



html
academy

интерактивные
онлайн-курсы

3 раздел: шаблонизация и организация кода

Содержание вебинара

Содержание вебинара

- Два подхода к встраиванию PHP-кода в HTML

Содержание вебинара

- Два подхода к встраиванию PHP-кода в HTML
- Подключение файлов в PHP-сценариях

Содержание вебинара

- Два подхода к встраиванию PHP-кода в HTML
- Подключение файлов в PHP-сценариях
- Как организовывать код в проекте

Содержание вебинара

- Два подхода к встраиванию PHP-кода в HTML
- Подключение файлов в PHP-сценариях
- Как организовывать код в проекте
- Шаблонизация: отделение логики от интерфейса

Содержание вебинара

- Два подхода к встраиванию PHP-кода в HTML
- Подключение файлов в PHP-сценариях
- Как организовывать код в проекте
- Шаблонизация: отделение логики от интерфейса
- Буферизация вывода

Содержание вебинара

- Два подхода к встраиванию PHP-кода в HTML
- Подключение файлов в PHP-сценариях
- Как организовывать код в проекте
- Шаблонизация: отделение логики от интерфейса
- Буферизация вывода
- Боремся с XSS-атаками (межсайтовый скриптинг)

Два способа генерации HTML

Два способа генерации HTML

PHP используется для **динамического** создания HTML кода:

Два способа генерации HTML

PHP используется для **динамического** создания HTML кода:



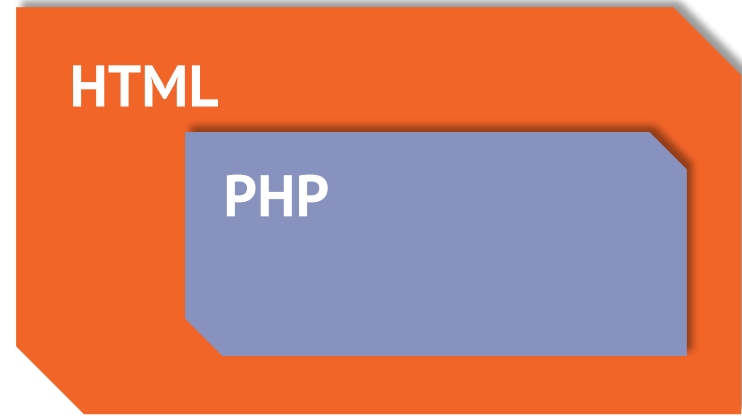
PHP-сценарий сам генерирует HTML код

Два способа генерации HTML

PHP используется для **динамического** создания HTML кода:



PHP-сценарий сам генерирует HTML код



В готовый HTML встраиваются фрагменты PHP

Пример: PHP внутри HTML

Применяется когда веб-страница достаточно сложная и уже полностью сверстана в HTML


```
<!doctype html>
<html>
<head>
  <title>Пример PHP в HTML</title>
</head>
<body>
<h1>Текущее время: <?php print (date("H:i:s")); ?></h1>
<h2>Текущая дата: <?=date("d.m.Y"); ?></h2>
</body>
</html>
```

Пример: PHP внутри HTML

Применяется когда веб-страница достаточно сложная и уже полностью сверстана в HTML

```
<!doctype html>
<html>
<head>
  <title>Пример PHP в HTML</title>
</head>
<body>
<h1>Текущее время: <?php print(date("H:i:s")); ?></h1>
<h2>Текущая дата: <?=date("d.m.Y"); ?></h2>
</body>
</html>
```

PHP-код выполняется
внутри блока «<?php ..?>»



Пример: PHP внутри HTML

Применяется когда веб-страница достаточно сложная и уже полностью сверстана в HTML

```
<!doctype html>
<html>
<head>
  <title>Пример PHP в HTML</title>
</head>
<body>
<h1>Текущее время: <?php print(date("H:i:s")); ?></h1>
<h2>Текущая дата: <?=date("d.m.Y"); ?></h2>
</body>
</html>
```

PHP-код выполняется
внутри блока «<?php ..?>»

Краткая форма
записи - «шорт-тег»

