Верстка HTML
Скрипты JS

Машалов Никита

Аспирнат

Инновационная педагогика

План

- Устройство пакета HTTP;
- HTML элементы и DOM-дерево;
- Событийная модель браузера JS;

HTTP

- Текстовый протокол;
- Поверх TCP соединения;
- Работает как запрос/ответ;
- Вверху заголовки, внизу содержание (body).

```
GET /api/v1/data HTTP/1.1
Host: api.example.com
Accept: application/json
Connection: close
```

```
HTTP/1.1 200 OK
Date: Mon, 01 May 2023 13:15:20 GMT
Content-Type: application/json
Content-Length: 47
Connection: close

{
  "data": [
    {
      "id": 1,
      "name": "Item 1"
    },
    {
      "id": 2,
      "name": "Item 2"
    }
  ]
}
```

HTTP клиент

- Method
  - GET используется для получения данных (используется бразуером для получения вебстранички);
  - POST для отправки данных для обработки.
- Host – имя сервиса (соединение устанавливается с физической машиной, которая может обслуживать много веб-страниц).
- Path – название вашего метода. Структурируется как в файловой системе.
- Connection – нужно ли обрывать tcp соединение после ответа.

```
1 GET /api/v1/data HTTP/1.1
2 Host: api.example.com
3 Accept: application/json
4 Connection: close
```

HTTP сервер

- Status code
  - 200 – OK;
  - 400 – вы отправили неправильный запрос;
  - 500 – ошибся в сервер.
- Content-Length – длина body поля (body может не влезть в кадр TCP, не всегда понятно стоит ли его ожидать);
- Body – набор байтов через двойной отступ после заголовков.

```
HTTP/1.1 200 OK
Date: Mon, 01 May 2023 13:15:20 GMT
Content-Type: application/json
Content-Length: 47
Connection: close

{
  "data": [
    {
      "id": 1,
      "name": "Item 1"
    },
    {
      "id": 2,
      "name": "Item 2"
    }
  ]
}
```

HTML - документ

- Файл с версткой веб-страницы;
- Содержит заголовок и нагрузку;
- Верстка выполняется с помощью html элементов, заключенных в скобки <>;
- HTML задает каркас страницы, анимация и оформление выполняются каскадными стилями CSS и скриптами JS.

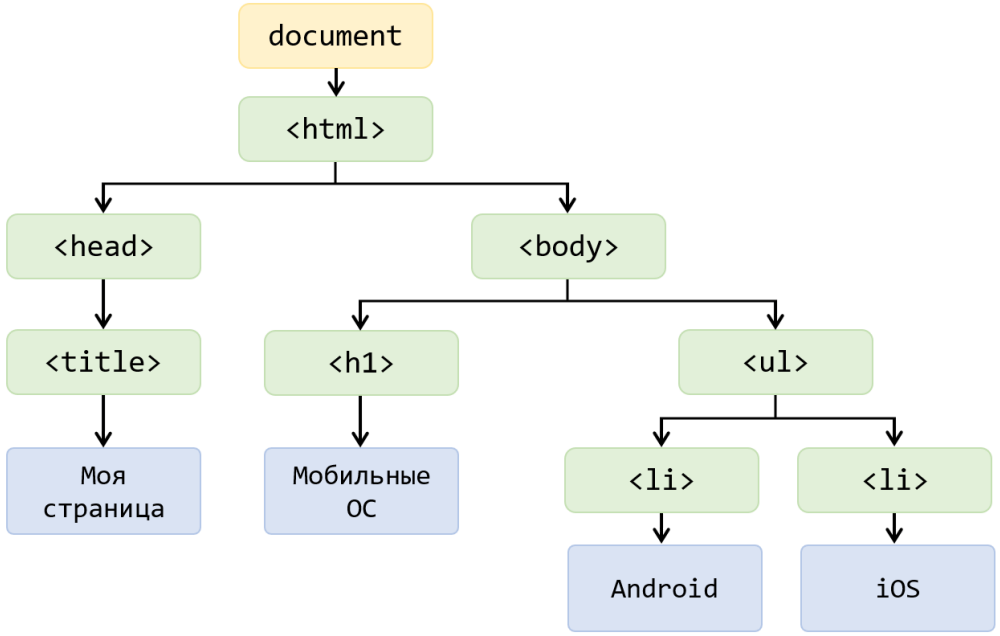
```
<html>
<head>
<title> Simple HTML Document </title>
</head>

<body>

This is a very simple HTML document.

</body>
</html>
```

HTML элементы

- <btn> - тэг кнопки;
- <btn class=“btn”> - аттрибут элемента с названием class и значением btn;
- <btn></btn> - закрывающий тэг (в нем писать аттрибуты не нужно);
- <btn>Старт</btn> - содержание (innerHTML) элемента. В данном случае текст Старт;
- Можно создавать свои собственный элементы, используя WebComponents.

DOM (document object model)

- HTML упорядочен иерархически в виде древа;
- Эффекты родителя, как сжатие, буду распространяться на дочерние узлы;
- Все алгоритмы поиска и вставки будут использовать структуру DOM-дерева;
- Язык JS позволяет вставлять новые узлы динамически.



JavaSript (Js)

- Похож на Python. Интерпретируем, имеет переменные, классы, функции и тд;
- Имеет функционал для взаимодействия с браузером WebAPI для модификации Dom-дерева;
- Обеспечивает безопасность пользователя, использующего ваш скрипт, ограничивая веб-запросы и работу с файловой системой;
- Имеет продвинутую событийную модель, позволяющую регистрировать обработчики на взаимодействие пользователя с браузером.

