Министерство образования Белгородской области
Областное государственное автономное
профессиональное образовательное учреждение
«Белгородский индустриальный колледж»

Рассмотрено
предметно-цикловой комиссией
Протокол заседания № 1
От «31»августа 2022
Председатель цикловой комиссии
_____/ Третьяк И.Ю.

Методические рекомендации
по выполнению самостоятельной работы по **МДК 09.01**
Проектирование и разработка веб-приложений
для специальности

09.02.07 Информационные системы и программирование

Разработчик:

Солдатенко М.Н. преподаватель
специальных дисциплин БИК

Белгород 2022

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

МДК 09.01 Проектирование и разработка веб-приложений профессионального модуля «Разработка дизайна веб-приложений» является специальным, формирующим базовые умения и компетенции для получения выпускником профессиональных умений и компетенций.

Методические рекомендации по выполнению самостоятельной работы студентов специальности 09.02.07 Информационные системы и программирование разработаны в соответствии с рабочей ПМ.09 Проектирование, разработка и оптимизация веб-приложений в части МДК 09.01 Проектирование и разработка веб-приложений, соответствуют требованиям Федерального государственного образовательного стандарта по специальностям среднего профессионального образования.

Методические рекомендации по выполнению самостоятельной работы студентов содержат информацию о том, сколько и какие темы выносятся на самостоятельное изучение, основную и дополнительную литературу, вопросы для самопроверки.

Целью методических рекомендаций по выполнению самостоятельной работы студентов является организация и управление самостоятельной работой студентов в процессе изучения данного МДК.

Форму самостоятельной работы студент выбирает согласно рабочей программе (реферат, презентация, решение задач). К каждой теме предложен план, вопросы проверки и самопроверки.

Методические рекомендации предназначены для студентов очной формы обучения специальности 09.02.07 Информационные системы и программирование. По учебному плану по МДК 09.01 Проектирование и разработка веб-приложений на самостоятельную работу студентов отводится 6 часов, на консультации - 6 часов.

Выполненная работа позволит приобрести не только знания, но и умения, навыки, компетенции, а также поможет выработать свою методику подготовки, что очень важно в дальнейшем процессе обучения.

ОБЩИЕ ПОЛОЖЕНИЯ

МДК 09.01 Проектирование и разработка веб-приложений профессионального модуля «Проектирование, разработка и оптимизация веб-приложений» является специальным, устанавливающим базовые знания для получения выпускником профессиональных умений, и преподается студентам специальности 09.02.07 Информационные системы и программирование.

Методические рекомендации по выполнению самостоятельной работы по МДК 09.01 Проектирование и разработка веб-приложений разработаны в соответствии с рабочей программой ПМ.09 Проектирование, разработка и оптимизация веб-приложений.

Содержание методических рекомендаций по выполнению самостоятельной работы по данному МДК соответствует требованиям Федерального государственного образовательного стандарта по специальностям среднего профессионального образования.

По учебному плану в соответствии с рабочей программой учебной ПМ.09 Проектирование, разработка и оптимизация веб-приложений студентами очной формы обучения предусмотрено самостоятельных занятий – 6 часов, консультаций - 6 часов.

Целью методических рекомендаций является обеспечение эффективности самостоятельной работы студентов с литературой на основе организации её изучения.

Задачами методических рекомендаций по самостоятельной работе являются:

- активизация самостоятельной работы студентов;
- содействие развития творческого отношения к данному МДК;
- выработка умений и навыков рациональной работы с литературой;
- управление познавательной деятельностью студентов.

Функциями методических рекомендаций по самостоятельной работе являются:

- определение содержания работы студентов по овладению программным материалом;
- установление требований к результатам изучения МДК.

Сроки выполнения и виды отчётности самостоятельной работы определяются преподавателем и доводятся до сведения студентов.

МДК 09.01 Проектирование и разработка веб-приложений базируется на знаниях, умениях и навыках, полученных студентами при изучении дисциплины «Информатика», МДК 08.01 Проектирование и разработка интерфейсов Использование междисциплинарных связей обеспечивает системность изучения материала дисциплины и исключение дублирования.

В соответствии с рабочей программой ПМ.09 Проектирование, разработка и оптимизация веб-приложений студент должен:

иметь практический опыт:

- в использовании специальных готовых технических решений при разработке веб-приложений;
- выполнении разработки и проектирования информационных систем;
- модернизации веб-приложений с учетом правил и норм подготовки информации для поисковых систем;
- реализации мероприятий по продвижению веб-приложений в сети Интернет;

уметь:

- разрабатывать программный код клиентской и серверной части веб-приложений;
- осуществлять оптимизацию веб-приложения с целью повышения его рейтинга в сети Интернет;
- разрабатывать и проектировать информационные системы

знать:

- языки программирования и разметки для разработки клиентской и серверной части веб-приложений;
- принципы функционирования поисковых сервисов и особенности оптимизации веб-приложений под них;
- принципы проектирования и разработки информационных систем