Пример: HTML внутри PHP

Применяется когда сценарий относительно простой, а HTML кода мало


```
<?php
$time = date("H:i:s");
$date = date("d.m.Y");

print("<h1>Текущее время: " . $time . "</h1>");
print("<h2>Текущая дата: $date</h2>");
```


Пример: HTML внутри PHP

Применяется когда сценарий относительно простой, а HTML кода мало

Для показа переменной
внутри строки используем
конкатенацию



```
<?php
$time = date("H:i:s");
$date = date("d.m.Y");

print("<h1>Текущее время: " . $time . "</h1>");
print("<h2>Текущая дата: $date</h2>");
```

Пример: HTML внутри PHP

Применяется когда сценарий относительно простой, а HTML кода мало

Для показа переменной
внутри строки используем
конкатенацию

```
<?php
```

```
$time = date("H:i:s");
```

```
$date = date("d.m.Y");
```

```
print("<h1>Текущее время: " . $time . "</h1>");
```

```
print("<h2>Текущая дата: $date</h2>");
```

Если используются двойные кавычки, то
можно показывать переменную внутри
строки без необходимости конкатенации

Какой подход выбрать

Какой подход выбрать полностью зависит от задачи. Возможен гибридный подход.

Какой подход выбрать

Какой подход выбрать полностью зависит от задачи. Возможен гибридный подход.

PHP в HTML

- Лучше воспринимается и легче поддерживать
- Готовая верстка
- Мало PHP кода

Какой подход выбрать

Какой подход выбрать полностью зависит от задачи. Возможен гибридный подход.

PHP в HTML

- Лучше воспринимается и легче поддерживать
- Готовая верстка
- Мало PHP кода

HTML в PHP

- Сложнее в поддержке
- Нет верстки, нужно оформить вывод
- Для простых скриптов

Подключение файлов

RNR позволяет включать одни сценарии в другие. Так можно объединять множество разных сценариев в один.

Зачем дробить РНР-сценарии

Зачем дробить РНР-сценарии

В любом веб-проекте есть множество повторяющейся логики и/или данных:

Зачем дробить РНР-сценарии

В любом веб-проекте есть множество повторяющейся логики и/или данных:

- общие пользовательские функции

Зачем дробить РНР-сценарии

В любом веб-проекте есть множество повторяющейся логики и/или данных:

- общие пользовательские функции
- конфигурация сайта

Зачем дробить РНР-сценарии

В любом веб-проекте есть множество повторяющейся логики и/или данных:

- общие пользовательские функции
- конфигурация сайта
- общие данные

Зачем дробить РНР-сценарии

В любом веб-проекте есть множество повторяющейся логики и/или данных:

- общие пользовательские функции
- конфигурация сайта
- общие данные
- **шаблоны страниц**

Зачем дробить РНР-сценарии

В любом веб-проекте есть множество повторяющейся логики и/или данных:

- общие пользовательские функции
- конфигурация сайта
- общие данные
- **шаблоны страниц**

Смысл разделения РНР-сценариев на отдельные файлы в возможности **повторного использования**

