Tutorial 4 Html CSS JS

> Version 3
>
> Version1 : Designed by Yueming ZHU (in 2017)
>
> Version2: Modified by Zhengchang HUA (in 2018)
>
> Version3: Modified by Xinghe YAO and Yueming ZHU(in 2020)
>
> Version4: Add vue guideline by Yueming ZHU (in 2023)

Software used

Any text editor you like (e.g. Notepad++, Visual Studio Code, Sublime, etc.), or IDEs (e.g. Jetbrains WebStorm)

We recomend to use **WebStrom**

Experiment Objective

Learn basic usage of HTML, CSS, JS by building a web front end of flight management system

Why called HTML

1. HTML is not a programming language! It's a kind of markup language
2. HTML uses **tag** to describe website

   e.g.

   ```
   <label>content</label>
   <p>It is a paragraph</p>
   ```

3. different tag plays a different role in HTML

Basic HTML structure

```
<!DOCTYPE html> <!-- Type of webpage -->
<html> <!-- The entire webpage is wrapped with html tags -->
<head> <!-- The meta information of the webpage, this part of the content will
not be displayed on the webpage -->
  <title></title> <!-- The title -->
</head>
<body> <!-- The main content of the page is displayed in the body -->

</body>
</html>
```

Also you might notice that annotations in html is <!-- 注释内容 -->

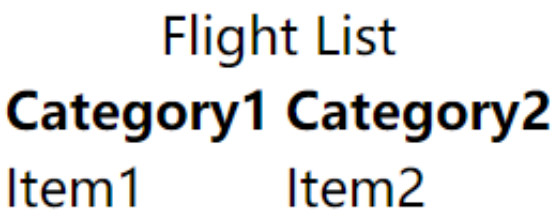
Begin the Exercise(HTML/CSS/JS)

1. First Try - HTML Table

Try following code:

```
<table>
    <caption>Flight List</caption>
    <tr>
      <th>Category1</th>
      <th>Category2</th>
    </tr>
    <tr>
      <td>Item1</td>
      <td>Item2</td>
    </tr>
</table>
```

You put these content into **body** tag and and rename the file to `xxx.html`, you will get



Some explanations about tags above

1. `caption`: the title of table
2. `tr`: table row, could contain multiple `<th>` and `<td>` tag
3. `th`: table head, as you see Category is bold
4. `td`: table item

Now let's try more code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<table>
    <caption>
        <h2>Flight List</h2>
    </caption>
    <tbody>
    <tr>
        <th>Flight No.</th>
        <th>Airline Company</th>
        <th>From</th>
        <th>To</th>
        <th>Date</th>
        <th>Departure Time</th>
        <th>Arrival Time</th>
    </tr>
    <tr>
        <td>ZH1858</td>
        <td>Shenzhen Airlines</td>
        <td>SHA</td>
        <td>PEK</td>
        <td>2020/09/25</td>
        <td>07:55</td>
        <td>10:10</td>
    </tr>
    <tr>
        <td>MU5479</td>
        <td>Eastern Airlines</td>
        <td>TAO</td>
        <td>WUH</td>
        <td>2020/09/25</td>
        <td>13:40</td>
        <td>15:55</td>
    </tr>
    <tr>
        <td>CA5795</td>
        <td>Air China</td>
        <td>KMG</td>
        <td>CKG</td>
        <td>2020/09/20</td>
        <td>06:50</td>
        <td>08:25</td>
    </tr>
```

```
    <tr>
        <td>HU7707</td>
        <td>Hainan Airlines</td>
        <td>PEK</td>
        <td>SZX</td>
        <td>2020/09/20</td>
        <td>22:00</td>
        <td>01:30+1</td>
    </tr>
    <tr>
        <td>SC4837</td>
        <td>Shandong Airlines</td>
        <td>CKG</td>
        <td>TAO</td>
        <td>2020/09/20</td>
        <td>07:00</td>
        <td>09:25</td>
    </tr>
    </tbody>
</table>
</body>
</html>
```

Flight List

Flight No.	Airline Company	From	To	Date	Departure Time	Arrival Time
ZH1858	Shenzhen Airlines	SHA	PEK	2020/09/25	07:55	10:10
MU5479	Eastern Airlines	TAO	WUH	2020/09/25	13:40	15:55
CA5795	Air China	KMG	CKG	2020/09/20	06:50	08:25
HU7707	Hainan Airlines	PEK	SZX	2020/09/20	22:00	01:30+1
SC4837	Shandong Airlines	CKG	TAO	2020/09/20	07:00	09:25

That's a fairly simple and crude page, but we can add some styles on it and make it have a nicer look.

A tag could always has many attributes to make it prettier, such as background color, border, distance to other items ans so on.

Create a new file named `table2.html`, and input the following

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <title>Flight list</title>
 </head>
 <body>
```

```
<table border="1" cellpadding="10" cellspacing="5">
    <caption>
        <h2>Flight List</h2>
    </caption>
    <tbody>
    <tr>
        <th>Flight No.</th>
        <th>Airline Company</th>
        <th>From</th>
        <th>To</th>
        <th>Date</th>
        <th>Departure Time</th>
        <th>Arrival Time</th>
    </tr>
    <tr bgcolor="#F9E3AF">
        <td>ZH1858</td>
        <td>Shenzhen Airlines</td>
        <td>SHA</td>
        <td>PEK</td>
        <td>2020/09/25</td>
        <td>07:55</td>
        <td>10:10</td>
    </tr>
    <tr bgcolor="#F9E3AF">
        <td>MU5479</td>
        <td>Eastern Airlines</td>
        <td>TAO</td>
        <td>WUH</td>
        <td>2020/09/25</td>
        <td>13:40</td>
        <td>15:55</td>
    </tr>
    <tr bgcolor="#F9E3AF">
        <td>CA5795</td>
        <td>Air China</td>
        <td>KMG</td>
        <td>CKG</td>
        <td>2020/09/20</td>
        <td>06:50</td>
        <td>08:25</td>
    </tr>
    <tr bgcolor="#F9E3AF">
        <td>HU7707</td>
        <td>Hainan Airlines</td>
        <td>PEK</td>
        <td>SZX</td>
        <td>2020/09/20</td>
        <td>22:00</td>
        <td>01:30+1</td>
```

```
        </tr>
        <tr bgcolor="#F9E3AF">
            <td>SC4837</td>
            <td>Shandong Airlines</td>
            <td>CKG</td>
            <td>TAO</td>
            <td>2020/09/20</td>
            <td>07:00</td>
            <td>09:25</td>
        </tr>
        </tbody>
    </table>
 </body>
</html>
```

Explanation about some attributes

1. `border`: a border around html element
2. `cellpadding`: The cellpadding attribute specifies the space, in pixels, between the cell wall and the cell content.
3. `cellspacing`: The cellspacing attribute specifies the space, in pixels, between cells.
4. `bgcolor`: The bgcolor attribute specifies a background color of a table.