1. ТЕМАТИЧЕСКИЙ ПЛАН ВИДОВ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Наименование и краткая характеристика	**Количество часов**	**Вид работы**
1	**2**	**3**
Проработка учебной и специальной технической литературы Подготовка к лабораторным работам, оформление лабораторных работ, отчетов и подготовка к их защите	**6**	Подготовка докладов Оформление отчета, подготовка к защите
Консультации по МДК: 1. Программирование сложных структур с использованием JSON 2. Использование библиотеки jQuery 3. Применение технологии AJAX	**6** 2 2 2	Создание сценариев, программирование сложных структур

2. ПОРЯДОК ВЫПОЛНЕНИЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

По каждому вопросу, выносимому на самостоятельную работу студентам, приведены методические рекомендации.

1. Проработка учебной и специальной технической литературы.

Оснащение: рекомендуемая литература, методические указания по выполнению самостоятельной работы.

*Задание:*подготовка докладов.

Порядок выполнения задания

На основании основной и дополнительной специальной технической литературы, рекомендуемой к выполнению самостоятельной работы студентам необходимо рассказать о сетях передачи данных, о методах кодирования применяемых при передаче данных, о мобильных сетях первого, второго, третьего и четвертого поколений, о современном оборудовании сетей, модели взаимодействия открытых систем, технологии глобальных систем.

Подготовка к лабораторным работам, оформление лабораторных работ, отчетов и подготовка к их защите.

Оснащение: рекомендуемая литература, методические указания по выполнению самостоятельной работы, методические указания по выполнению лабораторных работ, требования к оформлению отчета.

Задание: оформление отчета, подготовка к защите

Порядок выполнения практического задания

На основании основной и дополнительной литературы, рекомендуемой к выполнению самостоятельной работы, а также на основаниииметодических указаний по выполнению лабораторных работ, требований к оформлению отчетастудентам необходимо подготовиться к лабораторной работе, оформить отчет и подготовиться к защите.В процессе подготовки студент должен усвоить теоретический материал, относящийся к лабораторной работе, изучить и ясно представить себе содержание и порядок выполнения лабораторной работы, знать ответы на приведенные в методическом руководстве контрольные вопросы.

Консультации

1. Программирование сложных структур с использованием JSON

JSON - простой, основанный на использовании текста, способ хранить и передавать структурированные данные. С помощью простого синтаксиса вы можете легко хранить все, что угодно, начиная от одного числа до строк, массивов и объектов, в простом тексте. Также можно связывать между собой массивы и объекты, создавая сложные структуры данных.

После создания строки JSON, ее легко отправить другому приложению или в другое место сети, так как она представляет собой простой текст.

JSON имеет следующие преимущества:

- Он компактен.
- Его предложения легко читаются и составляются как человеком, так и компьютером.
- Его легко преобразовать в структуру данных для большинства языков программирования (числа, строки, логические переменные, массивы и так далее)

- Многие языки программирования имеют функции и библиотеки для чтения и создания структур JSON.

Название JSON означает JavaScript Object Notation (представление объектов JavaScript). Как и представляет имя, он основан на способе определения объектов (очень похоже на создание ассоциативных массивов в других языках) и массивов.

Наиболее частое распространенное использование JSON - пересылка данных от сервера к браузеру. Обычно данные JSON доставляются с помощью AJAX, который позволяет обмениваться данными браузеру и серверу без необходимости перезагружать страницу.

Пример:

1. Пользователь нажимает миниатюру продукта в онлайн магазине.
2. JavaScript, выполняющийся на браузере, генерирует запрос AJAX к скрипту PHP, запущенному на сервере, передавая ID выбранного продукта.
3. Скрипт PHP получает название продукта, описание, цену и другую информацию из базы данных. Затем составляет из данных строку JSON и отсылает ее браузеру.
4. JavaScript, выполняющийся на браузере, получает строку JSON, декодирует ее и выводит информацию о продукте на странице для пользователя.

Также можно использовать JSON для отправки данных от браузера на сервер, передавая строку JSON в качестве параметра запросов GET или POST. Но данный метод имеет меньшее распространение, так как передача данных через запросы AJAX может быть упрощена. Например, ID продукта может быть включен в адрес URL как часть запроса GET.

Библиотека jQuery имеет несколько методов, например, getJSON() и parseJSON(), которые упрощают получение данных с помощью JSON через запросы AJAX.

Как создать строку JSON?

