МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Воронежский государственный технический университет»

Кафедра автоматизированного оборудования
машиностроительного производства

ИНФОРМАТИКА

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ для студентов
направления 15.03.05 «Конструкторско-технологическое обеспечение
машиностроительных производств» (профили «Технология
машиностроения», «Металлообратывающие станки и комплексы»
и «Конструкторско-технологическое обеспечение
кузнечно-штамповочного производства»), всех форм обучения

Воронеж 2021

УДК 621.01(07)
ББК 34.5я7

Составитель канд. техн. наук, доц. Д. Е. Пачевский

Информатика: методические указания к выполнению лабораторных работ для студентов направления 15.03.05 «Конструкторско-технологическое обеспечение машиностроительных производств» (профили «Технология машиностроения», «Металлообратывающие станки и комплексы» и «Конструкторско-технологическое обеспечение кузнечно-штамповочного производства»), всех форм обучения / ФГБОУ ВО «Воронежский государственный технический университет»; сост.: Д. Е. Пачевский. – Воронеж: Изд-во ВГТУ, 2021. – 23 с.

В методических указаниях изложены общие вопросы к выполнению лабораторных работ, сформулированы задания и представлен теоретический материал. Выполнение лабораторных работ дает студентам возможность освоить офисные технологии и программные средства, изучить их возможности, основы алгоритмизации, формы представления данных и форматы при работе с информацией.

Предназначены для студентов, обучающихся по направлению 15.03.05 «Конструкторско-технологическое обеспечение машиностроительных производств» (профили «Технология машиностроения», «Металлообратывающие станки и комплексы» и «Конструкторско-технологическое обеспечение кузнечно-штамповочного производства»), всех форм обучения.

Методические указания подготовлены в электронном виде и содержатся в файле МУ_ЛР_Информ.pdf.

Библиогр.: 6 назв.

УДК 621.01(07)
ББК 34.5я7

Рецензент – С. Ю. Жачкин, д-р техн. наук, проф. кафедры автоматизированного оборудования машиностроительного производства ВГТУ

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

ВВЕДЕНИЕ

В настоящее время информатика и ее практические результаты становятся важнейшим двигателем научно-технического прогресса и развития человеческого общества.

Скорость развития средства обработки и передачи информации поразительна, в истории человечества этому бурно развивающемуся процессу нет аналога. Сведения, касающиеся прикладной области быстро устаревают. На смену одним технологиям приходят другие, более совершенные и более сложные. Специалисты в области информационных технологий должны непрерывно обучаться и повышать свою квалификацию. Однако иметь теоретические и практические знания в области информатики в наше время стало необходимостью для всех, потому что общество, в котором мы живем, является информационным обществом.

Кроме того, решение ряда производственных задач нередко требует от инженера знания офисных технологий и программных средств, их возможностей, основ алгоритмизации, форм представления данных и форматов при работе с информацией. Перечисленные выше проблемы решаются в рамках изучаемой дисциплины. В рассматриваемых методических указаниях приведен краткий теоретический материал, дополняющий лекционный курс по изучению указанной выше дисциплине и даны задания для выполнения лабораторных работ. Требования составлены в соответствии с государственным стандартом и учебным планом выпускающей кафедры.

ЛАБОРАТОРНАЯ РАБОТА № 1
ИЗУЧЕНИЕ ТЕГОВ HTML, ФОРМАТИРОВАНИЕ ДОКУМЕНТА

Цель работы: получить базовые знания и приобрести практические навыки работы по форматированию гипертекстовых документов.

Задачи изучения и приобретения практических навыков:

1. Изучить назначение и атрибуты тегов для форматирования текстовых блоков, применить на практике полученные знания.
2. Изучить назначение и атрибуты тегов для форматирования блочных элементов, применить на практике полученные знания.

Теоретические сведения

HTML (Hypertext Markup Language) – это код, который используется для структурирования и отображения веб-страницы и её контента. Например, контент может быть структурирован внутри множества параграфов, маркированных списков или с использованием изображений и таблиц данных. Как видно из названия, эта статья даст вам базовое понимание HTML и его функций.

HTML не является языком программирования; это язык разметки, и используется, чтобы сообщать вашему браузеру, как отображать веб-страницы, которые вы посещаете. Он может быть сложным или простым, в зависимости от того, как хочет веб-дизайнер. HTML состоит из ряда элементов, которые вы используете, чтобы вкладывать или оборачивать различные части контента, чтобы заставить контент отображаться или действовать определённым образом. Ограждающие теги могут сделать слово или изображение ссылкой на что-то ещё, могут сделать слова курсивом, сделать шрифт больше или меньше и так далее.

Главными частями HTML элемента являются:

– открывающий тег (Opening tag). Состоит из имени элемента (в данном случае, "p"), заключённого в открывающие и закрывающие угловые скобки. Открывающий тег указывает, где элемент начинается или начинает действовать, в данном случае — где начинается абзац.

– закрывающий тег (Closing tag). Это то же самое, что и открывающий тег, за исключением того, что он включает в себя косую черту перед именем элемента. Закрывающий элемент указывает, где элемент заканчивается, в данном случае — где заканчивается абзац. Отсутствие закрывающего тега является одной из наиболее распространённых ошибок начинающих и может приводить к странным результатам.