Note: The attributes above are not supported in HTML5 standard. This is for demonstration only, do not use them on production

And the page then looks like this.

Flight List

Flight No.	Airline Company	From	To	Date	Departure Time	Arrival Time
ZH1858	Shenzhen Airlines	SHA	PEK	2020/09/25	07:55	10:10
MU5479	Eastern Airlines	TAO	WUH	2020/09/25	13:40	15:55
CA5795	Air China	KMG	CKG	2020/09/20	06:50	08:25
HU7707	Hainan Airlines	PEK	SZX	2020/09/20	22:00	01:30+1
SC4837	Shandong Airlines	CKG	TAO	2020/09/20	07:00	09:25

2. CSS (Cascading Style Sheet)

CSS is short for cascading style sheet, which provides a place to integrate all html tags together, and helps html to look better.

Create a new file called `table.css` and copy the following code.

```
@CHARSET "UTF-8";

#a {
    font-size: 20px;
    font-style: italic;
}

table {
    border: 2px solid black;
    border-collapse: separate;
    border-spacing: 5px;
    margin: 0 auto;
}

th {
    border: 1px solid black;
    padding: 15px;
    text-align: center;
    font-size: 15px;
}

td {
    border: 1px solid black;
    padding: 15px;
    background-color: #F9E3AF;
    text-align: center;
    // 更加灵活的位置摆放参考flex布局
    font-size: 15px;
}
```

Don't forget to add the following line to your table.html, inside <head> tag:

```
<link rel="stylesheet" type="text/css" href="table.css"/>
```

In this stylesheet, there are several CSS Selectors:

Each of them took effect on <table>, <th>, <td> elements, respectively. #a is an **ID selector**, which set the style for a tag with given id attribute. The #a selector in CSS corresponds with the id="a" attribute. In table.html, we edit the code of table caption as following

```
<caption id="a"><h2>Flight List</h2></caption>
```

Then page will look like this

Flight List

Flight No.	Airline Company	From	To	Date	Departure Time	Arrival Time
ZH1858	Shenzhen Airlines	SHA	PEK	2020/09/25	07:55	10:10
MU5479	Eastern Airlines	TAO	WUH	2020/09/25	13:40	15:55
CA5795	Air China	KMG	CKG	2020/09/20	06:50	08:25
HU7707	Hainan Airlines	PEK	SZX	2020/09/20	22:00	01:30+1
SC4837	Shandong Airlines	CKG	TAO	2020/09/20	07:00	09:25

3. Second Try - HTML Form

If we want to add some new flight information to this table, we need to create a user interface for users to input flight information manually, and create an item in the table. The HTML <form> element is used to create a HTML form.

Form can contain the <input> element. E.g., the text box, the check box, the radio button, and the submit button, etc. Then, let's create a form for submitting flight information.

Add the following code in your `table.html`, between the tag </table> and </body>

```
    <div>
        <form name="form1">
            Flight No.:
            <label>
                <input type="text" name="flight-no"/>
            </label>
            <br/> Airline Company:
            <label>
                <input type="text" name="airline-company"/>
            </label>
            <br/> From:
            <label>
                <input type="text" name="from"/>
            </label>
            <br/> To:
            <label>
                <input type="text" name="to"/>
            </label>
            <br/> Date:
            <label for="year"></label><select id="year">
```

```
                <option>2021</option>
                <option>2020</option>
                <option>2019</option>
            </select> year
            <label for="month"></label><select id="month"></select> month
            <label for="day"></label><select id="day"></select> day
            <br/> Departure Time:
            <label for="dhour"></label><select id="dhour"></select>:
            <label for="dminute"></label><select id="dminute"></select>
            <br/> Arrival Time:
            <label for="ahour"></label><select id="ahour"></select>:
            <label for="aminute"></label><select id="aminute"></select>
            <br/>
            <input type="button" value="Add Flight"/>
            <br/>
        </form>
    </div>
```

In the input elements in the form, different type values stands for different input boxes. text are the text boxes, select is the drop down menu, option are the options under the drop down menu. button is the submit button.

The aim of element is to create divisions in the whole HTML document. They can separate the whole document to independent, different parts. It can be used for layout management, and do not need any special styles attached to it.

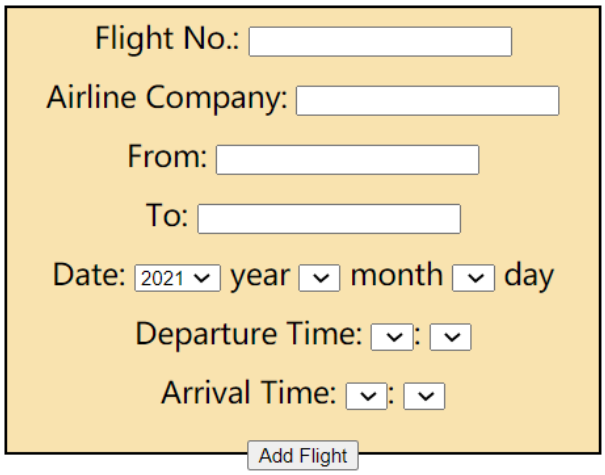
At the same time, we should add some styles to the div element. We'll also achieve this using CSS selectors in stylesheet

```
div {
    line-height: 40px;
    width: 400px;
    height: 300px;
    text-align: center;
    background-color: #F9E3AF;
    font-size: 20px;
    margin: 40px auto 0;
    border: 2px solid black;
}
```

Then the page will look like this

Flight List

Flight No.	Airline Company	From	To	Date	Departure Time	Arrival Time
ZH1858	Shenzhen Airlines	SHA	PEK	2020/09/25	07:55	10:10
MU5479	Eastern Airlines	TAO	WUH	2020/09/25	13:40	15:55
CA5795	Air China	KMG	CKG	2020/09/20	06:50	08:25
HU7707	Hainan Airlines	PEK	SZX	2020/09/20	22:00	01:30+1
SC4837	Shandong Airlines	CKG	TAO	2020/09/20	07:00	09:25



4. JS - Add actions

It is the first try that we connect the html file to the javaScript file.