Схема разделения ответственности

Схема разделения ответственности

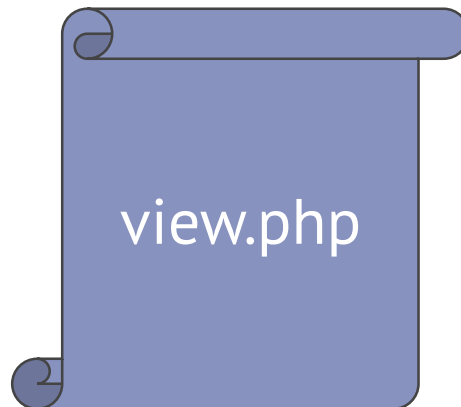
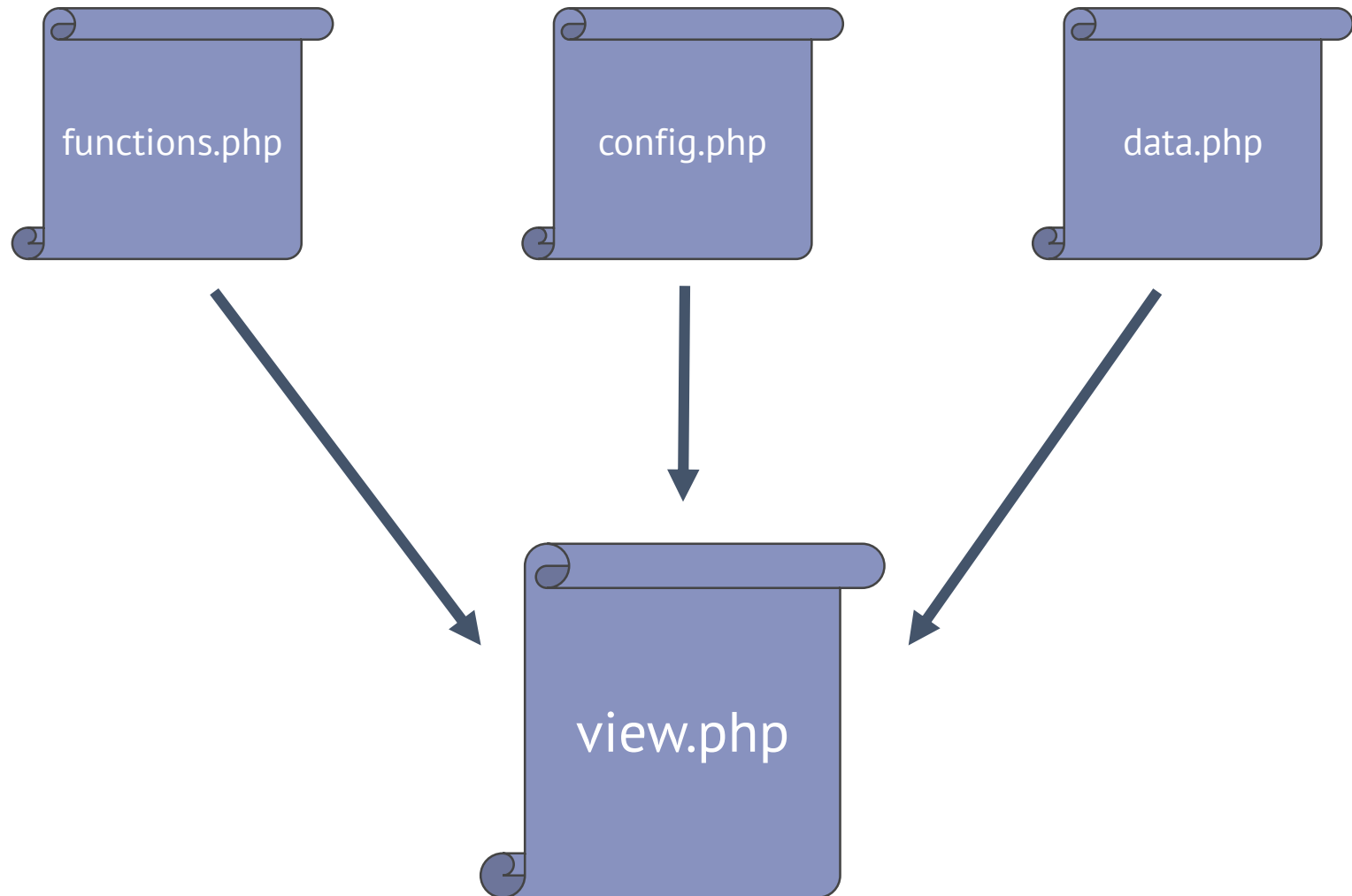


Схема разделения ответственности



Функции для подключения

Функции для подключения

```
require ( 'filename.php' ) ;
```

Функции для подключения

```
require ( ' filename.php ' ) ;
```

Подключает сценарий. Если файл сценария не существует, то вызывает фатальную ошибку и останавливает выполнение.

