



Framework Vue.js

Bartosz Rogowski
Kinga Pyrek





Czym jest Vue.js?

Vue.js to framework JavaScript, pozwalający tworzyć aplikacje webowe z dynamicznym interfejsem użytkownika. Vue.js posiada szereg dodatków i funkcji, które są ogromnym wsparciem dla developerów. Rozbudowuje standardowy HTML, CSS i JavaScript i wykorzystuje model programowania oparty o komponenty.

Jest świetną alternatywą dla skomplikowanych i rozbudowanych frameworków takich jak Angular i React. W przeciwieństwie do nich Vue.js nie jest wspierany i rozwijany przez duże firmy takie jak Google czy Facebook, dlatego nie jest często wybierany przez duże korporacje. Jego twórcą jest były pracownik Google - Evan You.



Single-File Component

Single-File Component to styl pisania aplikacji w bibliotekach JavaScript, polegający na tym, że każdy pojedynczy plik reprezentuje jeden komponent. Ma on rozszerzenie `.vue` i jest specjalnym formatem, który pozwala na rozszerzenie klasycznego HTML, CSS i JS - tagi `<template>`, `<script>`, `<style>` zawierają i łączą widok, logikę i styl komponentu w tym samym pliku.

```
<script>
export default {
  data() {
    return {
      greeting: 'Hello World!'
    }
  }
}
</script>

<template>
  <p class="greeting">{{ greeting }}</p>
</template>

<style>
.greeting {
  color: red;
  font-weight: bold;
}
</style>
```



Options API vs Composition API

Komponenty Vue.js mogą być tworzone w dwóch różnych stylach API: Options API and Composition API.

Pierwszy z nich polega na definiowaniu logiki komponentu z użyciem obiektu opcji. Własności definiowane przez opcje są widoczne w tych funkcjach wewnętrznych, które wskazują na instancję komponentu - `this.nazwa_zmiennej`

Drugi natomiast opiera się na idei importowania funkcji z API. W stylu SFC, Composition API zazwyczaj jej używane z tagiem `<script setup>`. To w nim pisany jest reaktywny kod.

Options API

```
<script>
export default {
  // Properties returned from data() becomes reactive state
  // and will be exposed on `this`.
  data() {
    return {
      count: 0
    }
  },

  // Methods are functions that mutate state and trigger updates.
  // They can be bound as event listeners in templates.
  methods: {
    increment() {
      this.count++
    }
  },

  // Lifecycle hooks are called at different stages
  // of a component's lifecycle.
  // This function will be called when the component is mounted.
  mounted() {
    console.log(`The initial count is ${this.count}.`)
  }
}
</script>

<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

Composition API

```
<script setup>
import { ref, onMounted } from 'vue'

// reactive state
const count = ref(0)

// functions that mutate state and trigger updates
function increment() {
  count.value++
}

// lifecycle hooks
onMounted(() => {
  console.log(`The initial count is ${count.value}.`)
})
</script>

<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

**Ale zacznijmy od podstaw...
czyli jak stworzyć aplikację?**

Vue cli

Vue cli to globalnie instalowany pakiet npm, który udostępnia polecenie vue w terminalu. Udostępnia możliwość szybkiego tworzenia szkieletu nowego projektu za pomocą vue create. Możliwe jest także zarządzanie projektami za pomocą graficznego interfejsu użytkownika za pośrednictwem vue ui.

- `npm install -g @vue/cli`
- `vue create app_name`
- `npm run serve`



Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project, check out the [vue-cli documentation](#).

Installed CLI Plugins

[babel](#) [eslint](#)

Essential Links

[Core Docs](#) [Forum](#) [Community Chat](#) [Twitter](#) [News](#)

Ecosystem

[vue-router](#) [vuex](#) [vue-devtools](#) [vue-loader](#) [awesome-vue](#)