– контент (Content). Это контент элемента, который в данном случае является просто текстом.

– элемент(Element). Открывающий тег, закрывающий тег и контент вместе составляют элемент.

Элементы также могут иметь атрибуты. Атрибуты содержат дополнительную информацию об элементе, которую вы не хотите показывать в фактическом контенте. В данном случае, class это имя атрибута, a editor-note это значение атрибута. Класс позволяет дать элементу идентификационное имя, которое может позже использоваться, чтобы обращаться к элементу с информацией о стиле и прочих вещах.

Атрибут всегда должен иметь:

– пробел между ним и именем элемента (или предыдущим атрибутом, если элемент уже имеет один или несколько атрибутов).

– имя атрибута, за которым следует знак равенства.

– значение атрибута, заключённое с двух сторон в кавычки.

Элементы должны открываться и закрываться правильно, поэтому они явно располагаются внутри или снаружи друг друга. Если они перекрываются, как в примере выше, ваш веб-браузер будет пытаться сделать наилучшее предположение на основе того, что вы пытались сказать, что может привести к неожиданным результатам. Так что не стоит этого делать!

Пустые элементы.

Некоторые элементы не имеют контента, и называются пустыми элементами. Возьмём элемент <img>, который уже имеется в нашем HTML:

<img src="images/firefox-icon.png" alt="Моё тестовое изображение">

Он содержит два атрибута, но не имеет закрывающего тега </img>, и никакого внутреннего контента. Это потому, что элемент изображения не оборачивает контент для влияния на него. Его целью является вставка изображения в HTML страницу в нужном месте.

Анатомия HTML документа.

Мы завершили изучение основ отдельных HTML элементов, но они не очень полезны сами по себе. Теперь мы посмотрим, как отдельные элементы объединяются в целую HTML страницу. Давайте вернёмся к коду, который мы записывали в наш index.html (с которым мы впервые встретились в статье Работа с файлами):

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Моя тестовая страница</title>
  </head>
  <body>
    <img src="images/firefox-icon.png" alt="Моё тестовое изображение">
  </body>
</html>
```

Здесь мы имеем:

<!DOCTYPE html> — доктайп. В прошлом, когда HTML был молод (около 1991/1992), доктайпы должны были выступать в качестве ссылки на набор правил, которым HTML страница должна была следовать, чтобы считаться хорошим HTML, что могло означать автоматическую проверку ошибок и другие полезные вещи. Однако в наши дни, никто не заботится об этом, и они на самом деле просто исторический артефакт, который должен быть включён для того, что бы все работало правильно. На данный момент это все, что вам нужно знать.

<html></html> — элемент <html>. Этот элемент оборачивает весь контент на всей странице, и иногда известен как корневой элемент.

<head></head> — элемент <head>. Этот элемент выступает в качестве контейнера для всего, что вы пожелаете включить на HTML страницу, но не являющегося контентом, который вы показываете пользователям вашей страницы. К ним относятся такие вещи, как ключевые слова и описание страницы, которые будут появляться в результатах поиска, CSS стили нашего контента, кодировка и многое другое.

<body></body> — элемент <body>. В нем содержится весь контент, который вы хотите показывать пользователям, когда они посещают вашу страницу, будь то текст, изображения, видео, игры, проигрываемые аудиодорожки или что-то ещё.

<meta charset="utf-8"> — этот элемент устанавливает UTF-8 кодировку вашего документа, которая включает в себя большинство символов из всех известных человечеству языков. По сути, теперь документ может обрабатывать любой текстовый контент, который вы в него вложите. Нет причин не устанавливать её, так как это может помочь избежать некоторых проблем в дальнейшем.

<title></title> — элемент <title>. Этот элемент устанавливает заголовок для вашей страницы, который является названием, появляющимся на вкладке браузера загружаемой страницы, и используется для описания страницы, когда вы добавляете её в закладки/избранное.

Задания на самостоятельную работу

Отформатировать документ, полученный у преподавателя, таким образом, чтобы в браузере отображалась исходная структура текстовых блоков, использовать все изученные на лекции HTML теги. Создать блочные элементы и поместить туда соответствующее исходное содержимое.

Отчет должен содержать в себе следующие разделы:

– название работы;

– цель работы и задачи;

– описание процесса выполнения работы с иллюстрациями (скриншотами);

– выводы о проделанной работе.

ЛАБОРАТОРНАЯ РАБОТА № 2
ИЗУЧЕНИЕ ТЕГОВ HTML, ФОРМАТИРОВАНИЕ ДОКУМЕНТА

Цель работы: получить базовые знания и приобрести практические навыки работы по работе с каскадными таблицами стилей CSS

Задачи изучения и приобретения практических навыков:

1. Изучить синтаксис CSS.
2. Изучить назначение и применения классов и идентификаторов в CSS.
3. Изучить применения псевдоклассов CSS.
4. Изучить назначение команд CSS для форматирования HTML элементов, применить на практике полученные знания.

Теоретические сведения

CSS (Cascading Style Sheets) — это код, который вы используете для стилизации вашей веб-страницы. *Основы CSS* помогут вам понять, что вам нужно для начала работы. Как и HTML, CSS на самом деле не является языком программирования. Это не язык разметки - это *язык таблицы стилей.* Это означает, что он позволяет применять стили выборочно к элементам в документах HTML.

Вся структура называется **набором правил** (но зачастую для краткости "правило"). Отметим также имена отдельных частей:

Селектор (Selector)

Имя HTML-элемента в начале набора правил. Он выбирает элемент(ы) для применения стиля (в данном случае, элементы p). Для стилизации другого элемента, просто измените селектор.

Объявление (Declaration)

Единственное правило, например color: red; указывает, какие из **свойств** элемента вы хотите стилизовать.

Свойства (Properties)

Способы, которыми вы можете стилизовать определённый HTML-элемент (в данном случае, color является свойством для элементов <p>). В CSS вы выбираете, какие свойства вы хотите затронуть в вашем правиле.

Значение свойства (Property value)

Справа от свойства, после двоеточия, у нас есть значение свойства, которое выбирает одно из множества возможных признаков для данного свойства (существует множество значений color, помимо red).

Обратите внимание на важные части синтаксиса:

Каждый набор правил (кроме селектора) должен быть обёрнут в фигурные скобки ({}).

В каждом объявлении необходимо использовать двоеточие (:), чтобы отделить свойство от его значений.

В каждом наборе правил вы должны использовать точку с запятой (;), чтобы отделить каждое объявление от следующего.

Таким образом, чтобы изменить несколько значений свойств сразу, вам просто нужно написать их, разделяя точкой с запятой, например так:

```
p {
  color: red;
  width: 500px;
  border: 1px solid black;
}
```

Выбор нескольких элементов

Вы также можете выбрать несколько элементов разного типа и применить единый набор правил для всех из них. Добавьте несколько селекторов, разделённых запятыми. Например:

```
p,li,h1 {
```

```
  color: red;
}
```

Теперь, когда мы изучили некоторые основы CSS, давайте добавим ещё несколько правил и информацию в наш файл style.css, чтобы наш пример хорошо выглядел. Прежде всего, давайте сделаем, чтобы наши шрифты и текст выглядели немного лучше.

Прежде всего, вернитесь и найдите вывод из Google Fonts, который вы уже где-то сохранили. Добавьте элемент <link> где-нибудь внутри шапки вашего index.html (снова, в любом месте между тегами <head> и </head>). Это будет выглядеть примерно так:

```
<link href='http://fonts.googleapis.com/css?family=Open+Sans' rel='stylesheet' type='text/css'>
```

Этот код связывает вашу страницу с таблицой стилями, которая загружает семейство шрифтов Open Sans вместе с вашей страницей и позволяет вам применять их к вашим HTML-элементам используя свою собственную таблицу стилей.

Затем, удалите существующее правило в вашем style.css файле. Это был хороший тест, но красный текст, на самом деле, не очень хорошо выглядит.

Добавьте следующие строки в нужное место, заменив строку placeholder актуальной font-family строкой, которую вы получили из Google Fonts. (font-family просто означает, какой шрифт(ы) вы хотите использовать для вашего текста). Это правило устанавливает глобальный базовый шрифт и размер шрифта для всей страницы (поскольку <html> является родительским элементом для всей страницы, и все элементы внутри него наследуют такой же font-size и font-family):

```
html {
  font-size: 10px; /* px значит 'пиксели': базовый шрифт будет 10 пикселей в высоту  */
  font-family: placeholder: здесь должно быть имя шрифта из Google fonts
}
```

Примечание: Все в CSS документе между /* и */ является CSS комментарием, который браузер игнорирует при исполнении кода. Это место, где вы можете написать полезные заметки о том, что вы делаете.

Теперь мы установим размер шрифта для элементов, содержащих текст внутри HTML тела (<h1> (en-US), <li>, и <p>). Мы также отцентрируем текст нашего заголовка и установим некоторую высоту строки и расстояние между буквами в теле документа, чтобы сделать его немного более удобным для чтения:

```
h1 {
  font-size: 60px;
  text-align: center;
}
```

```
p, li {
  font-size: 16px;
  line-height: 2;
  letter-spacing: 1px;
}
```

Позиционирование и стилизация нашего заголовка главной страницы

```
h1 {
  margin: 0;
  padding: 20px 0;
  color: #00539F;
  text-shadow: 3px 3px 1px black;
}
```

Вы, возможно, заметили, что есть ужасный разрыв в верхней части тела. Это происходит, потому что браузеры применяют некоторый стиль по умолчанию для элемента <h1> (en-US) (по сравнению с другими), даже если вы не применяли какой-либо CSS вообще! Это может звучать как плохая идея, но мы хотим, чтобы веб-страница без стилей имела базовую читаемость. Чтобы избавиться от разрыва, мы переопределили стиль по умолчанию, установив margin: 0;.

Затем мы установили заголовку верхний и нижний padding на 20 пикселей, и сделали текст заголовка того же цвета, как и цвет фона html.

Здесь, мы использовали одно довольно интересное свойство - это text-shadow, которое применяет тень к текстовому контенту элемента. Оно имеет следующие четыре значения:

Первое значение пикселей задаёт горизонтальное смещение тени от текста — как далеко она движется поперёк: отрицательное значение должно двигать её влево.

Второе значение пикселей задаёт вертикальное смещение тени от текста — как далеко она движется вниз, в этом примере: отрицательное значение должно переместить её вверх.

Третье значение пикселей задаёт радиус размытия тени — большее значение будет означать более размытую тень.

Четвёртое значение задаёт основной цвет тени.

И вновь попробуйте поэкспериментировать с различными значениями, чтобы посмотреть, что вы можете придумать.

Центрирование изображения

