

Кафедра информационных систем и технологий (№ 42)

Дисциплина «Web-программирование»
Автор – ст. преподаватель каф. 42 В.А. Ушаков

Лабораторная работа № 2

ОБРАБОТКА ДАННЫХ HTML-ФОРМ В NODE.JS

Цель работы – Получение навыков программирования на языке программирования JavaScript в среде Node.JS. Изучение возможностей получения данных из HTML-формы и их обработки в среде Node.JS.

Краткие методические сведения

Одним из основных способов передачи данных web-сайту является обработка HTML-форм. Формы представляют специальные элементы разметки HTML, которые содержат в себе различные элементы ввода – текстовые поля, кнопки и т.д. И с помощью данных форм можно ввести некоторые данные пользователя и отправить их на сервер. А сервер уже обрабатывает эти данные.

Создание HTML-форм состоит из следующих аспектов:

- создание элемента `<form>` в разметке HTML;
- добавление в этот элемент одного или нескольких полей ввода;
- установка метода передачи данных: GET или POST;
- установка адреса, на который будут отправляться введенные данные.

Методы передачи данных GET или POST

GET – это данные, которые передаются в адресной строке браузера, например, когда пользователь нажимает на ссылку.

POST – это данные, которые передаются после нажатия пользователем кнопки в HTML-форме.

Чтобы передать данные методом **GET** не надо создавать в HTML-странице HTML-форму – достаточно ссылки на документ с добавлением строки запроса, которая может выглядеть как «переменная=значение». Пары объединяются с помощью амперсанта «&», а к URL страницы строка присоединяется с помощью вопросительного знака «?». Преимущество передачи параметров таким способом заключается в том, что клиенты, которые не могут использовать метод **POST** (например, поисковые машины), все же смогут, просто пройдя по ссылке, передать параметры скрипту и получить содержимое. Поэтому метод **GET** следует использовать:

- для доступа к общедоступным страницам с передачей параметров (повышение функциональности);
- для передачи информации, не влияющей на уровень безопасности.

Недостаток же этого метода заключается в том, что просто изменив параметры в адресной строке, пользователь может изменить ход сценария непредсказуемым образом, что в свою очередь создает «дыру» в безопасности. Поэтому метод **GET** не следует использовать:

- для доступа к защищенным страницам с передачей параметров;
- для передачи информации, влияющей на уровень безопасности;
- для передачи информации, не подлежащей модифицированию пользователем.

Передать данные методом **POST** можно только с помощью HTML-формы на web-странице. Основное отличие **POST** от GET в том, что данные передаются не в заголовке запроса, а в теле запроса, следовательно, пользователь их не видит. Модифицировать данные возможно только изменив саму форму. **Преимущество** передачи данных методом POST заключается в большей безопасности данных и функциональности запросов с помощью HTML-форм в сравнении с методом GET. Поэтому метод **POST** следует использовать:

- для передачи большого объема информации (текст, файлы, и т.д.);
- для передачи важной информации;
- для ограничения доступа.

Какой способ следует применять?

- Если HTML-форма служит для запроса информации, например – при поиске, то ее следует отправлять методом **GET**. Чтобы можно было обновлять страницу, можно было поставить закладку и/или послать ссылку другому пользователю.
- Если в результате отправки HTML-формы данные записываются или изменяются на сервере, то следует их отправлять методом **POST**. Также, **POST** может понадобиться, если на сервер надо передать большой объём данных (у GET объём передаваемых данных сильно ограничен), а также, если не следует «светить» передаваемые данные в адресной строке (при вводе логина и пароля, например).

Задание

- 1 Создать папку с проектом, содержащую страницу с HTML-формой, каскадной таблицей стилей (CSS) и JS-скриптом в среде Node.JS.
 - 2 Разработать web-страницу, которая должна предлагать пользователю HTML-форму, включающую различные элементы (текстовые поля, списки, кнопки и т.д.). HTML-форма должна содержать не менее 10 различных элементов. HTML-форма должна иметь дизайн, разработанный средствами CSS. Тематика, для которой создается форма, определяется в соответствии с приложением А. HTML-форма должна иметь обязательные поля (input type="text") в соответствии с разделом «Варианты для лабораторных работ».
- РЕКОМЕНДУЕТСЯ** использовать форму из Лабораторной работы № 5 «Создание форм в web-документах» по дисциплине «Web-технологии».

Таблица 1 – Тематика HTML-формы

№ варианта	Тематика HTML-формы
1	Дистанционное обучение
2	Электронный магазин бытовой техники
3	Электронный магазин компьютерной техники
4	Фитнес-клуб
5	Бронирование мест в гостинице
6	Туристическое агентство
7	Агентство недвижимости (продажа объектов)
8	Агентство недвижимости (аренда объектов)
9	Заказ билетов в театр
10	Заказ билетов на самолет (поезд)

3 Разработать JS-скрипт в среде Node.JS, который получает данные из HTML-формы методом POST и, в зависимости от полученного содержимого, формирует новую страницу. Итоговая страница должна содержать текстовые и графические элементы, связанные с результатом заполненной HTML-формы. **Количество элементов новой страницы должно быть не меньше количества полей в HTML-форме.** Для поля типа radio или checkbox предусмотреть возможность изменения минимум одного изображения в HTML-форме в зависимости от выбранного пользователем значения поля.

4 JS-скрипт должен проверить полноту введенных пользователем данных и корректность их ввода. Также полученная с помощью JS-скрипта страница должна иметь разработанный дизайн средствами CSS.

Примечания к выполнению лабораторных работ:

В ходе выполнения ЛР следует использовать HTML5, CSS3 (в частности, блочную верстку или *FlexBox*) и среда Node.JS 17 или новее.

В ходе выполнения ЛР следует выносить код каскадной таблицы стилей и сценариев в файлы *.css и *.js соответственно. Код скриптов должен располагаться в файлах *.js. *Не следует использовать Javascript, JQuery и другие технологии для тех задач, которые можно решить в среде Node.JS.*

Рекомендуется использовать относительные пути при создании *.html-, *.css-, *.js- файлов для обращения к рисункам, файлам *.css и т.д.

Содержание отчета

- 1 Титульный лист
- 2 Цель работы
- 3 Задание к лабораторной работе
- 4 Код web-страниц (*.html и *.css) и/или скриптов (*.js)

Листинг кода должен содержать комментарии

- 5 Примеры web-страниц
- 6 Выводы по лабораторной работе

Требования к оформлению отчета о лабораторной работе

При оформлении отчета о лабораторной работе следует пользоваться ГОСТ 7.32-2017 издания 2017 года. Правила оформления текстовых документов по ГОСТ 7.32-2017, а также титульные листы лабораторных работ представлены на сайте ГУАП (<https://guap.ru/regdocs/docs/uch#rules>).

Рекомендуемые источники информации:

- 1 <https://nodejsdev.ru/>
- 2 <https://nodejs.org/ru/docs/guides>
- 3 Сухов К.К. Node.js. Путеводитель по технологии. – М.: ДМК Пресс, 2015. – 416 с.
- 4 Кантелон М., Хантер М., Головайчук Т., Райлих Н. Node.js в действии. – СПб.: Питер, 2014. – 548 с.
- 5 Пауэрс Ш. Изучаем Node. Переходим на сторону сервера. СПб.: Питер, 2017. – 304 с.