Inna opcja, czyli Vite


```
C:\Users\Kinga\Desktop\zti-test>npm init vue@latest
Need to install the following packages:
  create-vue@latest
Ok to proceed? (y) y
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'create-vue@3.1.10',
npm WARN EBADENGINE   required: { node: '^14.13.1 || >=16.0.0' },
npm WARN EBADENGINE   current: { node: 'v15.8.0', npm: '7.5.1' }
npm WARN EBADENGINE }

Vue.js - The Progressive JavaScript Framework

√ Project name: ... example1
√ Add TypeScript? ... No / Yes
√ Add JSX Support? ... No / Yes
√ Add Vue Router for Single Page Application development? ... No / Yes
√ Add Pinia for state management? ... No / Yes
√ Add Vitest for Unit Testing? ... No / Yes
√ Add Cypress for both Unit and End-to-End testing? ... No / Yes
√ Add ESLint for code quality? ... No / Yes

Scaffolding project in C:\Users\Kinga\Desktop\zti-test\example1...

Done. Now run:

  cd example1
  npm install
  npm run dev
```

```
C:\Users\Kinga\Desktop\zti-test>cd example1
C:\Users\Kinga\Desktop\zti-test\example1>npm install

added 52 packages, and audited 53 packages in 36s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Kinga\Desktop\zti-test\example1>npm run dev

> example1@0.0.0 dev
> vite

vite v2.9.9 dev server running at:

> Local: http://localhost:3000/
> Network: use `--host` to expose

ready in 599ms.
```



You did it!

You've successfully created a project with
[Vite](#) + [Vue 3](#).



Documentation

Vue's [official documentation](#) provides you with all information you need to get started.



Tooling

This project is served and bundled with [Vite](#). The recommended IDE setup is [VSCode](#) + [Volar](#). If you need to test your components and web pages, check out [Cypress](#) and [Cypress Component Testing](#). More instructions are available in [README.md](#).



Ecosystem

Get official tools and libraries for your project: [Pinia](#), [Vue Router](#), [Vue Test Utils](#), and [Vue Dev Tools](#). If you need more resources, we suggest paying [Awesome Vue](#) a visit.



Community

Got stuck? Ask your question on [Vue Land](#), our official Discord server, or [StackOverflow](#). You should also subscribe to [our mailing list](#) and follow the official [@vuejs](#) twitter account for latest news in the Vue world.



Support Vue

As an independent project, Vue relies on community backing for its sustainability. You can help us by [becoming a sponsor](#).



Składnia Vue.js

Vue.js używa składni bazującej na HTML-u, która pozwala na deklaratywne powiązanie (ang. *binding*) renderowanego drzewa DOM z komponentem. Pod spodem Vue.js kompiluje szablony do zoptymalizowanego kodu JavaScript. Vue.js jest w stanie ustalić minimalną liczbę komponentów do ponownego zrenderowania i zaaplikowania minimalnej liczby manipulacji DOM kiedy stan aplikacji ulega zmianie. Wyświetlanie zawartości zmiennych we Vue odbywa się dzięki tzw. wąsom. Można wykorzystać również operatory czy funkcje korzystając ze składni JavaScriptu.

- wstawianie (interpolacja) tekstu

```
<span>Message: {{ msg }}</span>
```

example2

- wyrażenia JavaScript

```
{{ number + 1 }}  
  
{{ ok ? 'YES' : 'NO' }}  
  
{{ message.split('').reverse().join('') }}
```



Dyrektywy

Vue.js posiada dyrektywy czyli specjalne atrybuty z przedrostkiem `v-`. Wartością takiej dyrektywy jest wyrażenie JavaScript. Dyrektywy pozwalają na reaktywne aktualizowanie drzewa DOM za każdym razem gdy wartość przekazanego wyrażenia się zmieni.

Dyrektywa `v-html` sprawia, że renderowany jest kod HTML, a nie zachodzi zwykłe podstawienie tekstu.

```
<p>Using text interpolation: {{ rawHtml }}</p>  
<p>Using v-html directive: <span v-html="rawHtml"></span></p>
```

Using text interpolation: This should be red.

Using v-html directive: This should be red.

Dyrektywy - c.d.

