

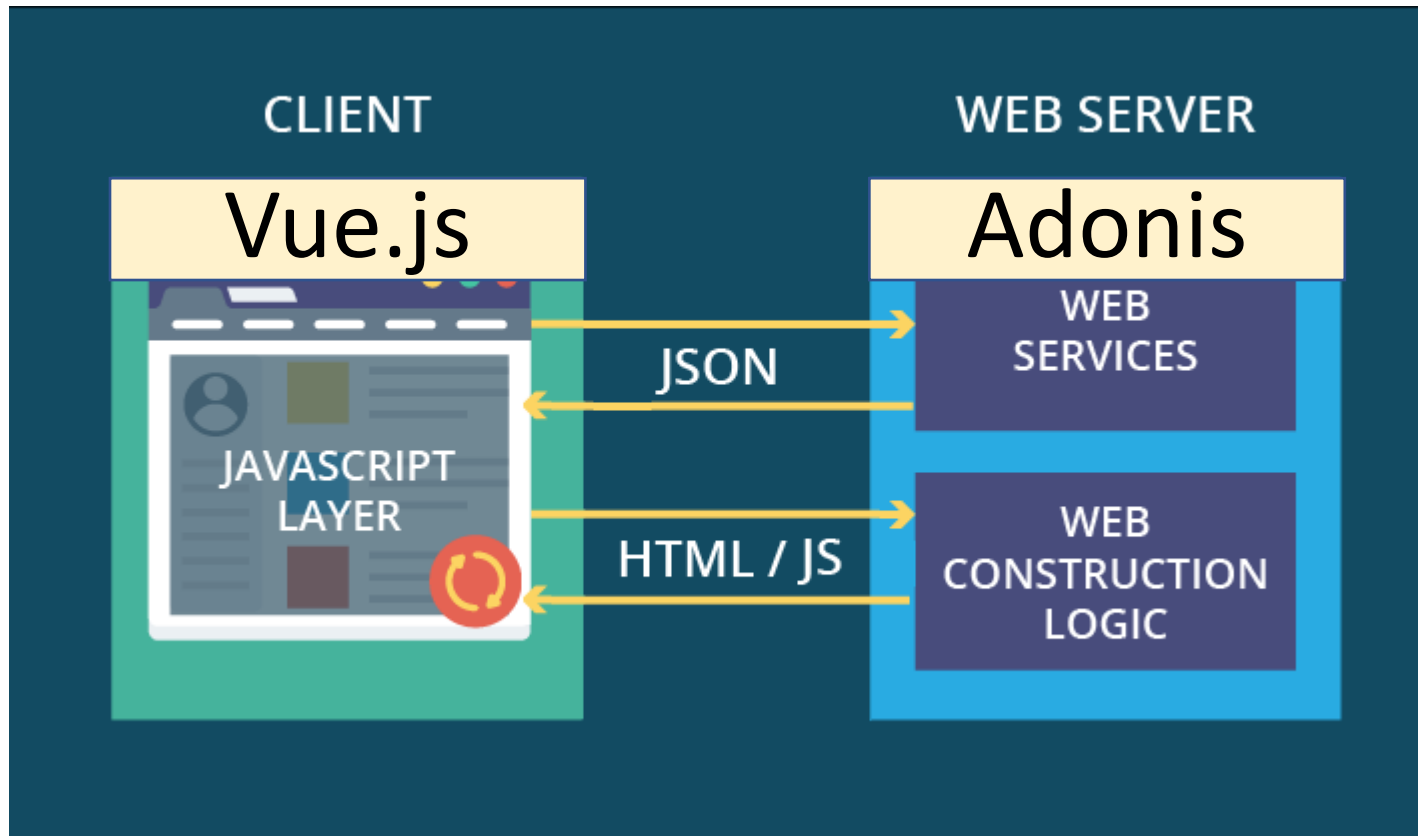


SSR, SSG, Hydration, PWA

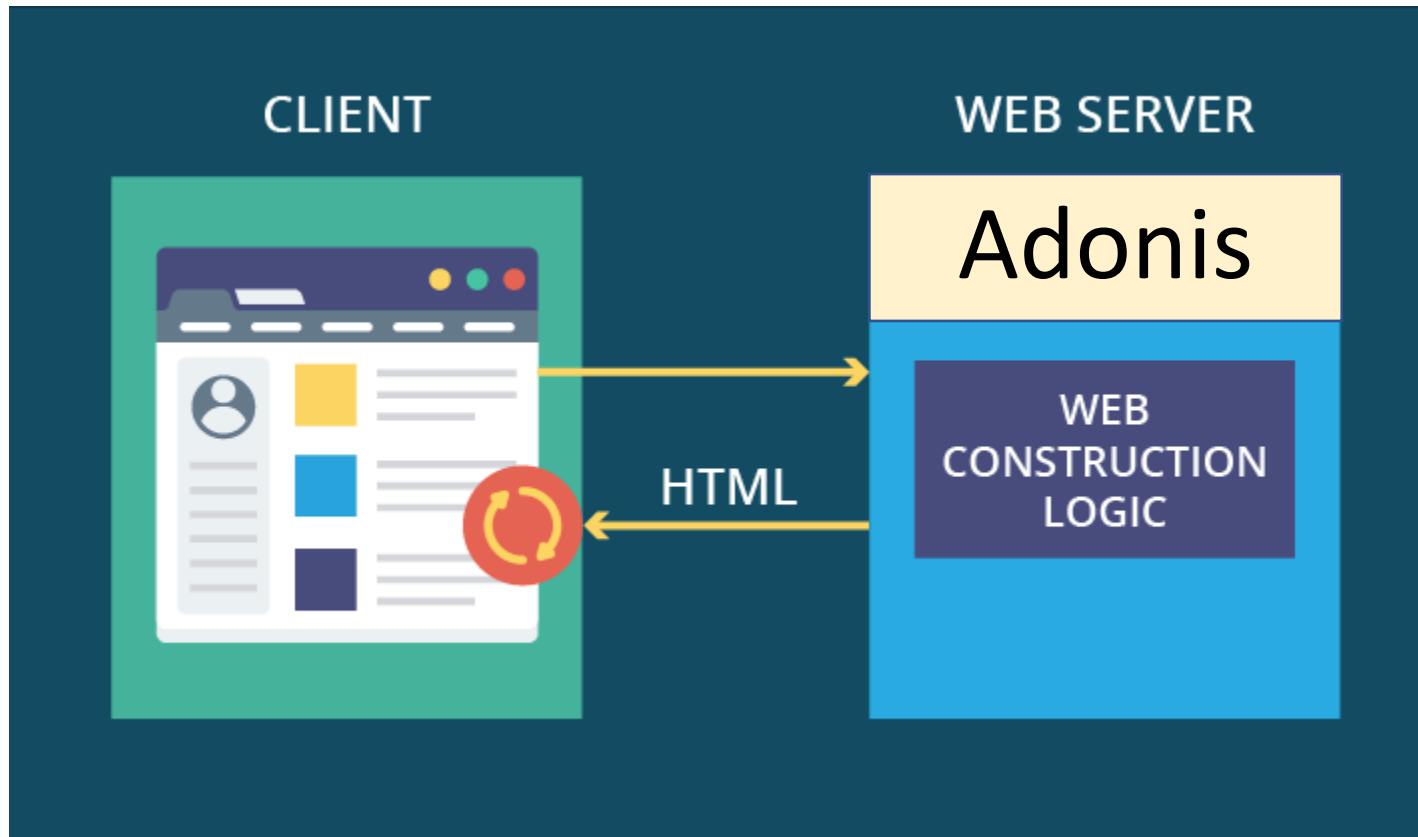
kurz Vývoj progresívnych
webových aplikácií

Eduard Kuric

Vue.js SPA – Adonis.js



SSR Node.js (Adonis.js/Laravel)



Vieme, Vue.js ...

- Je rámec na vytváranie aplikácií na strane klienta
- Vue komponenty
 - produkujú
 - a manipulujú DOM
 - vo webovom prehliadači.

Vue.js ako SSR

- Komponenty môžeme zostaviť na strane servera do HTML
- Vygenerované HTML vrátiť webovému prehliadaču
- HTML zhydratovať do reaktívnej Vue aplikácie
- **Načo Vue SSR?**
 - SPAčky si pýtajú veľa pri prvom doručení obsahu (pozrite [Core Web Vitals Metrics](#))
 - Lepšie SEO optimalizácie, webové prehliadače vidia priamo zostavený HTML

Ako Vue SSR?

