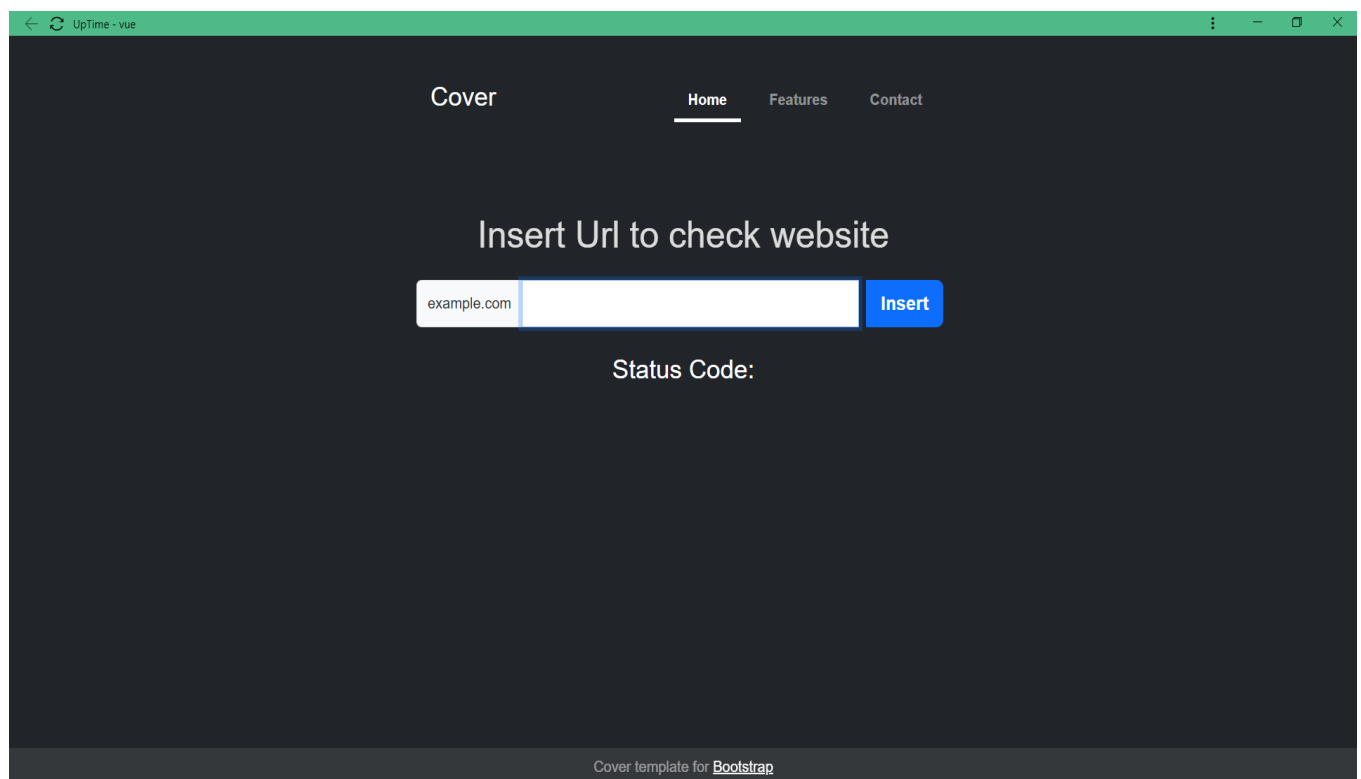
