
Exercise. Todo (part 1) – Creating basic app with JavaScript

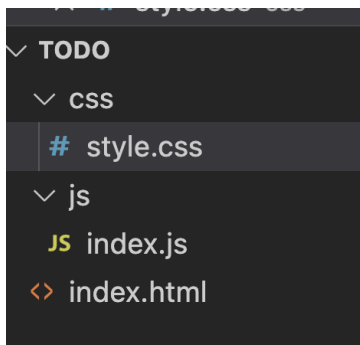
This is the first part of exercise series, where you create a simple Todo app using JavaScript, NodeJS, Express and PostgreSQL. During the first part a simple web app without any backend is created. User can add new tasks and there are displayed on browser. If user refreshes browser, data is cleared, since it is not saved on database yet.

Visual Studio is used as a development tool.

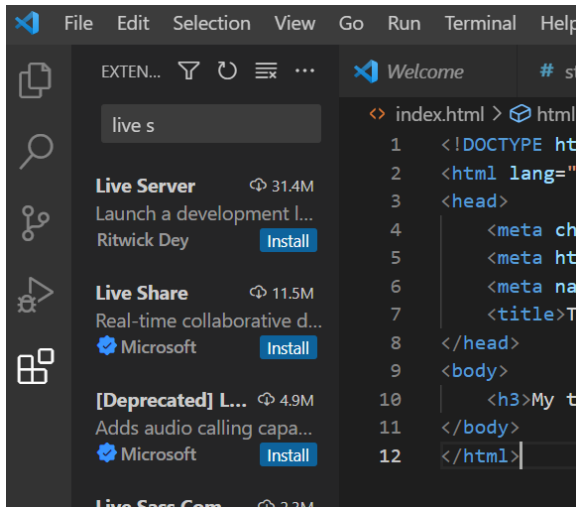
Key learnings for this part are:

- Create a new project for frontend and divide files into separate folders
- Add Bootstrap to a website
- Implement front-end logic for displaying and adding tasks

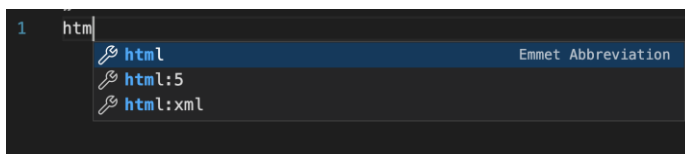
Create an empty folder under name todo to your workstation and create subfolders and files for basic web application.



Install live server (in case not already installed) for testing through Visual Studio Code extensions. Search for live server and press blue Install button.



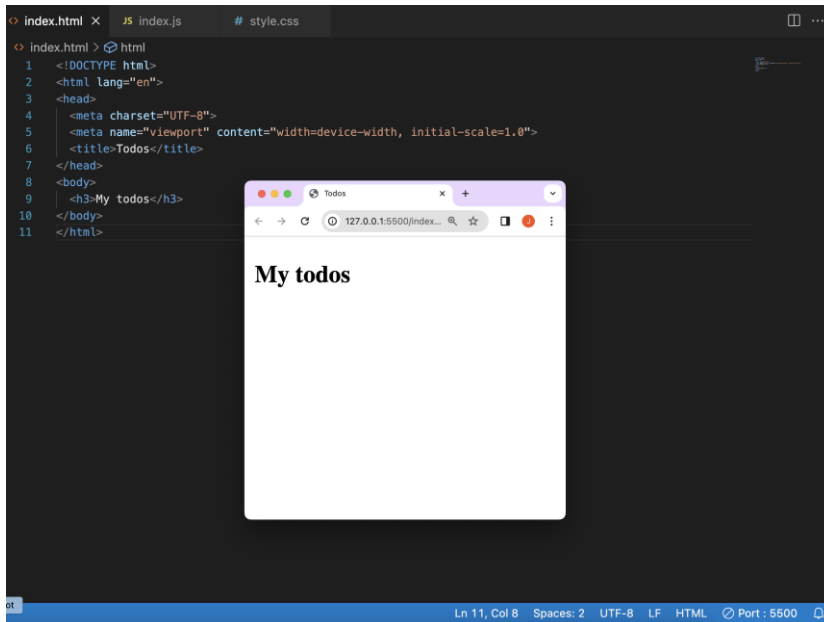
Add basic HTML file. Use Visual Studio Code to generate basic HTML skeleton (start typing html and select html:5).



