

# Разработка Web- приложений

Преподаватель: Вильданов  
Вадим  
Кадирович

# Возможности PHP

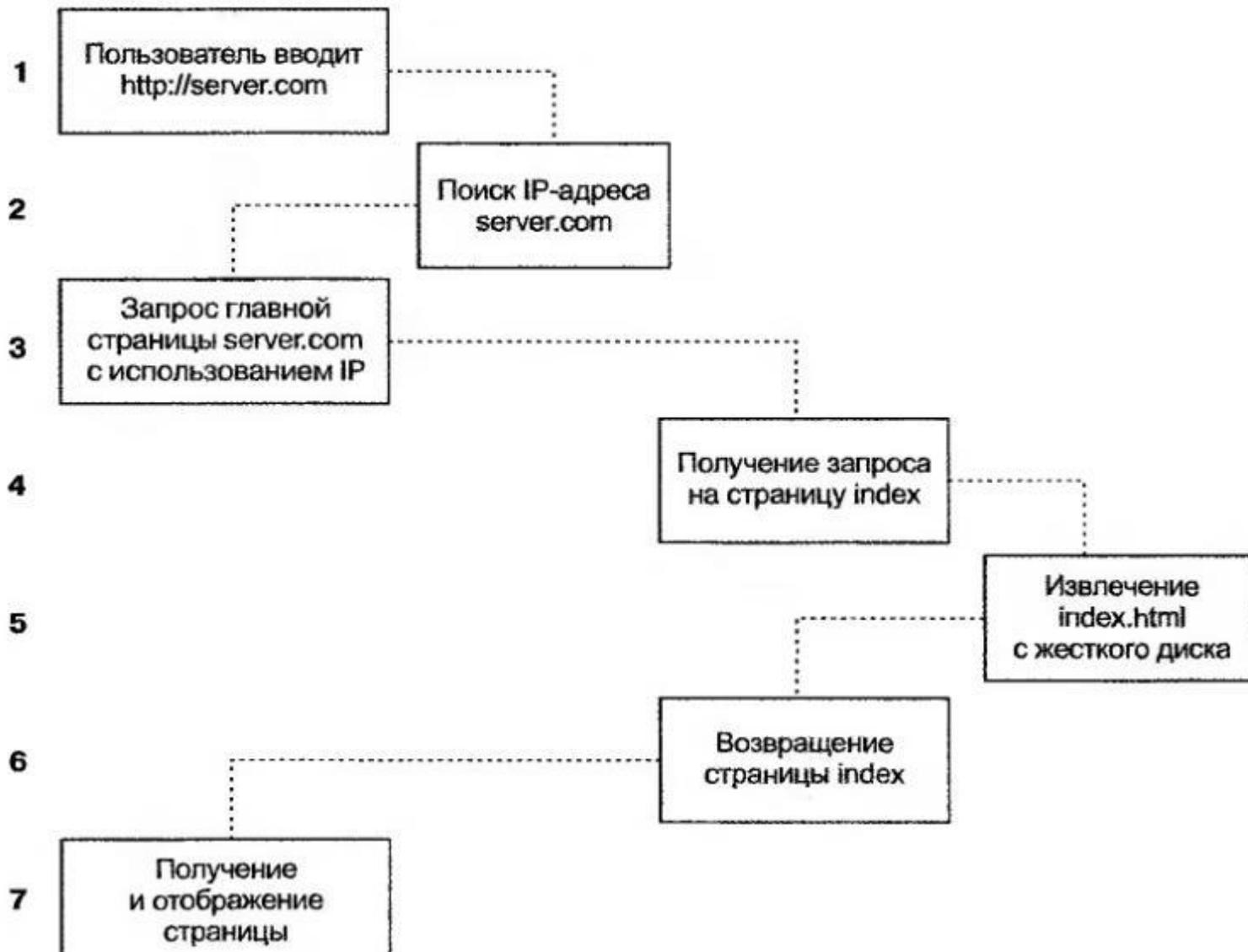
- Создание приложений ( *скриптов* ), которые исполняются на стороне *сервера*.
- Создание *скриптов*, выполняющихся в *командной строке*. То есть с помощью *PHP* можно создавать такие *скрипты*, которые будут исполняться, вне зависимости от *web-сервера* и браузера, на конкретной машине.
- Создание *GUI* -приложений (графических интерфейсов), выполняющихся на стороне клиента.