First, create a file named `table.js` under the same directory of `table.html`

```
function onClickAddFlight() {
    let flightNo = document.querySelector('form input[name="flight-
no"]').value;
    alert(flightNo);
}
```

This function means to display the flight Number that being entered by user when clicking the "Add Flight" button, through which we can check whether our script is correctly linked to the HTML document.

The way we get the information entered by user is to use **querySelector()** method. The **QuerySelector()** method can return the first sub-element that is matched to the specific group of selectors. **Document** can be regarded as the root node of the html page, from which, we can acquire all sub-elements from the html page.

After that, in the <head> section of HTML document, adding the following line:

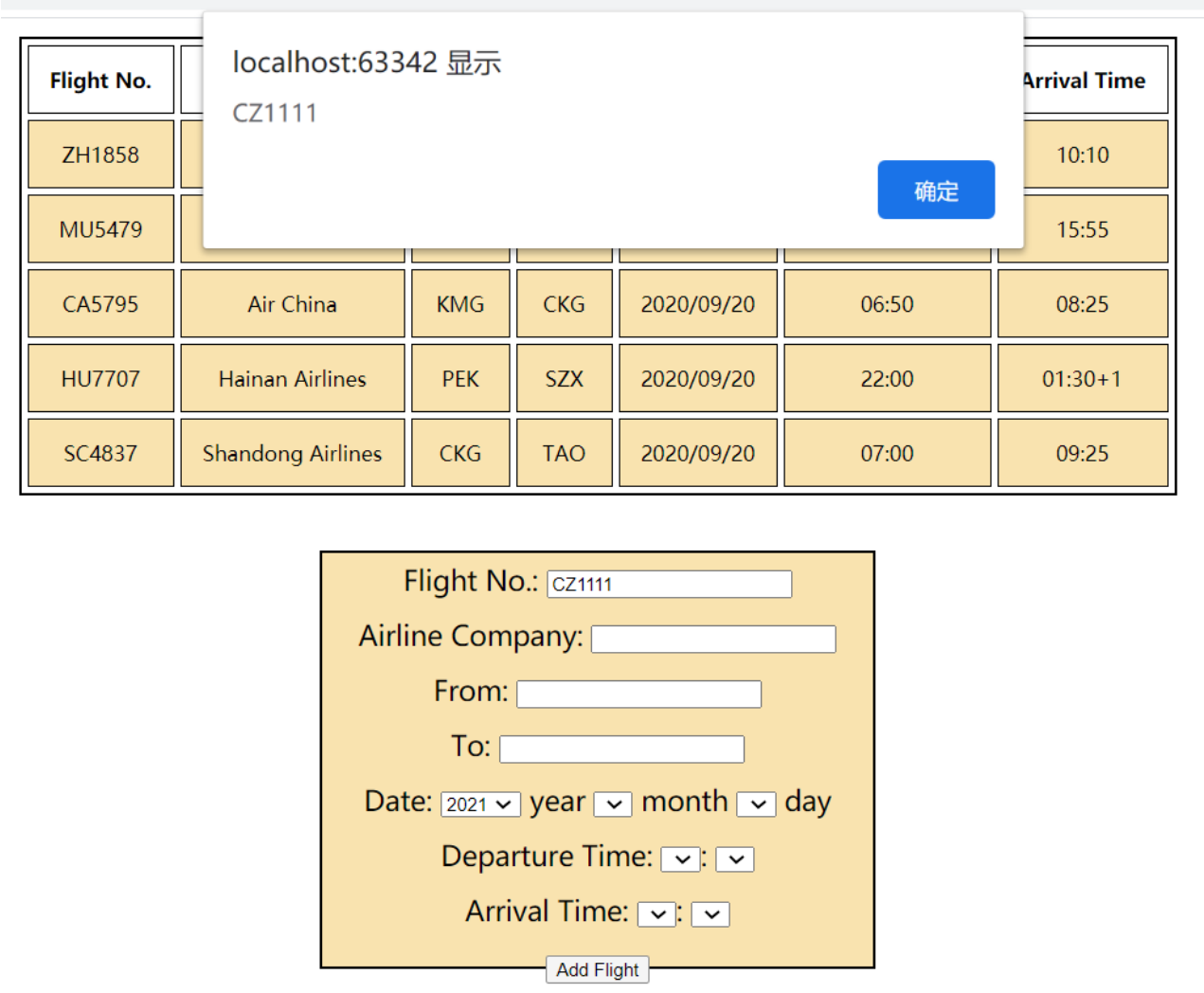
```
<script type="text/javascript" src="table.js"></script>
```

Then, modify the submit button, add a onClick attribute:

```
<input type="button" value="Add Flight" onclick="onClickAddFlight()"><br>
```

Note: This approach to add event listener for click is to test whether the html file is connected to js file only. DO NOT hard code event handlers in HTML attributes in real cases.

Then we can open `table.html` in browser, input a flight number, and click "Add Flight" button to see the effect:



5. Using regular expression to validate user input

Regular Expressions are patterns that used to match character combinations in strings. You can use regular expressions to match the patterns you want, e.g., e-mail addresses, IP addresses, phone numbers, etc.

Note: Regular expression is not a required part in this lecture. But it is better for you to understand some basic concepts and how to use them.

Here are some examples of using regular expressions in JavaScript:

```
/[A-Z0-9]+/ // matches one or more single upper case letter from A to Z, or a digit from 0-9
/[A-Z]{3}/ // matches exactly three uper case letters, each letter in the range of A to Z
/^Flight[A-Z]+\d+$/ // matches a line which starts with "Flight", then following one or more letters from A to Z, then one or more digits from 0-9, then the end of line.
```

Here are some explanations to the regular expressions above:

\d: Stands for digits (0-9)

[]: Stands for a character in given range

{N}: Repeat the pattern for N times

+: Appears no less than once

^: Start of a line

$: End of a line

If you want to learn more about regular expression, the following websites may be of help:

RegExr

Regexper

We have the following requirements for user input:

- Flight No. only contains upper case letters and digits. The first part is the code of airline companies, which is 2 characters long and can only contain digits and upper case letters. The second part is the sequence number, which is 3-4 digits long and can only contain numbers.

  ```
  /^[A-Z0-9]{2}\d{3,4}$/
  ```

- The origin and destination airport code ("From" and "To" rows in the table) can only be 3 characters long, and can only contain upper case letters.