Функции для подключения

```
require ( ' filename.php ' ) ;
```

Подключает сценарий. Если файл сценария не существует, то вызывает фатальную ошибку и останавливает выполнение.

```
require_once ( ' filename.php ' ) ;
```

Функции для подключения

```
require ( ' filename.php ' ) ;
```

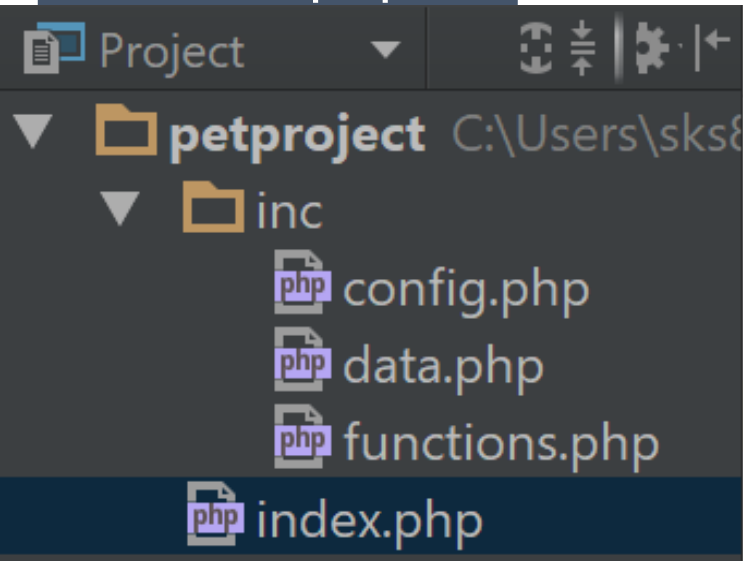
Подключает сценарий. Если файл сценария не существует, то вызывает фатальную ошибку и останавливает выполнение.

```
require_once ( ' filename.php ' ) ;
```

Подключает сценарий только один раз. При повторном вызове не будет снова его подключать.

Пример подключения

index.php



```
1 <?php
2 require_once 'inc/config.php';
3 require_once 'inc/data.php';
4 require_once 'inc/functions.php';
5
6 print("Hello world!");
```

Переменные

Все переменные во включаемых сценариях становятся доступными внутри общего сценария

Общая область видимости

Все подключаемые сценарии как будто склеиваются в один большой и получают общую область видимости переменных

Общая область видимости

Все подключаемые сценарии как будто склеиваются в один большой и получают общую область видимости переменных

index.php

data.php

```
$cats = ['Игры'];
```

functions.php

```
function cut_text() {};
```

```
cut_text();  
print($cats[0]);
```

Включение внутри функций

Если подключить сценарий внутри функции, то все определенные в нём переменные останутся в локальной области видимости функции

inc/data.php

```
<?php  
$foo = "bar";
```

inc/data.php

```
<?php  
function get_template($file) {  
    require_once($file);  
    print($foo);  
}  
  
get_template("inc/data.php");  
print($foo);
```

Включение внутри функций

Если подключить сценарий внутри функции, то все определенные в нём переменные останутся в локальной области видимости функции

inc/data.php

```
<?php  
$foo = "bar";
```

Здесь переменная
доступна

inc/data.php

```
<?php  
function get_template($file) {  
    require_once($file);  
    print($foo);  
}  
  
get_template("inc/data.php");  
print($foo);
```

Включение внутри функций

Если подключить сценарий внутри функции, то все определенные в нём переменные останутся в локальной области видимости функции

inc/data.php

```
<?php  
$foo = "bar";
```

Здесь переменная
доступна

Вызовет ошибку:
переменная не
определена

inc/data.php

```
<?php  
function get_template($file) {  
    require_once($file);  
    print($foo);  
}  
  
get_template("inc/data.php");  
print($foo);
```

Шаблонизация



Шаблонизация

деление всей верстки сайта на отдельные блоки – шаблоны.

В сценариях такие шаблоны будут подключены по необходимости и заполнены данными



TRUST WORTHY AND TIMELY HEALTH & MEDICAL

[GET STARTED](#)

True Multiple Designs

Dolore magna aliquautenim ad minim
veniam nostru exercitation uamco laboris
minim veniam exercitation ullamco.



Power Packed Themes

Dolore magna aliquautenim ad minim
veniam nostru exercitation uamco laboris
minim veniam exercitation ullamco.



More Themes in Feature

Dolore magna aliquautenim ad minim
veniam nostru exercitation uamco laboris
minim veniam exercitation ullamco.

