Оглавление

ПРЕДИСЛОВИЕ

Данное учебно-методическое пособие к выполнению курсовой работы составлено в соответствии с требованиями Федерального государственного образовательного стандарта высшего образования и рабочей программой «Web-технологии в информационных системах» для обучающихся по направлению подготовки 09.03.02 Информационные системы и технологии.

Целью дисциплины «Web-технологии в информационных системах» является углубление уровня освоения компетенций обучающегося в области применения web-технологий. Освоение дисциплины включает курс лекций, компьютерные практикумы, групповые и индивидуальные консультации по курсовым работам и самостоятельную работу обучающихся по следующим разделам:

- введение в современные web-технологии;
- принципы организации сети «Интернет» и виды web-ресурсов;
- организация информационного обмена инженерными данными;
- безопасность в сети «Интернет»;
- системы управления контентом CMS.

В учебно-методическом пособии содержатся сведения об основах создания динамических web-ресурсов, а также о принципах безопасности при работе в открытом пространстве сети «Интернет» в рамках подготовки курсовой работы, выполнение которой заключается в последовательном решении шести представленных в пособии практических заданий и объединении полученных результатов в прототип информационной системы. После каждого задания даны темы для самостоятельного изучения. В конце каждого раздела приведены вопросы для самоконтроля, которые позволяют закрепить и проверить полученные знания, умения и навыки.

Требования к форме выполнения, оформлению и представлению курсовой работы представлены в приложениях.

Цель курсовой работы заключается в разработке прототипа информационной системы в соответствии с полученным техническим заданием.

Выполнение курсовой работы заключается в последовательном решении шести практических заданий и объединении полученных результатов в прототип информационной системы.

Практическое задание 1. Создание web-ресурсов на языке разметки HTML.

Цель: получение знаний о базовых принципах создания web-ресурсов с использованием языка разметки гипертекста HTML.

Результат работы: сверстанные статические страницы для прототипа информационной системы.

Практическое задание 2. Создание динамических web-ресурсов с элементами программирования.

Цель: получение знаний о базовых принципах создания динамических web-ресурсов с использованием скриптовых языков программирования.

Результат работы: сверстанные статические страницы с добавленными динамическими элементами.

Практическое задание 3. Web-анимация.

Цель: получение знаний о базовых принципах создания анимированных web-ресурсов.

Результат работы: сверстанные статические страницы с добавленными анимированными элементами.

Практическое задание 4. Создание прототипа информационной системы.

Цель: создание прототипа информационной системы.

Результат работы: прототип информационной системы.

Практическое задание 5. Публикация web-ресурсов.

Цель: получение знаний о базовых принципах размещения созданных ранее web-ресурсов в сети «Интернет» и организация общего доступа к ресурсам.

Результат работы: публикация прототипа информационной системы в сети «Интернет».

Практическое задание 6. Основы безопасной работы в сети «Интернет».

Цель работы: получение знаний о базовых принципах безопасной работы в сети «Интернет».

Результат работы: настройка безопасности прототипа информационной системы.

1. ВВЕДЕНИЕ В HTML

Для того чтобы пользователи Интернета видели одну и туже картинку при отображении сайтов, используется единый язык HTML, воспринимаемый различными устройствами. Таким образом, HTML (англ. *HyperText Markup Language* — язык гипертекстовой разметки) — стандартизированный язык разметки документов, используемый в сети «Интернет». Отметим, что язык HTML позволяет пользователю переходить от одной части текста к другой, при том что этот текст может храниться на разных устройствах.

Большинство существующих web-страниц содержит описание разметки на языке HTML, который переводится браузерами в отформатированный текст и отображается на экране монитора компьютера или мобильного устройства.

Основой языка HTML являются HTML-теги, которые используются для разграничения начала и конца элементов в разметке. Любой HTML-документ состоит из дерева HTML-элементов и текста. Каждый HTML-элемент обозначается начальным (открывающим) и конечным (закрывающим) тегами. Открывающий и закрывающий теги содержат имя тега. Простой пример кода, написанного на языке HTML, представлен на рис. 1.

```
<html>

  <head>
    <meta content="text/html; charset=utf-8">
    <title>Прототип информационной системы</title>
  </head>

  <body>
    <h1>
      Заголовок страницы
    </h1>
    <div>
      Содержание страницы
    </div>
  </body>

</html>
```

Рис. 1. Пример кода на HTML

Вместе с тем запомнить все теги и их параметры при использовании языка HTML сложно, однако существует множество справочников и руководств, к которым можно обращаться, чтобы получить ответ на появившийся вопрос.

