VUEJS 3+

SZKOLENIE:

- Plan szkolenia i cele
- Aktualne obowiązki
- Pytania, dyskusje, potrzeby
- Elastyczny program szkoleniowy

TRENER:

MATEUSZ KULESZA

- Senior Software Developer,
- Team Leader, Scrum Master
- Project Manager
- Consultant and Trainer

PROFESSIONAL EXPERIENCE

- HTML5, CSS3, SVG, EcmaScript 5 i 6
- jQuery, underscore, backbone.js
- canjs, requirejs, dojo …
- Grunt, Gulp, Webpack, Karma, Jasmine ...
- Angular.JS, Angular2, React, RxJS, VueJS

VUE.JS WPROWADZENIE

- Js Application Framework
- Przyjęcie przyrostowe
- Skoncentrowany tylko na warstwie widoku
- Ekosystem bibliotek i narzędzi

HELLO VUE

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

<div id="app">{{ message }}</div>

<script>
  Vue.createApp({
    data() {
      return {
        message: "Hello Vue!",
      };
    },
  }).mount("#app");
</script>
```

VUE ARCHITECTURE

Pod maską Vue kompiluje szablony do funkcji renderowania Virtual DOM. W połączeniu z systemem reaktywności Vue jest w stanie inteligentnie obliczyć minimalną liczbę składników do ponownego renderowania i zastosować minimalną ilość manipulacji DOM, gdy zmienia się stan aplikacji.

"RUSZTOWANIE" DLA PROJEKTU Z VITE

Create Vite

```
npm create vite@latest
```

lub

```
npm create vite@latest my-vue-app --template vue-ts
```

NUXT METAFRAMEWORK

- npx nuxi init myapp
- npx nuxi help
- npx nuxi add api|plugin|component|composable|middleware|layout|page

PLIKI .VUE TODO

```
<!-- App.vue -->
<template></template>

<style></style>

<script lang="ts"><script>
```

```
import App from "./App.vue";
Vue.createApp(App).mount("#app");
```

3 API DEFINIOWANIA KOMPONENTÓW

- Options API (vue 2.7+)
- Setup API (vue 3+)
- Setup Script API (vue 3.2+)

OPTIONS API

```
<script lang="ts">
import { defineComponent } from 'vue'

export default defineComponent({
    data: () => ({/* ... */}), // must be function!
    props:[/* ... */],
    methods: { /* ... */}
    watch:{ /*  ... */},
    computed:{ /* ... */}
})
<script/>
```

SETUP API FUNCTION IN OPTIONSAPI

SETUP SCRIPT

MOŻNA UŻYĆ TEŻ W VUE2.7

```
// If using Vue 2 with Composition API plugin configured:
import { ref, computed } from "@vue/composition-api";

export default {
  setup() {
    // ...
```

REAKTYWNE API

```
<script lang="ts" setup>
  import { ref } from "vue";

  let zmienna = 12; // Not Reactive!!!
  const count = ref(12);

  const zmien = () => {
    zmienna += 10; // Not Reactive!!!
    count.value += 10;
  };
</script>

<template>
  <div>
    <p>{{ zmienna }} {{ count }}</p>
    <button @click="zmien">Zmien</button>
  </div>
</template>
```

DYREKTYWY

Vue rozszerza html o dodatkowe "atrybuty", które dodają logikę specyficzną dla Vue do kodu szablonu:

v-text	v-if	v-for	v-on
v-html	v-else	v-cloak	v-once
v-show	v-else-if	v-pre	v-model

BINDOWANIE DO WŁASNOŚCI / ATTRIBUTÓW

```
<div id="app">
  <span v-bind:title="message">Hover</span>
  <!-- or shorter -->
  <span :title="message">Hover</span>
</div>
```

```
var app2 = Vue.createApp({
  data: {
    message: "You loaded this page on " + new Date().toLocaleString(),
  },
}).mount("#app");
```

BINDOWANIE STYLI I KLAS

```
<div
  class="static"
  v-bind:class="{ active: isActive, 'text-danger': hasError }"
></div>
<!-- styles -->
<div v-bind:style="{ color: activeColor, fontSize: fontSize + 'px' }"></div>
```

###

Także w formie tablicy:

```
<div v-bind:class="[activeClass, errorClass]"></div>
<div v-bind:class="[isActive ? activeClass : '', errorClass]"></div>
<div v-bind:class="[{ active: isActive }, errorClass]"></div>
<!-- styles -->
<div v-bind:style="[baseStyles, overridingStyles]"></div>
```

V-SHOW, V-IF, V-ELSE

```
<div id="app">
  <span v-show="type == 'A'">Only hidden</span>
  <span v-if="type == 'B'">Completly removed!</span>
  <span v-else-if="type == 'C'">Must be sibling</span>
  <span v-else>Must also be sibling</span>
</div>
```

DIREKTYWA V-FOR

```
<ul id="app">
  <li v-for="todo in todos">{{ todo.text }}</li>
</ul>
```

```
var app = Vue.createApp({
  data: {
    todos: [
      { text: "Learn JavaScript" },
      { text: "Learn Vue" },
      { text: "Build something awesome" },
    ],
  },
});
```

V-FOR I OBIEKTY

```
<div v-for="(item, index) in items"></div>
<div v-for="(val, key) in object"></div>
<div v-for="(val, key, index) in object"></div>
```

V-FOR I ZAKRESY

v-for może również przyjmować liczbę całkowitą. W takim przypadku wielokrotnie powtarza szablon.

```
<div>
  <span v-for="n in 10">{{ n }} </span>
</div>
```

“KEY” ATTRIBUTE

```
<div v-for="item in items" :key="item.id">{{ item.text }}</div>
```

KIEDY UŻYWAĆ TAGU SZABLONU

Czasami używanie dyrektyw generuje nieprawidłowy html. Znacznik szablonu pomaga, będąc elementem "neutralnym"

```
<ul>
  <template v-for="item in items">
    <li>{{ item.msg }}</li>
    <li class="divider"></li>
  </template>
</ul>
```

```
<ul v-if="todos.length">
  <li v-for="todo in todos">{{ todo }}</li>
</ul>
<p v-else>No todos left!</p>
```

OBSŁUGA ZDARZEŃ UŻYTKOWNIKA

```
<div id="app">
  <p>{{ message }}</p>
  <button v-on:click="message = 'direct change!'">Change Message</button>
  <!-- or call method -->
  <button v-on:click="changeMessage">Change Message</button>
  <!-- or shorter -->
  <span @click="changeMessage">Change Message</button>
</div>
```

```
var app = Vue.createApp({
  el: "#app",
  data: {
    message: "Hello Vue.js!",
  },
  methods: {
    changeMessage: function () {
      this.message = "Changed Message!";
    },
  },
});
```

DOSTĘP DO INFORMACJI O ZDARZENIU

Oryginalne zdarzenie jest dostępne jako zmienna $event lub jako argument metody

```
<div id="app">
  <p>{{ message }}</p>
  <input @input="message = $event.target.value" />
  <input @input="updateMessage" />
</div>
```

```
var app = Vue.createApp({
  el: "#app",
  data: {
    message: "Hello Vue.js!",
  },
  methods: {
    updateMessage: function (event) {
      this.message = event.target.value;
    },
  },
});
```

MODYFIKATORY

- `stop` - wywołanie event.stopPropagation ().
- `prevent` - wywołaj event.preventDefault ().
- `capture` - dodaj detektor zdarzeń w trybie przechwytywania.
- `self` - tylko wywoływaj procedurę obsługi jeśli zdarzenie zostało wywołane z tego elementu.
- `{keyCode | keyAlias}`- tylko wyzwalaj dla niektórych klawiszy.
- `once` - obsłuży wyzwalacz co najwyżej raz.
- `left` - uruchamia tylko program obsługi zdarzeń myszy lewego przycisku.
- `right` - uruchamia tylko program obsługi zdarzeń dla myszy z prawym przyciskiem myszy.
- `middle` - uruchamia tylko program obsługi zdarzeń środkowego przycisku myszy.
- `passive` - dołącza zdarzenie DOM z {passive: true}.