Podstawienie zmiennej za pomocą nawiasów klamrowych do kodu HTML nie zadziała. W tym celu należy użyć dyrektywy `v-bind`, która wiąże zmienną z atrybutem..

```
<div v-bind:id="dynamicId"></div>
```

Z uwagi na częste użycie, nazwę tej dyrektywy można pominąć:

```
<div :id="dynamicId"></div>
```

```
1  <script>
2  export default {
3    data() {
4      return {
5        titleClass: 'title'
6      }
7    }
8  }
9  </script>
10
11 <template>
12   <h1 :class="titleClass">Make me red</h1>
13 </template>
14
15 <style>
16   .title {
17     color: red;
18   }
19 </style>
```

Make me red



Obsługa zdarzeń

Dyrektywa `v-on` odpowiada za nasłuchiwanie zdarzeń i uruchamianie kodu JavaScript w momencie, gdy zdarzenie to występuje (np. kliknięcie myszy na przycisk). Z uwagi na częste użycie tej dyrektywy, wprowadzono prostszy zapis za pomocą symbolu `@`. Można podać także nazwę funkcji, która ma być wyzwolona.

```
<button v-on:click="count++">Add 1</button>
```

```
<button @click="count++">Add 1</button>
```

<https://vuejs.org/guide/essentials/event-handling.html>



Dyrektywy - c.d.

Dostępne są także dyrektywy sterujące:

- `v-if`
- `v-else`
- `v-else-if`
- `v-for` (przykład w vs codzie)

```
1  <script>
2  export default {
3    data() {
4      return {
5        seen: true
6      }
7    },
8    methods: {
9      toggle() {
10       this.seen = !this.seen
11     }
12   }
13 }
14 </script>
15
16 <template>
17   <button @click="toggle">toggle</button>
18   <h1 v-if="seen">If you don't wanna see me</h1>
19   <h1 v-else>Now you don't</h1>
20 </template>
```

Formularze

Formularze stanowią częstą formę komunikacji aplikacji z użytkownikiem. Do synchronizowania stanu elementów wejściowych formularza z odpowiadającym im stanem w JavaScript służy dyrektywa `v-model`. Ręczne tworzenie powiązań wartości i zmiana detektorów zdarzeń jest możliwa, jednak bywa kłopotliwa.

```
1 <script>
2 export default {
3   data() {
4     return {
5       text: ''
6     }
7   },
8   methods: {
9     onInput(e) {
10       this.text = e.target.value
11     }
12   }
13 }
14 </script>
15
16 <template>
17   <input :value="text" @input="onInput" placeholder="Type here">
18   <p>{{ text }}</p>
19 </template>
```

```
1 ∨ <script>
2 ∨ export default {
3 ∨   data() {
4 ∨     return {
5       text: ''
6     }
7   }
8 }
9 </script>
10
11 ∨ <template>
12   <input v-model="text" placeholder="Type here">
13   <p>{{ text }}</p>
14 </template>
```



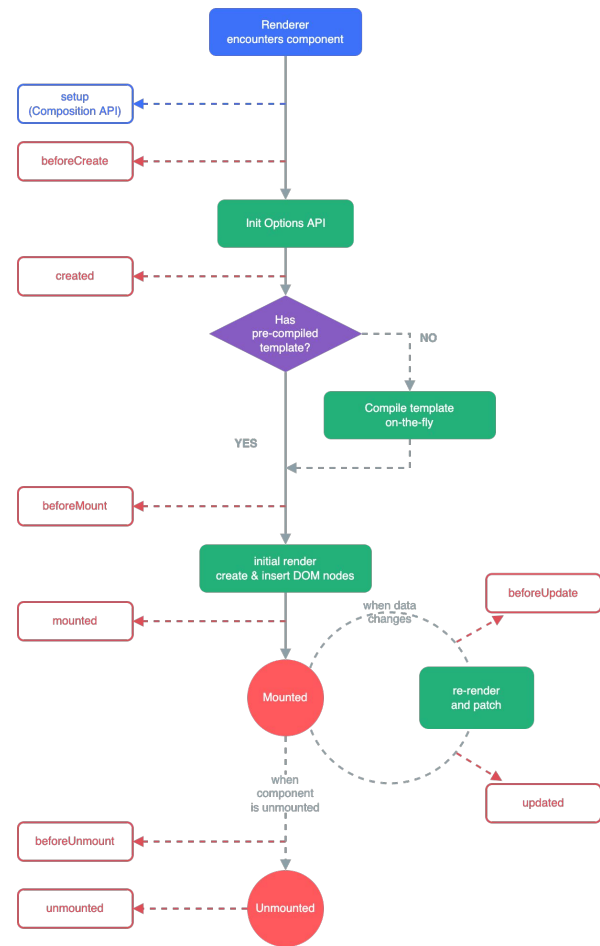

Computed properties

Są ideowo zbliżone do mechanizmu cache'owania. Polega to na tym, że obliczenia są wstępnie przeprowadzane i zapisywane tak, by później móc się szybko odwołać do ich wyniku. Jest to przydatne w sytuacjach, gdy mamy do czynienia z dużą liczbą danych (np. obiekty w liście) lub gdy często wywoływana jest dana funkcja, której wynik jest stały.