Кроме того, учитывая большой спрос на создание различных страниц в Интернете, разработано большое количество редакторов, позволяющих упростить работу с HTML-языком.

Все HTML-редакторы можно разделить на две основные категории:

1) WYSIWYG HTML — редактор кода, с помощью которого можно создавать web-страницы без знания языков программирования, что следует из названия: What You See Is What You Get (что видишь, то и получаешь). Преимуществом является отсутствие углубления непосредственно в процесс разработки страницы, однако это же является и недостатком.

Редакторы такого типа, как правило, формируют объемные HTML-коды, в результате чего документ получается громоздким и время его загрузки увеличивается;

2) редактор HTML-тегов. Работая с программами данного типа, пользователю доступен непосредственно код страницы, который при желании можно изменять. HTML-документ получается более компактным по сравнению с результатами работы редакторов первого типа.

Вместе с тем HTML-редакторы могут быть выполнены как в виде онлайн-сервисов, так и в виде полноценных программных пакетов.

Одним из первых и наиболее простых был редактор FrontPage, который входил в пакет Microsoft Office. К основным недостаткам данного ПО можно отнести снижение быстродействия, а также ограниченные возможности в дизайне. В Microsoft Office 2007 программа FrontPage была заменена на Microsoft Expression Web и Microsoft Office SharePoint Designer.

Позднее в Microsoft Office 2010 и Microsoft Office 2013 программа заменена на Microsoft Office SharePoint Designer.

К более продвинутым редакторам можно отнести HomeSite+ и Dreamweaver. Данное ПО позволяет создать как сайт-визитку, так и настоящий шедевр современного искусства. К минусам этого ПО можно отнести долгую обработку массивного кода. Кроме того, требуется знание HTML на порядок выше, чем при работе с SharePoint Designer.

Adobe HomeSite — редактор HTML, владельцем которого в настоящее время является Adobe Systems, не является WYSIWYG-редактором, таким как FrontPage или Dreamweaver, и в настоящее время существует только версия, названная HomeSite+, которая включена в пакет Dreamweaver MX 2004 и выше. HomeSite+ имеет дополнительные функции для прикладного расширения.

Программа Dreamweaver наибольшую популярность получила, начиная с версии MX, выпущенной компанией Macromedia в 2002 году. Седьмая версия программы получила название Dreamweaver MX 2004, а затем в 2005 г. была выпущена Dreamweaver 8. В 2007 г. уже компания Adobe выпустила более свежую версию под названием Dreamweaver CS3. Последней версией считается Dreamweaver CC 2019.

Популярная в настоящее время также программа Notepad++, которая является бесплатным редактором HTML и разрабатывается open Source-сообществом энтузиастов, обладает мультиязычной поддержкой, подсвечивает синтаксис самых распространенных языков программирования (например PHP, JavaScript, Python и др.).

Интерфейс данной программы выполнен по принципу вкладок, благодаря чему можно работать сразу с несколькими документами одновременно.

Для выполнения курсовой работы необходимы следующие инструменты: текстовый редактор, браузер для просмотра результатов и валидатор — программа для проверки синтаксиса HTML и выявления ошибок в коде. Рассмотрим данные инструменты подробнее.

Текстовый редактор — программа или компонент программного комплекса, предназначенная для создания и изменения текстовых данных в текстовых файлах.

HTML-документ можно создать практически в любом текстовом редакторе, даже в Блокноте, однако при выборе текстового редактора необходимо обратить внимание на то, чтобы в нем поддерживались следующие возможности:

– подсветка синтаксиса (выделение тегов, текста, ключевых слов и параметров разными цветами);

– работа с вкладками и проверка текущего документа на ошибки.

Браузер — программа, предназначенная для просмотра web-страниц.

