Автор: Остапенко Денис (sharp-c@yandex.ru)

Введение в веб-программирование

Веб-программирование заключается в создании **клиентских** и **серверных** частей веб-приложений разной степени интерактивности. Клиентская часть исполняется в браузере пользователя, отображая ему информацию, позволяя ему взаимодействовать с ней и серверной частью. Клиентская часть, как правило, состоит из текстовой и графической информации, ее разметки с помощью **HTML**, стилевого оформления с помощью **CSS** и активного содержимого — скриптов на языке программирования **JavaScript**. Серверная часть исполняется на веб-сервере, она может взаимодействовать с файловой системой сервера, базами данных, другими сетевыми серверами, прикладным и системным программным обеспечением сервера. В настоящее время наиболее популярным языком для серверного веб-программирования является **PHP**, а стандартом на взаимодействие с базами данных — **SQL**.

Клиентское веб-программирование

Программное обеспечение

Для написания кода на HTML, CSS и JavaScript достаточно иметь любой plain-текстовый редактор и любой веб-браузер. Рекомендуемым набором является текстовый редактор Notepad++, который содержит подсветку синтаксиса, для редактирования кода, веб-браузер Mozilla Firefox, наиболее корректно отображающий содержимое веб-страниц и плагин к нему Firebug, позволяющий интерактивно отлаживать верстку и скрипты на JavaScript. Иногда для отладки медленных скриптов стоит использовать браузер Google Chrome, имеющий один из наиболее быстрых скриптовых движков. Ссылки на упомянутое программное обеспечение можно найти в дополнении.

Язык разметки гипертекста HTML

Тэги

Современный HTML (здесь и далее рассматривается **XHTML 1.0**) берет начало из **XML**, то есть содержит древовидно организованную **тэгами** информацию. Тэг состоит из **открывающего** и **закрывающего тэга**, заключенных в угловые скобки:

```
<tag>Информация</tag>
```

Открывающий и закрывающий тэг могут содержать **атрибуты**, разделенные пробельными символами, например:

```
<tag attribute1="value1" attribute2="value2">Информация</tag>
```

В случае, если тэг не содержит информации, он может быть закрыт в тех же угловых скобках (исключения: doctype и комментарий):

`<tag></tag>` = `<tag />`

Тэги могут вкладываться, но не могут пересекаться:

`<a><b></b></a>`, но не `<a><b></a></b>`

Комментарии в HTML находятся внутри такого тэга:

```
<!-- Комментарий -->
```

Структура HTML-документа

HTML-документ состоит из **doctype**, описывающего схему документа и определенной последовательности тэгов:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="язык документа, например, ru" lang="язык документа">
<head>
	<!-- Внутри тэга head находятся тэги, описывающие страницу, подключающие стили CSS и скрипты -->
	<title>Заголовок страницы, видимый в заголовке браузера</title>
</head>
<body>
	<!--Внутри тэга body находится содержимое страницы -->
</body>
</html>
```

Некоторые важные тэги

Название тэга	Роль	Пример и важные атрибуты
a	Гиперссылка	`<a href="URL ссылки">Текст ссылки</a>`
h1, h2, h3, h4, h5, h6	Заголовки от самого крупного (h1) до самого мелкого (h6)	`<h1>Заголовок</h1>` `<h2>Подзаголовок</h2>`
img	Изображение	`<img src="URL изображения" alt="Описание изображения" />`
p	Абзац	`<p>Абзац</p>` `<p>Новый абзац</p>`
div, span	Блочный и текстовый элемент	`<div>Блочный элемент</div>` `<span>Текстовый элемент</span>`
b, i, u, strike	Полужирное (**bold**),	`<b>Полужирный</b>`

	курсивное (*italic*), подчеркнутое (<u>underline</u>) и зачеркнутое (~~strike~~) начертание. Вместо них рекомендуется использовать CSS	`<i>Курсивный</i>`
		`<u>Подчеркнутый</u>`
		`<strike>Зачеркнутый</strike>`

table, tr, td	Таблица, строка, ячейка	`<table>`
		`<tr>`
		`<td>Первая строка, первая ячейка</td>`
		`<td>Первая строка, вторая ячейка</td>`
		`</tr>`
		`<tr>`
		`<td>Вторая строка, первая ячейка</td>`
		`<td>Вторая строка, вторая ячейка</td>`
		`</tr>`
		`</table>`

ul, ol, li	Ненумерованный и нумерованный список, элемент списка	`<ul>`
		`<li>Первый элемент ненумерованного списка</li>`
		`<li>Второй элемент ненумерованного списка</li>`
		`</ul>`
		`<ol>`
		`<li>Первый элемент нумерованного списка</li>`
		`<li>Второй элемент нумерованного списка</li>`
		`</ol>`

br	Перевод строки	`Строка 1 ` `Строка 2`
sub, sup	Нижний и верхний индекс	`H<sub>2</sub>0` `2<sup>8</sup> = 256`

Язык стилевого оформления CSS

В настоящее время оформление веб-страниц осуществляется каскадными таблицами стилей на языке CSS. Таблицы стилей состоят из последовательности объявлений селекторов, указывающих, к каким элементам следует применить стили и списков применяемых стилей:

```
body {
    color: yellow;
}
a {
    text-decoration: none;
    background: rgb(208,200,196);
}
```

Стили можно включать в код HTML-страницы с помощью тэга style, размещенного внутри тэга head, указывать непосредственно (без селекторов) для определенного тэга с помощью атрибута style, либо писать в отдельном файле и подключать с помощью тега link:

```
<head>
    <style type="text/css">
        …
    </style>
</head>
либо
<div style="…">Элемент</div>
либо
<link rel="stylesheet" href="file.css" />
```

Селекторы

В таблице перечислены некоторые важные селекторы:

Селектор	Область применения стиля
tag	Все тэги tag в документе
.cls	Все тэги с атрибутом class равным cls
#identifier	Тэг с уникальным атрибутом id, равным identifier
selector1 selector2	Тэги, удовлетворяющие selector2, находящиеся внутри тэгов, удовлетворяющих selector1
selector1, selector2	Тэги, удовлетворяющие selector1 или selector2

Селекторы могут комбинироваться, например, table#all div.page { … } означает, что стиль будет применен к тэгам div, которые имеют атрибут class="page" и находятся в тэге table с id="all".

Стили

Стили состоят из списка свойств, разделенных точкой с запятой. Некоторые свойства могут задавать значения сразу для нескольких других свойств (**составные свойства**), например:

```
.cls {
    border: solid 1px red;
}
```
эквивалентно
```
.cls {
    border-style: solid;
    border-width: 1px;
    border-color: red;
}
```
В таблице перечислены некоторые важные стили:

Свойства	Возможные значения	Описание

color	#ff00ff
	rgb(255,0,255)
	rgb(1.0, 0.0, 0.0)
	violet
	#f0f
		Цвет текста. Задается в RGB, в шестнадцатеричной, десятичной или процентной форме, либо одним из нескольких предопределенных названий.

background	red url('pic.png') repeat-x bottom red url('pic.png')	Составное свойство. Фон. Может задавать свойства background-color, background-image, background-repeat, background-position
border	solid 1px gray 5px 10px 5px 10px	Составное свойство. Граница. Может задавать свойства border-style, border-width, border-color. В свою очередь, эти свойства тоже составные и задают, например, border-top-color (цвет верхней границы), border-right-color (правой), border-bottom-color (нижней), border-left-color (левой)
padding, margin	10px 5px 10px 5px 10px	Отступ от границ элемента до его содержимого; отступ от границ элемента до содержащего его элемента
width, height	50% 10px	Ширина и высота элемента
position, top, left	position: absolute; top: 50px; left: 50px;	Указывают для задания точной позиции сверху (top) и слева (left) элемента относительно страницы (position: absolute) или обычного местоположения (position: relative)
text-align	left center right justify	Выравнивание текста в элементе (влево, по центру, вправо или по ширине)
line-height	20px	Междустрочный интервал
font	8pt 'tahoma' 11px	Размер и название шрифта. Размер может задаваться в точках (pt — point), как в Windows, или в пикселях (px — pixel)

Советы верстальщику

HTML и CSS предоставляют большие возможности по оформлению страницы, но для удобного восприятия информации следует придерживаться некоторых правил.

Шрифты

Следует использовать только стандартные шрифты, входящие в состав большинства современных операционных систем, например (размер 18pt):

Серифные (**серифы** — маленькие черточки на концах букв): Times New Roman Georgia

Бессерифные: Arial Verdana Tahoma Trebuchet MS

Моноширинные: Courier New Lucida Console

Моноширинные шрифты используются чаще всего для оформления листингов кода или консольного вывода.
Не следует использовать больше 2 различных шрифтов для оформления страницы.

Цвета

Основными правилами является использование ярких цветов только для выделения важного текста, но не для выделения всего текста и не для фона. Цвета фона и текста должны быть достаточно контрастны для удобства чтения (более важная информация — контрастнее). На страницах с темным фоном предпочтительно использование бо**л**ьших размеров шрифтов.
Не следует использовать больше 4 различных цветов для оформления страницы.

Отступы и расстояния

Текст визуально удобнее для чтения, если междустрочный интервал (line-height) немного больше стандартного, абзацы разделены (margin-bottom), блоки имеют ощутимые отступы между собой (margin) и между своими границами и содержимым (padding).

Символы

В оформлении текста следует использовать специальные символы вместо их клавиатурных аналогов:

Символы	Запись в HTML	Название
& < >	& < >	Амперсанд, меньше (less than), больше (greater than) (следует использовать HTML-запись, поскольку эти символы имеют специальное значение в XML-подобных языках)
—	&mdash;	Тире (вместо дефиса с клавиатуры)
« »	« »	Кавычки-елочки (quote) (вместо символов дюйма с клавиатуры)
©	&copy;	Символ копирайта (вместо (c))
− ×	− ×	Минус, умножить (вместо дефиса, символа x или звездочки с клавиатуры)
← ↑ → ↓	← ↑ → ↓	Стрелки (arrow) влево (left), вверх (up), вправо (right), вниз (down)

Графика

Существует 3 основных графических формата, используемых в web:

Формат	Описание
JPG	Формат, сжимающий изображение с потерей качества. Хорошо подходит для фотографий, очень плохо для схем, чертежей, графиков и скриншотов.
PNG	Формат, сжимающий изображение без потери качества. Хорошо подходит для схем, чертежей, графиков, скриншотов, градиентов. Поддерживает полупрозрачность (альфа-канал).
GIF	Формат, сжимающий изображение без потери качества. Поддерживает максимум 256 цветов, только полную прозрачность (альфа-маска), поддерживает анимацию. Подходит только для простых анимированных изображений.

Не следует использовать в web форматы без сжатия (BMP, PPB), форматы графических пакетов (PSD, PSP), векторные форматы (WMF, EMF — в настоящее время на пути к стандартизации находится векторный формат SVG), графические форматы трехмерной графики (3DS, X — некоторые браузеры поддерживают VRML).

Дополнительно

Начинающему верстальщику также стоит ознакомиться с «Ководством» Артемия Лебедева, доступным по адресу http://www.artlebedev.ru/kovodstvo/sections/.

Язык программирования JavaScript

JavaScript — императивный, динамический, нестрого типизированный, интерпретируемый си-подобный язык клиентского веб-программирования. JavaScript исполняется браузером пользователя и не имеет доступа к файловой системе ни сервера, ни клиента. JavaScript предназначен для обеспечения интерактивности страницы, в частности для исполнения запросов к серверу без перезагрузки страницы (**Ajax**).

Тэг script

JavaScript включается в веб-страницу с помощью тега script:

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

либо из внешнего файла

```
<script type="text/javascript" src="имя_файла.js"></script>
```

Комментарии в JavaScript и других си-подобных языках могут начинаться на //, а заканчиваться на перевод строки, либо начинаться на /*, а заканчиваться на */:

```
// Комментарий
/* И это —
   тоже комментарий */
```

Ввод и вывод

JavaScript имеет доступ к **DOM** (Document Object Model) веб-страницы, на которой исполняется. Это позволяет ему, в частности, менять, добавлять и удалять любые элементы и стили на странице. Кроме этого, программа на JavaScript может использовать простые диалоговые окна для вывода и ввода информации, используя функции alert и prompt:

```
alert(“Вывод");
var result = prompt(“Ввод");
```

Переменные и типы

В императивных языках ключевым понятием является понятие **переменной** — некоторой поименованной области памяти, которая может хранить информацию какого-нибудь **типа** (например, строка или число) и подвергаться изменению в ходе работы программы.

Типы в JavaScript перечислены в таблице:

Тип	Описание	Пример записи значения
Number	Число (целое или вещественное)	Целые: `123`, `-234` Вещественные: `3.14`, `1e6` Шестнадцатеричные: `0xFF`
String	Строка Некоторые свойства и методы: str.length — длина строки str.charAt(5) — символ на 5-й позиции (считая с нуля) str.charCodeAt(5) — код символа на 5-й позиции str.substring(0, 2) — подстрока с 0-го по 2-й символ	`'Строка'` `"\tСтрока с escape-последовательностями\r\n\x0D\x0A"`
Boolean	Булевское значение (true или false)	`true` `false`
Array	Массив — последовательность значений, к элементам которой можно обращаться по числовому индексу Некоторые свойства и методы: arr.length — длина массива arr.push(123) — добавление элемента 123 в конец массива arr.sort()— сортировка массив	`['Нулевой элемент', 123, true, 0.05]` Пример обращения: `arr[0]` — строка «Нулевой элемент»
Object	Объект — последовательность значений, к элементам которой можно обращаться по числовому или строковому ключу	`{'key1': 123, key2: 234, 2: 345}` Пример обращения: `obj['key2']` или `obj.key2` — число 234

Function	Функция — группа инструкций, которую можно вызвать, передав набор параметров в переменные аргументы функции	`function(arg1, arg2) {`
		`  // Последовательность инструкций, использующая переданные параметры arg1 и arg2`
		`}`
		Пример вызова: `(function(arg1, arg2) { … })(123, 456);`

undefined, null	Два специальных значения, указывающих на то, что значение либо не определено, либо отсутствует	`undefined` `null`

Переменную можно преобразовать к какому-нибудь типу следующим образом:
var a = '100', b = '500';
var c = a + b; // c = '100500', потому что обе переменных в выражении — строки
var d = Number(a) + Number(b); // d = 600

Синтаксис

Программа на языке JavaScript состоит из **инструкций**. Основные инструкции:

Объявление переменной

```
var название_переменной;
```

При объявлении переменной ей можно сразу присвоить значение (**инициализация**):

```
var название_переменной = значение;
```

Выражение

Произвольное выражение, записанное в математической нотации с функциями, вызовами методов и полей объектов, скобками и операторами. Основные арифметические операторы: * / + - % (взятие остатка). Основные математические функции находятся в объекте Math, например, Math.sin (синус), Math.cos (косинус), Math.sqrt (корень), Math.exp (экспонента). В частности, выражение может состоять из одного вызова функции, например, `alert(название_переменной);` выводит значение название_переменной.

В выражениях, помимо переменных, могут использоваться конкретные значения (**литералы**).

Выражение вида a = a * 5 может быть сокращенно записано в виде a *= 5, а a += 1 и a-=1 можно коротко записать ++a и --a, что называется инкремент и декремент.

Присваивание

```
название_переменной = выражение;
```

Составная инструкция

Это ограниченная фигурными скобками последовательность инструкций:

```
{ инструкция 1; инструкция 2; инструкция 3; }
```

Условие

Условие это составной оператор, который выбирает одну из инструкций (возможно, составных) для исполнения в зависимости от значения условного выражения (условное выражение это выражение, которое может принимать значение true или false. В условном выражении могут использоваться логические операторы || (или), && (и), ! (не) и операторы сравнения (<, >, <=, >=, == (значения равны), === (равны значения и типы), != (значения не равны), !== (не равны значения или тип))).

```
if (условное_выражение) {
        набор_инструкций_которые_исполняются_если_условное_выражение_истинно
} else {
        набор_инструкций_которые_исполняются_если_условное_выражение_ложно
}
```

либо без else:

```
if (условное_выражение) {
        набор_инструкций_которые_исполняются_если_условное_выражение_истинно
}
```

Если в зависимости от условия какой-то переменной присваиваются какие-либо значения, удобно использовать тернарный оператор:
var str = n % 2 == 0 ? 'Число n четное' : 'Число n нечетное';
Эта инструкция эквивалентна
var str;
if (n % 2 == 0) str = 'Число n четное';
else str = 'Число n нечетное';

Цикл

Циклы предназначены для многократного исполнения одного набора инструкций (**тела цикла**) до тех пор, пока верно какое-то условие. Циклы бывают:
С предусловием (условие проверяется перед каждым исполнением набора инструкций):

```
while (условное_выражение) {
        тело цикла
}
```

С постусловием (условие проверяется после каждого исполнения набора инструкций):

```
do {
        тело цикла
} while (условное_выражение);
```

Цикл for позволяет помимо условия указать инструкции, которые будут выполняться перед первым запуском цикла и после каждого выполнения тела цикла (**итерации**):

```
for (перед_запуском_цикла ; условное_выражение ; после_каждой_итерации) {
        тело цикла
}
```

Специальная форма цикла for для объектов позволяет перечислить все ключи объекта:

```
for (название_переменной in объект) {
        тело_цикла
}
```

например:

```
var obj = {abc: 123, def: 345};
for (var k in obj) {
       alert(k + ' ' + obj[k]);
}
```

выведет

```
abc 123
def 345
```

Функция

Функция это поименованная группа инструкций, объединенных общими аргументами и логической задачей.

