Титульный лист материалов по дисциплине

ДИСЦИПЛИНА **Разработка клиент-серверных приложений**

полное название дисциплины без аббревиатуры

ИНСТИТУТ **Информационных технологий**

КАФЕДРА **Инструментального и прикладного программного**

полное название кафедры **обеспечения**

ГРУППА/Ы **ИКБО-01/02/03/12/13/16-18**

номер групп/ы, для которых предназначены материалы

ВИД УЧЕБНОГО МАТЕРИАЛА **Материал к практическим занятиям**

лекция; материал к практическим занятиям; контрольно-измерительные материалы к практическим занятиям; руководство к КР/КП, практикам

ПРЕПОДАВАТЕЛЬ **Строганкова Наталья Владимировна**

фамилия, имя, отчество

СЕМЕСТР **5**

указать номер семестра обучения

ПРАКТИЧЕСКАЯ РАБОТА № 3

1 Цель работы

Целью данной практической работы является знакомство с современными подходами создания клиентской части клиент-серверных приложений и разработка простого web-сайта, основанного на наработках предыдущей практической работы.

2 Теоретическая часть

2.1 Создание клиентской части

В учебно-методическом пособии кратко рассказано о том, с помощью каких технологий возможно оформлять клиентскую часть. В данном случае мы будем рассматривать процесс взаимодействия разных технологий на примерах HTML, CSS, JavaScript, JQuery, AJAX. HTML — это сам документ, а CSS и JavaScript — вспомогательные аппараты для упрощения оформления и интерактивности страницы. AJAX же позволяет обмениваться информацией с сервером асинхронными запросами.

Объединяет все эти технологии HTML, то есть ссылки на них (css-файлы и js-файлы) будут находиться в самом документе в виде следующих строк:

```
<script type="text/javascript" src="/js/example.js"></script>
<link rel="stylesheet" type="text/css" href="/css/example.css">
```

В фалах с расширением .js обычно содержится код JavaScript и элементы AJAX и JQuery.

В файлах с расширением .css содержатся таблицы каскадных стилей.

Приведем простой пример применения этих технологий на практике.

Пример HTML-файла:

```
1.  <html>
2.  <head>
3.  <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
```

```
4.  <script type="text/javascript" src="jquery-
1.6.2.min.js"></script>
5.  <script type="text/javascript" src="example.js"></script>
6.  <link rel="stylesheet" type="text/css" href="example.css">
7.  <title>HardSite</title>
8.  </head>
9.  <body>
10. <input type="text" id="text" name="text" class="panda-
searchfield">
11. <div class="b-top"><input type="submit" value="Перейти"
onclick="sendTextJQuery()" class="b-top-but"></div>
12. <div class="b-top"><input type="submit" value="Перейти"
onclick="sendText()" class="b-top-but"></div>
13. </body>
14. </html>
```

На 4–5 строках листинга представлено подключение js-файлов. Один из них — это библиотека JQuery, другой — это созданный файл. Строка 6 содержит подключение файла — таблицы каскадных стилей. Изначально, когда обращаются в адресной строке, к определенному сайту, отправляется HTTP GET-запрос, после принятия от сервера HTML-документа браузер смотрит на информацию, отмеченную в теге <head>, там обычно указывается служебная информация, по ней браузер понимает, какие еще файлы необходимо запросить от сервера, и будет поочередно отправлять запросы на сервер, требуя сначала файл jquery-1.6.2.min.js, затем example.js и заканчивая файлом каскадных стилей example.css. Изображения, которые могут использоваться в HTML-документе, также будут запрашиваться у сервера отдельными запросами.

2.2 Таблицы каскадных стилей (CSS)

Существует три способа добавления стилей в документ:

1) Внутренние стили определяются атрибутом style в тегах.

Пример:

```
<p style = "color: blue">Абзац с текстом синего цвета</p>
```

2) Глобальные стили располагаются в контейнере <style>…</style>, который в свою очередь находится в теге <head>…</head>.

Пример:

```
1.  <html>
2.  <head>
3.  … ….
4.  <style type="text/css">
5.  p {color:#808080;}
6.  </style>
7.  </head>
8.  <body>
9.  <p>Серый цвет текста во всех абзацах Web-страницы</p> 10. <p>Серый цвет текста во всех абзацах Web-страницы</p>
10. </body>
11. </html>
```

3) Внешние стили содержатся в отдельном файле с расширением .css. Внешние стили позволяют всем страницам сайта использовать одни и те же стили, тем самым позволяя страницам выглядеть единообразно.

Для связи с файлом стилей используется тег <link>, расположенный в теге <head>…</head>. В нем задается два атрибута: rel=«stylesheet» и href, определяющий адрес файла стилей.

Пример:

```
1. <html>
2. <head>
3.  … ….
4.  <link rel="stylesheet" href="style.css">
5.  … ….
6.  </head> 7. <body>
8.  … ….
9.  </body>
```

```
10. </html>
```

2.2.1 Подключение стилей

Правило подключения глобальных и внешних стилей состоит из селектора и объявлений стиля.