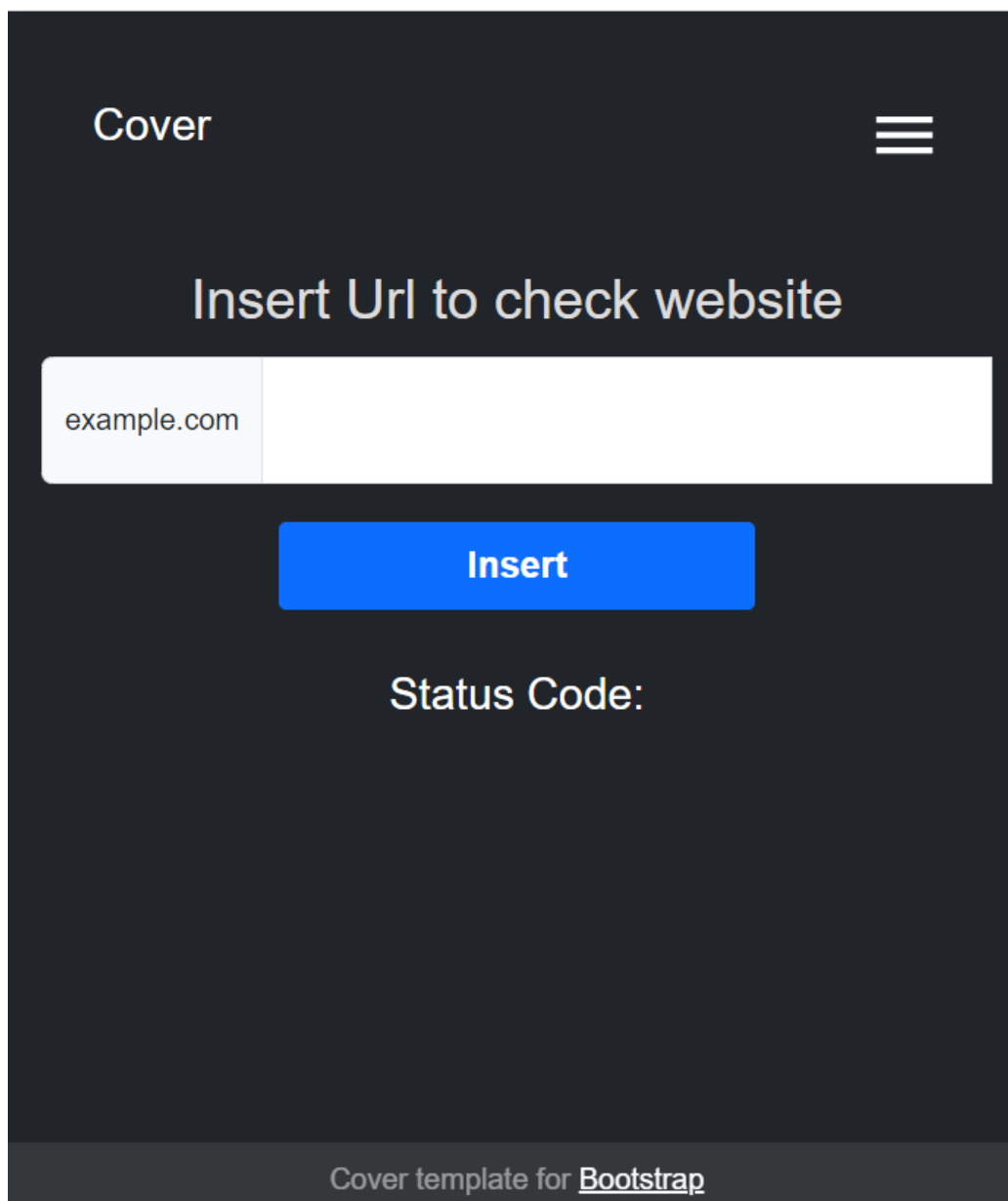