```
function название_функции(список_формальных_аргументов_разделенных_запятой)
{
        тело_функции
}
```

Функция может возвращать значение оператором return. После его исполнения происходит выход из функции.
Функции можно вызвать, используя следующий синтаксис:

```
название_функции(фактические_аргументы_разделенные_запятой);
```

Например:

```
function sum(a, b)
{
        return a + b;
}
```

вызов функции:

```
var a = sum(4, 7*8); // Вернет 60 и присвоит его переменной a
```

Функцию можно присвоить переменной и вызвать, используя имя этой переменной:

```
var название_переменной = function(список_формальных_аргументов_разделенных_запятой) {
        тело_функции
}
```

вызов:

```
название_переменной_содержащей_функцию(список_фактических_аргументов);
```

Работа с DOM страницы (jQuery)

Браузер представляет коду на JavaScript доступ к объектам (DOM), вызов методов и свойств которых позволяет менять вид страницы. Непосредственная работа с ними не очень удобна, поэтому рекомендуется использовать библиотеку **jQuery** (http://jquery.com).

Манипуляция с DOM

jQuery позволяет оперировать с элементами страницы по селектору, подобно CSS, предоставляя функцию $, которая возвращает объект jQuery. Например, такой код раскрасит текст во всех абзацах с классом cls в красный цвет:

```
$('p.cls').css('color', 'red');
```

Некоторые важные методы объекта, который возвращает функция $ и позволяет оперировать с элементами страницы, перечислены в таблице:

Метод	Описание
.attr(key), .attr(key, value)	Получить или установить в значение value атрибут key
.addClass(cls), .hasClass(cls), .removeClass(cls), .toggleClass(cls)	Добавить класс к элементам, проверить элемент на наличие класса, удалить класс, переключить класс (добавить, если не существует, удалить, если существует)

.html(), .html(str)	Получить или установить в значение str HTML-код, содержащийся внутри элемента
.val(), .val(str)	Получить или установить в значение str строку, содержащуюся в элементе формы
.css(key), .css(key, value)	Получить или установить в значение value CSS-свойство key
.show(), .hide(), .toggle()	Показать, скрыть или переключить видимость элемента

События

Обычно следует изменять страницу в ответ на какие-либо действия со стороны пользователя, например, клик или наведение мышью. Для этого указывают название функции, которая будет вызвана, когда действие (**событие**) произойдет, в соответствующем атрибуте элемента, реагирующего на событие, например:

```
<script type="text/javascript">
function func() { alert('Пользователь нажал на кнопку'); }
</script>
<button id="btn" onclick="func();">Кнопка</button>
```

Помимо указания кода обработки события в атрибуте тэга, можно использовать методы объекта jQuery, передавая им функцию, которая будет вызвана при возникновении события, например:

```
$(document).ready(function() {
    $('#btn').click(function() {
        alert('Пользователь нажал на кнопку');
    });
});
```

Следует обратить внимание, что jQuery должен оперировать с DOM только когда документ будет полностью загружен (для этого служит событие ready у глобального объекта document).

Веб-формы, протокол HTTP и Ajax

Для связи между клиентом и сервером используется **протокол HTTP**. С его помощью можно получить данные по их URL, а также передать на сервер какие-то данные со стороны клиента, например, формы и/или файлы. Кроме этого, используя JavaScript, можно отправлять данные на сервер, не перезагружая страницу (Ajax).

HTTP

Протокол HTTP описывает пакеты **HTTP-запроса** и **HTTP-ответа**. Клиент соединяется с веб-сервером по протоколу TCP, используя (как правило) 80-й порт удаленной машины, отправляет ему HTTP-запрос, веб-сервер обрабатывает его, возможно, передавая управления серверным скриптам, и отправляет клиенту HTTP-ответ. В основном используется два типа HTTP-запросов: **GET** и **POST**. GET-запрос передает на сервер запрашиваемый URL, возможно, с параметрами. Вот пример простейшего GET-запроса (в конце пакета идет два перевода строки \r\n\r\n):