```
img {
  display: block;
  margin: 0 auto;
}
```

В заключение мы отцентрируем изображение, чтобы оно лучше выглядело. Мы можем использовать margin: 0 auto уловку снова, как мы это делали раньше для body, но мы также должны сделать кое-что ещё. Элемент <body>

является блочным, это значит, что он занимает место на странице и может иметь margin и другие значения отступов, применяемых к нему. Изображения, наоборот, являются строчными элементами, то есть они этого не могут. Таким образом, чтобы применить margin к изображению, мы должны дать изображению блочное поведение с помощью display: block;.

Задания на самостоятельную работу

Отформатировать документ, полученный у преподавателя, таким образом, чтобы в браузере отображался внешний вид документа, представленный в исходном задании. Используйте дополнительный файлы изображений из пакета задания, а так же справочник команд CSS.

Отчет должен содержать в себе следующие разделы:

– название работы;

– цель работы и задачи;

– описание процесса выполнения работы с иллюстрациями (скриншотами);

– выводы о проделанной работе.

ЛАБОРАТОРНАЯ РАБОТА № 3
ИЗУЧЕНИЕ ОСНОВ JAVASCRIPT

Цель работы: получить базовые знания и приобрести практические навыки работы по работе с языком JavaScript.

Задачи изучения и приобретения практических навыков:

1. Изучить синтаксис JavaScript.
2. Применить полученные знания на практике.

Теоретические сведения

JavaScript – это язык программирования, который добавляет интерактивность на ваш веб-сайт (например: игры, отклик при нажатии кнопок или при вводе данных в формы, динамические стили, анимация).

JavaScript ("JS" для краткости) — это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах. Его разработал Brendan Eich, сооснователь проекта Mozilla, Mozilla Foundation и Mozilla Corporation.

JavaScript невероятно универсален и дружелюбен к новичкам. Обладая большим опытом, вы сможете создавать игры, анимированную 2D и 3D графику, полномасштабные приложения с базами данных и многое другое!

JavaScript сам по себе довольно компактный, но очень гибкий. Разработчиками написано большое количество инструментов поверх основного языка JavaScript, которые разблокируют огромное количество дополнительных функций с очень небольшим усилием. К ним относятся:

Программные интерфейсы приложения (API), встроенные в браузеры, обеспечивающие различные функциональные возможности, такие как динамическое создание HTML и установку CSS стилей, захват и манипуляция видеопотоком, работа с веб-камерой пользователя или генерация 3D графики и аудио сэмплов.

Сторонние API позволяют разработчикам внедрять функциональность в свои сайты от других разработчиков, таких как Twitter или Facebook.

Также вы можете применить к вашему HTML сторонние фреймворки и библиотеки, что позволит вам ускорить создание сайтов и приложений.

Тем не менее, с JavaScript немного более сложно освоиться, чем с HTML и CSS.

Переменные — это контейнеры, внутри которых вы можете хранить значения. Вы начинаете с того, что объявляете переменную с помощью ключевого слова var (не рекомендуется, продолжайте читать, чтобы получить объяснения) или let, за которым следует любое имя, которым вы захотите её назвать:

```
let myVariable;
```

После объявления переменной вы можете присвоить ей значение:

```
myVariable = 'Bob';
```

Вы можете сделать обе эти операции на одной и той же строке, если вы захотите:

```
var myVariable = 'Bob';
```

Вы можете получить значение, просто вызвав переменную по имени:

```
myVariable;
```

После установки значения переменной вы можете изменить его позже:

```
var myVariable = 'Bob';
myVariable = 'Steve';
```

Так для чего нам нужны переменные? Что ж, переменные должны были сделать что-нибудь интересное в программировании. Если значения не могли бы изменяться, то вы не могли бы ничего сделать динамическим, например, персонализировать приветственное сообщение или сменить изображение, отображаемое в галерее изображений.

Комментарии - это, по сути, короткие фрагменты текста, которые могут быть добавлены в код, и которые игнорируются браузером. Вы можете поместить комментарии в JavaScript-код, так же как вы делали это в CSS:

```
/*
    Всё, что находится тут - комментарий.
*/
```

Если ваш комментарий не содержит переноса строк, то зачастую легче поставить две косые черты, как тут:

// Это комментарий

Операторы - это математический символ, который производит результат, основанный на двух значениях (или переменных). В приведённой ниже таблице вы можете увидеть некоторые из наиболее простых операторов, наряду с некоторыми примерами, которые опробуете в JavaScript консоли.

Условия — это кодовые структуры, которые позволяют вам проверять, истинно или ложно выражение, а затем выполнить другой код в зависимости от результата. Самая распространённая форма условия называется, if ... else. Например:

```
var iceCream = 'chocolate';
if (iceCream === 'chocolate') {
  alert('Yay, I love chocolate ice cream!');
} else {
  alert('Awwww, but chocolate is my favorite...');
}
```

Выражение внутри if (...) — это проверка, которая использует тождественный оператор (как описано выше), чтобы сравнить переменную iceCream со строкой chocolate и увидеть равны ли они. Если это сравнение возвращает true, выполнится первый блок кода. Если нет, этот код пропустится и выполнится второй блок кода, после инструкции else.

Функции - способ упаковки функциональности, которую вы хотите использовать повторно. Всякий раз, когда вам нужна определённая процедура, вы можете просто вызвать функцию по её имени, а не переписывать весь код каждый раз. Вы уже видели некоторые функции, описанные выше, например:

```
var myVariable = document.querySelector('h1');
alert('hello!');
```

Эти функции, document.querySelector и alert, встроены в браузер для того, чтобы вы использовали их всякий раз, когда вам это необходимо.

Если вы видите что-то, что выглядит как имя переменной, но имеет после него скобки — (), скорее всего, это функция. Функции часто принимают аргументы — биты данных, которые им необходимы для выполнения своей работы. Они находятся в скобках, и разделяются запятыми, если присутствует более одного аргумента.

Например, функция alert() вызывает всплывающий блок, появляющийся в окне браузера, но мы должны дать ему строку в качестве аргумента, чтобы сказать функции, что писать во всплывающем блоке.

Хорошая новость заключается в том, что вы можете определить свои собственные функции — в следующем примере мы напишем простую функцию, которая принимает два числа в качестве аргументов и умножает их:

```
function multiply(num1,num2) {
  var result = num1 * num2;
  return result;
}
```

Попробуйте запустить вышеупомянутую функцию в консоли, затем попробуйте изменить аргументы, например:

multiply(4,7);
multiply(20,20);
multiply(0.5,3);

События. Для создания действительной интерактивности на веб-сайте вам необходимы события. События — это структура, которая следит за тем, что происходит в браузере, а затем позволяет вам запускать код в ответ на это. Наиболее очевидным является событие клика (en-US), которое вызывается браузером, когда мы щёлкаем по чему-то мышью. Для демонстрации этого события введите следующую команду в вашу консоль, а затем щёлкните по текущей веб-странице:

```
document.querySelector('html').onclick = function() {
    alert('Ouch! Stop poking me!');
}
```

Существуют множество способов прикрепить событие к элементу. Здесь мы выбираем <html> элемент и устанавливаем ему обработчик свойства onclick (en-US) анонимной функцией (т.е. безымянной) которая содержит код, который мы хотим запустить для события клика.

Обратите внимание, что