# Uptime

Апликација за проверка на availability/uptime на  
сервиси мобилна и веб



Првичен изглед на интерфејсот за РС



Првичен изглед на интерфејсот за мобилен телефон

Чекори кои треба да ги следиме за правилно да ја користеме апликацијата:

- Чекор 1: Внесуваме валидна URL адреса на input полето и притискаме на копчето "Insert".

- За пример ќе ја земеме URL адресата:

<https://joksim.github.io/en/calendar/>

The screenshot shows a web application with a dark theme. At the top, there is a navigation bar with links: "Cover", "Home" (underlined), "Features", and "Contact". The main heading is "Insert Url to check website". Below this is a form with a text input field containing "example.com", followed by a larger empty input field, and a blue "Insert" button. The result is displayed in green text: "Status Code: 200 OK". Below the status code is the URL "https://joksim.github.io/en/calendar/". There are three input fields for "Insert Tag", "Insert Attribute", and "Value". At the bottom, there are two blue buttons labeled "HTML" and "API". Below these buttons is the text "Exist:". At the very bottom, it says "Cover template for [Bootstrap](#)".

Во овој момент дознаваме дека Веб страницата е валидна бидејќи имаме **Status Code: 200 OK**

- Доколку веб страницата е валидна, автоматски ни се овозможуваат нови полиња кои треба да ги пополнеме за понатамошна проверка.
- Чекор 2: Во овој чекор треба да одлучиме дали сакаме да провериме вредности од API или од HTML страница.
  - Во нашиот случај избираме HTML. Во останатите input полиња ќе внесеме податоци за Tag: link, Attribute: rel, Value: alternate

```
← → ↻ ⓘ view-source:https://joksim.github.io/en/calendar/
Line wrap
1 <!DOCTYPE html>
2 <html lang="en-us">
3
4 <head>
5
6 <meta charset="utf-8">
7 <meta name="viewport" content="width=device-width, initial-scale=1">
8 <meta http-equiv="X-UA-Compatible" content="IE=edge">
9 <meta name="generator" content="Source Themes Academic 4.7.0">
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24 <meta name="author" content="Boban Joksimoski">
25
26
27
28
29
30
31 <meta name="description" content="Teaching & research assistant">
32
33
34 <link rel="alternate" hreflang="mk" href="https://joksim.github.io/mk/calendar/">
35
36 <link rel="alternate" hreflang="en-us" href="https://joksim.github.io/en/calendar/">
37
38
39
40
41
42
43
44 <meta name="theme-color" content="rgb(0, 136, 204)">
45
46
47
```

[Cover](#)[Home](#)[Features](#)[Contact](#)

Insert Url to check website

example.com

Insert

Status Code: 200 OK

https://joksim.github.io/en/calendar/

link

rel

alternate

HTML

API

Exist: True

Cover template for [Bootstrap](#)

Во овој момент дознаваме дека постои

```
<link rel="alternate">
```

Во HTML страницата бидејќи

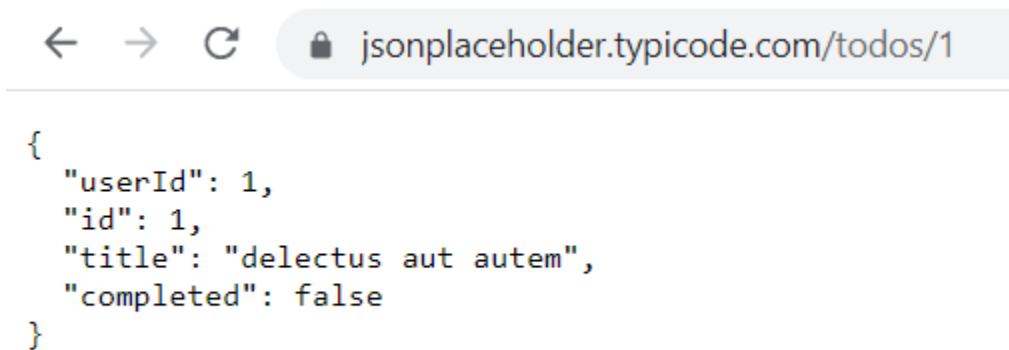
**Exist: True**

- **Чекор 3: Проверка на API - клуч: вредност**

- За пример ќе го земеме ова API:

<https://jsonplaceholder.typicode.com/todos/1>

- Имаме JSON формат:



- Следно ќе провереме дали постои
    - Key: "id", Value: "1"

The screenshot shows a web application with a dark theme. At the top, there is a navigation bar with links: 'Cover', 'Home' (which is underlined), 'Features', and 'Contact'. Below the navigation bar, the text 'Insert Url to check website' is displayed. There is a text input field containing 'example.com' and a blue button labeled 'Insert'. Below the input field, the text 'Status Code: 200 OK' is shown in green. Underneath, the URL 'https://jsonplaceholder.typicode.com/todos/1' is displayed. There are two input fields: one labeled 'id' containing '1' and another empty field. Below these fields are two blue buttons labeled 'HTML' and 'API'. At the bottom, the text 'Exist: True' is shown in green. The footer contains the text 'Cover template for [Bootstrap](#)'.

Добивме Status Code: 200 OK и Exist: True

Тоа значи дека имаме валидно API и постои клучот "id" со вредност "1"

Во најлошо сценарио можеме да добиеме Status Code: 400 Bad Request и Exist: False што значи дека имаме погрешно внесено URL адреса или имаме непостоечни вредности ставено во input полињата.