```
GET /script.php?param1=value1&param2=value2 HTTP/1.1
Host: host.ru
```

Этот запрос передает серверу, что требуется URL http://host.ru/script.php?param1=value1&param2=value2. Строка «GET» указывает на тип HTTP-запроса, строка “HTTP/1.1" сообщает версию протокола HTTP. Поскольку на сервере может располагаться несколько сайтов с разными доменными именами, нужно указывать HTTP-заголовок Host. Пример HTTP-ответа может быть таким:

```
HTTP/1.1 200 OK
Content-Length: 34

param1=value1<br />
param2=value2
```

Строка «HTTP/1.1» указывает на версию протокола, 200 OK — код возврата и его краткое описание (другими известными кодами являются 403 Forbidden, 404 Not Found, 500 Internal Server Error, 503 Service Unavailable). HTTP-заголовок Content-Length указывает длину возвращаемых данных в байтах. Через 2 перевода строки от заголовка HTTP-ответа идут сами возвращаемые данные указанной длины.

POST-запрос аналогичен GET-запросу, но помимо URL передает специальным образом закодированные данные, например, файл клиента или данные веб-формы.

Обычно GET-запрос используется для передачи коротких данных, на результат обработки которых можно будет сослаться по URL (например, поисковая выдача: http://yandex.ua/yandsearch?text=http), а POST-запрос для передачи больших или чувствительных данных (типа паролей), либо файлов.

Веб-формы

Для элементов, разрешающих пользовательский ввод с последующей передачей введенных пользователем параметров на сервер используются тэги input, textarea (окно для ввода текста), select (выпадающий список), заключенные в тэг form. Пример:

```
<form method="get" action="http://host.ru/script.php">
    <input type="text" name="param1" />
    <input type="password" name="param2" />
</form>
```

Атрибут method тэга form позволяет задавать тип запроса (get или post), атрибут action задает URL, по которому будут отправлены введенные данные (если он пуст, форма отправится по текущему адресу).

Атрибут type тэга input определяет внешний вид элемента формы. Он может принимать значения text (простое текстовое поле), checkbox (флажок), file (поле выбора файла), password (поле ввода пароля), radio (радиокнопка), submit (кнопка, по нажатию на которую происходит отправка формы). Для именования передаваемых значений используется атрибут name.

Ajax

Библиотека jQuery предоставляет удобный способ работы с Ajax: методы get и post у $. Она позволяет передавать на сервер данные в виде пар ключ-значение, записанных в виде объектов JavaScript вместо ручного кодирования параметров.

Первая буква Ajax означает «асинхронный». Передача данных через сеть и обработка их сервером может занимать достаточно долгое время, и было бы очень неудобно, если бы браузер «висел», пока она осуществляется. Поэтому программист, записывая обращение к серверу через Ajax, должен указать функцию, которая будет вызвана, когда сервер вернет данные. На вход этой функции будет передан аргумент, содержащий данные, которые вернул сервер. Пример GET-запроса, который передает на сервер значение, введенное в текстовое поле с id="in", и выдает alert с ответом сервера, когда он придет:

```
$.get(
    'http://server/script.php',
    {data: $('#in').val()},
    function(res) {
        alert(res);
    }
);
```

Введение в серверное веб-программирование

В настоящее время наиболее популярным языком серверного веб-программирования является PHP. PHP, как и JavaScript, императивный, динамический, нестрого типизированный, интерпретируемый си-подобный язык серверного веб-программирования. PHP исполняется на сервере, имеет доступ к файловой системе сервера, но не клиента. PHP предназначается для обработки переданных клиентом данных, их хранения, получения информации из сети, файловой системы или базы данных и выводе клиенту в HTTP-ответе. PHP предназначается для создания динамических страниц, в то время как статический контент (например, картинки) сервер может отдавать без его участия.

Программное обеспечение

Если для работы с HTML, CSS и JavaScript достаточно иметь любой современный веб-браузер, для работы с PHP и SQL следует установить, как минимум, веб-сервер, интерпретатор PHP и систему управления базами данных (СУБД). Наиболее популярным набором является веб-сервер Apache, интерпретатор PHP (предпочтительно не ниже версии 5.3) и СУБД MySQL (предпочтительно не ниже версии 5.0). Для целей обучения обычно достаточно пакета Denwer, в который интегрированы эти три программы, но на «боевых» серверах следует устанавливать и настраивать каждую программу по отдельности. Для удобства просмотра баз данных и отладки SQL рекомендуется установить HeidiSQL, для отладки регулярных выражений рекомендуется Regex Builder. Ссылки на упомянутое программное обеспечение можно найти в дополнении.
Для удобства разработки, скорее всего, придется настроить отображение ошибок в файле конфигурации php.ini, в частности, установить error_reporting = E_ALL | E_STRICT, display_errors = On, display_startup_errors = On, html_errors = On. Если же PHP устанавливался вместе с пакетом Denwer, следует отключить автоматическое экранирование специальных символов с помощью magic_quotes_gpc = Off.

Язык программирования PHP

Как PHP исполняется на сервере

Изначально PHP предназначался для вставки вывода динамического содержимого непосредственно в HTML-код. В настоящее время PHP и HTML принято разделять (читайте далее про MVC), но PHP-код по-прежнему заключается в HTML-подобные угловые скобки, начинающиеся на <?php, а заканчивающиеся на ?> (в коде можно встретить сокращенную запись <? ?>, но она не рекомендуется):

```
<?php
echo 'Hello, world';
?>
```

Если поместить файл index.php с этим содержимым в папку, определенную как корень сайта (далее будем называть ее www), после чего перейти в браузере по адресу http://localhost/, если веб-сервер и интерпретатор верно настроены, а веб-сервер запущен, можно увидеть надпись Hello, world на странице. Если веб-сервер не запущен, браузер не сможет соединиться с localhost (localhost, он же IP 127.0.0.1 — специальное имя и адрес для текущего компьютера). Если запущен, но в настройках указана не та папка, либо index.php не указан, как один из возможных индексных файлов, можно будет наблюдать ошибку 404, либо страницу, которую показывает сервер по умолчанию. И, наконец, если связь веб-сервера с интерпретатором PHP настроена неверно, файл index.php не исполнится, а будет предложен для скачивания.
Если все настроено верно, сервер получит запрос на индексный файл, лежащий в корне сайта, найдет index.php, по его расширению определит имя программы, которая его обрабатывает, и запустит интерпретатор PHP, передав ему имя файла index.php и все переданные через GET и POST параметры (если они есть). Интерпретатор PHP найдет все текстовые блоки внутри файла index.php, которые начинаются с <?php, а заканчиваются на ?> и последовательно исполнит их, возможно, подставив результаты вывода. Полученная строка будет передана обратно веб-серверу, который отправит ее клиенту. Таким образом, код

```
a<?php echo 1; ?>b<?php echo 2; ?>c
```

после исполнения на сервере будет отдан клиенту в виде строки a1b2c.

Переменные и типы

В отличие от JavaScript, в PHP имя переменной предваряется символом $, а объявление переменной происходит при первом присваивании, например

```
<?php
$a = 120 + 3; /* Присваивание, комментарии и арифметические операции в PHP такие же, как в JavaScript */
echo $a; // Выведет 123
echo $b; // Ошибка, переменная $b не была объявлена
?>
```

Тип	Описание	Пример записи значения
boolean	Аналог Boolean из JavaScript	true false
integer	Целое число (от -2^{63} до $2^{63}-1$)	Целые: 123, -234 Шестнадцатеричные: 0xFF
float	Вещественное число	3.14 1e-6
string	Строка strlen($str) — длина строки $str[5] — символ на 5-й позиции (считая с нуля) substr($str, 2, 3) — 3 символа строки, начиная со 2-го $str1 . $str2 — конкатенация (соединение) строк	'Строка' "\tСтрока с escape-последовательностями\r\n\x0D\x0A" "Строка с подстановкой значения переменной: $var"
array	Массив. В отличие от JavaScript, в PHP массив может быть	array(100, 200, 300)

	ассоциативным, т.е. его индекс может быть не только числовым от 0 до arr.length−1, но вообще любым числом или любой строкой count($arr) — количество элементов в массиве $arr[] = 123; — добавление элемента 123 в конец массива sort($arr) — сортировка массива array_key_exists('key', $arr) — проверка, существует ли в массиве $arr строковый индекс 'key'	Пример обращения: $arr[1] это 200 array('ok' => false, 'msg' => 'Сообщение', 404 => 'Not found') Пример обращения: $arr[404] это строка Not found, а $arr['ok'] это false
object	Объект в PHP это экземпляр класса, и он имеет те поля и методы, которые определены в объявлении класса.	Создание: $obj = new ClassName; Пример обращения к полю: $obj->field к методу: $obj->method($args)
resource	Ресурс это особый объект, хранящий в себе некоторое состояние внешней сущности, например, дескриптор файла или соединения с базой данных	
null	Специальное значение, указывающее на отсутствие значения	null

Переменную можно преобразовать к какому-нибудь типу следующим образом:

```
$a = '100 лунатиков';
$b = (int) $a; // Теперь $b содержит число 100
```

Следует заметить, что в false автоматически преобразовываются значения 0, 0.0, '', '0', array(), null.

Ввод и вывод

В отличие от JavaScript, PHP при исполнении не может вывести в браузере клиента диалоговое окно, чтобы о чем-нибудь спросить пользователя, или прочитать значение, записанное в элементе ввода. Все данные, которые PHP получает от клиента, передаются ему через GET или POST-запрос, они хранятся в предопределенных суперглобальных массивах $_GET и $_POST. Кроме этого, некоторую информацию о клиенте (например, его IP-адрес) может сообщить веб-сервер (через массив $_SERVER). Для вывода используется конструкция языка echo.

```
<?php
$num = (int) $_GET['num']; // Получаем переданный через GET параметр num и преобразовываем его из строки в число
$res = $num * 2;
echo $res . '<br />' . $_SERVER['REMOTE_ADDR']; // Выводим через перевод строки переданное значение, умноженное на два, и IP клиента
?>
```

Кроме этого, в отладке часто бывает полезна функция var_dump, которая выводит тип и подробное значение переменной, например:

```
<?php
$arr = array('sulfur' => true, 'carbon', 10 => 'nitre', 'boom');
var_dump($arr);
?>
```

выведет

```
array(4) {
  ["sulfur"]=>
  bool(true)
  [0]=>
  string(6) "carbon"
  [10]=>
  string(5) "nitre"
  [11]=>
  string(4) "boom"
}
```

Обратите внимание на следование числовых индексов при наличии явного указания числового индекса для какого-нибудь элемента.

Выражения

Выражения в PHP аналогичны выражениям в JavaScript, разве что математические функции определены глобально, а не в объекте Math: sin, cos, sqrt, exp.

Управляющие конструкции

Условия и циклы в PHP аналогичны условиям и циклам в JavaScript, за одним исключением. Вместо инструкции for...in в PHP для перебирания всех элементов, например, массива, используется инструкция foreach:

```
$arr = array('key1' => 'val1', 'key2' => 'val2');
foreach ($arr as $key => $val) {
       echo '[' . $key . '] = ' . $val . '<br />';
}
```

выведет

```
[key1] = val1
[key2] = val2
```

Функции

Функции в PHP аналогичны функциям в JavaScript, начиная с версии 5.3 их можно присваивать так же, как это делают в JavaScript.

Классы и объекты

Класс определяет, какие поля и методы будут иметь создаваемые объекты этого класса. Объекты позволяют объединить в одной переменной все данные и методы их обработки, относящиеся к какой-то одной сущности, а классы описать, какие данные и методы какая-либо сущность имеет.

Синтаксис класса и создания объекта
Поля и методы
Конструктор и деструктор
Модификаторы доступа
Статические поля и методы

Базы данных
Язык SQL
Работа с базой данных, SQL injection, XSS

Прикладное программирование
Серверная графика
Куки, авторизация, хэши
Дата, время
Работа с файловой системой: просмотр каталогов, чтение файла, чтение файла по URL
Загрузка файлов
Регулярные выражения
JSON

Проектирование
MVC
CRUD

Приложение для продвинутых читателей
Кодировки и методы кодирования в интернете (cp1251, utf-8, utf-16, urlencode)
Замыкания, scope, this
Клиентская графика
Введение в ExtJS?
Валидация HTML, CSS и JS (jslint)
Читабельные URL
Введение в Zend Framework?
Нормализация БД и join
Server-push

Приложения
Сайты с упомянутым программным обеспечением
Что еще стоит изучить