Для выполнения курсовой работы подойдет любой браузер. Наибольшей популярностью сегодня пользуются три браузера:

– Chrome — браузер, разрабатываемый компанией Google на основе свободного браузера Chromium и движка Blink;

– Firefox — свободный браузер на движке Gecko, разработкой и распространением которого занимается Mozilla Corporation;

– Safari — веб-браузер, разработанный корпорацией Apple и входящий в состав macOS и iOS.

Валидатор позволяет пользователям проверять документы HTML и XHTML на предмет правильной разметки для повышения технического качества web-страниц.

Валидация HTML-документа предназначена для выявления ошибок в синтаксисе web-страницы и расхождений со спецификацией HTML. Если есть доступ в Интернет, то можно проверить созданный документ онлайн с помощью службы валидации разметки Консорциума World Wide Web, введя в специальной форме путь к проверяемому документу или сайту. После проверки будут показаны возможные ошибки или появится надпись, что документ прошел валидацию успешно.

Для проверки локального HTML-файла или при отсутствии подключения к Интернету предназначена компьютерная программа и библиотека HTML Tidy, целью которой является исправление неверного HTML и улучшение внешнего вида и стиля отступов полученной разметки. Некоторые редакторы уже содержат встроенный модуль Tidy, и валидацию документа можно провести без привлечения дополнительных инструментов.

Создавать web-страницы достаточно просто. Для закрепления материала без целенаправленного изучения синтаксиса HTML проделаем следующие действия:

1) открываем программу Блокнот или любой другой редактор кода;

2) набираем код, представленный на рис. 1;

3) сохраняем готовый документ (Файл → Сохранить как...) под любым именем с расширением файла .html;

4) запускаем браузер и открываем с его помощью созданный файл.

Если все сделано правильно, то в браузере увидим результат, представленный на рис. 2.

Рис. 2. Отображение страницы в браузере

Вместе с тем, чтобы создать более сложную структуру разметки документа, в HTML существуют десятки различных тегов, атрибутов, селекторов. HTML — это только «каркас» web-страницы. Чтобы сделать страницу более красивой и добавить элементы анимации, необходимо воспользоваться CSS — формальным языком описания внешнего вида документа, написанного с использованием языка разметки.

Практическое задание 1

1. Создать структурные HTML-страницы для прототипа информационной системы.
2. Проверить их с помощью валидатора; при необходимости исправить все ошибки.

Темы для самостоятельного изучения

1. Быстрое прототипирование web-страниц с использованием различных библиотек кода (Bootstrap).
2. Средства оптимизации и минимизации HTML-кода.
3. Адаптивная верстка с помощью Flexbox, Grid.

Вопросы для самоконтроля

1. Что такое HTML?
2. Какие существуют виды редакторов HTML?
3. Что такое валидатор?
4. Что такое HTML Tidy?
5. Для чего нужен браузер?
6. Сколько существует HTML-тегов?
7. Основные инструменты, используемые при создании сайтов.
8. Наиболее популярные на сегодняшний день браузеры.

2. ВВЕДЕНИЕ В CSS. АНИМАЦИЯ

CSS (англ. *Cascading Style Sheets* — каскадные таблицы стилей) — это набор параметров форматирования, который применяется к элементам документа, чтобы изменить их внешний вид. CSS представляет собой мощную систему, расширяющую возможности дизайна и верстки web-страниц. Для добавления стилей на web-страницу существует несколько способов, которые различаются своими возможностями и назначением. Далее рассмотрим их подробнее.

Связанные стили

При использовании связанных стилей описание селекторов и их значений располагается в отдельном файле с расширением .css, а для связывания документа с этим файлом применяется тег <link>. Данный тег помещается в контейнер <head>, как показано на рис. 3.

```
<html>

  <head>
    <meta content="text/html; charset=utf-8">
    <title>Прототип информационной системы</title>
    <link rel="stylesheet" href="/main.css">
  </head>

  <body>
    <h1>
      Заголовок страницы
    </h1>
    <div>
      Содержание страницы
    </div>
  </body>

</html>
```

Рис. 3. Подключение связанных стилей

Значение атрибута тега <link> — rel остается постоянным. Значение href задает путь к CSS-файлу и может быть задано как относительно, так и абсолютно. Таким образом, существует возможность подключить таблицу стилей, которая находится на другом домене. Содержимое файла main.css, подключенного с помощью тега <link>, приведено на рис. 4.

```
CSS ▼

1  h1 {
2    color: #000;
3    font-family: Arial, Verdana, sans-serif;
4    text-align: center;
5  }
6
7  p {
8    padding-left: 20px;
9  }
```

