Министерство образования и науки Российской Федерации

Калужский филиал федерального государственного бюджетного образовательного учреждения высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ "Фундаментальные Науки"

КАФЕДРА "Программное Обеспечение ЭВМ, Информационные Технологии и Прикладная Математика"

ЛАБОРАТОРНАЯ РАБОТА No 1

ДИСЦИПЛИНА "Основы WEB-программирования "

КАФЕДРА "Язык разметки гипертекста HTML"

Калуга, 2014.

Цель

Изучить основные принципы разметки гипертекста с использованием языка HTML.

Время

2ч.

Порядок выполнения лабораторной работы

1. Изучить теоретический материал.
2. Выполнить задание согласно выданному варианту.
3. Ответить на вопросы для самоконтроля.
4. Подготовить отчет.
5. Защитить лабораторную работу.

Требования к лабораторной работе

1. Использовать пятую версию языка разметки (HTML5).
2. Создавать «валидную» разметку. Валидацию страниц проверять инструментом для валидации от w3c (http://validator.w3.org).
3. Визуальное оформление выносить во внешние css файлы.

Вопросы для самоконтроля

1. Как расшифровывается HTML?
2. Какое расширение должны иметь HTML документы?
3. Какие теги являются обязательными для HTML документа?
4. Для чего используется тег <!DOCTYPE> ?

5. Какой тег позволяет вставлять картинки в HTML документы?
6. С помощью какого тега можно сделать текст жирным?
7. С помощью каких тегов создаются заголовки?
8. С помощью какого тега создается упорядоченный список?
9. В чем различие между блочными и строчными элементами?
10. Элемент, созданный с помощью тега <p> является блочным или строчным?
11. Какие теги используются для создания таблиц?
12. Что такое CSS?
13. Что такое селектор?
14. Какие виды CSS селекторов вы знаете?

Требования к содержимому отчета

1. Титульный лист
2. Задание
3. Результаты выполнения
4. Листинг

Теоретический материал

1. Муссиано Ч., Кеннеди Б. HTML и XHTML. Подробное руководство. - Пер. с англ. - СПб: Символ-Плюс, 2008. - 752с., ил.
2. Седерхольм Д., Маркотт И. CSS ручной работы. - - Пер. с англ. - СПб: Питер, 2011. - 240с., ил.

Варианты

Вариант 1

Numbers
The most basic type of number is an *integer*, a **series of digits** which can start

with a **plus or minus sign.**

```
1 , 23 , and -10000 are examples.
```

Commas are not allowed in numbers, but underscores are. So if you feel the need to mark your thousands so the numbers are more readable, use an underscore.

```
population = 12_000_000_000
```

Decimal numbers are called *floats* in Ruby. Floats consist of numbers with **a decimal place** or **scientific notation.**

```
3.14 , -808.08 and 12.043e-04 are examples.
```

Вариант 2

Optimizing JavaScript

Lookup Tables

The `fastSin()` function divides 2π radians into the number of steps specified in the argument, and stores the sin values for each step in an array, which is returned.

Testing the JavaScript `Math.sin()` against a lookup table yields the results shown in Figure 1-2.