Cover

Home

Features

Contact

Insert Url to check website

example.com

invalidURL.invalid

Insert

Status Code: 400 Bad Request

invalidURL.invalid

invalid

url

HTML

API

Exist: False

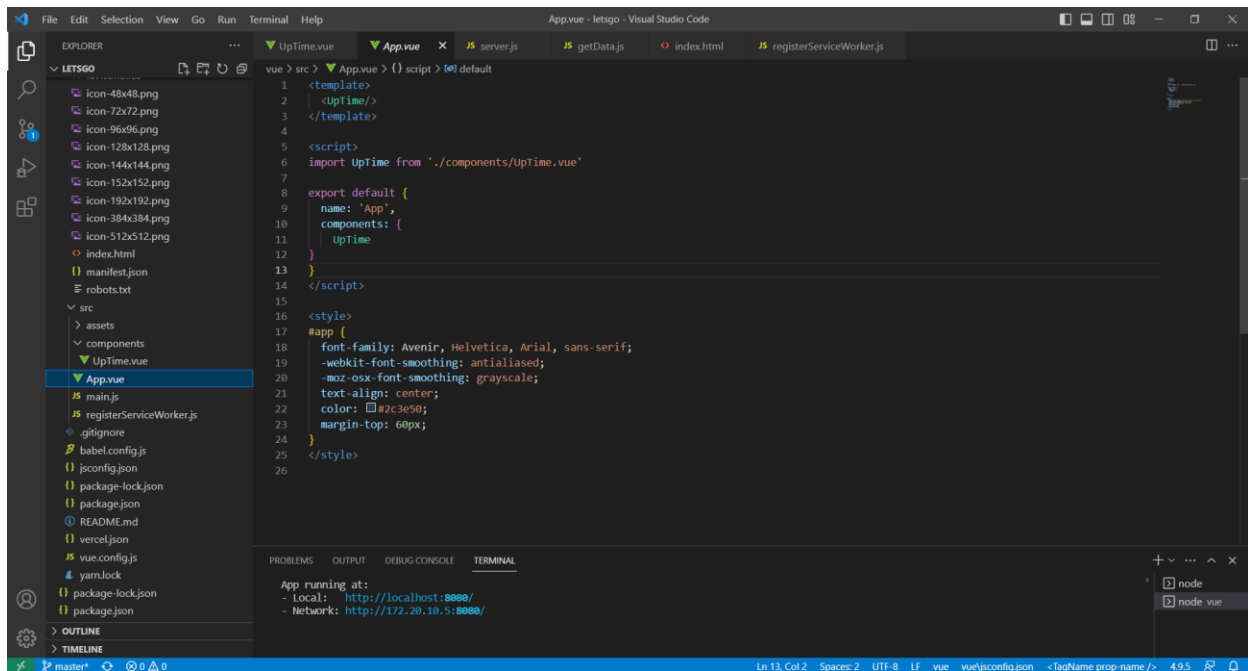
Cover template for [Bootstrap](#)

Вака изгледа кога имаме погрешни податоци  
внесено во апликацијата



# CODE:

- ✓ Frontend на апликацијата: Vue.js
- ✓ Користиме една компонента која е именувана како UpTime.vue



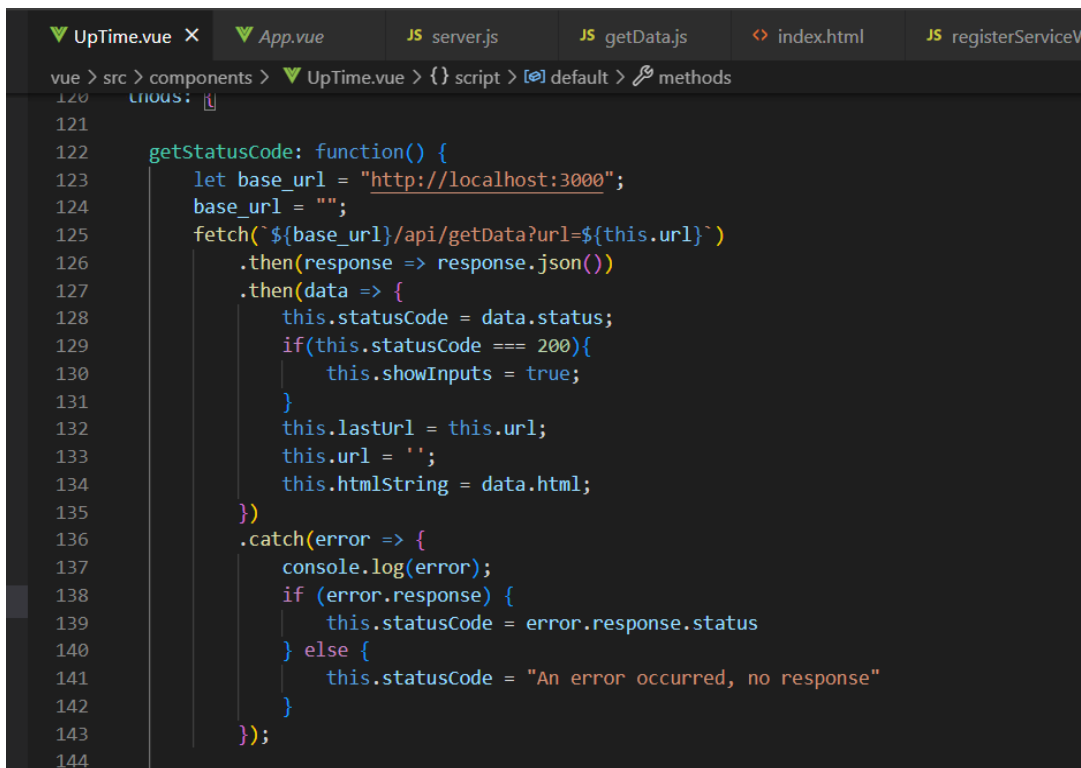
The screenshot shows the Visual Studio Code editor with the following details:

- Explorer Panel:** Shows the file structure of the project. The `src` directory is expanded, showing `assets`, `components`, and `App.vue` (which is selected).
- Editor Panel:** Displays the content of `App.vue`. The code is as follows:

```
1 <template>
2   <UpTime/>
3 </template>
4
5 <script>
6   import UpTime from './components/UpTime.vue'
7
8   export default {
9     name: 'App',
10    components: {
11      UpTime
12    }
13  }
14 </script>
15
16 <style>
17   #app {
18     font-family: Avenir, Helvetica, Arial, sans-serif;
19     -webkit-font-smoothing: antialiased;
20     -moz-osx-font-smoothing: grayscale;
21     text-align: center;
22     color: #2c3e50;
23     margin-top: 60px;
24   }
25 </style>
26
```
- Terminal Panel:** Shows the output of the application running. The text is:

```
App running at:
- Local: http://localhost:8080/
- Network: http://172.20.10.51:8080/
```

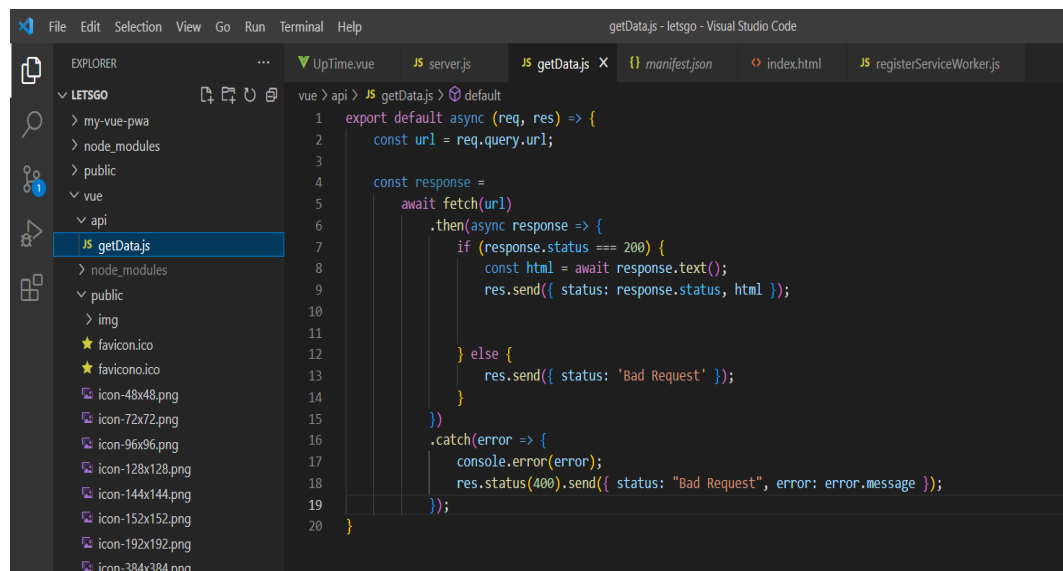
- ✓ Во UpTime.vue кога корисникот ќе внесе URL адреса и притисне на копчето "Insert", се повикува функцијата `getStatusCode()`



```
vue > src > components > UpTime.vue > {} script > default > methods
120   this.$emit('status', this.statusCode);
121
122   getStatusCode: function() {
123     let base_url = "http://localhost:3000";
124     base_url = "";
125     fetch(`${base_url}/api/getData?url=${this.url}`)
126       .then(response => response.json())
127       .then(data => {
128         this.statusCode = data.status;
129         if(this.statusCode === 200){
130           this.showInputs = true;
131         }
132         this.lastUrl = this.url;
133         this.url = '';
134         this.htmlString = data.html;
135       })
136       .catch(error => {
137         console.log(error);
138         if (error.response) {
139           this.statusCode = error.response.status
140         } else {
141           this.statusCode = "An error occurred, no response"
142         }
143       });
144   }
```