Есть несколько основных правил для создания строки JSON:

- Строка JSON содержит либо массив значений, либо объект (ассоциативный массив пар имя/значение).
- *Массив* заключается в квадратные скобки ([и]) и содержит разделенный запятой список значений.
- *Объект* заключается в фигурные скобки ({ и }) и содержит разделенный запятой список пар имя/значение.
- *Пара имя/значение* состоит из имени поля, заключенного в двойные кавычки, за которым следует двоеточие (:) и значение поля.
- *Значение* в массиве или объекте может быть:
  - Числом (целым или с плавающей точкой)
  - Строкой (в двойных кавычках)
  - Логическим значением (true или false)
  - Другим массивом (заключенным в квадратные скобки)
  - Другой объект (заключенный в фигурные скобки)
  - Значение null

Чтобы включить двойные кавычки в строку, нужно использовать обратную косую черту: \". Так же, как и во многих языках программирования, можно помещать управляющие символы и шестнадцатеричные коды в строку, предваряя их обратной косой чертой. Смотрите детали на сайте JSON.

Ниже приводится пример оформления заказа в формате JSON:

```
{
  "orderID": 12345,
  "shopperName": "Ваня Иванов",
  "shopperEmail": "ivanov@example.com",
  "contents": [
    {
      "productID": 34,
      "productName": "Супер товар",
      "quantity": 1
    },
    {
      "productID": 56,
      "productName": "Чудо товар",
      "quantity": 3
    }
  ],
  "orderCompleted": true
}
```

Рассмотрим строку подробно:

- Мы создаем объект с помощью фигурных скобок ({ и }).
- В объекте есть несколько пар имя/значение:

`"orderID": 12345`	Свойство с именем `"orderId"` и целочисленным значением `12345`
`"shopperName": "Ваня Иванов"`	свойство с именем `"shopperName"` и строковым значением `"Ваня Иванов"`
`"shopperEmail": "johnsmith@example.com"`	Свойство с именем `"shopperEmail"` и строковым значением `"ivanov@example.com"`
`"contents": [ ... ]`	Свойство с именем `"contents"`, значение которого является массивом
`"orderCompleted": true`	Свойство с именем `"orderCompleted"` и логическим значением `true`

- В массиве "contents" есть 2 объекта, представляющие отдельные позиции в заказе. Каждый объект содержит 3 свойства: productID, productName, и quantity.

Кстати, так как JSON основан на объявлении объектов JavaScript, то вы можете быстро и просто сделать выше приведенную строку JSON объектом JavaScript:

```
<script type="text/javascript">
var cart = {
  "orderID": 12345,
  "shopperName": "Ваня Иванов",
  "shopperEmail": "ivanov@example.com",
  "contents": [
    {
      "productID": 34,
      "productName": "Супер товар",
      "quantity": 1
    },
    {
      "productID": 56,
      "productName": "Чудо товар",
      "quantity": 3
    }
  ],
  "orderCompleted": true
};
</script>
```

Работаем со строкой JSON в JavaScript

JSON имеет простой формат, но создавать строку JSON вручную достаточно утомительно. Кроме того, часто нужно взять строку JSON, конвертировать ее содержание в переменную, которую можно будет использовать в коде.

Большинство языков программирования имеют инструменты для простого преобразования переменных в строки JSON и наоборот.

Создаем строку JSON из переменной

JavaScript имеет встроенный метод JSON.stringify(), который берет переменную и возвращает строку JSON, представляющую ее содержание. Например, создадим объект JavaScript, который содержит сведения о заказе из нашего примера, а затем создадим из него строку JSON:

```
<script type="text/javascript">

var cart = {
  "orderID": 12345,
  "shopperName": "Ваня Иванов",
  "shopperEmail": "ivanov@example.com",
  "contents": [
    {
      "productID": 34,
      "productName": "Супер товар",
      "quantity": 1
    },
    {
      "productID": 56,
      "productName": "Чудо товар",
      "quantity": 3
    }
  ],
  "orderCompleted": true
};

alert ( JSON.stringify( cart ) );

</script>
```

Данный код выдаст:

```
{"orderID":12345,"shopperName":"Ваня Иванов","shopperEmail":"ivanov@example.com","contents":[{"productID":34,"productName":"Супер товар","quantity":1},{"productID":56,"productName":"Чудо товар","quantity":3}],"orderCompleted":true}
```

Обратите внимание, что метод JSON.stringify() возвращает строку JSON без пробелов. Ее сложнее читать, но зато она более компактна для передачи через сеть.

Создаем переменную из строки JSON

Существует несколько способов разобрать строку JSON в JavaScript, но самый безопасный и надежный - использовать встроенный метод JSON.parse(). Он получает строку JSON и возвращает объект или массив JavaScript, который содержит данные. Например:

```
<script type="text/javascript">

var jsonString = '                          \
{                                           \
  "orderID": 12345,                         \
  "shopperName": "Ваня Иванов",             \
  "shopperEmail": "ivanov@example.com",     \
  "contents": [                             \
    {                                       \
      "productID": 34,                      \
      "productName": "Супер товар",         \
      "quantity": 1                         \
    },                                      \
    {                                       \
      "productID": 56,                      \
      "productName": "Чудо товар",          \
      "quantity": 3                         \
    }                                       \
  ],                                        \
  "orderCompleted": true                    \
}                                           \
';

var cart = JSON.parse ( jsonString );

alert ( cart.shopperEmail );
alert ( cart.contents[1].productName );

</script>
```

Мы создали переменную jsonString, которая содержит строку JSON нашего примера с заказом. Затем мы передаем данную строку методу JSON.parse(), который создает объект, содержащий данные JSON и сохраняет его в переменной cart. Остается только осуществить проверку, выведя свойства объекта shopperEmail и productName массива contents.

