МШП 20.35 УНИВЕРСИТЕТ КОД БУДУЩЕГО Минцифры России НАЦИОНАЛЬНЫЕ ПРОЕКТЫ РОССИИ ЦИФРОВАЯ ЭКОНОМИКА

Fullstack-разработка

NodeJS
Формы
GET-запросы

МШП

Цель урока:

Научимся отправлять данные на сервер через поля ввода

МШП

Разбор ДЗ

А было ли оно?)

МШП

Теория

Формы

МШП

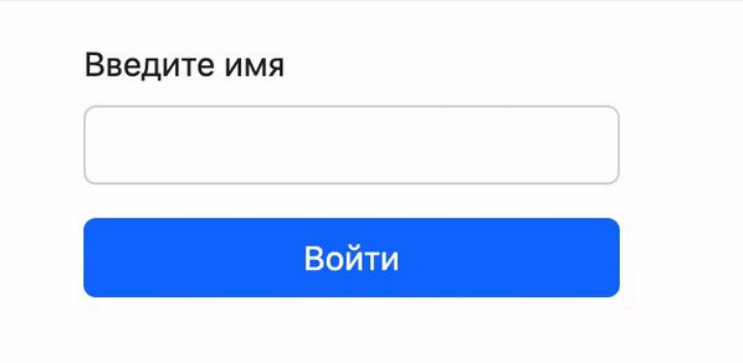
Параметры запроса

```
<a href="/news?topic=js">
    Новости о JS
</a>
```

До сих пор мы передавали данные на сервер через **тег <a> и query-параметры**

МШП

Настало время улучшить работу нашего сервера, добавив **пользовательский ввод**



МШП

Раньше

На клиенте мы уже работали с полями ввода:

```
<input type="text" id="username">

<button type="button" id="click-me">
  Отправить
</button>
```

Пользователь вводит значение через input, затем нажимает на button.

Дальше срабатывает клиентский JS код.

МШП

Атрибут id

Для работы сервера **атрибут id**
не играет никакой роли.
Поэтому его можно смело удалять!

```
<input type="text" id="username">

<button type="button" id="click_me">
  Отправить
</button>
```

МШП

Атрибут name

Вместо него появляется новый атрибут **name**, который обязательно нужно добавить каждому полю ввода.

```
<input type="text" name="username">

<button type="button">
  Отправить
</button>
```

Зачем? Узнаем немного позже...

МШП

Кнопка submit

Следующее отличие – у кнопки будет **новый тип.**

Тип **submit** наделяет кнопку суперсилой запускать отправку данных на сервер.

```
<input type="text" name="username">

<button type="submit">
  Отправить
</button>
```

МШП

Тег form

Тег **form** – последний ингредиент заклинания.

Он группирует поля ввода и кнопку,
а также задаёт **путь** для отправки данных.

```
<form action="/login">
    <input type="text" name="username">
    <input type="password" name="password">
    <button type="submit">
        Отправить
    </button>
</form>
```

МШП

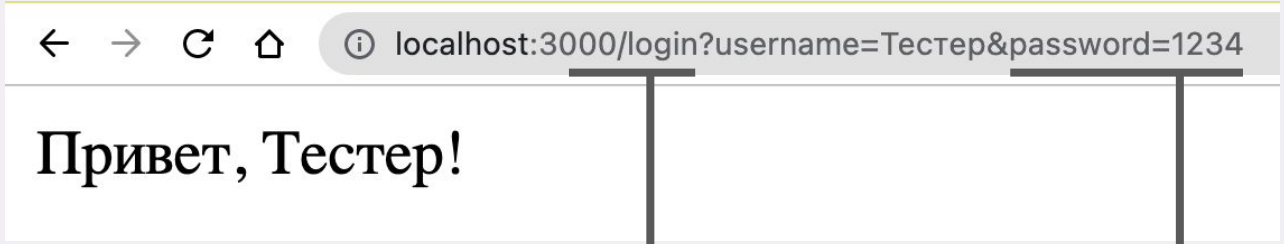
Как это работает

Пусть вёрстка, которую мы сейчас обсудили, хранится на сервере в шаблоне **form.hbs**, который открывается через роут **/login**

Пользователь вводит свои данные и нажимает "Войти"...

МШП

Как это работает



Тег **form** отправляет запрос на сервер
по пути из атрибута **action, а это опять /login**

Каждое поле ввода превращается в **query**-
параметр с названием
как в атрибуте **name**.

```
<input type="password" name="password">
```

МШП

Важен каждый пункт!

1. **form** с атрибутом **action**
2. **input** с атрибутом **name**
3. **button** с типом **submit**

GET-запрос с параметрами

/login?username=tester&password=1234

МШП

Теория

Обработка форм

МШП

Пример 1

Сервер проверяет, есть ли **query** параметры, и показывает либо **форму**, либо **ответ**.

```
app.get(`/login`, function (res, req) {
   let username = res.query.username;

   if (!username) {
       res.render(`form`);
   } else {
       res.send(`Привет, ${username}`);
   }
})
```

МШП

Пример 2

Иногда требуется много условий и даже несколько разных шаблонов для ответов.