Cykl życia

Każda instancja komponentu wykonuje serię kroków inicjalizujących kiedy jest tworzona. Po drodze uruchamia również funkcje zwane hookami cyklu życia, dając użytkownikom możliwość dodawania własnego kodu na określonych etapach.

```
1 <script>
2 export default {
3   mounted() {
4     this.$refs.p.textContent = 'mounted!'
5   }
6 }
7 </script>
8
9 <template>
10 <p ref="p">hello</p>
11 </template>
```





Watchers

Mechanizm ten pozwala na obserwowanie danej zmiennej i podejmowanie danej akcji, gdy jej stan się zmienia (na przykład mutacja DOM lub zmiana innego elementu stanu na podstawie wyniku operacji asynchronicznej).

```
export default {  
  data() {  
    return {  
      count: 0  
    }  
  },  
  watch: {  
    count(newCount) {  
      // yes, console.log() is a side effect  
      console.log(`new count is: ${newCount}`)  
    }  
  }  
}
```

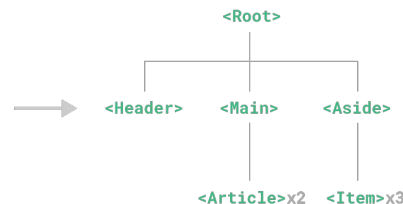
example5

Łączenie wielu komponentów

Komponenty pozwalają nam podzielić interfejs użytkownika na niezależne i wielokrotnego użytku części oraz myśleć o każdym kawałku osobno. Aplikacja często jest zorganizowana w drzewo zagnieżdżonych komponentów.

```
App.vue  ChildComp.vue x +
1 <script>
2 import ChildComp from './ChildComp.vue'
3
4 export default {
5   components: {
6     ChildComp
7   }
8 }
9 </script>
10
11 <template>
12   <ChildComp />
13 </template>
```

```
App.vue  ChildComp.vue x +
1 <template>
2   <h2>A Child Component!</h2>
3 </template>
```

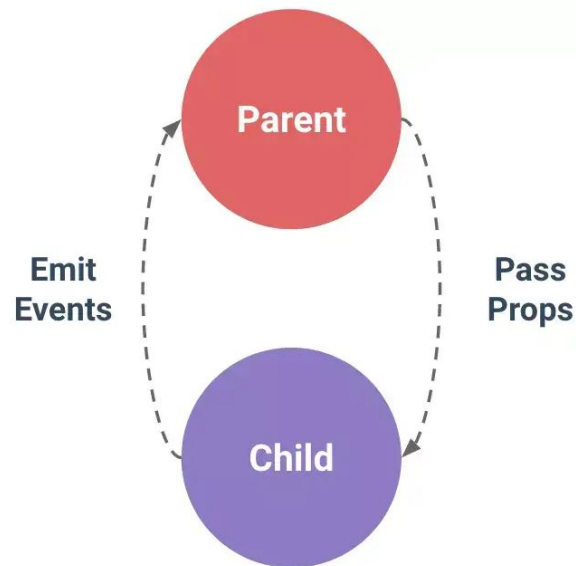


Przykładowe rozdzielenie na komponenty

Synchronizowanie zmiennych między komponentami

Jeśli tworzymy blog, prawdopodobnie będziemy potrzebować komponentu reprezentującego post na blogu. Chcemy, aby wszystkie posty na blogu miały ten sam układ wizualny, ale różniły się treścią. Taki komponent nie będzie przydatny, chyba że możesz przekazać do niego dane, takie jak tytuł i treść konkretnego posta, który chcemy wyświetlić. Właśnie tam wkraczają props, dzięki nim komponent *child* może przyjąć parametry z komponentu *parent*.