В результате мы получим следующий вывод:

```
ivanov@example.com
Чудо товар
```

В реальном приложении ваш JavaScript код будет получать заказ в виде строки JSON в AJAX ответе от скрипта сервера, передавать строку методу JSON.parse(), а затем использовать данные для отображения на странице пользователя.

JSON.stringify() и JSON.parse() имеют другие возможности, такие как использование возвратных функций для пользовательской конвертации определённых данных. Такие опции очень удобны для конвертации различных данных в правильные объекты JavaScript.

Работаем со строкой JSON в PHP

PHP, как и JavaScript, имеет встроенные функции для работы с JSON строками.

Создаем строку JSON из переменной PHP

Функция json_encode() принимает переменную PHP и возвращает строку JSON, представляющую содержание переменной. Вот наш пример с заказом, написанный на PHP:

```php
<?php
$cart = array(
  "orderID" => 12345,
  "shopperName" => "Ваня Иванов",
  "shopperEmail" => "ivanov@example.com",
  "contents" => array(
    array(
      "productID" => 34,
      "productName" => "Супер товар",
      "quantity" => 1
    ),
    array(
      "productID" => 56,
      "productName" => "Чудо товар",
      "quantity" => 3
    )
  ),
  "orderCompleted" => true
);

echo json_encode( $cart );
?>
```

Данный код возвращает абсолютно такую же строку JSON, как и в примере с JavaScript:

```
{"orderID":12345,"shopperName":"Ваня Иванов","shopperEmail":"ivanov@example.com","contents":[{"productID":34,"productName":"Супер товар","quantity":1},{"productID":56,"productName":"Чудо товар","quantity":3}],"orderCompleted":true}
```

В реальном приложении ваш скрипт PHP пришлет данную строку JSON как часть AJAX ответа браузеру, где JavaScript код с помощью метода JSON.parse()преобразует ее обратно в переменную для вывода на странице пользователя.

Вы может передавать различные флаги в качестве второго аргумента функции json_encode(). С их помощью можно изменять принципы кодирования содержания переменных в строку JSON.

Создаем переменную из строки JSON

Для преобразования строки JSON в переменную PHP используется метод json_decode(). Заменим наш пример для JavaScript с методом JSON.parse() на код PHP:

```php
<?php
$jsonString = '
{
  "orderID": 12345,
  "shopperName": "Ваня Иванов",
  "shopperEmail": "ivanov@example.com",
  "contents": [
    {
      "productID": 34,
      "productName": "Супер товар",
      "quantity": 1
    },
    {
      "productID": 56,
      "productName": "Чудо товар",
      "quantity": 3
    }
  ],
  "orderCompleted": true
}
';

$cart = json_decode( $jsonString );
echo $cart->shopperEmail . "<br>";
echo $cart->contents[1]->productName . "<br>";
?>
```

Как и для JavaScript данный код выдаст:

```
ivanov@example.com
Чудо товар
```

По умолчанию функция json_decode() возвращает объекты JSON как объекты PHP. Существуют обобщенные объекты PHP класса stdClass. Поэтому мы используем -> для доступа к свойствам объекта в примере выше.

Если вам нужен объект JSON в виде ассоциированного массива PHP, нужно передать true в качестве второго аргумента функции json_decode(). Например:

```php
$cart = json_decode( $jsonString, true );
echo $cart["shopperEmail"] . "<br>";
echo $cart["contents"][1]["productName"] . "<br>";
```

Данный код выдаст такой же вывод:

```
ivanov@example.com
Чудо товар
```

Также функции json_decode() можно передавать другие аргументы для указания глубины рекурсии и способов обработки больших целых чисел.

2. Использование библиотеки jQuery

Потратив некоторое время на попытки привнести динамическую функциональность на страницы вашего сайта, вы обнаружите, что постоянно следуете одному и тому же шаблону: сначала находите один или несколько элементов страницы, а затем выполняете над ними некоторые действия, такие как скрытие, изменение размеров, положения, прозрачности и т.д.

Используя обычный javascript для решения каждой из этих задач потребуется десятки строк программного кода. Автор jQuery разработал свою библиотеку таким образом, что наиболее общие задачи становятся тривиальными. Например, вот так, с помощью функции $() из библиотеки, можно находить элементы на странице по различным параметрам:

$("div")	вернет все div-элементы на странице.
$(".someBlock")	вернет все элементы с классом someBlock.
$("#content")	вернет элемент с идентификатором content.
$("#content2 div.someBlock")	вернет div-элементы с классом someBlock, которые находятся внутри элемента с идентификатором content2.
$("div:odd")	вернет div-элементы, находящиеся на странице под нечетными номерами.
$("[value = 5]")	вернет все элементы с атрибутом value, равным 5.