```
<form v-on:submit.prevent></form>
```

KEYBOARD MODIFIERS

- `{keyCode | keyAlias}` - tylko wyzwalaj dla niektórych klawiszy,np:
- `.enter`
- `.tab`
- `.delete` (“Delete” i “Backspace”)
- `.esc`
- `.space`
- `.up` `.down`
- `.left` `.right`

System keys : `.ctrl` `.alt` `.shift` `.meta` or any from `$event.key`

```
<input @keyup.page-down="onPageDownFunc" />
```

OBSERWATORY (WATCHERS)

```
<script setup lang="ts">
  const data = reacrive({
    firstName: "Foo",
    lastName: "Bar",
    fullName: "Foo Bar",
  });
  watch([firstName, lastName], ([firstName, lastName]) => {
    data.fullname.value = firstName + " " + lastName;
  });
</script>
```

ASYNC WATCHER + CLEAN UP

```
watch(params, (value, old, onCleanup) => {
  const handle = setTimeout(() => {
    // ...
  }, 1000);

  // Cancel operation if params changed or Unmounted:
  onCleanup(() => clearTimeout(handle));
});
```

WYLICZONE WŁAŚCIWOŚCI TODO

```
<script setup lang="ts">
  const nowFn = () => Date.now();
  const nowComp = computed(() => Date.now());
</script>
```

{{ nowFn() }} - kod jest wykonywany przy każdym renderowaniu
{{ nowComp }} - wynik jest buforowany na podstawie zależności

AUTOMATYCZNE ŚLEDZENIE

```
const count = ref(1);
const plusOne = computed(() => count.value + 1);

console.log(plusOne.value); // 2
```

```
<button @click="count += 1">Up</button> {{ plusOne }} // 3
```

DEBUGOWANIE COMPUTED / WATCHER

```
const count = ref(1);
const plusOne = computed(() => count.value + 1, {
  onTrack: console.log,
  onTrigger: console.log,
});
```

COMPUTED SETTER

Właściwości obliczane są domyślnie tylko typu getter, ale możesz także ustawić setter, gdy jest to potrzebne:

```
<script setup lang="ts">
  const fullName = computed({
    // getter
    get: function () {
      return this.firstName + " " + this.lastName;
    },
    // setter
    set: function (newValue) {
      var names = newValue.split(" ");
      this.firstName = names[0];
      this.lastName = names[names.length - 1];
    },
  });
  fullName.value = "Split This";
</script>
```

FORMS

```
<input v-model="message" placeholder="edit me" />
<p>Message is: {{ message }}</p>
```

```
<span>Multiline message is:</span>
<p style="white-space: pre-line;">{{ message }}</p>
<br />
<textarea v-model="message" placeholder="add multiple lines"></textarea>
```

2-WAY BINDING WITH V-MODEL

```
<div id="app">
  <p>{{ message }}</p>
  <input :value="message" @input="message" />
  <!-- or shorter -->
  <input v-model="message" />
</div>
```

RADIO, CHECKBOX AND SELECT

```
<input type="radio" v-model="picked" value="a" />
<input type="radio" v-model="picked" value="b" />
{{picked}}

<input type="checkbox" v-model="toggle" true-value="yes" false-value="no" />
{{toggle}}

<input type="checkbox" v-model="namesList" value="Jack" />
<input type="checkbox" v-model="namesList" value="Jim" />
<input type="checkbox" v-model="namesList" value="Johnny" />
{{namesList}}

<select v-model="selected">
  <option value="abc">ABC</option>
</select>
{{selected}}
```

V-MODEL MODIFIERS

`.lazy` - używaj zdarzenia change

```
<!-- synced after "change" instead of "input" -->
<input v-model.lazy="msg" />
```

`.number` - konwertuje na typ `number`

```
<input v-model.number="age" type="number" />
```

`.trim` - wycina białe znaki

```
<input v-model.trim="msg" />
```

COMPONENTS

```
// Define a new component called todo-item
app.component("todo-item", {
  template: "<li>This is a todo</li>",
});
```

Teraz możesz skomponować go w szablonie dowolnego innego komponentu:

```
<ol>
  <!-- Create an instance of the todo-item component -->
  <todo-item></todo-item>
</ol>

<table>
  <tr is="blog-post-row"></tr>
</table>
```

LOKALNA REJESTRACJA

```
import ComponentA from '...'
import ComponentB from '...'