Props to customowe atrybuty, które można zarejestrować w komponencie. Aby przekazać tytuł do naszego komponentu postów na blogu, musimy zadeklarować go na liście właściwości akceptowanych przez ten komponent, korzystając z opcji props:



Props - przykład

App.vue



```
1 <script>
2 import BlogPost from './BlogPost.vue'
3
4 export default {
5   components: {
6     BlogPost
7   },
8   data() {
9     return {
10       posts: [
11         { id: 1, title: 'My journey with Vue' },
12         { id: 2, title: 'Blogging with Vue' },
13         { id: 3, title: 'Why Vue is so fun' }
14       ]
15     }
16   }
17 }
18 </script>
19
20 <template>
21   <BlogPost
22     v-for="post in posts"
23     :key="post.id"
24     :title="post.title"
25   ></BlogPost>
26 </template>
```

BlogPost.vue



```
1 <script>
2 export default {
3   props: ['title']
4 }
5 </script>
6
7 <template>
8   <h4>{{ title }}</h4>
9 </template>
```

output



My journey with Vue

Blogging with Vue

Why Vue is so fun

Emits - przykład

Oprócz otrzymywania props, podrzędny komponent *child* może także wysyłać (emitować) zdarzenia do jego elementu nadrzędnego *parent*

```
App.vue  ChildComp.vue x +
1 <script>
2 import ChildComp from './ChildComp.vue'
3
4 export default {
5   components: {
6     ChildComp
7   },
8   data() {
9     return {
10       childMsg: 'No child msg yet'
11     }
12   }
13 }
14 </script>
15
16 <template>
17   <ChildComp @response="(msg) => childMsg = msg" />
18   <p>{{ childMsg }}</p>
19 </template>
```

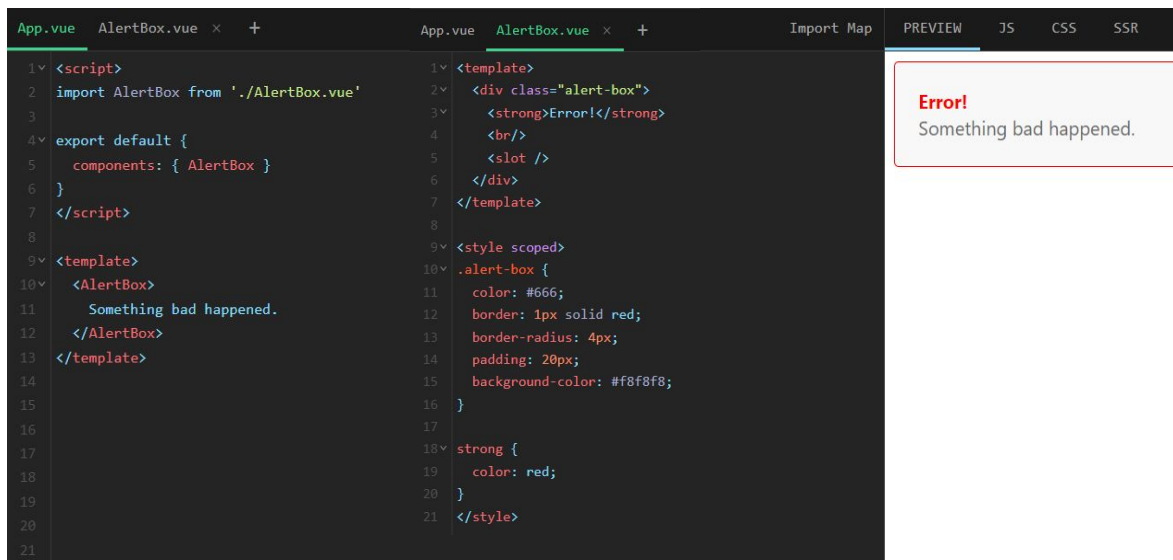
```
App.vue  ChildComp.vue x +
1 <script>
2 export default {
3   emits: ['response'],
4   created() {
5     this.$emit('response', 'hello from child')
6   }
7 }
8 </script>
9
10 <template>
11   <h2>Child component</h2>
12 </template>
13
14
15
16
17
18
19
```

Child component

hello from child

Sloty

Sloty są mechanizmem, który pozwala nam na tworzenie komponentów wielokrotnego użytku. Mechanizm ten ułatwia korzystanie z reguły DRY (Don't Repeat Yourself). Sloty umożliwiają umieszczanie lub zmianę treści. Sloty w Vue.js można rozumieć jako miejsce, w którym umieszcza się nową treść lub pozostawia tę domyślnie zadeklarowaną.