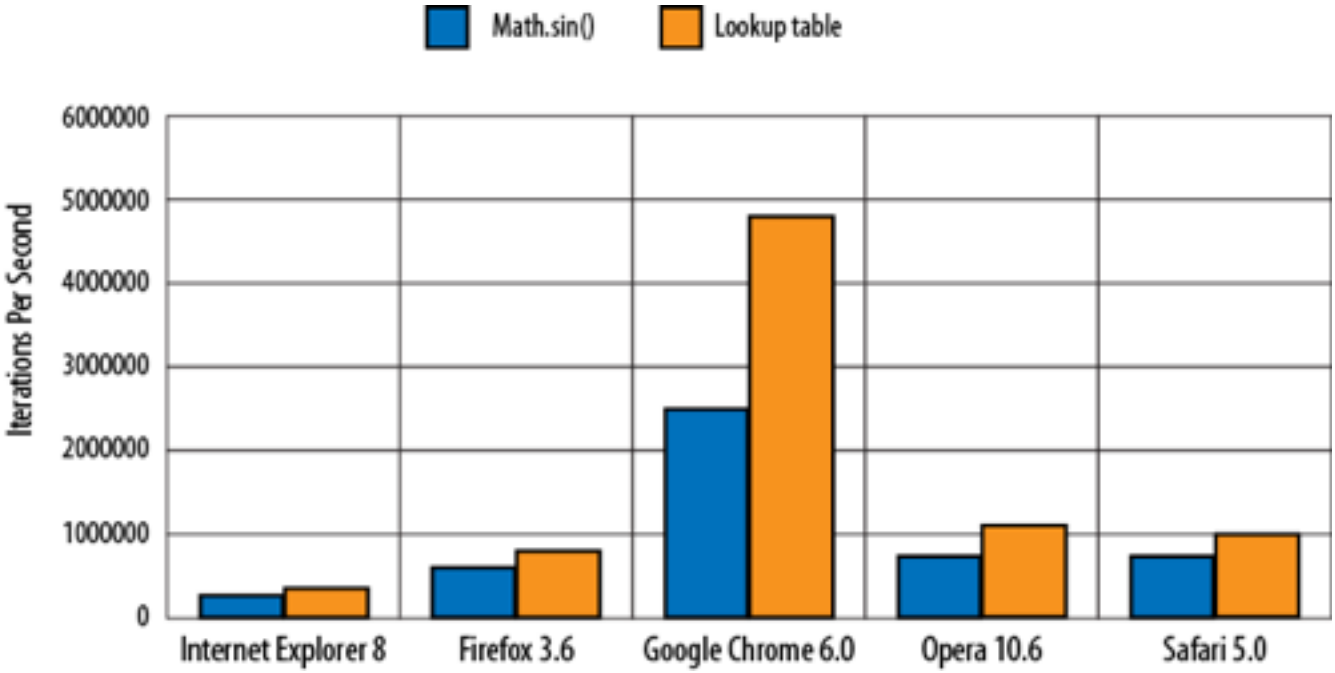


Figure 1-2. Math.sin() versus lookup table performance. Bigger is better.

Across most browsers, there appears to be an approximately 20% increase in perform- ance, with an even more pronounced improvement in Google Chrome. If the calculated values within the lookup table had come from a more complex function than `Math.sin()`, then the performance gains would be even more significant; the speed of accessing the lookup table remains constant regardless of the initial work required to fill in the values.

Вариант 3

Сторонние библиотеки

Рис. 6. HTML5 Please подсказывает решение проблем

Впрочем, это еще не все. Библиотека, несмотря на свой небольшой размер (она даже в полной комплектации занимает чуть более 15 Кб), довольно удачно спроектирована и имеет удобный механизм расширения – метод `Modernizr.addTest()`. Применяется он следую- щим образом:

```
Modernizr.addTest('bar', function(){
      var foo = document.createElement('foo');
      return typeof foo.addBar === 'function'
});
```

Наверное, пример слишком абстрактен, но основной механизм проиллюстрирован. Все, правда, так просто. Модель разработки modernizr открыта, и все стоящие расширения принимаются в ре- позиторий (в структуре проекта на GitHub).

В заключение можно упомянуть, что modernizr используют Twitter, Google, Microsoft (в пакете ASP.NET MVC 3 Tools Update Modernizr поставляется в комплекте с новыми приложениями ASP. NET MVC) и множество других компаний. Так что нам тоже не грех.

Вариант 4

3

Списки и массивы

Если скалярные значения представляют «единственное число» в Perl, как упоминалось в начале главы 2, «множественное число» представляется списками и массивами.

Список содержит упорядоченную коллекцию скалярных значений. *Массив* представляет собой переменную для хранения списка. В Perl эти два термина часто используются как синонимы. Но если выражаться точнее, список – это данные, а массив – переменная. В программе можно создать списочное значение, которое не хранится в массиве, но любая переменная массива содержит список (хотя этот список может быть пустым). На рис. 3.1 изображен список, хранящийся в массиве или нет.

Каждый *элемент* массива или списка представляет собой отдельную скалярную переменную с независимым скалярным значением. Элементы упорядочены, то есть хранятся в определенной последовательности от первого до последнего элемента. Каждому элементу массива