Vue.createApp({
  el: '#app'
  components: {
    'component-a': ComponentA,
    'component-b': ComponentB
  }
})
```

SETUP SCRIPT API

Wystarczy zaimportować komponenty aby były dostępne w szablonie.

```
<script setup lang="ts">
  import ComponentA from "...";
  import ComponentB from "...";
</script>

<template>
  <ComponentA />
  <ComponentB />
</template>
```

PASING DATA INTO COMPONENT WITH PROPS

```
Vue.component("blog-post", {
  // camelCase in JavaScript
  props: ["postTitle"],
  template: "<h3>{{ postTitle }}</h3>",
});
```

```
<!-- static binding -->
<blog-post post-title="hello!"></blog-post>

<!-- dynamic binding -->
<blog-post v-bind:post-title="getTitle()"></blog-post>

<!-- static expression binding -->
<blog-post
  v-bind:comments="{ id: 1, title: 'My Journey with Vue' }"
></blog-post>

<!-- dynamic binding multiple props -->
<blog-post v-bind="post"></blog-post>
```

PROPS - SETUP SCRIPT

Wystarczy zaimportować komponenty aby były dostępne w szablonie.

```
<script setup lang="ts">
  const props = defineProps<{
    items: Item[];
    selectedId?: Item["id"];
  }>();
</script>

<template> {{props.selectedId}} </template>
```

PROPS AND REACTIVITY

```
const props = defineProps<{title:string}>()

// turn `props` into an object of refs, then destructure
const { title } = toRefs(props)

// `title` is a ref that tracks `props.title`
console.log(title.value)

// OR, turn a single property on `props` into a ref
const title = toRef(props, 'title')
```

COMPONENT COMMUNICATION - CUSTOM EVENTS

```
<script setup lang="ts">
  const emit = defineEmits<{ (e: "enlarge-text", val: number): void }>();
</script>
<template>
  <button @click="emit('enlarge-text', 0.1)">Enlarge text</button>
</template>
```

COMPONENT LIFECYCLE

ATTRIBUTE INHERITANCE

```
Vue.component("bootstrap-date-input", {
  template: `<input type="date" class="form-control">`,
});
```

Możesz dodać dodatkowe atrybuty i zostaną one scalone z elementem głównym komponentu

```
<bootstrap-date-input
  data-date-picker="activated"
  class="date-picker-theme-dark"
></bootstrap-date-input>
```

wynikiem jest:

```
<input
  type="date"
  class="form-control date-picker-theme-dark"
  data-date-picker="activated"
/>
```

COMPONENT CONTENT PROJECTION WITH SLOTS

```
<a v-bind:href="url" class="nav-link">
  <slot></slot>
</a>
```

```
<navigation-link url="/profile">
  <!-- Add a Font Awesome icon -->
  <span class="fa fa-user"></span>
  Your Profile
</navigation-link>
```

NAMED SLOTS

```
<div class="container">
  <header>
    <slot name="header"></slot>
  </header>
  <main>
    <slot></slot>
  </main>
</div>
```

```
<base-layout>
  <h1 slot="header">Here might be a page title</h1>

  <main>
    <p>A paragraph for the main content.</p>
    <p>And another one.</p>
  </main>
</base-layout>
```

SCOPED SLOTS

```
<slot :title="valA" :propB="valB"></slot>
```

```
<Component v-slot="{title, propB}">
  <h1>{{title}}</h1>
  <div>{{propB}}</div>
</Component>
```

SLOTS AND SETUP

```
<script setup>
  import { useSlots, useAttrs } from "vue";

  const slots = useSlots();
  const attrs = useAttrs();
</script>
```

HEADLESS AND COMPOUND COMPONENTS

Sloty pozwalają oddzielić komponenty logiczne od prezentacji:

https://headlessui.com/vue/

DIRECTIVES