Селектор, расположенный в левой части правила, определяет элемент (элементы), для которых установлено правило. Далее в фигурных скобках перечисляются объявления стиля, разделенные точкой с запятой.

Пример:

```
1. p {
2. text-indent: 30px;
3. font-size: 14px;
4. color: #666;
5. }
```

Объявление стиля — это пара «свойство CSS: значение CSS».

Пример: `color: red`

color – свойство CSS, определяющее цвет текста;

red – значение CSS, определяющее красный цвет.

Существует несколько типов селекторов: селекторы CSS, селекторы тегов, селекторы идентификаторов и селекторы классов.

Селекторы тегов

В качестве селектора может выступать любой HTML-тег, для которого определяются правила стилевого оформления.

Пример: `h1 {color: red; text-align: center;}`

Селекторы идентификаторов

HTML предоставляет возможность присвоить уникальный идентификатор любому тегу. Идентификатор задается атрибутом id:

```
<div id="a1">…</div>
```

В CSS-коде селектор идентификатора обозначается знаком # (хэштег). Так как идентификатор id применяется только к уникальным элементам, название тега перед знаком # обычно опускают:

```
#a1 {color: green;}
```

Селекторы классов

Для стилевого оформления чаще всего используются селекторы классов. Класс для тега задается атрибутом class:

```
<div class="c1">…</div>
```

Если атрибут id применяется для уникальной идентификации, то при помощи атрибута class тег относят к той или иной группе.

В CSS-коде селектор класса обозначается знаком «точка» (.). Разные теги можно отнести к одному классу. В таком случае имя тега перед знаком «точка» (.) опускают:

```
i.green {color: #008000;}
b.red {color: #f00;}
.blue {color: #00f;}
```

Для тега можно одновременно указать несколько классов, перечисляя их в атрибуте class через пробел. В этом случае к элементу применяются стили каждого из указанных классов.

```
<div class="left w100">…</div>
```

Если некоторые из этих классов содержат одинаковые свойства стиля, но с разными значениями, то будут применены значения стиля класса, который в CSS-коде расположен ниже.

Селекторы CSS

Имеется набор стандартных селекторов, предоставляемый css. Например: E: focus; E: hover и т.д. Также он позволяет воспринимать структуры вида E#myid и E.myclass, которые позволяют более точно настроить стили.

E#myid позволяет применить стили к элементу с id равным myid, который обязательно должен находиться в теге E.

E.myclass позволяет применить стили к элементам класса myclass, но только те, которые находятся в теге E.

2.3 Отправка HTTP-запросов с помощью языка JavaScript

JavaScript позволяет разработчикам:

– изменять страницу, писать на ней текст, добавлять и удалять теги, менять стили элементов;

– реагировать на события: скрипт может ждать, когда что-нибудь случится (клик мыши, окончание загрузки страницы) и реагировать на это выполнением функции;

– выполнять запросы к серверу и загружать данные без перезагрузки страницы;

– устанавливать и считывать cookie, валидировать данные, выводить сообщения и многое другое.

Основное внимание в нашей работе уделяется взаимодействию клиента с сервером, в связи с этим разберем пример отправки и получения данных серверу.

В данном примере будет показано, как можно отправить POST-запрос на сервер с помощью JavaScript.

Пример использования объекта XMLHttpRequest:

```
1.  function sendText () {
2.  var text = document.getElementById("text").value;
3.  var xmlhttp = new XMLHttpRequest();
4.  xmlhttp.open ('POST', '/xhr/test.html', false);
5.  xmlhttp.send(text);
6.  xmlhttp.onreadystatechange = function(e) {
7.  if (this.readyState == 4 && this.status == 200) {
8.  …
9.  }
10. }
11. }
```

Пример использования объекта XMLHttpRequest:

```
1.  function sendTextJQuery ()
2.  {
3.  var text = $ ("#text").val ();
4.  $.ajax ({
5.  url: "/form.php",
6.  type: "post",//тип запроса
7.  data: text//отправляемые данные, в данный момент это простой
текст
8.  success: function (data, textStatus) {
9.  …
10.  }
11.  });
12.  }
```

Обращение к объектам документа можно производить через команду document.getElementById, при этом будет возвращаться тот элемент, которому соответствует указанный идентификатор, и в дальнейшем можно при желании извлечь из него данные для последующей обработки.

В листинге показаны способы отправления запросов с данными на сервер с помощью компонента XMLHttpRequest и AJAX.

Функция sendText осуществляет общение с сервером через объект XMLHttpRequest. Для того чтобы с ней работать, необходимо описать с кем и как необходимо соединиться, это позволяет сделать метод open с входными параметрами: тип запроса, url, асинхронной или синхронный тип запроса. После чего нужно послать сформированный HTTP-запрос. Делает это функция send, где входными параметрами являются данные, которые необходимо передать серверу. Далее необходимо ожидать ответа от сервера, это позволяет осуществлять метод onreadystatechange.