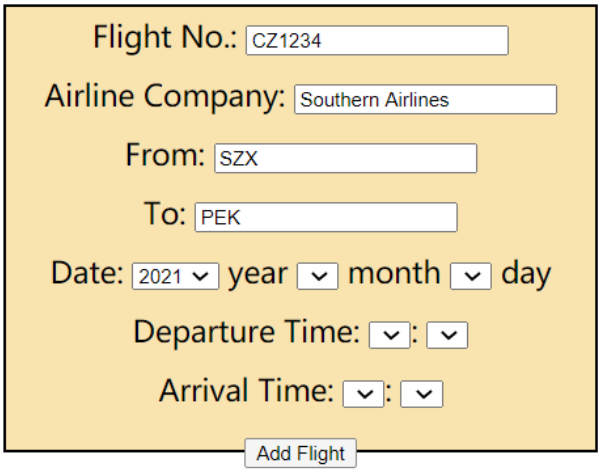
  ```
  /^[A-Z]{3}$/
  ```

We modify the `table.js` as follows:

```
function onClickAddFlight() {
    let flightNo = document.querySelector('form input[name="flight-
no"]').value;
    let origin = document.querySelector('form input[name="from"]').value;
    let destination = document.querySelector('form input[name="to"]').value;
    if (validateInput(flightNo, origin, destination)) {
        alert("Success!");
    }
}

function validateInput(flightNo, origin, destination) {
    let flightNoRegex = new RegExp(/^[A-Z0-9]{2}\d{3,4}$/);
    let airportCodeRegex = new RegExp(/^[A-Z]{3}$/);
    if (!flightNoRegex.test(flightNo)) {
        alert("Invalid Flight No.");
        return false;
    }
    if (!airportCodeRegex.test(origin)) {
        alert("Invalid origin airport code.");
        return false;
    }
    if (!airportCodeRegex.test(destination)) {
        alert("Invalid destination airport code.");
        return false;
    }
    return true;
}
```

If the input is valid, a message box with "Success!" will pop up.



6. More on <select> dropdown menu

Next, we'll add some items into the date and the time menu.

First, setting the year for a reasonable range (from 2000 to 2020).

Adding an `onload` attribute to the <body> tag, while delete the old <option> items:

```
<body onload="initial()">

...
        Date:
        <label for="year"></label><select id="year"></select> year
        <label for="month"></label><select id="month"></select> month
        <label for="day"></label><select id="day"></select> day
...

</body>
```

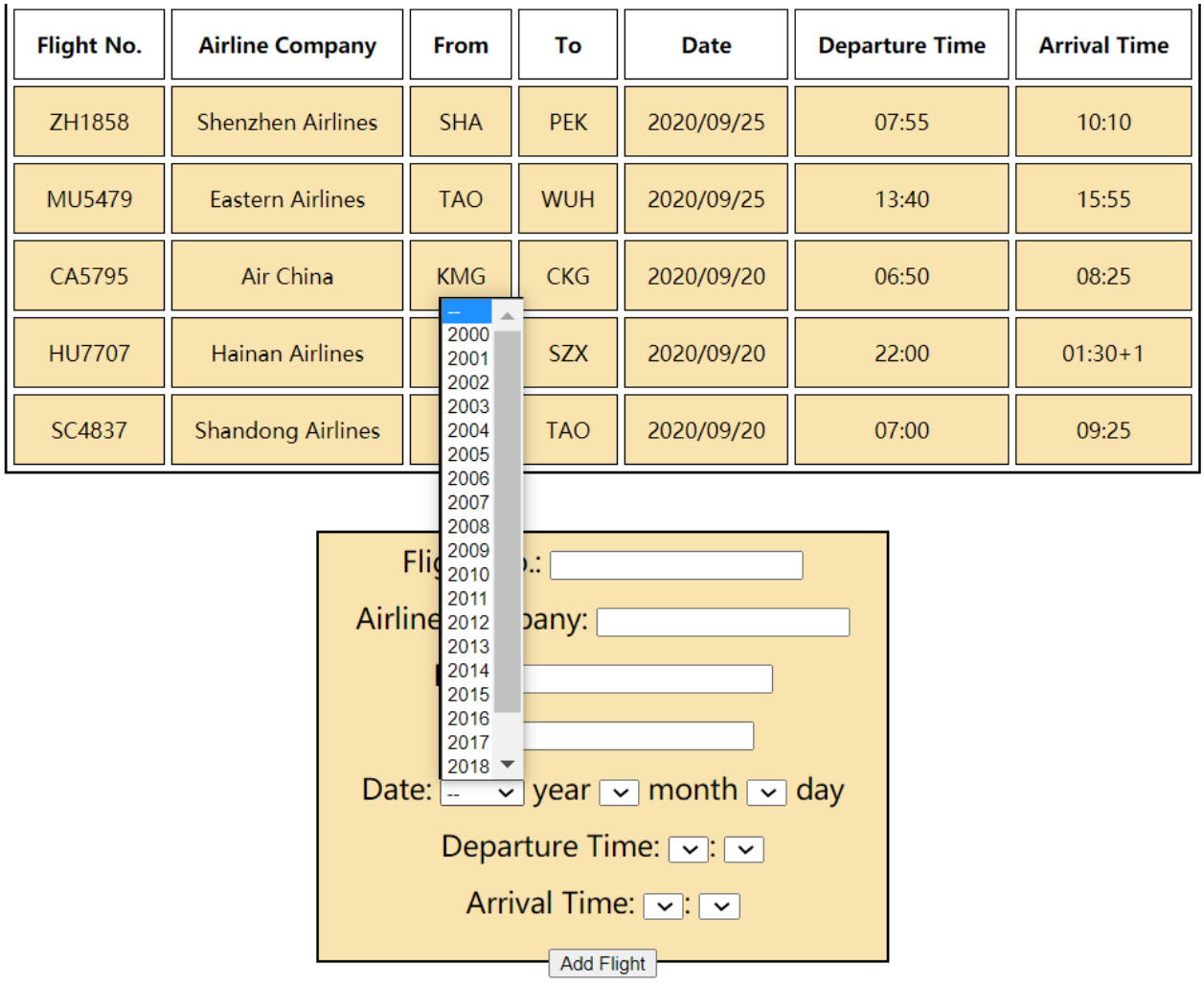
`onload` attribute stands for the code that will be executed when loading the element.