- Vytvorme si package.json súbor:

```
npm init -y
```

- Pridajme podporu ES modulov:

```
"type": "module"
```

- Nainštalujme vue:

```
npm install vue
```

example.js (Vue SSR)

```
import { createSSRApp } from 'vue'
import { renderToString } from 'vue/server-renderer'

const app = createSSRApp({
  data: () => ({ count: 1 }),
  template: `<button @click="count++">{{ count }}</button>`
})

renderToString(app).then((html) => {
  console.log(html)
})
```

example.js (Vue SSR)

- Spustime aplikáciu:

```
node example-vue-ssr.js
```

```
// vystup
```

```
<button>1</button>
```


Express framework

- Minimalistický webový rámec nad Node.js poskytující základný aparát na zjednodušení vývoje (MVC) aplikací
- Prináša podporu:
 - smerovanie (routing)
 - middleware
 - šablóny
 - integráciu s DB systémami
- <https://expressjs.com/>
- `npm install express`

Express framework (server.js)

[ukážka k prednáške]

```
node server-express.js
```

<http://localhost:3000>

Hydratácia (hydration)

- prijatý (statický) HTML obsah (DOM) môžeme na klientovi **hydratovať** na dynamický obsah (reaktívny virtuálny DOM)
- hydratácia je znovunavrátanie reaktívnej podoby statickému HTML obsahu (prijatý zo serveru)
- Vue nájde **mapovanie medzi DOMom vráteným zo servera a virtuálnym domom vygenerovaným klientskou aplikáciou**
 - tomuto hovoríme hydratácia

Hydratácia (hydration) /2

[ukážka k prednáške]

- client-hydration.js
- server-hydration.js
- app-hydration.js – univerzálny kód, zdieľaný kód klientom/serverom

Static Site Generation (SSG)

- za účelom zlepšenia SEO marketingových stránok (/contact, /about) môže postačiť pre-rendering
- miesto kompilácie komponentov serverom do HTML (on-the-fly), **prerendering vygeneruje statický HTML obsah (pri kompilácii app) pre špecifikované smerovanie (routes)**

Quasar podporuje SSR

- <https://quasar.dev/quasar-cli-vite/developing-ssr/introduction>
- <https://quasar.dev/quasar-cli-webpack/developing-ssr/configuring-ssr>
- <https://quasar.dev/quasar-cli-webpack/developing-ssr/preparation>

WEB API prehliadačov

- Manipulácia s dokumentmi (DOM)
- Získavanie údajov zo servera (AJAX, Fetch API)
- Vykreslenie a tvorbu grafiky (Canvas, WebGL)
- Audio a Video (HTMLMediaElement, Web Audio API, WebRTC)
- Prístup k zariadeniam (Geolocation, Notifications, Vibration API)
- Ukladanie údajov na strane klienta (Web Storage, IndexedDB)

Notification API

```
let promise = Notification.requestPermission();  
var notification = new Notification("Hi there!");
```

- <https://chrisdavidmills.github.io/notification-test/>

Push API

- Súčasťou WEB API

Push API umožňuje webovým aplikáciám prijímať správy zo servera

- bez ohľadu na to, či je webová aplikácia v popredí
- alebo je dokonca otvorená vo webovom prehliadači (je prehliadač zapnutý)

PWA – Progressive Web Apps

- Fungujú pre každého používateľa, bez ohľadu na prehliadač/zariadenie
- Vyzerajú ako natívne aplikácie, správajú sa tak, sú „súčasťou plochy“ – push notifikácie
- Vždy aktuálne (najnovšia verzia) bez potreby sťahovania celej aplikácie
- Bezpečné (HTTPS)
- Jednoducho zdieľateľné cez URL
- Umožňujú pracovať offline, alebo na sieťach s nízkou prenosovou rýchlosťou