Функция sendTextJQuery отправляет данные серверу с помощью технологий AJAX. Выполняется это функцией $.ajax, в которой указываются необходимые параметры: url, тип запроса и данные. Функция success позволяет получать ответ от сервера, который в дальнейшем необходимо обрабатывать.

XMLHttpRequest — это объект JavaScript, содержащий API, который обеспечивает клиентский скрипт функциональностью для обмена данными между клиентом и сервером. Может работать как асинхронно, так и синхронно с серверным приложением. Использует запросы HTTP или HTTPS напрямую к серверу и загружает данные ответа сервера напрямую в вызывающий скрипт. Позволяет осуществлять HTTP-запросы к серверу без перезагрузки страницы.

В основе работы AJAX лежит объект XMLHttpRequest. Только в отличие от XMLHttpRequest требует еще и библиотеку JQuery, но и имеет более гибкий функционал и более простое взаимодействие с сервером.

3 Порядок выполнения работы

3.1 Описание HTML-страниц (не менее трех страниц)

1) Первая страница «О себе» содержит информацию:

Работу выполнил: Иванов Иван Иванович

Номер группы: ИКБО-01-18

Номер индивидуального задания: 3

Текст индивидуального задания: «Создание калькулятора…»

2) Вторая страница — реализация индивидуального задания. На странице отображены следующие элементы:

а) поля для занесения информации, необходимой для вычисления;

б) кнопка для отправки результатов на сервер;

в) поле, в котором будет выводиться результат вычислений.

3) Третья страница — работа с таблицей, студент выбирает некоторую узкую тему, с которой он дальше будет работать. Например, таблица «Студенты». На странице отображается информация о студентах, при этом дана возможность добавлять нового студента, удалять и изменять данные уже существующего студента.

3.2 Изменения сервера в соответствии с требованиями

1) Сервер должен обрабатывать запросы и в соответствии с ними возвращать необходимую клиенту информацию.

2) От клиента будут запросы на загрузку данных о css, js и других файлах, используемых страницей. Пример: все страницы используют файл каскадных стилей (css), в HTML-документе ссылка на него. При загрузке страницы браузером он обнаруживает, что данная страница использует css-файл и начинает автоматически требовать его с сервера. Сервер должен в итоге отправить ему этот файл.

3) Для сохранения данных третьей страницы необходимо:

а) создать структуру таблицы. Например, таблица «Студенты» содержит информацию: Имя, Фамилия, Отчество, Год рождения. Для этого мы создаем новый класс под названием Student с описанием полей Id, Name, LastName, FirstName, YearBirth и их геттеры и сеттеры (getName (), setName ()…);

б) для хранения данных создаем список со структурой таблицы (Пример: List<Student>). Сохранение в файл или БД не требуется. То есть при каждом перезапуске сервера у нас будут обнуляться записи.

3.3 Технические требования

Требования к серверу:

1) все страницы должны храниться как HTML-страницы на сервере;

2) при первом подключении к серверу должна отобразиться первая HTML-страница, или если в адресной строке указан какой-то определенный ресурс, необходимо вывести информацию о той странице, которая соответствует данному запросу. При указании несуществующего ресурса система должна выводить первую HTML-страницу;

3) все поля ввода информации должны иметь проверку входных значений (с помощью JavaScript).

Требования к оформлению HTML-страниц:

1) общие требования:

а) у каждого студента должен быть уникальный стиль сайта и уникальная тема для третьей страницы;

б) все страницы должны быть красиво оформлены, стили описаны в отдельном css-файле;

в) у каждой страницы должно быть реализовано единое меню, у которого будет минимум три ссылки (переход на каждую созданную страницу);

2) индивидуальные требования к страницам:

а) первая страница: вывод статической информации;

б) вторая страница: для выполнения вычислений мы должны использовать возможность отправки данных на сервер POST-запросом и получать от сервера ответ, который должен помещаться в специально отведенное для этого поле. Данная функция должна быть реализована с помощью JavaScript и храниться отдельном jsфайле;

в) третья страница: при запуске сервера и при обращении к этой странице у нас будет отображаться пустая таблица (то есть сервер сохраненные данные не хранит постоянно, а только временно держит их в оперативной памяти). Данная страница должна позволять:

– добавлять строки и сохранять их на сервере;

– изменять строки и сохранять изменение на сервере;

– удалять строки и удалять информацию с сервера.

г) операции удаления, добавления и изменения должны быть реализованы с помощью JavaScript, данные операции должны быть реализованы на одной странице, действие каждой команды должно отправляться на сервер:

– операция удаления: напротив каждой строки должен быть значок удаления, после нажатия на который строка должна пропасть (удалиться со страницы и с сервера);

– операция создания: для создания должна быть отдельная кнопка, по нажатию на которую будет блокироваться основное окно с таблицей и появляться другое, в котором будут описаны поля, необходимые для создания новой строки, и кнопка сохранить, по нажатию на которую произведется добавление на сервер и на страницу;

– операция изменения: напротив каждой строки должен быть значок, нажатие которого должно работать по аналогии операции создания. Только все поля должны быть заполнены данными изменяемой строки.