The screenshot shows a code editor with two files: `App.vue` and `AlertBox.vue`. The `App.vue` file contains a script section with an import and a default export component that uses the `AlertBox` component. The `AlertBox.vue` file contains a template section with a `<div>` containing a `` tag, a `<slot />`, and another `<div>`. The `App.vue` file also contains a template section with a `<AlertBox>` component. The `AlertBox` component is rendered with the text "Something bad happened." inside the slot. The `AlertBox` component's template also includes a `` tag with the text "Error!". The `AlertBox` component's style section includes a `<style>` tag with a `.alert-box` class and a `strong` selector. The `Preview` tab on the right shows the rendered output, which is a red box with the text "Error! Something bad happened."

```
App.vue | AlertBox.vue | Import Map | PREVIEW | JS | CSS | SSR
1 <script>
2 import AlertBox from './AlertBox.vue'
3
4 export default {
5   components: { AlertBox }
6 }
7 </script>
8
9 <template>
10 <AlertBox>
11   Something bad happened.
12 </AlertBox>
13 </template>
14
15
16
17
18
19
20
21
```

```
AlertBox.vue
1 <template>
2 <div class="alert-box">
3   <strong>Error!</strong>
4   <br/>
5   <slot />
6 </div>
7 </template>
8
9 <style scoped>
10 .alert-box {
11   color: #666;
12   border: 1px solid red;
13   border-radius: 4px;
14   padding: 20px;
15   background-color: #f8f8f8;
16 }
17
18 strong {
19   color: red;
20 }
21 </style>
```

Error!
Something bad happened.

Routing

Aby skomunikować poszczególne komponenty ze sobą, trzeba je odpowiednio zaimportować. Routing pozwala użytkownikowi na przełączanie się pomiędzy stronami bez konieczności odświeżania strony. Ułatwia to nawigację w samej aplikacji. Do bardziej wyszukanych przypadków można zastosować oficjalną bibliotekę `vue-router`.

```
1 <script>
2 import Home from './Home.vue'
3 import About from './About.vue'
4 import NotFound from './NotFound.vue'
5
6 const routes = {
7   '/': Home,
8   '/about': About
9 }
10
11 export default {
12   data() {
13     return {
14       currentPath: window.location.hash
15     }
16   },
17   computed: {
18     currentView() {
19       return routes[this.currentPath.slice(1) || '/'] || NotFound
20     }
21   },
22   mounted() {
23     window.addEventListener('hashchange', () => {
24       this.currentPath = window.location.hash
25     })
26   }
27 }
28 </script>
29
30 <template>
31   <a href="#/">Home</a> |
32   <a href="#/about">About</a> |
33   <a href="#/non-existent-path">Broken Link</a>
34   <component :is="currentView" />
35 </template>
```

—

App time !



Zalety korzystania z Vue.js

- szybszy niż React czy Angular
- posiada dobrą przejrzystą dokumentację
- prostota
- niski próg wejścia - świetny dla początkujących
- Vue cli, Vite - szybkie tworzenie projektów



Wady

- elastyczność, ponieważ może prowadzić do większej liczby błędów w kodzie
- mała społeczność
- brak wsparcia ze strony dużych korporacji



Bibliografia

- <https://typeofweb.com/pierwszy-komponent-vue-js>
- <https://poradniktechnologiczny.pl/vuejs-jak-wymieniac-dane-miedzy-komponentami>
- <https://vuejs.org/guide/introduction.html>
- <https://medium.com/quick-code/how-to-create-a-simple-vue-js-app-in-5-minutes-f74fb04adc01>
- <https://smartbees.pl/blog/vuejs-czyli-wszystko-co-warto-wiedziec-o-tej-bibliotece-javascript>
- <https://www.moonbite.pl/blog/technologie-informatyczne/czym-jest-vuejs-i-dlaczego-warto-z-niego-korzystac>

Dziękujemy za uwagę