Рис. 4. Содержание файла main.css

Файл со стилем не хранит никаких данных, кроме синтаксиса CSS, а HTML-документ содержит только ссылку на файл со стилем — таким способом реализован принцип разделения кода и оформления сайта. Использование связанных стилей является наиболее универсальным и удобным методом добавления стиля на сайт. Необходимо помнить, что стили хранятся в одном файле, а в HTML-документах указывается только ссылка на него.

Глобальные стили

При использовании глобальных стилей свойства CSS описываются в самом документе и располагаются в заголовке web-страницы. По своей гибкости и возможностям этот способ добавления стиля уступает связанному, но позволяет хранить стили в одном месте, в данном случае прямо на той же странице с помощью контейнера <style>, как показано на рис. 5.

```
<html>

  <head>
    <meta content="text/html; charset=utf-8">
    <title>Прототип информационной системы</title>
    <link rel="stylesheet" href="/main.css">

    <style>
      h1 {
        font-family: Arial, Roboto, sans-serif;
        color: #333;
        text-align: left
      }
    </style>

  </head>

  <body>
    <h1>
      Заголовок страницы
    </h1>
    <div>
      Содержание страницы
    </div>
  </body>

</html>
```

Рис. 5. Пример использования глобального стиля

CSS-стили, подключенные описанным выше способом, загружаются при каждой повторной загрузке страницы, т.е. могут оказывать влияние на скорость загрузки, поэтому на практике данный способ используют очень редко и только в исключительных случаях. К примеру, если необходимо отправить заказчику шаблон страницы, то проще будет предоставить предварительный результат, представленный на одной странице.

Внутренние стили

Внутренний стиль является расширением для одиночного тега, используемого на текущей web-странице. Для определения стиля используется атрибут style, а его значением выступает набор стилевых правил (рис. 6).

Рис. 6. Пример использования внутреннего стиля

Внутренние стили рекомендовано применять ограниченно или лучше вообще отказаться от их использования. Применение таких стилей увеличивает общий объем файлов, что в свою очередь ведет к повышению времени загрузки страницы в браузере, а также усложняет внесение изменений. Однако в случае, если у вас нет доступа к CSS-файлам или необходимо применить правила только для одного элемента, рекомендовано использовать данный вид стиля.

Описанные методы использования CSS могут быть применены в сочетании друг с другом, но в этом случае необходимо помнить об их иерархии. Первым имеет приоритет внутренний стиль, затем глобальный и в последнюю очередь — связанный.

Импорт CSS

В случае если у вас уже есть готовый CSS-файл с описанными стилями, то можно не копировать код, а импортировать содержимое CSS-файла с помощью команды @import.

Этот метод допускается использовать совместно со связанными или глобальными стилями, но нельзя использовать с внутренними стилями.

Пример синтаксиса:

```
@import url("имя файла");
@import "имя файла";
```

После ключевого слова @import указывается путь к стилевому файлу одним из двух приведенных способов — с помощью url или без него. На рис. 7 и 8 показано, как можно импортировать стили.

```HTML
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Импорт</title>
  <style>
   @import url("style/header.css");
   h1 {
    font-family: Arial, Helvetica, sans-serif;
    color: red;
   }
  </style>
 </head>
 <body>
  <h1>Заголовок 1</h1>
  <h2>Заголовок 2</h2>
 </body>
</html>
```

Рис. 7. Пример использования импорта CSS

```CSS
@import "/style/side.css";
@import "/style/footer.css";

body {
 font-family: Arial, Verdana, Helvetica, sans-serif;
 background: white;
 color: black;
}
```

Рис. 8. Пример использования импорта CSS в связанном CSS-файле

Анимация с помощью CSS

Все современные браузеры поддерживают переходы CSS transitions и CSS animations, которые позволяют реализовать анимацию средствами CSS без использования скриптов, написанных на JavaScript.