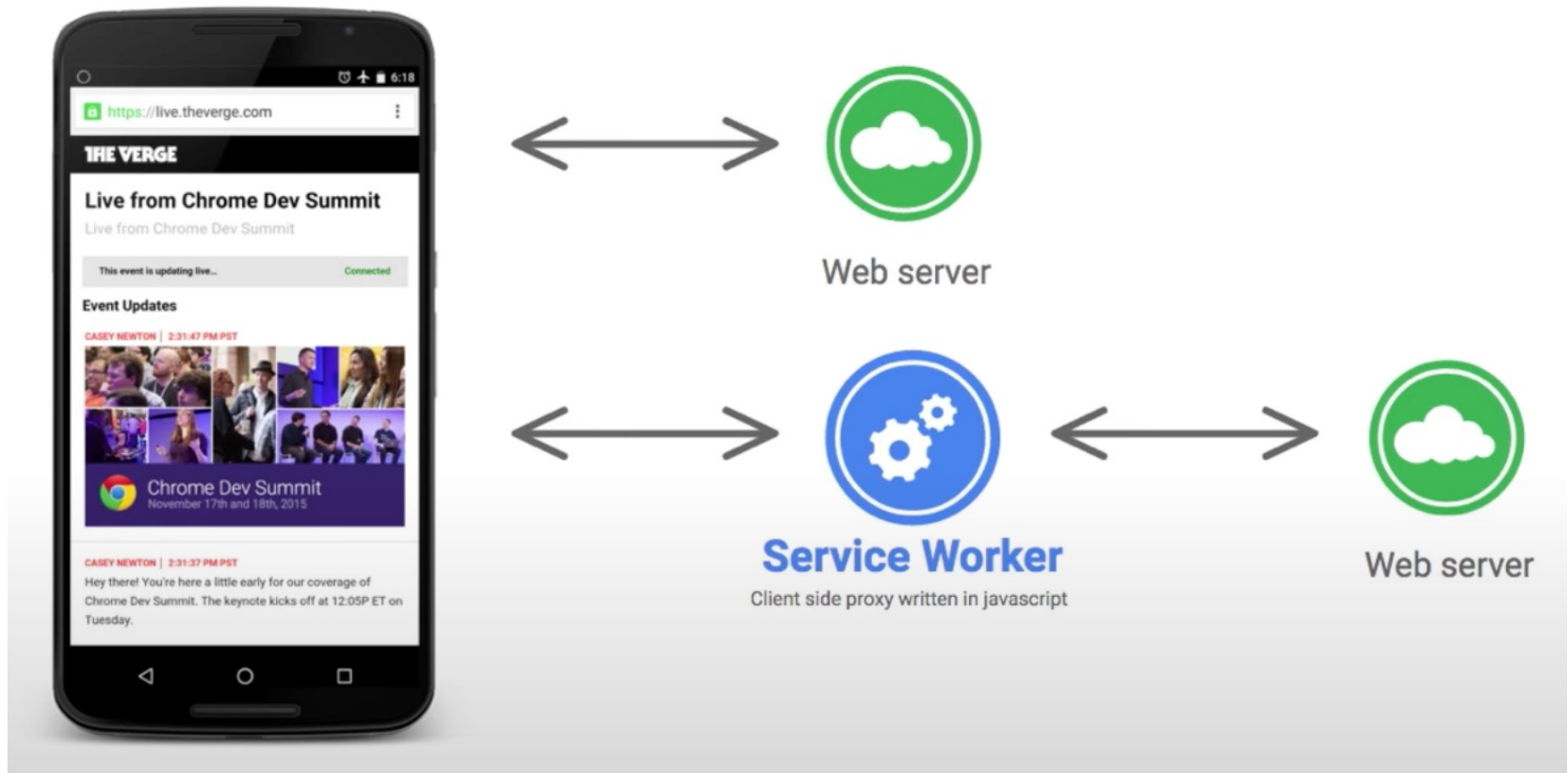
PWA – Manifest

- JSON súbor
- Obsahuje základné informácie o aplikácii:
 - Názov
 - Ikony
 - Štartovaciu URL pri spustení aplikácie
 - ...
- [Príklad manifestu](#)

PWA – Service worker

- Service Worker API - [súčasťou WEB API](#)
- Proxy, ktoré nám umožňuje riadiť a obsluhovať požiadavky medzi aplikáciou (klientom) a web serverom
- Beží v samostatnom vlákne, nezávislé od hlavnej aplikácie
- Vyžaduje HTTPS protokol
- Scope – určuje, pre ktoré “súbory” (cesty) bude zachytávať a obsluhovať požiadavky

PWA – Service worker /2



PWA – Service worker /3

- Cacheovanie obsahu (v režime offline), napríklad obrázky
- Notifikácie/Push notifikácie (aj keď je prehliadač vypnutý)
- [Introduction to Service Worker](#)
- [Using service worker](#)
- [Príklady service workerov](#)

Quasar PWA

- `quasar mode add pwa`
- `quasar dev -m pwa`
- `quasar build --mode pwa`

Preparation for PWA

PWA Build Commands

Sass - Syntactically Awesome Style Sheets

- CSS preprocesor
 - skriptovací jazyk, ktorý rozširuje možnosti písania CSS
 - napísaný kód sa “kompiluje” do CSS
- ďalšie známe preprocesory sú Less a Stylus
- pôvodná - indented **syntax** - `.sass`
- nová syntax (Sass 3) Sassy CSS - `.scss`

Sass - Syntactically Awesome Style Sheets

```
<style lang="scss" scoped>
.q-btn {
  &::v-deep {
    .q-btn__content {
      color: turquoise;
    }
  }
}
</style>
```