create date 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

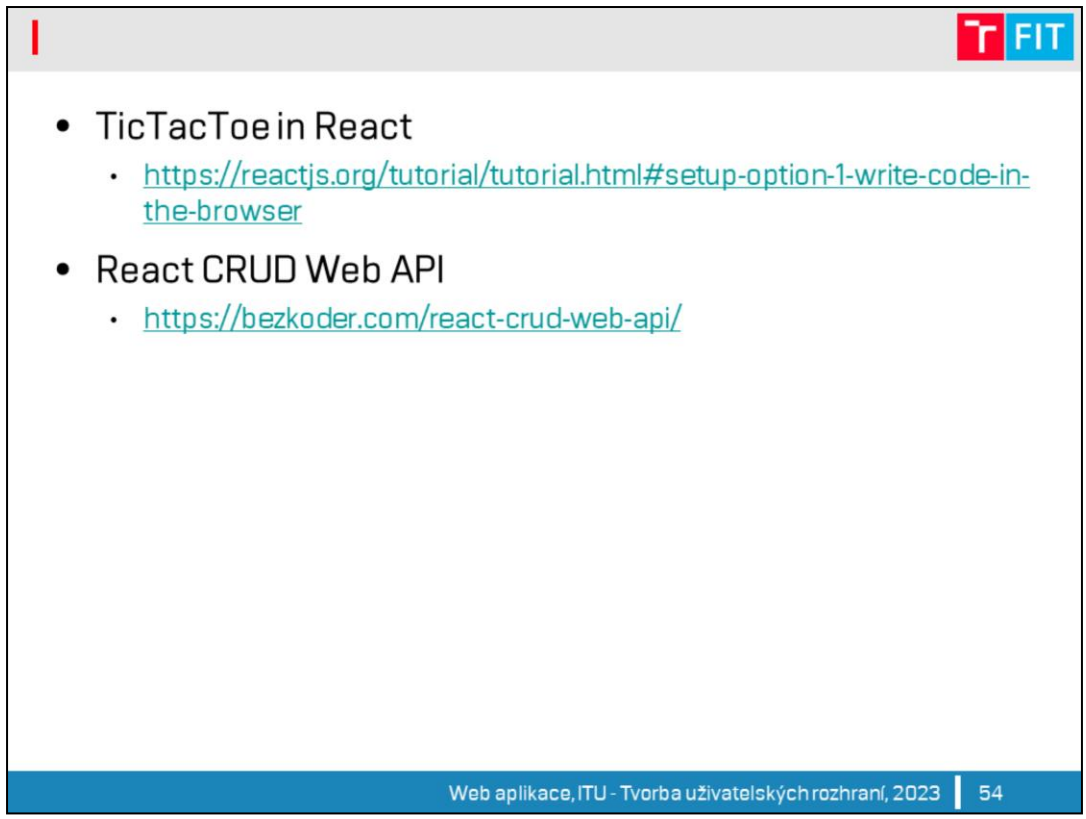


Task: web application for Friends management

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


FIT
• TicTacToe in React
• https://reactjs.org/tutorial/tutorial.html#setup-option-1-write-code-in-the-browser
• React CRUD Web API
• https://bezkoder.com/react-crud-web-api/
Web aplikace, ITU - Tvorba uživatelských rozhraní, 2023 54