Веб-браузер

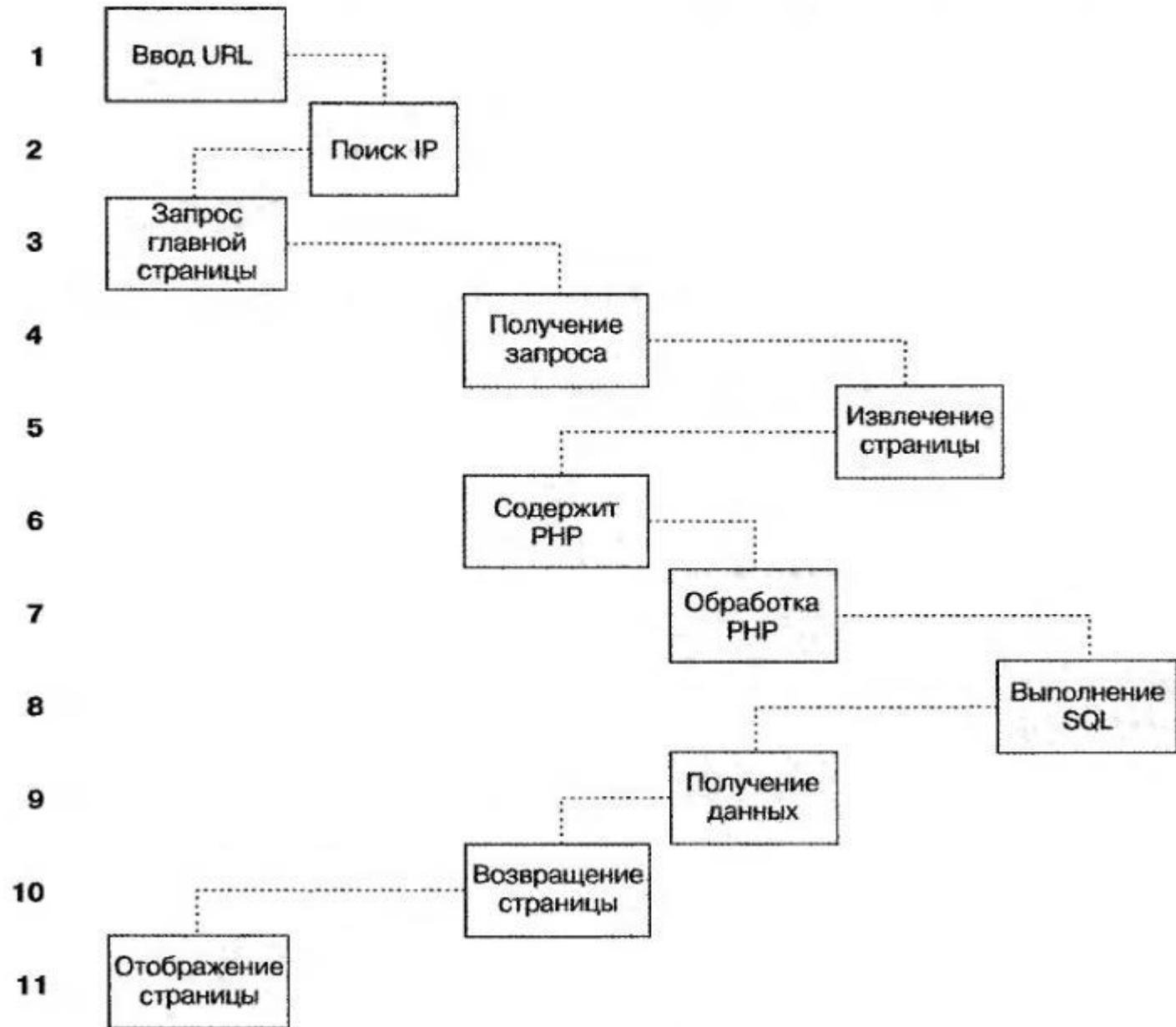
Интернет

Веб-сервер  
на server.com

Жесткий диск  
на server.com



Веб-браузер    Интернет    Веб-сервер    Процессор PHP    Жесткий диск    База данных MySQL



# Динамические web-страницы

Следует различать:

- Динамические эффекты страниц  
(определены заранее)
- Динамическое содержимое страниц  
(зависит от внешних условий)

# Разделение задач web-приложения

- Обработка событий, динамика и эффекты на странице – JavaScript
- Обработка запросов на стороне сервера, взаимодействие – PHP
- Хранение и обработка данных - MySQL

# Методы создания динамических web-страниц

- Ручной
- Онлайн конструктор
- CMS

# Курс «Разработка Web-приложений»

Включает в себя изучение:

- общих принципов организации сетевого взаимодействия компьютеров;
- истории появления и развития Интернета;