Note: This approach to add event listener is for demonstration only. DO NOT hard code event handlers in HTML attributes in real cases.

And define initial() function in table.js:

```
function initial() {
    let year = document.getElementById("year");
    year.innerHTML = "";
    year.options.add(new Option("--", null));
    for (let i = 2000; i <= 2020; i++) {
        year.options.add(new Option(i, i));
    }
}
```

The page will then look like this:



Next, we use the same approach to modify month, day, and two times.

First, add `onchange` attribute to the `dropdown` menu of year and month. The event will be triggered when the selected item changes.

```
<select id="year" onchange="setMonth()"></select> year
<select id="month" onchange="setDay()"></select> month
```

Note: This approach to add event listener is for demonstration only. DO NOT hard code event handlers in HTML attributes in real cases.

Add the following code to `table.js`:

```
function setMonth() {
    let month = document.getElementById("month");
    month.innerHTML = "";

    month.options.add(new Option("--", null));
    for (let i = 1; i <= 12; i++) {
        month.options.add(new Option(i, i));
    }
}

function setDay() {
    let year = document.getElementById("year").value;
    let month = document.getElementById("month").value;
    let day = document.getElementById("day");
    let data = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31];
    // clear the items
    day.innerHTML = "";

    // add new items
    day.options.add(new Option("--", null));
    for (let i = 1; i <= data[month - 1]; i++) {
        day.options.add(new Option(i, i));
    }
    if (((year % 4 === 0 && year % 100 !== 0) || year % 400 === 0) && month ===
2) {
        day.options.add(new Option(29, 29));
    }
}
```

Please complete this part by yourself.

To ensure that date and time are correctly entered, `onClickAddFlight` function should be modified. When user fills all the selections for date and time, the form will pop-up a "success", otherwise it may inform user to input date and time.

7. Adding <tr> elements dynamically

Before starting this part, I suggest to understant Document Object Model (DOM)

After receiving and validating user's input, we need to implement the function to add new flight into the table. Here we'll use a `<tbody>` tag, which represents the body part of a table. A `<tbody>` element must include one or more `<tr>` tags.

In this exercise, we use the following method: Add a table line <tr> element to <tbody>, as a template, and hide it by setting the style to `display:none`. Every time we add a new line, we display the newly added table line.

Delete the old data, and add a <tbody> tag, set its style attribute to `dispaly: none`:

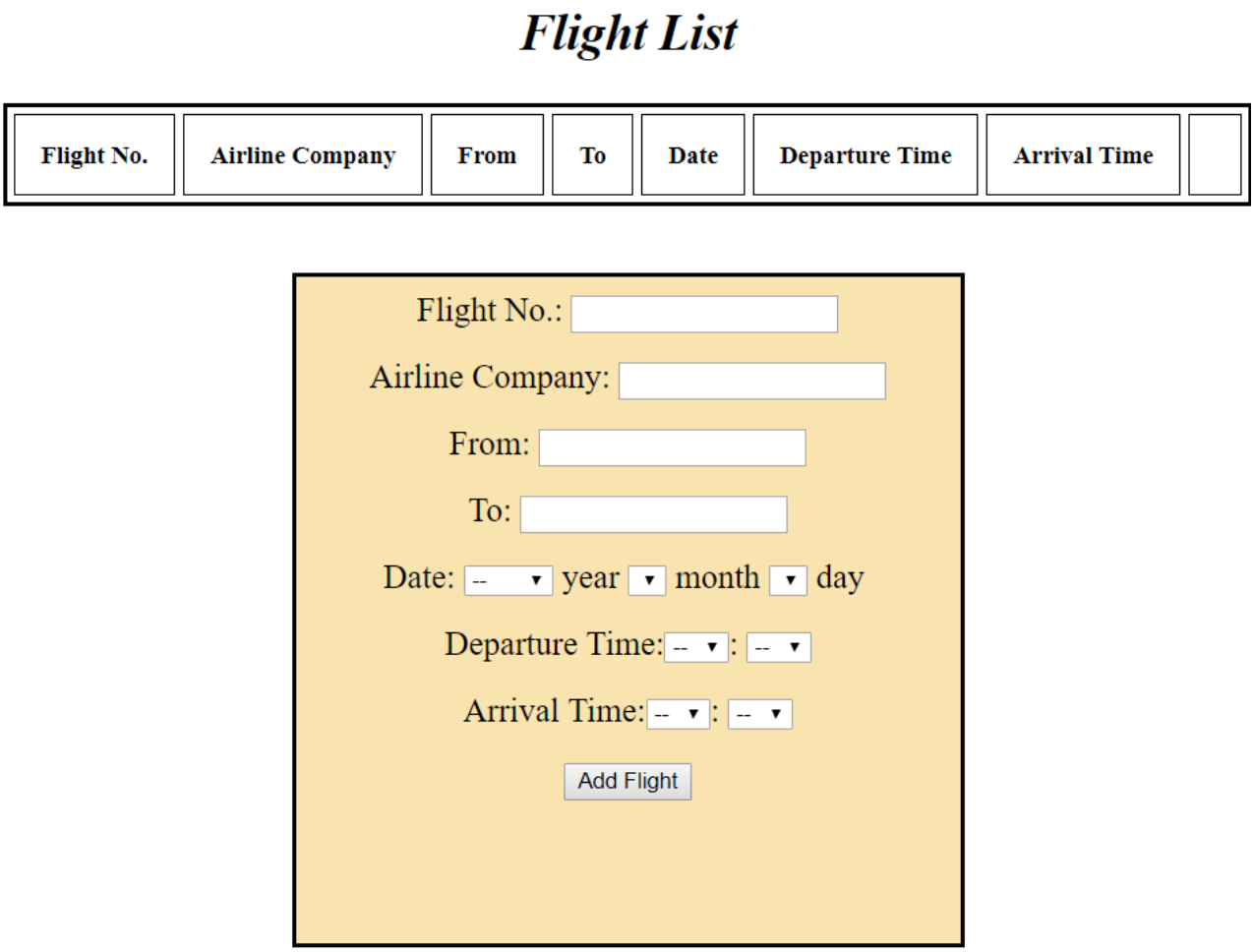
```
<table>
    <caption id="a"><h2>Flight List</h2></caption>
    <tr>
        <th>Flight No.</th>
        <th>Airline Company</th>
        <th>From</th>
        <th>To</th>
        <th>Date</th>
        <th>Departure Time</th>
        <th>Arrival Time</th>
        <th></th>
    </tr>
    <tbody id="tbody" style="display: none">
  <tr>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td>
            <input type="button" value="Delete">
        </td>
    </tr>
    </tbody>
</table>
```