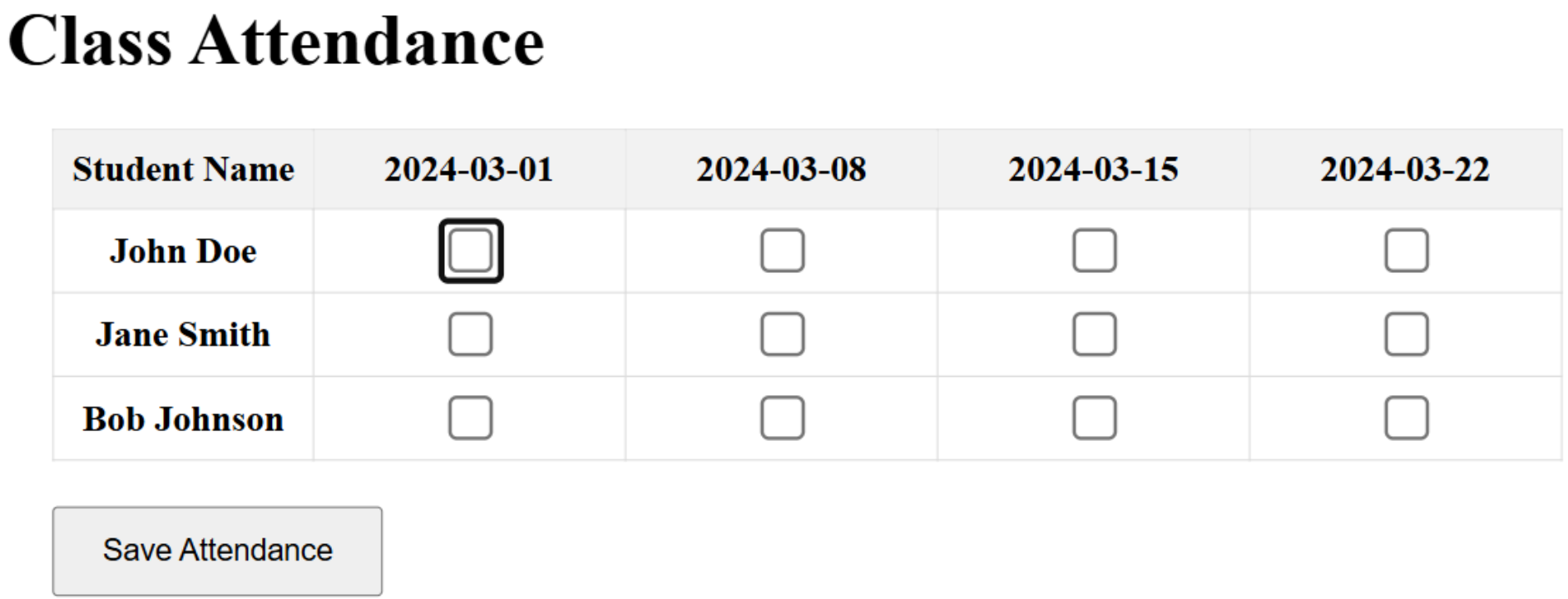
Событийная модель

- Движок браузера постоянно опрашивает элементы DOM-дерева об изменении их состояния (event loop).
- Если событие произошло, то происходит его диспетчиризация в слушателей
- Как правило слушатели это функции (callback), который принимают переменную события и обрабатывают согласно вашей логики

```
document.querySelector('.button').addEventListener('
click', () => { alert('Button clicked!'); });
```

Практика

Макет приложения



Обсуждение: зачем своя таблица посещаемости или журнал записи

- *Приватность.* Другие участники не должны видеть вашу посещаемость.
- *Оповещение.* Регистрация оповещений в телеграм куратору курса, если ученик не появляется.
- *Интеграция.* Ваша таблица может быть встроена в систему LMS (learning management system)

Наивный подход

- Для каждого посещения ученика создаем клетку
- Клетки однотипны
- При добавлении даты/ученика, придется каждый раз писать все заново
- Нужна динамическая генерация по студентам и датам

```
<table>
    <tr class="fixed-header">
        <th>Student Name</th>
        <th class="header-date">2024-03-01</th>
        <th class="header-date">2024-03-08</th>
        <th class="header-date">2024-03-15</th>
        <th class="header-date">2024-03-22</th>
    </tr>

    <tr>
        <td class="student-name">John Doe</td>
        <td><input type="checkbox"></td>
        <td><input type="checkbox"></td>
        <td><input type="checkbox"></td>
        <td><input type="checkbox"></td>
    </tr>
```

WebComponents (Lit)

- Используем “обертку” над API web component - Lit
- Его язык шаблонизации позволяет компактно описать повторяющие зависимости;
- Регистрируем новый элемент используя customElements
- Используем новый элемент в <body>

```
${this.students.map(student => html`
    <tr>
        <td class="student-name">${student}</td>
        ${this.dates.map(() => html`
            <td><input type="checkbox"></td>
        `)}
    </tr>
`)}
```

```
customElements.define(
    'attendance-table',
    AttendanceTable
);
```

```
<body>
    <attendance-table></attendance-table>
</body>
```

Домашнее задание

Домашнее задание

Описано на Stepik:

- установить все необходимое ПО для разработки;

- пройти тесты по html и js;

- выполнить задание в репозитории.