```
document.querySelector('html').onclick = function() {};
```

эквивалентно

```
var myHTML = document.querySelector('html');
myHTML.onclick = function() {};
```

Задания на самостоятельную работу

Изучить примеры и выполнить все задачи, полученные у преподавателя.

Отчет должен содержать в себе следующие разделы:

– название работы;
– цель работы и задачи;
– описание процесса выполнения работы с иллюстрациями (скриншотами);
– выводы о проделанной работе.

ЛАБОРАТОРНАЯ РАБОТА № 4
РАБОТА С JOOMLA

Цель работы: получить базовые знания и приобрести практические навыки работы по работе с системой управления сайтом Joomla.

Задачи изучения и приобретения практических навыков:

1. Изучить назначение CMS Joomla
2. Познакомится с инструментарием
3. Применить полученные знания на практике.

Теоритические сведения

Joomla – это система управления контентом с открытым исходным кодом, написанная на языке программирования PHP, с элементами JavaScript. Название “Joomla” сформировано из слова “Jumla”, африканского языка суахили, что в переводе означает «все вместе». Кстати по этому поводу разработчики, очень удачно выбрали название, так как CMS очень популярна, и имеет огромное сообщество пользователей, как в рунете так и в интернете. При этом Joomla активно развивается и постоянно совершенствуется как в плане функционала и удобства, так и в плане безопасности и быстродействия. На сегодняшний момент, а это уже 10 лет как создана первая версия движка, CMS Joomla занимает лидирующую позицию в рейтинге популярнейших CMS по всему миру, уступая только WordPress.

Возможности CMS Joomla. Богатая функциональность, которая к тому же значительно расширяется дополнительными расширениями (компоненты, модули и плагины).

Гибкая и простая система шаблонов, благодаря которой, очень легко изменять внешний вид сайта: позиции модулей, шрифты цвет фона и т.д. За время существования Joomla, создано огромное количество шаблонов, как бесплатных, так и платных – для различных версий CMS. При этом шаблоны Joomla достаточно просты в понимании, а значит вполне можно самостоятельно создать собственный шаблон, обеспечив, таким образом, уникальность дизайна создаваемого сайта. На нашем сайте опубликован мини-курс по данной теме – ссылка.

С версии 1.6 добавлена поддержка многоязычности.

Начиная с версии 2.5 расширена поддержка баз данных. Реализована поддержка Microsoft SQL Server, а с версии 3.0 — PostgreSQL. В дальнейшем планируется добавить поддержку Oracle, SQLite.

Встроенный медиа-менеджер, который обеспечивает возможность, закачки файлов на сайт.

Гибкий и функциональный механизм разделения пользователей по правам доступа к элементам сайта.

Поддержка человеко-понятных URL (ЧПУ).

Возможность установки срока начала и окончания публикации материалов.

Возможность создания закрытых областей сайта, доступ к которым разрешен только отдельным группам пользователей.

Возможность администрирования из пользовательской части сайта при наличии соответствующих прав доступа.

Преимущества CMS Joomla

Основные преимущества CMS Joomla:

Интуитивно-понятный интерфейс панели администратора, благодаря чему даже новичок с легкостью сможет создать сайт на данной CMS.

Удобный механизм создания и отображения меню, неограниченного уровня вложенности.

Открытый исходный код.

Богатый по функционалу менеджер материалов, который позволяет публиковать неограниченное количество материалов, причем с разделением по категориям.

Удобная система модулей, благодаря которой можно отображать в различных позиция необходимые данные.

Гибкость и расширяемость сторонними компонентами. Как было описано выше, функционал CMS можно значительно расширить дополнительными расширениями и адаптировать для решения практически любой поставленной задачи.

Универсальность и простота настройки. Каждый элемент системы – легко настраивается под каждого пользователя.

Широкое сообщество пользователей и постоянные обновления, которые исправляют найденные ошибки и повышают безопасность системы.

Многоязычность – как было описано выше в CMS введена поддержка многоязычности, что позволяет отображать на страницах сайта контент на различных языках.

Простота обновления.

Задания на самостоятельную работу

Создать простой сайт, использую инструментарий Joomla по заданиям, полученным у преподавателя.

Отчет должен содержать в себе следующие разделы:

– название работы;

– цель работы и задачи;

– описание процесса выполнения работы с иллюстрациями (скриншотами);

– выводы о проделанной работе.

ЛАБОРАТОРНАЯ РАБОТА № 5
АДАПТИВНАЯ ВЁРСТКА ШАБЛОНОВ JOOMLA

Цель работы: получить базовые знания и приобрести практические навыки работы по созданию адаптивных шаблонов под Joomla.

Задачи изучения и приобретения практических навыков:

1. Изучить основные принципы адаптивной верстки.
2. Познакомится со структурой шаблона Joomla.
3. Применить полученные знания на практике.

Теоретические сведения

10-15 лет назад большинство пользователей интернета пользовались компьютерами, а мобильный трафик был маленьким. С активным распространением смартфонов и планшетов появилась необходимость адаптировать контент для удобного просмотра на сенсорных экранах.

В статье разберёмся, какие особенности есть у адаптивной вёрстки, чем она отличается от фиксированной и резиновой и кратко расскажем о 4 инструментах для верстальщиков, которые хотят прокачать навыки.

Что такое адаптивная вёрстка. Во времена, когда о корректном отображении сайта на смартфонах никто не задумывался, использовались разные приёмы адаптации под мобильные устройства. В некоторых случаях у пользователей, заходивших на сайт открывалась так называемая PDA-версия.

Она представляет собой облегчённый шаблон десктопной версии. Пользоваться такой версией проекта относительно удобно, потому что она адаптирована для просмотра на маленьких разрешениях. Следующий этап развития — мобильная версия. Верстальщики создавали отдельный сайт для сенсорных девайсов. В большинстве случаев она полностью отличалась от десктопа. Поддерживать два ресурса одновременно не всегда выгодно, поэтому поиск решений продолжился.