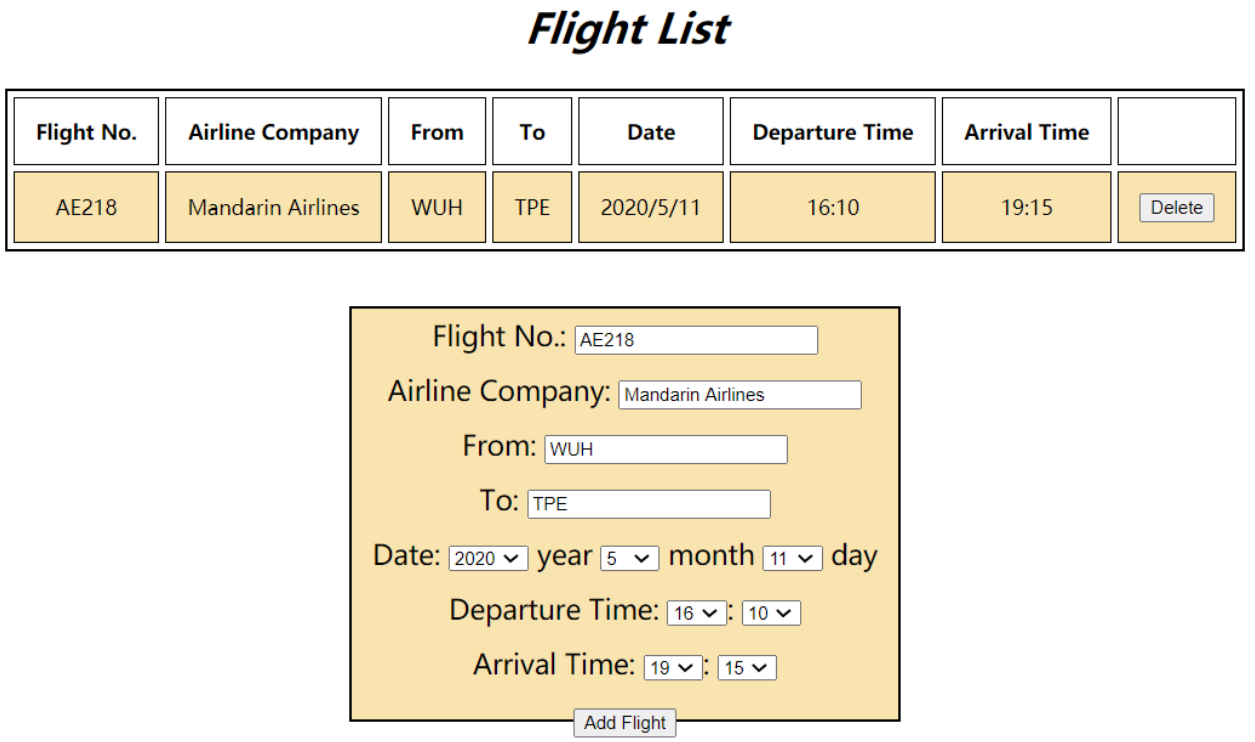
Modify `table.js`, add a function with dynamically add row to table.

```
function addRow() {
    let bodyObj = document.getElementById("tbody");
    if (!bodyObj) {
        alert("Body of Table not Exist!");
        return;
    }
    let year = document.getElementById("year").value;
  let month = document.getElementById("month").value;
    let day = document.getElementById("day").value;
    let dhour = document.getElementById("dhour").value;
    let dminute = document.getElementById("dminute").value;
    let ahour = document.getElementById("ahour").value;
    let aminute = document.getElementById("aminute").value;
    let rowCount = bodyObj.rows.length;
    let cellCount = bodyObj.rows[0].cells.length;
    bodyObj.style.display = ""; // display the tbody
```

```
    let newRow = bodyObj.insertRow(rowCount++);
    newRow.insertCell(0).innerHTML = document.forms[0]["flight-no"].value;
    newRow.insertCell(1).innerHTML = document.forms[0]["airline-
company"].value;
    newRow.insertCell(2).innerHTML = document.forms[0].from.value;
    newRow.insertCell(3).innerHTML = document.forms[0].to.value;
    newRow.insertCell(4).innerHTML = year + "/" + month + "/" + day;
    newRow.insertCell(5).innerHTML = dhour + ":" + dminute;
    newRow.insertCell(6).innerHTML = ahour + ":" + aminute;
    newRow.insertCell(7).innerHTML = bodyObj.rows[0].cells[cellCount -
1].innerHTML; // copy the "delete" button
    bodyObj.rows[0].style.display = "none"; // hide first row
}
```

Then delete the alert("Success!") in function `OnClickAddFlight`, and add the call to `addRow()` function.





8. Deleting <tr> elements dynamically

Add `removeRow()` function to table.js:

```
function removeRow(inputobj) {
    if (!inputobj) return;
    let parentTD = inputobj.parentNode;
    let parentTR = parentTD.parentNode;
    let parentTBODY = parentTR.parentNode;
    parentTBODY.removeChild(parentTR);
}
```

We use the button itself to find the parent <tr> element (which is the row to be removed). Then delete it in <tbody>.

Add the function to event listener of "delete" button via `onclick` attribute:

```
<input type="button" value="Delete" onclick="removeRow(this)">
```

Note: This approach to add event listener is for demonstration only. Do NOT hardcode event handlers in HTML attributes in real cases.