И это только малая часть всех возможностей для поиска элементов. После выбора элементов, можно сразу же приступить к манипуляциями над ними, jQuery предоставляет для этого широкий ассортимент методов. Приведем несколько наиболее востребованных:

$("#bigIt").css("height")	возвратит значение высоты у элемента с идентификатором bigIt.
$("div").css("width", "20px")	установит новое значение ширины всем div-элемента на странице.
$("#bigIt").attr("class")	возвратит значение класса элемента с id = bigIt.
$("#bigIt").attr("class", "box")	установит новое значение атрибута class у элемента с id = bigIt.
$("#bigIt").html(Новье!)	изменит все html-содержимое элемента с id = bigIt, на заданное в методе html.
$("#bigIt").text()	возвратит текст, находящийся внутри элемента с id = bigIt.

Таким образом, всего одной строкой кода, мы можем узнать или изменить css-значения, атрибуты, html и текстовое содержимое любых элементов на странице.

Помимо манипуляций с выбранными элементами, jQuery позволяет работать с самим набором: изменять его, а так же работать с элементами по отдельности. Продемонстрируем часть возможностей:

$("div").parent()	вернет родительские элементы всех div-ов.
$("div").children()	вернет дочерние элементы всех div-ов.
$("#someId").next()	вернет элемент, лежащий сразу после someId.
$("div").prev()	вернет элементы, лежащие перед div'ами.
$("div").eq(i)	вернет div-элемент, с индексом i в наборе.

$("div").get(i)	вернет DOM-объект div'a, с индексом i.
$("div").get()	вернет массив DOM-объеков всех div-ов.
$("div").size()	вернет размер набора (количествово div-ов).

Отметим различия методов get(i) и eq(i). Первый возвращает непосредственно DOM-объект элемента, идущего под номером i в наборе (кстати, нумерация начинается с 0). К такому элементу вы не сможете применить методы jQuery, зато сможете применить стандартные javascript методы. Метод eq(i) наоборот, возвращает i-й элемент в таком виде, что к нему можно применять методы jQuery. Вообще, для того, чтобы к элементам можно было применять методы библиотеки jQuery, они должны находиться в так называемом объекте jQuery, именно его возвращает функция $().

Рассмотрим отдельно метод .each(), предназначенный для поэлементной обработки набора. В качестве параметра, этот метод принимает пользовательскую функцию, которая будет автоматически вызвана, для каждого элемента набора:

```
      var heights = [];  // переменная, которая будет хранить высоты элементов
       $("div").each(function(indx, element){ // indx - номер элемента в наборе,
element - сам элемент
        heights.push($(element).height());
      });
       // в итоге, в переменную heights будут помещены значения высот всех div-
элементов
```

- Карта функций jQuery http://jquery.page2page.ru/index.php5

- Документация на русском также доступна по адресу http://jquerybook.ru/api/

Функция $()

Функция с лаконичным именем $() является самой главной во всей библиотеке. С ее помощью можно находить элементы на странице, добавлять "на лету" новый html:

```
      // создадим div-элемент и добавим его в конец элемента с идентификатором
content
      $("<div><p>Ба-бах!</p></div>").appendTo("#content");
```

Кроме этого, $() позволяет привязать всю функциональность jQuery к уже существующим объектам DOM-элементов:

```
      // найдем элемент с идентификатором some_id, средствами обычного javascript
      var el = document.getElementById("some_id");
```

```
      // установим у этого элемента новое значение css-свойства margin
      $(el).css("margin", "5px");
```

Начало работы скрипта

Прежде чем запускать js-скрипт, необходимо быть уверенным, что часть страницы, с которой этот скрипт будет работать уже загружена. Большинство программистов используют для этого событие onload, которое происходит по окончанию загрузки всей страницы:

```
window.onload = function(){
  // вызов нужных функций скрипта
}
```

Однако onload происходит после того, как страница сформирована полностью, включая загрузку всех изображений, флеш-баннеров и видеороликов. В то время как структура дерева DOM (элементов страницы), с которой обычно и работает скрипт, оказывается готова гораздо раньше. В результате скрипт запускается значительно позднее чем мог бы. На этот случай в jQuery есть метод ready, вызов которого осуществляется в момент готовности дерева DOM:

```
$(document).ready( function(){
  // вызов нужных функций скрипта
});
```

Примеры

Создайте HTML документ

```
<!DOCTYPE>
<html>
<head>
    <title>Jquery</title>
</head>
<body>
      <section id="news" class="news">
        <div id="article_main" class="article" data-article-id="7">
            <div class="header">
                <p>article #2</p>
            </div>
            <div class="body">
                <p>Hello</p>
            </div>
        </div>
        <div class="article" data-article-id="8">
            <div class="header">
                <p>article #2</p>
            </div>
            <div class="body">
                <p>Hey</p>
            </div>
        </div>
    </section>
    <script src="jquery-3.2.1.min.js"></script>
    <script>
         <!-- СЮДА КОПИРОВАТЬ КОД -->
    </script>
</body>
</html>
```