Welcome to Medical and Healthcare Center

Web page editors now use Lorem Ipsum as their default model text, and a search for lorem ipsum will many web sites still in their recently with desktop publishing software like fancy various versions have evolved over the years.

Что такое шаблон

готовая верстка страницы или блока, которая состоит только из оформления и не содержит никакого контента (полезной информации)

Что такое «рыба»?

Что такое «рыба»?



Какая
рыба?!

Что такое «рыба»?

Рыба – это заполнитель.

Бессмысленный текст, который используется в верстке, чтобы показать как будет выглядеть страница, наполненная контентом.



Какая
рыба?!

Типовой процесс верстки

Типовой процесс верстки

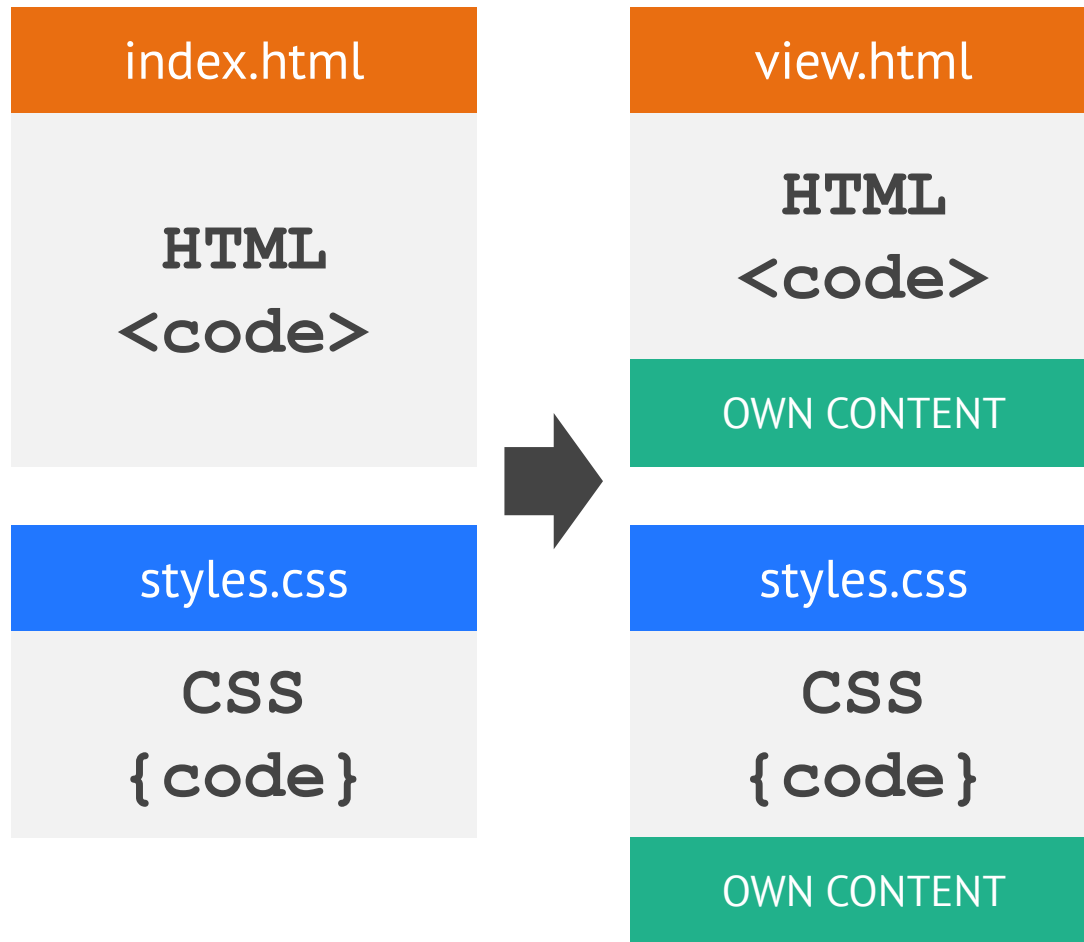
index.html

HTML
<code>

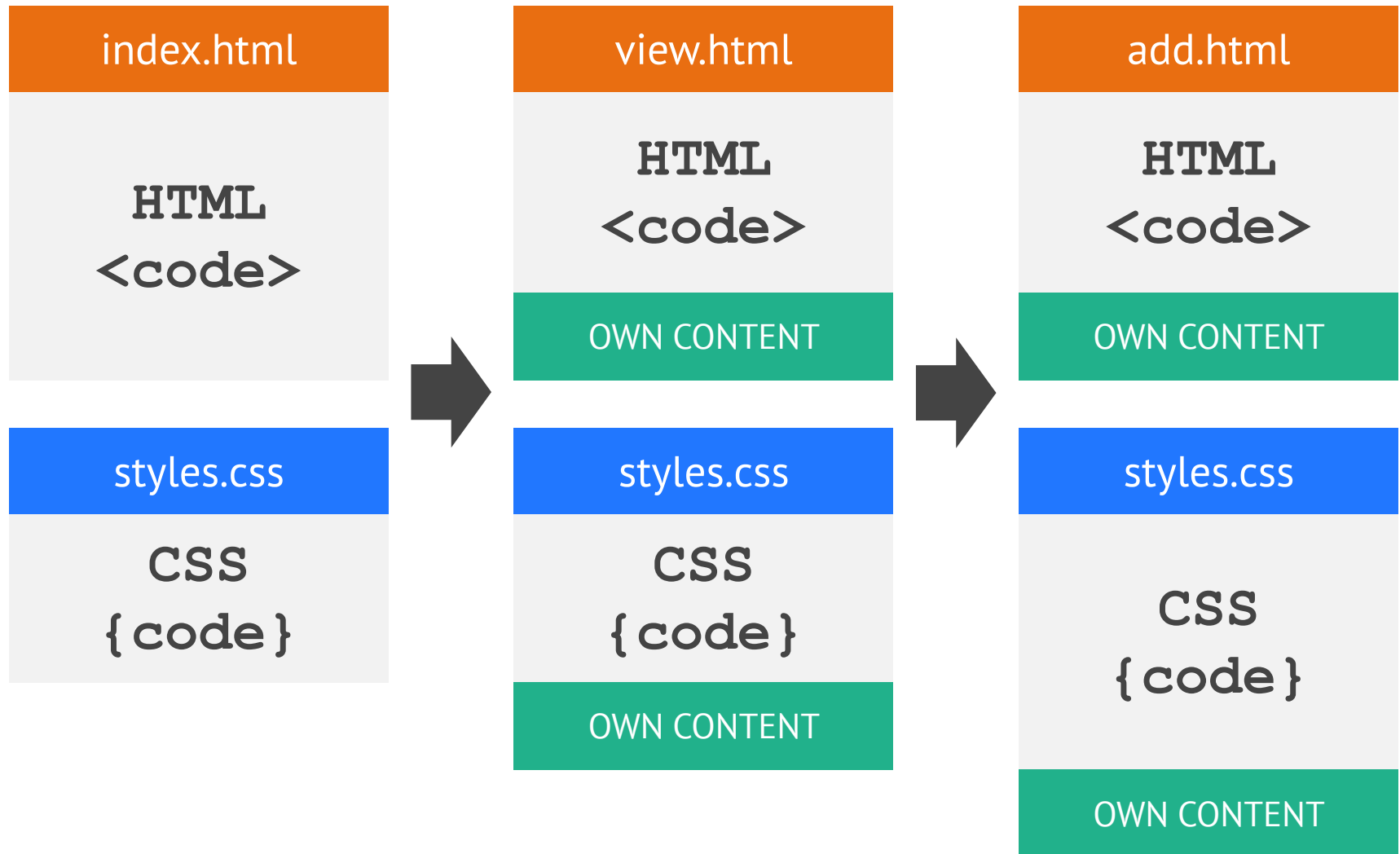
styles.css

CSS
{ code }

Типовой процесс верстки



Типовой процесс верстки



Правки в верстке

Правки в верстке



Добавь меню
сверху и
поменяй лого

Правки в верстке



Добавь меню
сверху и
поменяй лого

10 раз правит
шапку сайта



Правки в верстке



Добавь меню
сверху и
поменяй лого

10 раз правит
шапку сайта



Нужно
добавить лого
платежек в
футер



Правки в верстке



Добавь меню
сверху и
поменяй лого

10 раз правит
шапку сайта



Нужно
добавить лого
платежек в
футер

10 раз правит
футер сайта



Последствия правок в верстке

Последствия правок в верстке



А почему у нас
в половине
страниц меню
съехало?