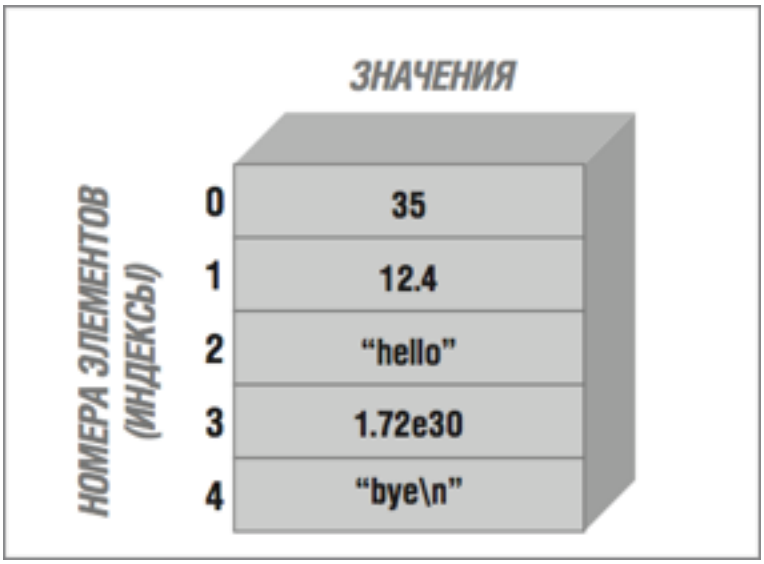
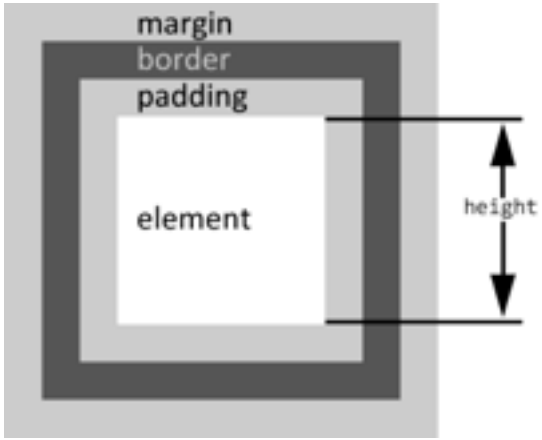


Рис. 3.1. *Список с пятью элементами*

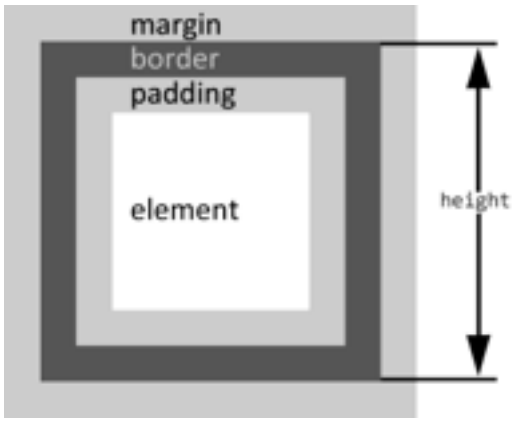
Вариант 5

О размерах блочных элементов

Ещё хотел отдельно остановиться на вычислении ширины и высоты блочных элементов, ведь тут есть один нюанс, по умолчанию, высота и ширина элементов считается **без учёта толщины границ** и **внутренних отступов**, т.е. как-то так:



Эта блочная модель называется *content-box*, и вот в CSS3 появилась возможность изменять блочную модель, указывая атрибут *box-sizing*. Отлично, теперь мы можем выбирать между двумя значениями *content-box* и *border-box*, первый я уже описал, а вот второй вычисляет высоту и ширину **включая внутренние отступы** и **толщину границ**:



Такая блочная модель была свойственна IE6 в *«quirks mode»*

Вариант 6

Объекты

На объекты в JavaScript возложено две роли:

— хранилище данных

— функционал объекта

Первое предназначение можно описать следующим кодом:

```
var user = {
      name: "Ivan",
     age: 32

};

alert(user.name); // Ivan
alert(user.age);  // 32
```

Вариант 7

HTML5's Element Family

So far, this chapter has focused on the changes to HTML5's syntax. But more important are the additions, subtractions, and changes to the *elements* that HTML supports. In the following sections, you'll get an overview of how they've changed.

Added Elements

In the following chapters, you'll spend most of your time learning about new elements—ingredients that haven't existed in web pages up until now. Table 1-1 has a preview of what's in store (and where you can read more about it).

TABLE 1-1 *New HTML5 elements*

CATEGORY	ELEMENTS	DISCUSSED IN…
Semantic elements for structuring a page	<article>, <aside>, <figcaption>, <figure>, <footer>, <header>, <nav>, <section>, <details>, <summary>	Chapter 2
Web forms and interactivity	<input> (not new, but has many new subtypes) <datalist>, <keygen>, <meter>, <progress>, <command>, <menu>, <output>	Chapter 3
Canvas	<canvas>	Chapter 4

Вариант 8

The insert() Function

The `insert()` function in LIB_mysql simplifies the process of inserting a new entry into a database by passing the new data in a keyed array. For example, if you have a table like the one in Figure 7-3, you can insert another row of data with the script in Listing 7-2, making it look like the table in Figure 7-4.

ID	NAME	CITY	STATE	ZIP
1	Kelly Garrett	Culver City	CA	90232
2	Sabrina Duncan	Anaheim	CA	92812

***Figure 7-3**: Example table people before the insert()*

```
$data_array['NAME'] = "Jill Monroe";
$data_array['CITY'] = "Irvine";
$data_array['STATE'] = "CA";
$data_array['ZIP'] = "55410";
insert(DATABASE, $table="people", $data_array);
```

***Listing 7-2**: Example of using insert()*

Вариант 9

PART II

PROJECTS

This section expands on the concepts you learned in the previous section with simple yet demonstrative projects. Any of these projects, with further develop- ment, could be transformed from a simple webbot concept into a potentially marketable product.