CSS transitions предоставляют способ контролировать скорость анимации при изменении CSS-свойств, т.е. некоторое свойство будет анимироваться при помощи специальных CSS-правил. Далее, при изменении этого свойства браузер будет сам обрабатывать анимацию.

Например, CSS, представленный ниже, три секунды анимирует свойство background-color.

```CSS
.animated {
 transition-property: background-color;
 transition-duration: 3s;
}
```

Существует четыре основных свойства, задающих анимацию:

1) transition-property;
2) transition-duration;
3) transition-timing-function;
4) transition-delay.

Далее рассмотрим их подробнее, однако стоит заметить, что общее свойство transition может перечислять их все в порядке: property duration timing-function delay, а также задавать анимацию нескольких свойств сразу.

transition-property

Список свойств, которые будут анимироваться, например: left, margin-left, height, color. Стоит помнить, что анимировать можно не все свойства. Список свойств, которые можно анимировать, возможно узнать в справочниках или официальном руководстве Консорциума Всемирной паутины W3C.

Заметим при этом, что значение all означает «анимировать все свойства».

transition-duration

Продолжительность анимации, задается в формате CSS time, т.е. в секундах (s) или миллисекундах (ms).

transition-delay

Задержка до анимации. Например, если transition-delay: 1s, то анимация начнется через 1 с после смены свойства.

Можно указывать отрицательные значения, при этом анимация начнется с середины.

transition-timing-function

Временная функция, которая задает, как процесс анимации будет распределен во времени. Например, начнется ли анимация медленно, чтобы потом ускориться или наоборот. Самое сложное, но при детальном изучении — вполне очевидное свойство. У него есть два основных вида значения: кривая Безье и «по шагам» [1].

Сложные анимации делаются объединением простых при помощи CSS-правила @keyframes.

В нем задается «имя» анимации и правила: что, откуда и как анимировать. Затем при помощи свойства «animation: имя параметры» эта анимация подключается к элементу, задается время анимации и дополнительные параметры, как ее применять. Пример кода приведен на рис. 9.

```css
@keyframes go-left-right {
  from {
    left: 0px;
  }
  to {
    left: calc(100% - 50px);
  }
}


.progress {
  animation: go-left-right 3s infinite alternate;
  position: relative;
  border: 2px solid green;
  width: 50px;
  height: 20px;
  background: lime;
}
```

Рис. 9. Пример анимации с помощью @keyframes

Практическое задание 2

1. Создать CSS-стиль для прототипа информационной системы.
2. Добавить анимированный «загрузчик» для прототипа информационной системы.
3. Подключить стили для всех страниц прототипа информационной системы.

Темы для самостоятельного изучения

1. Средства оптимизации и минимизации CSS-кода.
2. Препроцессоры SASS, LESS.

Вопросы для самоконтроля

1. Что такое CSS?
2. Каким образом можно подключить CSS?
3. Основные свойства, задающие анимацию.
4. Импорт CSS.

3. ДИНАМИЧЕСКИЕ WEB-РЕСУРСЫ

Динамический web-ресурс — это ресурс, страницы которого сгенерированы программно. В отличие от статичных страниц, являющихся файлами, хранящимися на сервере, web-сервер генерирует HTML-код для обработки браузером. Различают статические и динамические HTML-страницы.

Статические HTML-страницы: страница выглядит всегда одинаково, независимо от действий пользователя. Например, меню организовано ссылками на отдельные страницы, а не выпадающим списком.

Динамические HTML-страницы: такие страницы уже могут реагировать на действия пользователя и изменяться. Например, при щелчке по тексту может показываться всплывающий блок текста с переводом слова.

Динамика web-ресурсов реализована при помощи скриптов, которые выполняются браузером. Многие элементы языка HTML поддерживают определение обработчиков событий. Например, можно задать обработку события «нажатия кнопки мыши» на картинке. Тогда, если пользователь кликнет на картинку, вызовется определенный для этого обработчик.

Самый распространенный язык для создания динамики web-ресурсов — JavaScript.

Если web-ресурс содержит часто меняющийся контент (англ. *content* — информация на ресурсе), то необходимо использовать скрипты, выполняющиеся на сервере. Работает это следующим образом:

1) браузер запрашивает у сервера документ;

2) сервер определяет, что документ является скриптом и запускает его на выполнение;

3) скрипт генерирует HTML-страницу;

4) сервер отправляет сгенерированную страницу браузеру.