Последствия правок в верстке



А почему у нас
в половине
страниц меню
съехало?

#%\$@!!111



Общие части страниц



Поиск гифки...

НАЙТИ

МОЙ GIFTUBE

Регистрация

Вход для своих

КАТЕГОРИИ

Видеоигры

Животные

Люди

Наука

Приколы

Спорт

Фейлы

Фильмы и анимация

Видеоигры

НАЗАД



[Unstoppable](#)

@frexin

4 ❤️



[Battlefield 1](#)

@frexin

0 ❤️

Если у вас вдруг возникли вопросы, свяжитесь с нами по почте: info@giftube.com.

Сохранение смешных гифок разрешено только для личного использования.



Общие части страниц



Поиск гифки...

НАЙТИ

МОЙ GIFTUBE

Регистрация

Вход для своих

КАТЕГОРИИ

Видеоигры

Животные

Люди

Наука

Приколы

Спорт

Фейлы

Фильмы и анимация

Видеоигры

НАЗАД



[Unstoppable](#)

@frexin

4 ❤️



[Battlefield 1](#)

@frexin

0 ❤️

Если у вас вдруг возникли вопросы, свяжитесь с нами по почте: info@giftube.com.

Сохранение смешных гифок разрешено только для личного использования.



Общие части страниц



Поиск гифки...

НАЙТИ

МОЙ GIFTUBE

Регистрация

Вход для своих

КАТЕГОРИИ

Видеоигры

Животные

Люди

Наука

Приколы

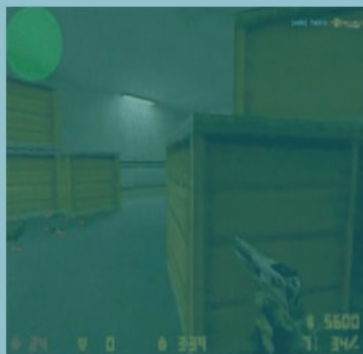
Спорт

Фейлы

Фильмы и анимация

Видеоигры

НАЗАД



[Unstoppable](#)

@frexin

4



[Battlefield 1](#)

@frexin

0



Если у вас вдруг возникли вопросы, свяжитесь с нами по почте: info@giftube.com.

Сохранение смешных гифок разрешено только для личного использования.





Шапка сайта

НАЙТИ

МОЙ GIFTUBE

Регистрация
Вход для своих

КАТЕГОРИИ

Видеоигры

Животные

Люди

Наука

Приколы

Спорт

Фейлы

Фильмы и анимация

Приколы

НАЗАД



[Конец рабочего дня](#)

@frexin

1 ❤️



[Колесо опаздывает на встречу](#)

@Чукча

1 ❤️

Контент страницы

Если у вас вдруг возникли вопросы, свяжитесь с нами по почте: info@giftube.com.

Подвал

Эти гифки разрешено использовать только для личного использования.



Плюсы шаблонизации

Плюсы шаблонизации

- Сложные сайты состоят из множества общих блоков: шапка, подвал, боковое меню, и т.д.

Плюсы шаблонизации

- Сложные сайты состоят из множества общих блоков: шапка, подвал, боковое меню, и т.д.
- Разделение логики сценария и отображения

Плюсы шаблонизации

- Сложные сайты состоят из множества общих блоков: шапка, подвал, боковое меню, и т.д.
- Разделение логики сценария и отображения
- Легче читать и поддерживать