Подключите библиотеку jQuery (https://code.jquery.com/jquery-3.2.1.min.js)

Она доступна по адресу https://jquery.com/download/

Изучите следующие примеры

```
$(function () {
    var newsElement = $('#news');
    console.log(newsElement);
 });
```

```
$(function () {
    var newsElement = $('.news');
    console.log(newsElement);
 });
```

```
$(function () {
    var headers = $('.header');
    console.log(headers);
    for(let header of headers){
        console.log(header);
    }
 });
```

```
$(function () {
    var headers = $('#article_main .header');
    console.log(headers);
    $.each(headers, (index, value) => {
        console.log(value);
    });
 });
```

```
$(function () {
    var articles = $('#news > .article');
    console.log(articles);
 });
```

```
$(function () {
    var article = $('*[data-article-id="8"]');
    console.log(article);
    var articleId = article.data("article-id");
    console.log(articleId);
 });
```

```
$(function () {
    var articles = $('.article');
    var headers = articles.find('.header');
    console.log(headers);
});
```

```
$(function () {
    var article = $('#article_main');
    console.log(article.children());
});
```

```
$(function () {
    var article = $('#article_main');
    console.log(article.children('.body'));
});
```

```
$(function () {
    var mainArticle = $('#article_main');
    console.log(mainArticle.next()[0]);
```

```
});
```

Задание по jQuery 1

Есть дерево из тегов /<li>.

Напишите код, который для каждого элемента <li> выведет:

- Текст непосредственно в нём (без подразделов).
- Количество вложенных в него элементов <li> – всех, с учётом вложенных.
- Создайте кнопку с текстом «Дерево», по нажатию не неё дерево <ul> должно анимировано исчезать, при повторном нажатии анимировано появляется.

Обработчик клика на элементе с идентификатором main

```
var $first = $('#main')
$first.on('click',function(){
    $first.css({
        'border':'1px solid red',
        'color':'blue'
    });
});


<!DOCTYPE HTML>
<html>
<head>  <meta charset="utf-8"></head>
<body>
  <ul>
    <li>Животные
      <ul>
        <li>Млекопитающие
          <ul>
            <li>Коровы</li>
            <li>Ослы</li>
            <li>Собаки</li>
            <li>Тигры</li>
          </ul>
        </li>
        <li>Другие
          <ul>
            <li>Змеи</li>
            <li>Птицы</li>
            <li>Ящерицы</li>
          </ul>
        </li>
      </ul>
    </li>
    <li>Рыбы
      <ul>
        <li>Аквариумные
          <ul>
            <li>Гуппи</li>
            <li>Скалярии</li>
          </ul>
        </li>
        <li>Морские
          <ul>
            <li>Морская форель</li>
          </ul>
```

```
            </li>
          </ul>
        </li>
      </ul>
      <script>

      </script>
    </body>
    </html>
```

Задание по jQuery 2

Подключить 2 плагина jQuery

- Слайдер. Должны переключаться изображения https://owlcarousel2.github.io/OwlCarousel2/
- Datatables. Должна работать сортировка https://datatables.net/

3. Применение технологии AJAX

AJAX - технология разработки веб-интерфейсов, которая дает возможность браузеру взаимодействовать с веб-сервером без видимой для пользователя перезагрузки страницы. Т.е. при обновлении каких-то данных перезагружается только часть страницы.

Технология AJAX базируется на использовании объекта XMLHttpRequest(), который позволяет отправлять и получать информацию в различных форматах включая XML и HTML. Реализация технологии состоит из клиентской и серверной частей. Клиентская часть выполняется в браузере пользователя и пишется на JavaScript, а серверная выполняется на веб-сервере и пишется на любом языке веб-программирования: php, asp, perl и др.

Клиентский JavaScript-код - это основной код, выполняющий Ajax-приложение и обеспечивающий взаимодействие с сервером. Клиентская часть выполняет сбор информации для отправки запроса серверу, устанавливает соединение с сервером, посылает запрос, получает ответ и обрабатывает ответ сервера. Рассмотрим реализацию клиентской части подробней.

1. Создание объекта запроса XMLHttpRequest.