Существует несколько языков программирования, на которых могут быть написаны скрипты, генерирующие «динамические» страницы. Самые распространенные из них:

– PHP — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений;

– Python — высокоуровневый язык программирования общего назначения с минималистичным синтаксисом, ориентированный на повышение производительности разработчика и читаемости кода;

– Java — строго типизированный объектно-ориентированный язык программирования.

Каждый из этих языков имеет свои особенности применения. Писать скрипты можно на любом языке. Главное, знать его сильные и слабые стороны и использовать их эффективно.

Стоит заметить, что сейчас достаточно популярна гибридная система AJAX (Asyncronous JavaScript And XML). Эта технология позволяет скриптам на JavaScript обращаться к какому-либо скрипту на сервере и получать информацию с него, что, в свою очередь, дает пользователю гибкость и позволяет перезагружать только часть содержимого страницы, а не полностью всю страницу.

JavaScript

JavaScript проектировался для того, чтобы сделать web-ресурсы «живыми». Программы на этом языке называются скриптами. В браузере они подключаются напрямую к HTML и, как только загружается страница, тут же выполняются. Программы на JavaScript — обычный текст, не требующий какой-то специальной подготовки. В этом плане JavaScript сильно отличается от Java.

JavaScript может выполняться не только в браузере, но и с помощью специальной программы — интерпретатора. Процесс выполнения скрипта называют *интерпретацией*.

Для выполнения программ, неважно на каком языке, существуют два способа: «компиляция» и «интерпретация».

Компиляция — когда исходный код программы при помощи специального инструмента иной программы, которая называется «компилятор», преобразуется в другой язык, как правило, — в машинный код. Этот машинный код затем распространяется и запускается. Вместе с тем исходный код программы остается у разработчика.

Интерпретация — когда исходный код программы получает другой инструмент, который называют «интерпретатор», и выполняет его «как есть». При этом распространяется именно сам исходный код (скрипт). Такой подход применяется в браузерах для JavaScript.

Современные интерпретаторы перед выполнением преобразуют JavaScript в машинный код или близко к нему оптимизируют, а уже затем выполняют. И даже во время выполнения стараются оптимизировать, поэтому JavaScript работает очень быстро.

Во все основные браузеры встроен интерпретатор JavaScript, именно поэтому они могут выполнять скрипты на странице. Но, разумеется, JavaScript можно использовать не только в браузере. Это полноценный и «безопасный» язык программирования общего назначения, который, однако, не предоставляет низкоуровневых средств работы с памятью и процессором.

Что же касается остальных возможностей — они зависят от окружения, в котором запущен JavaScript. В браузере JavaScript умеет делать все, что относится к манипуляции со страницей, взаимодействию с посетителем и в какой-то мере с сервером:

– создавать новые HTML-теги, удалять существующие, менять стили элементов, прятать, показывать элементы и т.п.;

– реагировать на действия посетителя, обрабатывать клики мыши, перемещения курсора, нажатия на клавиатуру и т.п.;

– посылать запросы на сервер и загружать данные без перезагрузки страницы (с помощью AJAX);

– получать и устанавливать cookie, запрашивать данные, выводить сообщения.

JavaScript — быстрый и мощный язык, но браузер накладывает на его исполнение некоторые ограничения. Это сделано для безопасности пользователей, чтобы злоумышленник не мог с помощью JavaScript получить личные данные или как-то навредить компьютеру пользователя. Ограничений нет там, где JavaScript используется вне браузера, например на сервере. Большинство возможностей JavaScript в браузере ограничено текущим окном и страницей.

JavaScript не может читать / записывать произвольные файлы на жесткий диск, копировать их или вызывать программы. Он не имеет прямого доступа к операционной системе.

Современные браузеры могут работать с файлами, но эта возможность ограничена специально выделенной директорией, работающий в одной вкладке; не могут общаться с другими вкладками и окнами, за исключением случая, когда они сами открыли это окно или несколько вкладок из одного источника (одинаковый домен, порт, протокол).

Основные особенности JavaScript:

– полная интеграция с HTML / CSS;

– поддерживается всеми распространенными браузерами и включен по умолчанию.