# Вопросы для самостоятельной ПОДГОТОВКИ

1. Определение процесса Web программирования и его составляющие;
2. Почему World Wide Web характеризуется как проект распределённой гипертекстовой системы;
3. Назначение и основные характеристики современных языков гипертекстовой разметки;
4. Основные характеристики технологической среды, в которой функционирует Web приложение;
5. Определение понятий сервера и клиента и концепции «клиент-сервер»;
6. Дайте формулировку роли сервера данных;
7. Определение тонкого клиента и его места в компьютерных технологиях;
8. Основные признаки и составные части трёхуровневой Web ориентированной информационной системы. Схема передачи и обработки данных в такой системе.
9. Функции обработки информации в двух- и трёхзвенных клиент/серверных системах;
10. Назначение протокола CGI;
11. Определение и характеристики основных протоколов, используемых в сети WWW;
12. Понятие унифицированного локатора ресурса;
13. Состав стека семейства протокола TCP/IP;
14. Краткая характеристика и функциональное назначение сервисных протоколов TCP/IP;
15. Основные положения протокола HTTP;
16. Назвать и дать краткую характеристику трёх компонентов веб-технологии;

# Введение в PHP. Включение PHP в HTML

- `<?php`
- `echo "Hello world!";`
- `?>`
  
- `<?`
- `echo "Hello world!";`
- `?>`

# Использование комментариев

- Однострочные
  1. //пример комментария
  2. #пример комментария
- Многострочные
  1. /\*Пример
  2. Многострочного
  3. комментария\*/

# Особенности PHP

- Язык php регистрозависимый
- Команды языка заканчиваются символом ;
- Символ \$ используется для обозначения имен переменных
- Присваивание:
  1. `<?php`
  2. `$mycount=1;`
  3. `$mystring="word";`
  4. `$myarray=array("one","two","three");`
  5. `?>`

# Правила присваивания имен переменных

- Имена должны начинаться с буквы или с символа подчеркивания
- Могут содержать a-z, A-Z, 0-9 и \_
- Имена не должны содержать пробелов
- Имена переменных регистрозависимы

# Присваивание значений переменным

- Присваивание по значению  
1. `$second = $first;`
- Присваивание по ссылке  
1. `$second = &$first;`

# Объявление константы

- Константы не имеют приставки \$ в своем имени.
- Получить значение константы можно указав её имя или используя функцию **constant("Имя\_константы")**  
**define("Имя\_константы",**  
**"Значение\_константы",**  
**[Нечувствительность\_к\_регистру])**

# Арифметические операторы

## Обозначение Название Пример

+ Сложение  $\$a + \$b$

- Вычитание  $\$a - \$b$

\* Умножение  $\$a * \$b$

/ Деление  $\$a / \$b$

% Остаток от деления  $\$a \% \$b$

# Строковые операторы

- *Конкатенация ( сложение строк )*
- **$\$c = \$a . \$b$**  (это строка, состоящая из  $\$a$  и  $\$b$  )

# Операторы присваивания

Обозначение	Пример	Описание
=	<code>\$a=3;</code>	<code>\$a=3;</code>
+=	<code>\$a+=3;</code>	<code>\$a=\$a+3;</code>
-=	<code>\$a-=3;</code>	<code>\$a=\$a-3;</code>
.=	<code>\$a.=\$b;</code>	<code>\$a=\$a.\$b;</code>
*=	<code>\$a*=3;</code>	<code>\$a=\$a*3;</code>
/=	<code>\$a/=3;</code>	<code>\$a=\$a/3;</code>

# Операторы сравнения

Обозначение	Название	Описание	Пример
==	Равенство	Значения <i>переменных</i> равны	\$a == \$b
===	Эквивалентность	Равны значения и <i>типы переменных</i>	\$a === \$b
!= или <>	Неравенство	Значения <i>переменных</i> не равны	\$a != \$b или \$a<>\$b
!==	Неэквивалентность	<i>Переменные</i> не эквивалентны	\$a !== \$b

# Операторы сравнения

Обозначение	Название	Пример
>	Больше	$\$a > \$b$
<	Меньше	$\$a < \$b$
>=	Больше либо равно	$\$a \geq \$b$
<=	Меньше либо равно	$\$a \leq \$b$

# Логические операторы

Обозначение	Название	Пример
and &&	И	
or 	Или	
xor	Исключающее или	
!	Инверсия	

# Инкремент и декремент

Обозначение	Название	Описание
<code>++\$a</code>	Пре-инкремент	Увеличивает <code>\$a</code> на единицу, возвращает <code>\$a</code>
<code>\$a++</code>	Пост-инкремент	Возвращает <code>\$a</code> , увеличивает <code>\$a</code> на единицу
<code>--\$a</code>	Пре-декремент	Уменьшает <code>\$a</code> на единицу, возвращает <code>\$a</code>
<code>\$a--</code>	Пост-декремент	Возвращает <code>\$a</code> , уменьшает <code>\$a</code> на единицу

# Типы данных

*PHP* поддерживает восемь простых типов данных.