Сейчас разработчики делают сайты адаптивными. Горизонтальной полосы прокрутки нет и пользователям не надо использовать жесты масштабирования. Контент моментально подстраивается под разрешение экрана и становится доступен для взаимодействия.

Адаптивная вёрстка удобна тем, что не надо создавать отдельный сайт для мобильных пользователей. Часть десктопного контента может быть скрыта, но основные инструменты не будут отличаться.

Для проектов с большим количеством сложных модулей создание адаптивной вёрстки может превратиться в сложное испытание. Нужны отдельные макеты под разные типы устройств и масштабная переработка структуры. На смартфонах с маленьким разрешением экрана полезной площади мало. Приходится либо избавляться от инструментов, либо компоновать их для экономии пространства.

Адаптивный дизайн и вёрстка — идеальное решение для всех сайтов, которые можно переработать для просмотра на устройствах с маленьким разрешением без критического ущерба для возможностей.

Некоторые проекты в стандартном виде будут смотреться на смартфонах очень плохо и пользователи потеряют много инструментов, если будут пользоваться ей. В таком случае стоит честно сказать аудитории, что на сайт лучше

заходить с компьютера. Adaptive вёрстка, как кресло, которое подстраивается под форму тела сидящего. Открываешь сайт с компьютера — пользуешься широкоформатной версией, достаёшь смартфон — контент выглядит немного по-другому, но главные возможности сохраняются.

Раньше верстальщики делили устройства на категории и писали код для каждой группы девайсов. Они учитывали особенности портретной и альбомной ориентации, анализировали популярные разрешения и держали в уме особенности некоторых моделей устройств.

При таком подходе время работы над проектом сильно увеличивается. Даже если надо адаптировать под смартфоны и планшеты сайт на несколько страниц без нестандартных блоков.

Адаптивная вёрстка позволяет сократить время разработки благодаря использованию простых инструментов. Например, можно задать ширину меню в процентах и блок всегда будет находиться по центру. И на десктопах, и на смартфонах ничего не поменяется.

Основа любого контента — изображения. На первых этапах появления адаптивной вёрстки больше всего проблем было с картинками. На помощь приходит CSS свойство max-width, но оно не работает в Internet Explorer.

Сделать картинки адаптивными через указание ширины в процентах можно очень быстро, но вес файла останется прежним. Если он весит несколько мегабайт, то время загрузки страницы на смартфонах увеличится, а при открытии изображения в лайтбоксе всё равно придётся использовать жест масштабирования.

Adaptive вёрстка давно стала привычным решением для адаптации контента под устройства с разными разрешениями экрана, но новичкам на первых этапах будет тяжело постигать базовые основы технологии. С опытом станет легче, но если планируете работать со сложными проектами, придётся часто находить выход из нестандартных ситуаций.

3 основных типа вёрстки

Новички часто путают адаптивную вёрстку с другими видами и не могут понять, в чём разница. Если всерьёз настроены на долгосрочную карьеру HTML-верстальщика, обязательно разберитесь в нюансах и научитесь докапываться до сути.

Кроме адаптивной вёрстки, ещё есть фиксированная и резиновая. Фиксированная вёрстка предполагает одинаковый размер элементов макета на любых разрешениях. Если блоки не помещаются в экран, появляется полоса прокрутки.

В основе резиновой вёрстки лежит масштабирование размера элементов при изменении разрешения. Разработчики задают блокам относительные единицы измерения в процентах.

Адаптивная вёрстка — оптимальный вариант для адаптации контента под разные устройства. Медиа-запросы, которые лежат в основе технологии, меняют масштаб элементов, а CSS свойства дают полную свободу действий.

Относительные единицы измерений — хорошее решение, но в большинстве проектов возникает задача изменить расположение блоков и тогда на помощь приходят медиа-запросы.

Ещё один аргумент в пользу адаптивной вёрстки — упрощённая работа с текстом. Если задать размер абзаца в процентах, размер шрифта не будет автоматически подстраиваться под размер экрана. В таких случаях лучше использовать медиа-запросы и задать значения под конкретный диапазон разрешений.

Многие верстальщики для ускорения создания адаптивной версии используют фреймворки. Самым популярным является Bootstrap, который активно развивается с 2011 года и объединил миллионы энтузиастов.

Адаптивная вёрстка — необходимый навык для всех разработчиков, которые хотят связать свою карьеру с фронтендом. Почти в каждом ТЗ на разработку сайта есть упоминание о создании отзывчивой структуры, которая будет подстраиваться под разные разрешения экранов.

Зачем нужна adaptive вёрстка

Ответ очевиден — чтобы владельцы смартфонов, планшетов и мониторов с разной диагональю могли взаимодействовать с контентом без проблем. И им не приходилось переключаться на десктопную версию, потому что мобильная спроектирована некачественно.

Адаптивными сайтами пользоваться одно удовольствие, если разработчик сделал всё правильно и не забыл, что есть устройства с нестандартными разрешениями. Если на этапе разработки верстальщик исправил критические баги и уделил время отладке, серьёзных проблем с юзабилити не должно быть.

В 2021 году пользователи ожидают, что сайты по умолчанию будут адаптированы для просмотра на любых устройствах. Они могут зайти на него даже с телевизора и с помощью Bluetooth мышки взаимодействовать с контентом.

Если на сайте нет адаптации под смартфоны, он перестаёт существовать для большинства пользователей. Доля мобильного трафика с каждым годом растёт и в последние 5 лет наметился чёткий тренд на mobile-first.

Многие проекты сейчас изначально разрабатываются под смартфоны, а только потом создаётся версия для десктопов. Ещё через 5-10 лет вполне может случиться полный переход на мобильную вёрстку. И тогда десктопную версию сайта вообще перестанут разрабатывать.

Хотя 5-10% аудитории всё равно будут выбирать широкоформатные мониторы. Потому что, десктопная версия в большинстве случаев гораздо удобнее мобильной и помогает экономить время на выполнении стандартных задач.