Change title and add header. Press Go Live button on the bottom right corner to start test web server and open webpage on browser.



Web page is launched and running on 127.0.0.1:5500/, which is an address for local web development server (127.0.0.1 = localhost) and port 5500 is used (in case you launch multiple live servers, port will be 5001, 5002 and so on).



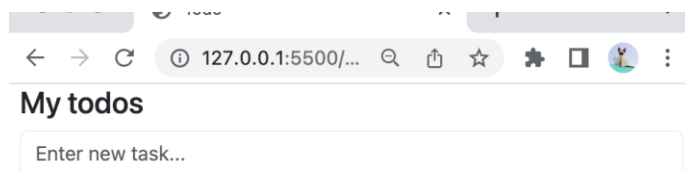
Add [Bootstrap](#) to your project. Copy links from the documentation and do not try to type code based on the image below. Link your own stylesheet as well and provide access to local JavaScript file. Content for these files will be created later.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/d
  <link href="./css/style.css" rel="stylesheet">
  <title>Todos</title>
</head>
<body>
  <h3>My todos</h3>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/c
  <script src="./js/index.js"></script>
</body>
</html>
```

Add a simple form which is used to add new task and a list, where tasks are displayed. Bootstrap classes are used for styling form and list. See documentation for more details how Bootstrap uses classes for style definitions.

```
<body>
  <div class="container-fluid">
    <h3>My todos</h3>
    <form>
      <input class="form-control" placeholder="Add new task..." />
    </form>
    <ul class="list-group"></ul>
  </div>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/boc
```

Test out web page. You can clearly see that Bootstrap is used and classes container-fluid, form-control and list-group are taking effect.



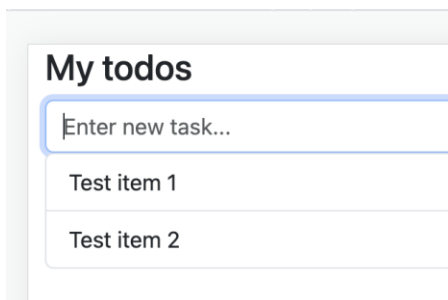
Implement basic functionality using JavaScript. On index.js first get references to UI elements list and input since you need to be able to add new rows to the list and read input from the input field. QuerySelector is used to get elements and store references into variables, which then can be used to manipulate HTML elements on the document.

EventListener for input is also added. Whenever user presses Enter key on the input field, a new row with input text is added to the todo list. In case there is no text, nothing is added. New line to list is added by creating new li element inside list (which is ul). List-group-item class is used (this is Bootstrap class and provides nice appearance to the list item). Text from input is displayed on the row. After adding new row input is set to empty string, so it is convenient for user to add another text (without need to erase previous text).

```
const list = document.querySelector('ul')
const input = document.querySelector('input')

input.addEventListener('keypress', (event) => {
  if (event.key === 'Enter') {
    event.preventDefault()
    const task = input.value.trim()
    if (task !== '') {
      const li = document.createElement('li')
      li.setAttribute('class', 'list-group-item')
      li.innerHTML = task
      list.append(li)
      input.value = ''
    }
  }
})
```

Test out web page. Now you should be able to add new tasks to the list.



My todos

Enter new task...

Test item 1

Test item 2

Provide styling (style.css) to add some whitespace around header (top and bottom 1em, left and right 0) and form (bottom 1em). Check out that styles are applied to the page.

```
h3 {
  margin: 1em 0;
}

form {
  margin-bottom: 1em;
}
```