Chapter 8: Price-Monitoring Webbots

The first project describes webbots that collect and analyze online prices from a mock store that exists on this book's website. The prices change periodically, creating an opportunity for your webbots to analyze and make purchase decisions based on the price of items.

Since this example store is solely for your experimentation, you'll gain confidence in testing your webbot on web pages that serve no commercial purpose and haven't changed since this book's publication. This environ- ment also gives you the freedom to make mistakes without obsessing over the crumbs your webbots leave behind in an actual online store's server log file.

Вариант 10

CSS-Only Scrolling Effects

Example 3-1. CSS parallax scrolling

```
<!DOCTYPE html>
<html>
	<head>
	<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
	<title>CSS Parallax</title>
		<style type="text/css">
			body {
				padding:0px;
				margin:0px;
			}
			.layer {
				position:absolute;
				width:100%;
				height:256px;
			}
			#back {

				background: #3BB9FF url(back1.png) 20% 0px;
			}
			#middle{
				background: transparent url(back2.png) 30% 0px ;
			}
			#front{
				background: transparent url(back3.png) 40% 0px;
			}
		</style>
	</head>
	<body>
		<div id = "back" class = "layer">

		</div>
		<div id = "middle" class = "layer">

		</div>
		<div id = "front" class = "layer">

		</div>
	</body>
</html>
```

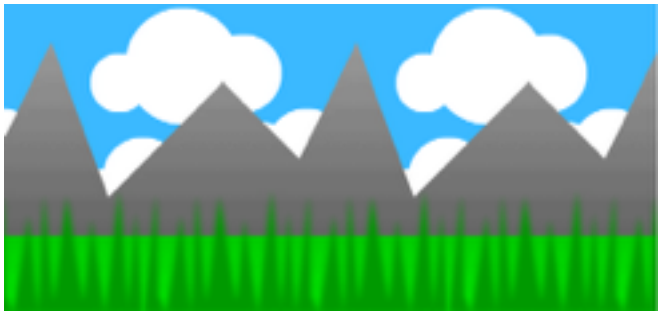


***Figure 3-3**. Parallax CSS effect with three layers*

Вариант 11

ДОБАВЛЕНИЕ ФОНА И ИЗМЕНЕНИЕ ГРАНИЦ

Теперь добавим сложное объявление, чтобы одновременно оформить элементы `<input>` и `<textarea>`. Зададим шири- ну (здесь я выбрал произвольное значение `400px`), добавим отступ, увеличим размер шрифта, удалим границы по умол- чанию и сделаем цвет чуть более темным, чем фон страницы (рис. 2.27).

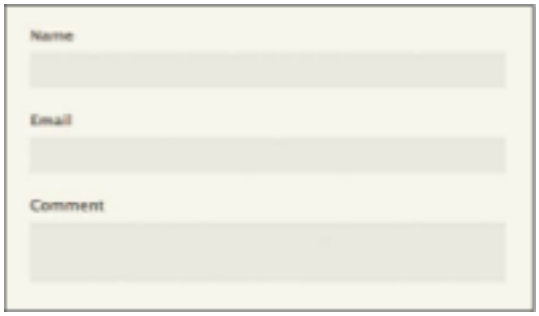


Рис. 2.27

```
#comment-form fieldset input,
#comment-form fieldset textarea {
    width: 400px;
    padding: 5px;
    font-size: 1.4em;
    border: none;
    background: #e2e1d7;
}
```

Вариант 12

Звуки audio

Аудиосопровождение веб-страниц имеет довольно давнюю, но со- всем не примечательную историю. Изначально звуковой контекст мог загружаться в браузер с помощью тега <BGSOUND>, позволяющего сопроводить просмотр ресурса. Например, фоновой музыкой. Этим приемом довольно быстро наигрались – эффект получался очень навязчивым, и создатели веб-ресурсов, за редким исключением, данную возможность просто игнорировали.

Звуки вернулись на веб-страницы с технологией flash, естествен- но, со всеми ей свойственными способностями и ограничениями. Сам же HTML долгое время оставался безмолвным. Пока в рамках HTML5 не появился специальный тег <audio>.

В самом простом виде этот тег выглядит следующим образом:

```
<audio src="sound.mp3" controls >
```

И этого уже вполне хватает для того, чтобы на странице появился аудиоплеер, проигрывающий указанный src-файл (рис. 89).

Рис. 89. Аудиоплеер одним тегом