- ✓ Со функцијата `GetStatusCode()` праќаме `Get Request` до `/api/getData` со тоа што ја праќаме URL адресата од `input` полето. Потоа ги зема податоците кои ни се вратени од `/api/getData` и правиме проверка дали статус кодот е 200, доколку не е статус кодот 200 праќаме **error message**, во спротивно праѓаме статус код: 200 и ги овозможуваме останатите `input` полиња.

- ✓ Зошто ни е потребно да праќаме URL адреса преку `/api/getData`?
- ✓ Ако пратеме `get request` преку фронтенд, т.е преку `UpTime.vue`, нè блокираат Веб страниците со **CORS**, а доколку пратеме `get request` од `getData.js` која е извршена како **ламбда функција** од **Vercel** успешно се праќа `get` барањето. Со тоа успешно го добиваме статус кодот од `get` барањето од Веб страницата.



```
1 export default async (req, res) => {
2   const url = req.query.url;
3
4   const response =
5     await fetch(url)
6     .then(async response => {
7       if (response.status === 200) {
8         const html = await response.text();
9         res.send({ status: response.status, html });
10      }
11    })
12    .catch(error => {
13      console.error(error);
14      res.status(400).send({ status: "Bad Request", error: error.message });
15    });
16 }
17
18
19
20 }
```

✓ Во `/api/getData` ја земаме URL адресата од `request`-от и праќаме `get` барање то таа URL адреса. Доколку добиеме статус код 200, тогаш го праќаме назад кон `UpTime.vue` како `response` со што го праќаме и целиот HTML/API документ. Доколку не добиеме статус код 200, враќаме назад како `response status code 400` и **error message**.

✓ Во `UpTime.vue` компонентата имаме уште 2 функции: `checkHtml()` и `checkApi()`

# checkHtml()

```
UpTime.vue X JS server.js JS getData.js {} manifest.json <> index.html JS registre
vue > src > components > UpTime.vue > {} script > default > methods
146
147   checkHtml: function(htmlString,tag,att,value) {
148
149       this.isHtml = true;
150       this.exist = false;
151
152
153       if(value === '' || att === '' || tag === ''){
154           return
155       }
156
157       const parser = new DOMParser();
158       const doc = parser.parseFromString(htmlString, 'text/html');
159       const hrefs = doc.querySelectorAll(['+att +']);
160
161
162
163       for(let h of hrefs){
164
165           if(h.tagName.toLowerCase() === tag && h[att].includes(value)){
166               this.exist = true;
167               return 1;
168           }
169       }
170
171       return 0;
172
173   }
```



Оваа функција прима 4 аргументи:

htmlString, tag, attribute, value. Од дадениот HTML документ ги листа сите tags и проверува дали на одреден tag постои соодветен attribute со соодветна value.

# checkApi()

```
vue > src > components > UpTime.vue > {} script > [default] > methods > checkApi
172
173     },
174
175     checkApi: function(htmlString,att,value2) {
176
177         this.exist = false;
178         this.isHtml = false;
179
180
181         if(value2 === '' || att === ''){
182             return
183         }
184
185         //https://dog.ceo/api/breeds/image/random
186         // https://jsonplaceholder.typicode.com/todos/1
187         const parser = new DOMParser();
188         const doc = parser.parseFromString(htmlString, 'text/html');
189         const doc2 = doc.body.textContent;
190         const jsonObject = JSON.parse(doc2);
191         this.exist = false;
192         for (let [key,value] of Object.entries(jsonObject)) {
193             if(key == att && value == value2){
194                 console.log(jsonObject[att] == value2)
195                 this.exist = true;
196             }
197         }
198     }
199
200 }
201 },
```

✓ Оваа функција прима 3 аргументи:

htmlString, key, value. Од даденото API ги листа сите вредности и проверува дали за одреден key постои соодветна value.