В Internet Explorer и других браузерах создание объекта XMLHttpRequest отличается. Во всех браузерах кроме Internet Explorer этот объект создается очень просто:`xmlHttp = new XMLHttpRequest();` Браузер Internet Explorer для создания объекта запроса использует анализатор MSXML. Кроме того существует две разных версии MSXML. Универсальный метод создания объекта запроса XMLHttpRequest для всех браузеров выглядит следующим образом:

```
        /*переменная для хранения объекта запроса*/
var xmlHttp=null;
try
{
 //создаем объект запроса для Firefox, Opera, Safari
 xmlHttp = new XMLHttpRequest();
}
catch (e)
```

```
{
 //создаем объект запроса для Internet Explorer
 try
 {
  xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
 }
 catch (e)
 {
  xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
 }
}
```

Создание объекта запроса происходит в строках `xmlHttp = new XMLHttpRequest();`, `xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");`и `xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");`. Остальные строки обрабатывают ошибки.

2. Выполнение запроса.

Для выполнения запроса необходимо открыть соединение с сервером, установить функцию, которая выполниться после ответа сервера и передать запрос.

Чтобы указать функцию которая будет обрабатывать ответ, необходимо свойству `onreadystatechange` присвоить имя JavaScript функции: `xmlHttp.onreadystatechange = MyFunc;`. Причем имя функции указывается без скобок.

Чтобы открыть соединение с сервером необходимо вызвать функцию `open()`. Эта функция имеет три обязательных параметра:

- метод запроса HTTP - обычно используют `'GET'` или `'POST'`, но можно использовать любой другой метод в соответствии с HTTP стандартами;
- url запроса - адрес скрипта на сервере, который обрабатывает запрос;
- асинхронность запроса - TRUE или FALSE, если TRUE, то запрос асинхронный и пользователь сможет продолжать работу со страницей.

Для отправки запроса необходимо вызвать метод `send()`, параметром которого могут быть любые данные, которые вы хотите отправить на сервер. Данные должны быть сформированы в строку запроса `param1=1¶m2=2¶m3=3`.
Если данные отправляются методом POST, то необходимо изменить MIME-тип запроса: `xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');`

Выполнение запроса с использованием метода GET:

```
/*Устанавливаем соединение*/
xmlHttp.open("GET", "example.php?param1=1¶m2=2", true);
/*Указываем функцию*/
xmlHttp.onreadystatechange = MyFunc;
/*Отправляем запрос*/
xmlHttp.send(null);
```

И с использование метода POST:

```
/*Устанавливаем соединение*/
xmlHttp.open("POST", "example.php", true);
/*Меняем MIME-тип*/
xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-
urlencoded");
/*Указываем функцию*/
xmlHttp.onreadystatechange = MyFunc;
/*Отправляем запрос*/
xmlHttp.send('param1=1¶m2=2');
```

3. Обработка ответа сервера.

Отправляя запрос, мы указали имя функции, обрабатывающей ответ сервера: `xmlHttp.onreadystatechange = MyFunc;`. Для начала, эта функция должна проверять статус запроса. Это можно сделать с помощью свойства `readyState`. Если `xmlHttp.readyState = 4`, то ответ от сервера получен и можно приступать к его обработке. Далее необходимо проверить статус HTTP-ответа с помощью свойства `status`. Если `xmlHttp.status = 200` значит все в порядке и можно продолжить обработку данных. Получить доступ к данным можно с помощью свойств `responseText` - в виде текста, либо `responseXML` - в виде объекта XMLDocument.

Полный пример работы Ajax-приложения:

```
<script type="text/javascript" language="javascript">
/*переменная для хранения объекта запроса*/
var xmlHttp=null;
/*создание объекта запроса*/
function createRequest()
{
  try{
    //создаем объект запроса для Firefox, Opera, Safari
    xmlHttp = new XMLHttpRequest();
  } catch(e){
    //создаем объект запроса для Internet Explorer
    try{
      xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
    } catch(e){
      try{
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
      } catch(e){
        xmlHttp = null;
      }
    }
  }
  if(xmlHttp == null)alert("Браузер не поддерживает AJAX!");
}
/*отправка запроса*/
function sendRequest()
{
/*получаем объект запроса*/
  createRequest();
  /*Устанавливаем соединение*/
  xmlHttp.open("GET", "ajax.php?param1=1¶m2=2", true);
  /*Указываем функцию*/
  xmlHttp.onreadystatechange = MyFunc;
  /*Отправляем запрос*/
  xmlHttp.send(null);
}
/*обрабатываем ответ*/
function MyFunc()
{
  if(xmlHttp.readyState == 4) {
    if (xmlHttp.status == 200) {
      alert(xmlHttp.responseText);
    } else {
      alert("Ошибка обработки запроса!");
    }
  }
}
```

```
</script>
<input value="сделать запрос" type="button" onClick="sendRequest();" />
```

Серверная часть может быть написана на чем угодно. Возвращаемое значение xmlHttp.responseText формируется стандартным выводом. Пример серверной части на php:

```
<?php
/*Получаем параметры*/
$param1 = $_GET['param1'];
$param2 = $_GET['param2'];
/*Отсылаем ответ клиенту*/
echo("Вы прислали $param1 и $param2");
?>
```

И в дополнение некоторые справочные материалы по объекту **XMLHttpRequest**:

Методы класса XMLHttpRequest

Метод	Описание
abort()	отменяет текущий запрос
getAllResponseHeaders()	возвращает полный список HTTP-заголовков в виде строки
getResponseHeader(headerName)	возвращает значение указанного заголовка
open(method, url, async, userName, password)	определяет метод, URL, асинхронность и другие параметры запроса
send(data)	отправляет запрос на сервер, если данных нет то параметр null
setRequestHeader(label, value)	добавляет HTTP-заголовок к запросу
overrideMimeType(mimeType)	позволяет указать mime-type документа; **Внимание:** метод отсутствует в Internet Explorer!