```
<div id="hook-arguments-example" v-demo:foo.a.b="message"></div>
```

```
Vue.directive("demo", {
  // bound to element
  bind: function (el, binding, vnode) {
    console.log(
      binding.name,
      binding.value,
      binding.expression,
      binding.arg,
      binding.modifiers,
      vnode
    );
  },
  inserted() {}, // gdy wstawiony do rodzica
  update() {}, // gdy rodzic jest aktualizowany
  componentUpdated() {}, // gdy dzieci są aktualizowane
  unbind() {}, // gdy element jest niszczony
});
```

PLUGINS

```
MyPlugin.install = function (Vue, options) {
  // 1. add global method or property
  Vue.myGlobalMethod = function () {
    // something logic ...
  };
};

// Calls MyPlugin.install
Vue.use(MyPlugin, { someOption: true });
```

https://github.com/vuejs/awesome-vue#components--libraries

ROUTING

```
const Foo = { template: "<div>foo</div>" };
const Bar = { template: "<div>bar</div>" };

const routes = [
  { path: "/foo", component: Foo },
  { path: "/bar", component: Bar },
];

const router = new Router({
  routes: routes,
});

const app = createApp(App);
app.use(router); /// !!
```

```
<router-link to="/some-path">Home</router-ling>
<router-view></router-view>
```

ROUTER PARAMS

```
const router = new VueRouter({
  routes: [
    // segmenty dynamiczne zaczynają się dwukropkiem
    { path: "/user/:id", component: User },
  ],
});
```

```
<template> <div>{{$route.params.id}}</div> </template>
```

ROUTER COMPOSABLE API

```
const router = useRouter();
const route = useRoute();

// fetch the user information when params change
watch(() => route.params.id /* ... */);

function pushWithQuery(query) {
  router.push({
    name: "search",
    query: {
      ...route.query,
    },
  });
}
```

ROUTER GUARDS

```
// same as beforeRouteLeave option with no access to `this`
onBeforeRouteLeave((to, from) => {
  const answer = window.confirm(
    "Do you really want to leave? you have unsaved changes!"
  );
  // cancel the navigation and stay on the same page
  if (!answer) return false;
});

const userData = ref();

// same as beforeRouteUpdate option with no access to `this`
onBeforeRouteUpdate(async (to, from) => {
  // only fetch the user if the id changed as maybe only the query or the hash changed
  if (to.params.id !== from.params.id) {
    userData.value = await fetchUser(to.params.id);
  }
});
```

ZAGNIEŻDZONE ROUTES

```
const User = {
  template:
    <div class="user">
      <h2>User {{ $route.params.id }}</h2>
      <router-view></router-view>
    </div>

}
const router = new VueRouter({
  routes: [
    { path: '/user/:id', component: User,
      children: [
        {
          // UserProfile wyrenderowany wewnątrz User
          // w <router-view>
          // gdy adres /user/:id/profile jest aktywny
          path: 'profile',
          component: UserProfile
        },
    }
```

WSPÓŁDZIELENIE DANYCH POMIĘDZY KOMPONENTAMI

```
// Defined outside of composable
const someGlobalStore = reactive({
  theme: {},
  /*  config: {},   session: {}, ... */
});

export function useTheme() {
  const theme = toRef(someGlobalStore, "theme");
  const switchToDark = () => {
    /* ... */
  };

  return {
    theme,
    switchToDark,
  };
}
```

Be carefull about sharing private user data in SSR!

MAGAZYNY STANU GLOBALNEGO - PINIA

```
import { createApp } from "vue";
import { createPinia } from "pinia";
import App from "./App.vue";

const pinia = createPinia();
const app = createApp(App);

app.use(pinia);
app.mount("#app");
```

PINIA - STORE

```
export const useCounterStore = defineStore("counter", {
  state: () => ({ count: 0, name: "Eduardo" }),
  getters: {
    doubleCount: (state) => state.count * 2,
  },
  actions: {
    increment() {
      this.count++;
    },
  },
});
```

PINIA - SETUP API STORE

```
export const useCounterStore = defineStore("counter", () => {
  const count = ref(0);
  const name = ref("Eduardo");
  const doubleCount = computed(() => count.value * 2);
  function increment() {
    count.value++;
  }

  return { count, name, doubleCount, increment };
});
```

PINIA - COMPOSABLES API

```
<script setup lang="ts">
  const store = useCounterStore();
  // `name` and `doubleCount` are reactive refs