Одно из главных преимуществ адаптивной вёрстки — бонус при продвижении в поисковых системах. Сайты, которые некорректно отображаются на смартфонах и планшетах, редко получают мобильный трафик. Им приходится рассчитывать только на десктопных пользователей.

Преимущества adaptive вёрстки:

Улучшение поведенческих факторов. Пользователи смогут взаимодействовать с контентом через смартфоны и планшеты.

Можно частично автоматизировать задачу. Время адаптации заметно сокращается благодаря фреймворкам, сервисам тестирования и другим инструментам.

Минусы:

Повышается стоимость проекта. Для сайтов с нестандартными особенностями адаптация может стоить очень дорого.

Нужна помощь опытного разработчика. Если проект постоянно дорабатывается, придётся нанимать специалиста в штат или заключать договор на удалённую работу.

Подходит не для всех проектов. Некоторые сайты сложно адаптировать для мобильных устройств, потому что встроенные инструменты слишком «тяжёлые» и надо будет сильно переработать структуру.

Адаптивная вёрстка уже давно считается привычной техникой создания сайтов. Если хотите стать конкурентоспособным на рынке разработчиков, придётся в совершенстве изучить технологию и пользоваться инструментами автоматизации для экономии времени.

Особенности адаптивной вёрстки

Отзывчивые сайты создаются по своим правилам. Если нарушить хотя бы одно из них, работа будет поставлена под угрозу. Ознакомьтесь с главными особенностями и постарайтесь запомнить их, чтобы не допускать ошибки в процессе решения задач клиентов.

Однородность

Мы уже несколько раз говорили, что адаптивный сайт примерно одинаково выглядит на устройствах с разным разрешением экрана. Часть возможностей с десктопа может быть недоступна на смартфонах, но «фундамент» остаётся на месте.

Мобильная версия должна быть практически идентична с основной, но не стоит жертвовать скоростью загрузки ради интеграции лишних инструментов. Лучше добавить в интерфейсе уведомление, что для использования калькулятора надо открыть сайт на компьютере.

Универсальность

Опытные верстальщики знают, что у разных устройств есть свои особенности. Например, сайт на Android и iOS смартфонах может выглядеть совершенно по-разному. Надо учитывать нюансы и внимательно проанализировать все точки соприкосновения пользователей с интерфейсом. В процессе обучения адаптивной вёрстке лучше заручиться поддержкой опытного наставника, чтобы быстрее освоить методику и узнать подробности, которые становятся известны только после нескольких тысяч часов активной разработки.

Кроссбраузерность

Некоторые верстальщики ориентируются исключительно на самый популярный браузер Google Chrome и забывают, что есть пользователи, которые принципиально используют Opera, Firefox или Яндекс.Браузер.

Задания на самостоятельную работу

Сверстать адаптивный шаблон под Joomla. Варианты взять у преподавателя.

Отчет должен содержать в себе следующие разделы:

– название работы;

– цель работы и задачи;

– описание процесса выполнения работы с иллюстрациями (скриншотами);

– выводы о проделанной работе.

ЛАБОРАТОРНАЯ РАБОТА № 6
ТЕСТИРОВАНИЕ НА ЮЗАБИЛИТИ

Цель работы: получить базовые знания и приобрести практические навыки работы по оптимизации содержимого веб-приложения.

Задачи изучения и приобретения практических навыков:

1. Изучить основные принципы оптимизации страницы для поисковых роботов.
2. Познакомится с основными принципами инфографики.
3. Настроить свой адаптивный Joomla шаблон с использованием полученных знаний.

Теоритические сведения

Юзабилити – это качественный показатель простоты и удобства использования сайта.

Юзабилити веб сайта оценивается по 6 основным качественным компонентам:

Ориентация: насколько просто новым посетителям веб-сайта совершать элементарнейшие действия, т.е. не мешают ли им дизайн и кукуева туча всплывашек, аддонов, рекламы и видюхи в режиме автоплей.

Эффективность: насколько быстро юзер может ориентироваться на сайте и совершать необходимые ему действия.

Запоминаемость: насколько легко будет юзеру сориентироваться на сайте после продолжительного отсутствия на сайте.

Ошибки: количество ошибок, совершенных посетителем сайта, способы ликвидации этих лагов и простота в устранении последствий этих ошибочных действий.

Удовлетворенность: субъективная степень «довольности» юзера, эмоциональное восприятие сайта: совокупность информации+дизайна+навигации +отображения в браузере.

Полезность: показатель функциональности сайта для юзера: дает ли веб-сайт пользователю то, что он ищет.

Задания на самостоятельную работу

Настроить свой адаптивный Joomla шаблон с использованием полученных знаний.

Отчет должен содержать в себе следующие разделы:

– название работы;

– цель работы и задачи;

– описание процесса выполнения работы с иллюстрациями (скриншотами);

– выводы о проделанной работе.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Острейковский, В. А. Информатика. Теория и практика: учеб. пособие / В. А. Острейковский, И. В. Полякова. – М.: Оникс, 2008. – 608 с.

2. Петров, М. В. Информатика: метод. указания / сост. М. В. Петрова. – Ульяновск: УлГТУ, 2011. – 67 с.

3. Молоков, К. А. Информатика: метод. указания к лабораторным работам / сост. К. А. Молоков – В 2 ч. Ч.1. – Владивосток: Изд-во ДВГТУ, 2009. – 61 с.

4. Терехов, А. В. Информатика: учеб. пособие / А. В. Терехов, А. В. Чернышов, В. Н. Чернышов. – Тамбов: Изд-во Тамб. гос. техн. ун-та, 2007. – 128 с.

5. Информатика. Практикум по технологии работы на компьютере / под ред. Н.В. Макаровой. – 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2005. – 256 с.

6. Румянцева, Е. Л. Информационные технологии / Е. Л. Румянцева, В. В. Слюсарь. – М.: Форум, Инфра -М, 2007. – 256 с.

ОГЛАВЛЕНИЕ

ИНФОРМАТИКА

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ для студентов направления 15.03.05 «Конструкторско-технологическое обеспечение машиностроительных производств» (профили «Технология машиностроения», «Металлообратывающие станки и комплексы» и «Конструкторско-технологическое обеспечение кузнечно-штамповочного производства»), всех форм обучения

Составитель
Пачевский Денис Евгеньевич

Компьютерный набор Е. Д. Зотовой

Редактор Е. А. Четвертухина

Подписано к изданию 28.10.2021.
Уч.-изд. л. 1,4.

ФГБОУ ВО «Воронежский государственный технический университет»
394026 Воронеж, Московский просп., 14