Свойства класса XMLHttpRequest

Свойство	Описание
onreadystatechange	обработчик события, которое происходит при каждой смене состояния объекта
readyState	возвращает текущее состояние объекта (0—неинициализирован, 1—открыт, 2—отправка данных, 3—получение данных и 4—данные загружены)
responseText	текст ответа на запрос
responseXML	текст ответа на запрос в виде XML
status	возвращает HTTP-статус в виде числа (404 — «Not Found», 200 — «OK» и т. д.)
statusText	возвращает статус в виде строки («Not Found», «OK» и т. д.)

Перечень рекомендуемых учебных изданий, Интернет-ресурсов, дополнительной литературы

1. Основные источники литературы:

2. Бенкен, Е.С. PHP, MySQL, XML: программирование для интернета / Е.С. Бенкен. – СПб. : БХВ–Петербург, 2014. – 336 с.: ил.+CD–ROM

3. Ляпин, Д.А. PHP это просто. Начинаем с видеоуроков / Д.А. Ляпин, А.В. Никитин. – СПб. : БХВ–Петербург, 2013. – 176 с.: ил.+CD–ROM.

4. Маркин, А.В. Основы web-программирования на PHP: учебное пособие / А.В. Маркин, С.С. Шкарин. – М. : Диалог-МИФИ, 2014. – 252 с.

5. Храмцов, П.Б. Основы Web-технологий: учебное пособие / П.Б. Храмцов, С.А. Брик, А.М. Русак, А.И. Сурин. – 3–е изд., испр. – М. : Интернет–Университет Информационных технологий; БИНОМ. Лаборатория знаний, 2014. – 512 с.

6. Федорова, Г.Н.., Рудаков, А.В. Технология разработки программных продуктов. Практикум: учебное пособие / Г.Н. Федорова, А.В. Рудаков. – М. : .Academia, 2014. – 192 с.

2. Дополнительные источники:

1. Дунаев, В. Самоучитель JavaScript / В. Дунаев. 2–е изд. – СПб. : Питер, 2012. – 400 с.

2. Кузнецов, М.В. PHP 5. Практика разработки Web-сайтов / М.В. Кузнецов, И.В. Симдянов, С. В. Голышев. – СПб. : БХВ-Петербург, 2012. – 960 с.: ил.

3. Котеров, Д. PHP 5 в подлиннике / Д. Котеров, А. Костарев. – СПб : Символ – Плюс, 2014. – 1120 с., ил.

5. Рудаков, А.В. Технология разработки программных продуктов: учебник. / А.В. Рудаков. – М. : .Academia, 2013. – 208с.

6. Савельева, Н.В. Основы программирования на PHP: курс лекций./ Н.В. Савельева. – М.: ИНТУИТ.РУ «Интернет–университет информационных технологий», 2012. – 264 с.

3. Интернет ресурсы:

1. Журнал веб-дизайн – уголок профессионала. [Электронный ресурс] – режим доступа: http://www.webmagazine.biz.

2. Система федеральных образовательных порталов Информационно – коммуникационные технологии в образовании. [Электронный ресурс] – режим доступа: http://www.ict.edu.ru.

3. Методы и средства инженерии программного обеспечения: Учебник. Автор/создатель Лавришева Е.М., Петрухин В.А. Единое окно доступа к образовательным ресурсам. http://window.edu.ru/catalog/pdf2txt/699/41699/18857

4. http://ru.wikipedia.org

5. http://www.pstut.ru/

6. htpp://www.php.spb.ru

7. http://www.javaportal.ru

8. От модели объектов - к модели классов. Единое окно доступа к образовательным ресурсам. http://real.tepkom.ru/Real_OM-CM_A.asp

Критерии оценки самостоятельной работы:

Оценка «5» - тема раскрыта в достаточной мере, отражены ключевые определения по теме, сделаны выводы, оформление соответствует требованиям, недочетов нет.

Оценка «4» - тема раскрыта в достаточной мере, отражены не все ключевые определения по теме, сделаны выводы, есть небольшие недочеты в оформлении.

Оценка «3» - тема раскрыта не в полной мере, отражены не все ключевые определения по теме, выводы недостаточно глубокие, есть недочеты в оформлении.

Оценка «2» - тема раскрыта не в полной мере, не отражены ключевые определения по теме, выводы не сделаны, есть ошибки в оформлении