9. Use hyperlinks to jump across the pages

Create a new HTML document named `myPage.html`, input the following:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>myPage</title>
</head>
<body>
<a href="table.html">
    <h1>click here to enter the flight list</h1>
</a>
</body>
</html>
```

That's a simple and dumb page. If you click the hyperlink, you'll be redirected to the flight list page (table.html)

API document

https://developer.mozilla.org/zh-CN/docs/Web/API

https://www.runoob.com/html/html-tutorial.html

VUE learning guideline

If you want to complete a website project with front-end and back-end, simply knowing HTML/CSS/JS is not enough. We recommend vue framework as your front-end tool.
For novices, it is more appropriate to use vue2.0 to get started. If you already have project experience about vue2.0 , you can try vue3.0.

You can refer to the following suggestions for learning vue。

1. **Understand those basic concepts**:
    - vue instance
    - Commands includes

```
v-text, v-html, {{}}
v-model
v-on
v-if v-else
v-show
v-for
v-bind
```

    - compute
    - watch
    - Component (template, prop...)
    - router
    - Axios

2. **Using vue+elementUI to complete this exercise and homework**

   We recommand to using Vue cli, and we provide a demo for it.

3. **Study and complete a project step by step from a learning site**

Install Vue

1. Install node.js

- Download from here

After finishing installation, try

```
node -v
npm -v
```

You have installed it successfully, if it can return the version number.

- Then change npm mirror to taobao

  Install nrm

  ```
  npm install nrm -g
  ```

  If an error info occurred `rollbackFailedOptional`, you can try

  ```
  npm config set registry http://registry.npm.taobao.org
  ```

- After install nrm, you can try

  ```
  nrm ls
  ```

  Result:

```
  npm ---------- https://registry.npmjs.org/
  yarn --------- https://registry.yarnpkg.com/
  tencent ------ https://mirrors.cloud.tencent.com/npm/
  cnpm --------- https://r.cnpmjs.org/
  taobao ------- https://registry.npmmirror.com/
  npmMirror ---- https://skimdb.npmjs.com/registry/
```

then set taobao as the npm mirror

```
nrm use taobao
```

Test the npm speed

```
nrm test npm
```

2. HTML file import vue scripts

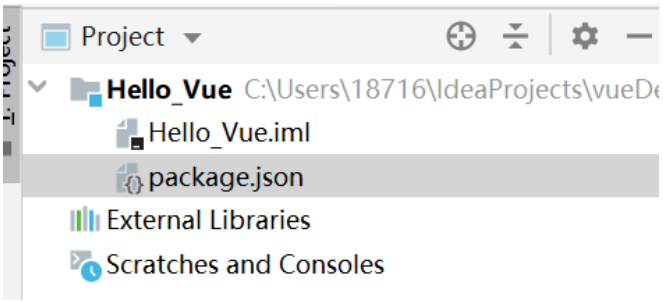
- Open webstorm and create a web project, named Hello_Vue, and then go to the root path of Hello_Vue

```
npm init -y
```

It can returns:

```
{
  "name": "Hello_Vue",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

In project structure, a package.json has been created



- Install vue

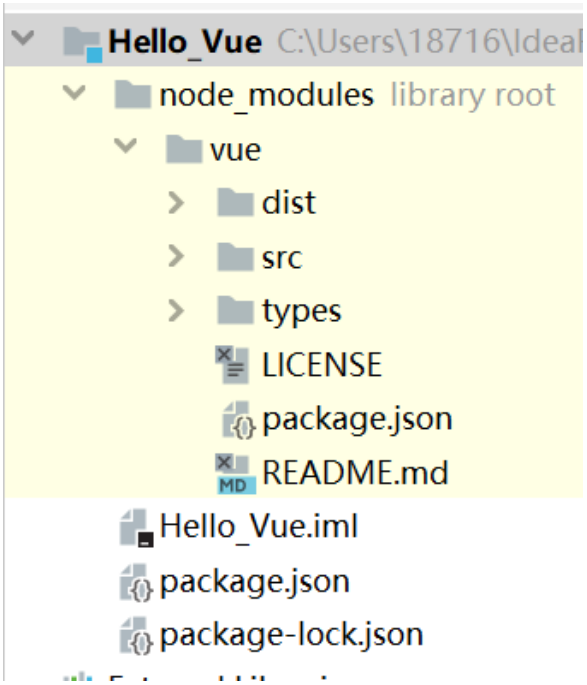
```
npm install vue --save
```

then it returns

```
npm notice created a lockfile as package-lock.json. You should commit this
file.
npm WARN Hello_Vue@1.0.0 No description
npm WARN Hello_Vue@1.0.0 No repository field.

+ vue@2.6.11
added 1 package from 1 contributor in 0.975s
```

After that a document `node_modules` has been created



3. Hello world demo

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <title>Hello world</title>
    <script type="text/javascript" src="../node_modules/vue/dist/vue.js">
</script>
</head>
<body>
<div id="app">
    {{msg}}
</div>
<script>
    var vm = new Vue({
        el:'#app',
        data:{
            msg: 'hello world'
        }
    })
</script>
</body>
</html>
```

4. Build vue cli project

- Install:

  ```
  npm install -g @vue/cli
  ```

- Check stall:

  ```
  vue -V
  ```

  if it returns version number, vue cli has been installed, such as:

  `@vue/cli 4.5.14`

- create a project

  ```
  vue create [project name]
  ```

  it will return

```
Vue CLI v4.5.14
┌─────────────────────────────────────────────┐
│                                             │
│   New version available 4.5.14 → 5.0.8      │
│     Run npm i -g @vue/cli to update!        │
│                                             │
└─────────────────────────────────────────────┘

? Please pick a preset: (Use arrow keys)
❯ proj ([Vue 3] dart-sass, babel, router, vuex, eslint)
  Default ([Vue 2] babel, eslint)
  Default (Vue 3) ([Vue 3] babel, eslint)
  Manually select features
```

- select vue2 or vue3 as your project module.

  If we select vue2, then it returns:

```
Vue CLI v4.5.14
┌─────────────────────────────────────────────┐
│                                             │
│   New version available 4.5.14 → 5.0.8      │
│     Run npm i -g @vue/cli to update!        │
│                                             │
└─────────────────────────────────────────────┘

? Please pick a preset: Default ([Vue 2] babel, eslint)
? Pick the package manager to use when installing dependencies: (Use arrow
keys)