- Четыре скалярных типа:
  - *boolean* (логический) ;
  - *integer* (целый) ;
  - *float* (с плавающей точкой) ;
  - *string* (строковый).
- Два смешанных типа:
  - *array* (массив) ;
  - *object* (объект).
- И два специальных типа:
  - *resource* (ресурс) ;
  - *NULL*.

# Тип `boolean` (булев или логический тип)

- Этот *тип* выражает истинность значения, то есть *переменная* этого *типа* может иметь только два значения – истина `TRUE` или ложь `FALSE`.
- Чтобы определить булев *тип*, используют ключевое слово `TRUE` или `FALSE`. Оба регистронезависимы.

1. `<?php`

2. `$test = True;`

3. `?>`

# Тип integer (целые)

- Этот *тип* задает число из множества целых чисел  $Z = \{\dots, -2, -1, 0, 1, 2, \dots\}$ .
- Целые могут быть указаны в десятичной, шестнадцатеричной или *восьмеричной системе счисления*, по желанию с предшествующим знаком " - " или " + ".
- Для использования восьмеричной системы счисления, нужно предварять число **0** (нулем), для использования шестнадцатеричной системы нужно поставить перед числом **0x**.

# Тип float (числа с плавающей точкой)

- *Числа с плавающей точкой* (они же числа двойной точности или *действительные числа*) могут быть определены при помощи любого из следующих синтаксисов:

1. `<?php`

2. `$a = 1.234;`

3. `$b = 1.2e3;`

4. `$c = 7E-10;`

5. `?>`

# Тип string (строки)

- **Строка** – это набор символов.
- В PHP символ занимает один байт.
- Это также означает, что PHP не имеет встроенной поддержки Unicode.

*Строка* в PHP может быть определена тремя различными *способами*:

- с помощью *одинарных кавычек* ;
- с помощью *двойных кавычек* ;
- *heredoc-синтаксисом*.

# Одинарные кавычки

- Простейший способ определить строку – это заключить ее в одинарные кавычки
- Чтобы использовать одинарную кавычку внутри строки, перед ней необходимо поставить символ обратной косой черты " \ ", т. е. экранировать ее.
- В строках, заключенных в одинарные кавычки, переменные и управляющие последовательности для специальных символов не обрабатываются .

# Двойные кавычки

- Если *строка* заключена в *двойные кавычки* " " ", то РНР распознает большее количество *управляющих последовательностей* для **СПЕЦИАЛЬНЫХ СИМВОЛОВ**.

# Управляющие последовательности

Последовательность	Значение
<code>\n</code>	Новая строка ( LF или 0x0A (10) в ASCII)
<code>\r</code>	Возврат каретки ( CR или 0x0D (13) в ASCII)
<code>\t</code>	Горизонтальная табуляция ( HT или 0x09 (9) в ASCII)
<code>\\</code>	Обратная косая черта
<code>\\$</code>	Знак доллара
<code>\"</code>	Двойная кавычка

# *Heredoc*

- *Heredoc* -текст ведет себя так же, как и *строка* в *двойных кавычках*
- При использовании такого синтаксиса нет необходимости экранировать кавычки
- Можно использовать *управляющие последовательности*.
- *Переменные* внутри *heredoc* тоже *обрабатываются*.

# *Heredoc*

- **`$str = <<<EOD` Большой блок текста, охватывающий несколько строчек, с использованием heredoc-синтаксиса `EOD;`**

# Тип array (массив)

- **Массив** в PHP представляет собой упорядоченную последовательность *значений* и *ключей*.
- Определить *массив* можно с помощью конструкции `array ()` или непосредственно задавая *значения* его элементам.

# Определение при помощи array()

```
$MyArr=array (key => value, key1 => value1, ... )
```

*Значение элемента массива можно получить, указав после имени массива в квадратных скобках ключ искомого элемента.*

```
$MyArr[key]
```

# Создание массива заданием элементов

- Создать *массив* можно, просто записывая в него *значения*.
- Если указать новый *ключ* и новое *значение*, например, `$book["new_key"]="new_value"`, то в *массив* добавится новый элемент.
- Если не указывать *ключ*, а только присвоить *значение* `$book[]="new_value"`, то новый элемент *массива* будет иметь *числовой ключ*, на единицу больший максимального существующего.
- Нумерация элементов массива начинается с нуля.

# Изменение и удаление элементов массива

- Для того чтобы изменить конкретный элемент *массива*, нужно просто присвоить ему с его *ключом* новое значение.
- Чтобы *удалить элемент массива*, нужно использовать функцию ***unset*** () .