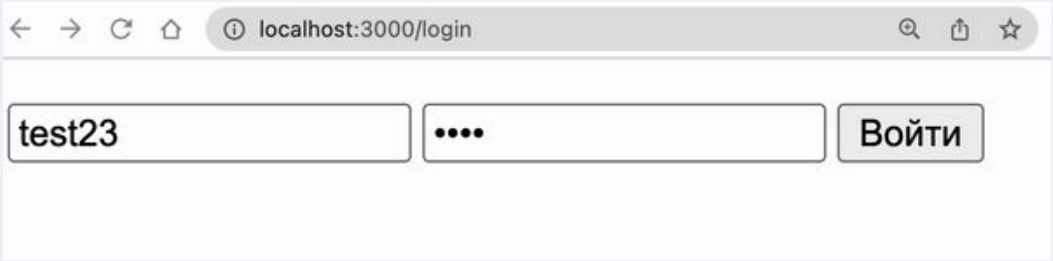
```
app.get(`/login`, function (res, req) {
    let username = res.query.username;
    let password = res.query.password;

    if (username == `test` && password == `1234`) {
        res.render(`welcome`);
    } else if (!username || !password) {
        res.render(`form`);
    } else {
        res.render(`error`);
    }
})
```

МШП

Пример 2

Иногда требуется много условий и даже несколько разных шаблонов для ответов.



МШП

Пример 3

```
app.get(`/login`, function (res, req) {
    let username = res.query.username;
    let password = res.query.password;
    let text = ``;

    if (username == `test` && password == `1234`) {
        text = `Добро пожаловать!`;
    } else if (!username || !password) {
        text = `Введите данные`;
    } else {
        text = `Данные неверны`;
    }

    res.render(`login`, {text: text});
})
```

МШП

Пример 3

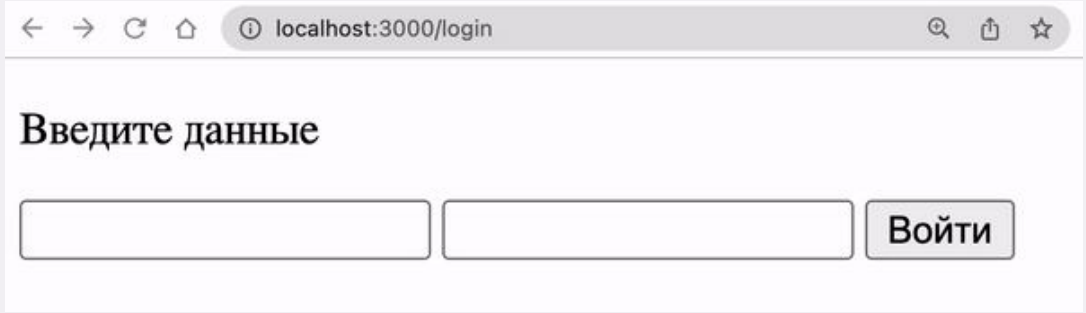
```
<p>{{text}}</p>

<form action="/login">
    <input type="text" name="username">
    <input type="password" name="password">
    <button type="submit">Войти</button>
</form>
```

В этом случае сообщение с ответом сервера подставляется в тот же шаблон, в котором находится форма.

МШП

Пример 3



Этот приём помогает создавать иллюзию "обновления" страницы в ответ на действия пользователя.

МШП

Практика

Простые формы

МШП

Теория

Поля ввода

МШП

Мы уже много работали с полями ввода в прошлом модуле.

Правда, они были стилизованы через Bootstrap.

В реальности они выглядят вот так...

МШП

Поля ввода

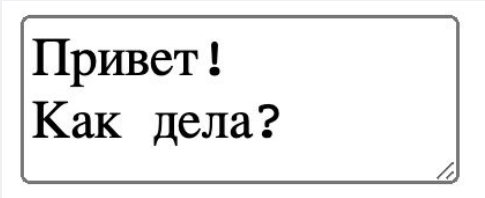
Однострочное:

```
<input type="text" name="username">
```

?username=Тестер

Многострочное:

```
<textarea name="message"></textarea>
```



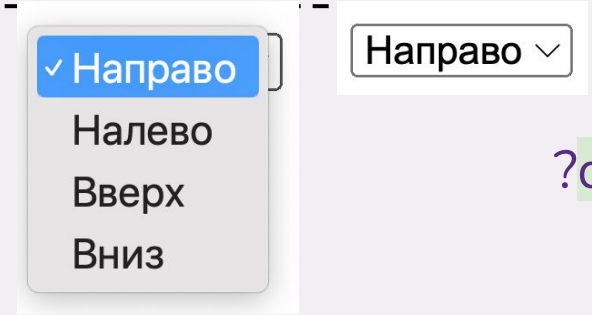
?message=Привет%21%0D%0AКак+дела%3F

МШП

Поля ввода

С выбором:

```
<select name="direction">
  <option value="right">Направо</option>
  <option value="left">Налево</option>
  <option value="up">Вверх</option>
  <option value="down">Вниз</option>
</select>
```



?direction=right

МШП

Поля ввода

Чекбокс отличается от других полей ввода:

```
<input type="checkbox" name="agree">
```

Если чекбокс отметить, подставить значение **on**

?agree=on

Если чекбокс не отметить, параметр не передаётся

?

МШП

Итак, все поля имеют атрибут name и передают данные на сервер через query-параметры.

Давайте используем их на практике

МШП

Практика

Формы посложнее

МШП

Цель урока:

✓ Научились отправлять данные на сервер через поля ввода

МШП

Домашнее задание