❯ Use Yarn
  Use NPM
```

- Select package manager to use.

  If we select NPM, it will returns:

```
Vue CLI v4.5.14
✨  Creating project in /Users/yuemingzhu/vue-cli-project.
🗃  Initializing git repository...
⚙️  Installing CLI plugins. This might take a while...


added 1328 packages in 43s
🚀  Invoking generators...
📦  Installing additional dependencies...
```

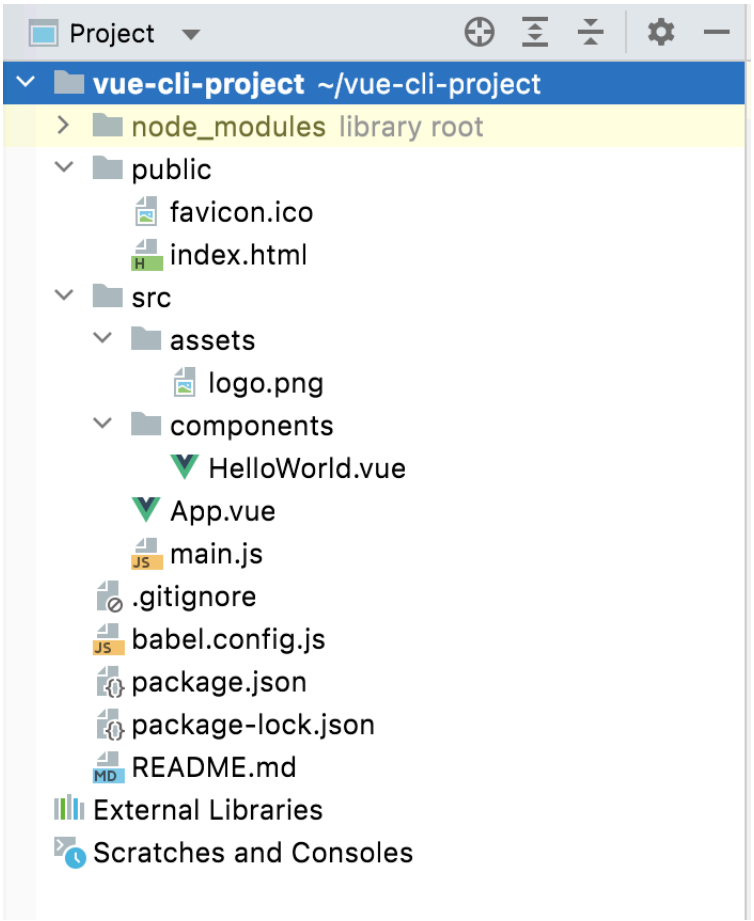
```
added 61 packages in 6s
⚓  Running completion hooks...

📄  Generating README.md...

🎉  Successfully created project vue-cli-project.
👉  Get started with the following commands:

 $ cd vue-cli-project
 $ npm run serve
```

- open it by webstrom



- Run it

  Click package.json -> click this button

Command will returns:

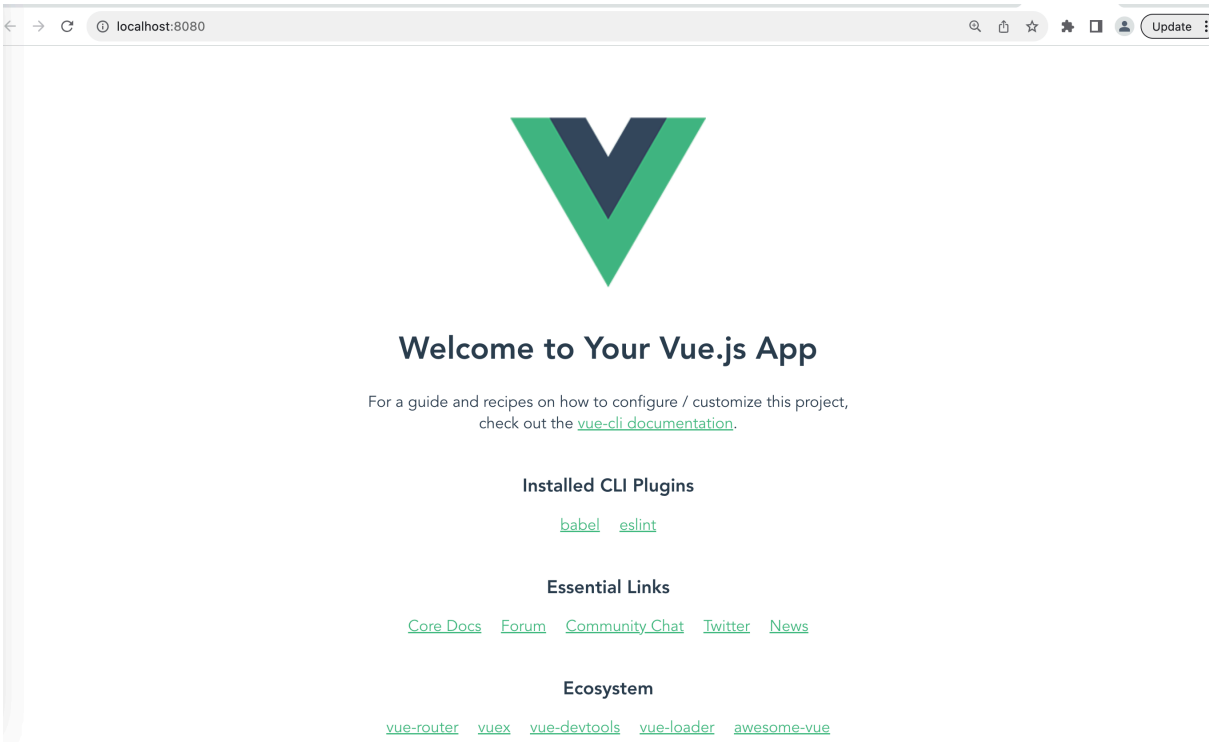
```
DONE  Compiled successfully in 5980ms
                                                5:49:38 PM


 App running at:
 - Local:   http://localhost:8080/
 - Network: http://192.168.1.4:8080/

 Note that the development build is not optimized.
 To create a production build, run npm run build.
```

Then Click http://localhost:8080/

It will return


localhost:8080
Update
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