  // This will also create refs for properties added by plugins
  // but skip any action or non reactive (non ref/reactive) property
  const { name, doubleCount } = storeToRefs(store);

  // the increment action can just be extracted
  const { increment } = store;
</script>
```

PINIA - SUBSCRIBE TO STORE

```
cartStore.$subscribe((mutation, state) => {
  // import { MutationType } from 'pinia'
  mutation.type; // 'direct' | 'patch object' | 'patch function'
  // same as cartStore.$id
  mutation.storeId; // 'cart'
  // only available with mutation.type === 'patch object'
  mutation.payload; // patch object passed to cartStore.$patch()

  // persist the whole state to the local storage whenever it changes
  localStorage.setItem("cart", JSON.stringify(state));
});
```

NUXT

NUXT

- Routing oparty na plikach
- Code-splitting
- Automatyczne importowanie
- Narzędzia do pobierania danych z obsługą SSR/CSR
- Obsługa TypeScript bez konfiguracji
- Skonfigurowane narzędzia do kompilacji i HMR

SERVER-SIDE RENDERING

- Szybszy czas początkowego ładowania strony wolniejszych urządzeniach.
- Ulepszone SEO (Web Vitals)
- Lepsza wydajność na urządzeniach o niskim poborze mocy
- Lepsza dostępność
- Łatwiejsze cacheowanie strony

INIT PROJECT

```
npx nuxi init <project-name>
```

STRUCTURE

- app.vue
- components/YourComponent.vue
- pages/some/example/path/YourPage.vue `<NuxtPage/>`
- layouts/default.vue `<NuxtLayout/>`

FILESYSTEM ROUTING

Pliki mapowane są na ścieżki adresu URL:

```
pages/
--| index.vue  // domain.com/
--| about.vue  // domain.com/about
--| posts/
----| index.vue // domain.com/posts
----| [id].vue // domain.com/posts/123
...
```

HYBRYDOWA NAWIGACJA I PREFETCH

```
<ul>
  <li><NuxtLink to="/about">About</NuxtLink></li>
  <li><NuxtLink to="/posts/1">Post 1</NuxtLink></li>
  <li><NuxtLink :to="'/posts/' + post.id">Post {{post.id}}</NuxtLink></li>
</ul>
```

Nuxt Link dodaje prefetch komponentów i przechodzi bez przeładowania dokumentu

PARAMETRY ŚCIEŻKI

```
<script setup>
  const route = useRoute();
  // Owierając /posts/1 parametr route.params.id będzie miał wartość 1
  console.log(route.params.id);
</script>
```

TRYBY RENDEROWANIA

- SPA / CSR - Single page app - Client side rendering
- SSR - Server side rendering
- SSG - Static generated content (nuxi generate)

- ISR - Incremental Static Regeneration

- Universal - SSR + CSR

- Hybrid - SSG + SSR + CSR (routeRule)
- Edge - SSG/ISR + SSR on the Edge (CDN)

TRYBY ŁADOWANIA DANYCH

- CSR - Client side fetch
- SSR - Server side
- Hydration - SSR + CSR

POBIERANIE DANYCH - KLIENT

```
<script setup>
  const { data: count } = await useFetch('/api/count')
</script>
<template> Page visits: {{ count }} </template>
```

SERVER API

```
server/routes/callback.ts:
server/api/hello.ts:
server/api/posts/[...].ts:
```

```ts
export default defineEventHandler((event) => {
  const query = getQuery(event); // event.context.params

  return {
    message: `Hello, ${query.name}!`,
  };
});
```

POST REQUESTS

server/api/hello.get.ts: server/api/hello.post.ts:

```
export default defineEventHandler(async (event) => {
  const body = await readBody(event);
  return { body };
});
```

MIDDLEWARE

```
// server/middleware/log.global.ts
export default defineEventHandler((event) => {
  console.log("New request: " + event.node.req.url);
});

// server/middleware/auth.ts
export default defineEventHandler((event) => {
  event.context.auth = { user: 123 };
});
```

DZIĘKUJĘ