Плюсы шаблонизации

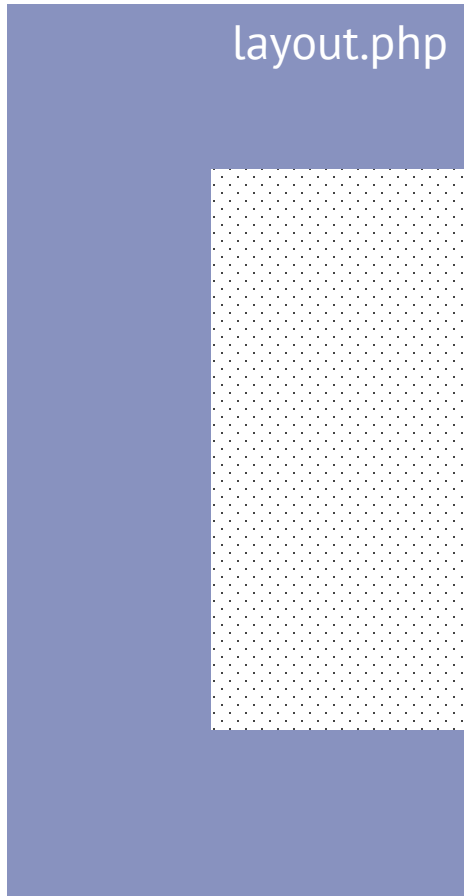
- Сложные сайты состоят из множества общих блоков: шапка, подвал, боковое меню, и т.д.
- Разделение логики сценария и отображения
- Легче читать и поддерживать
- Разделение труда: верстальщик – верстает, программист – программирует

Собираем из шаблонов страницу

Собираем из шаблонов страницу

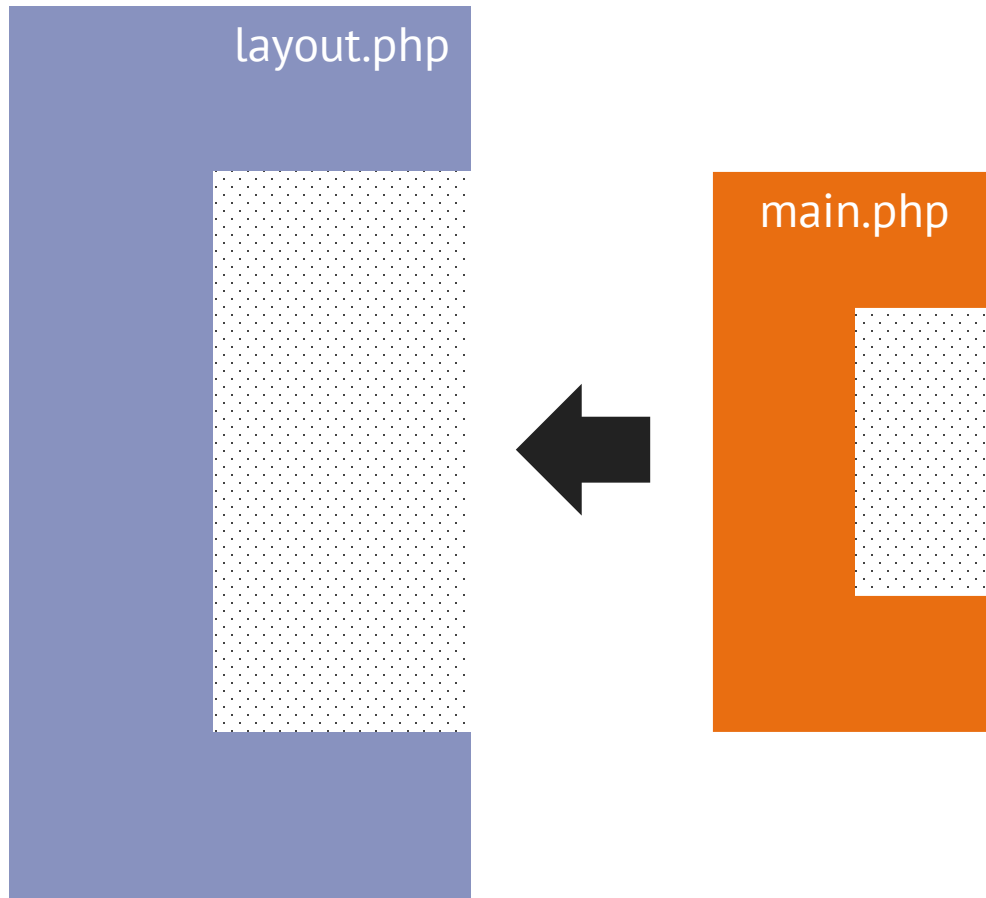
index.php

Собираем из шаблонов страницу



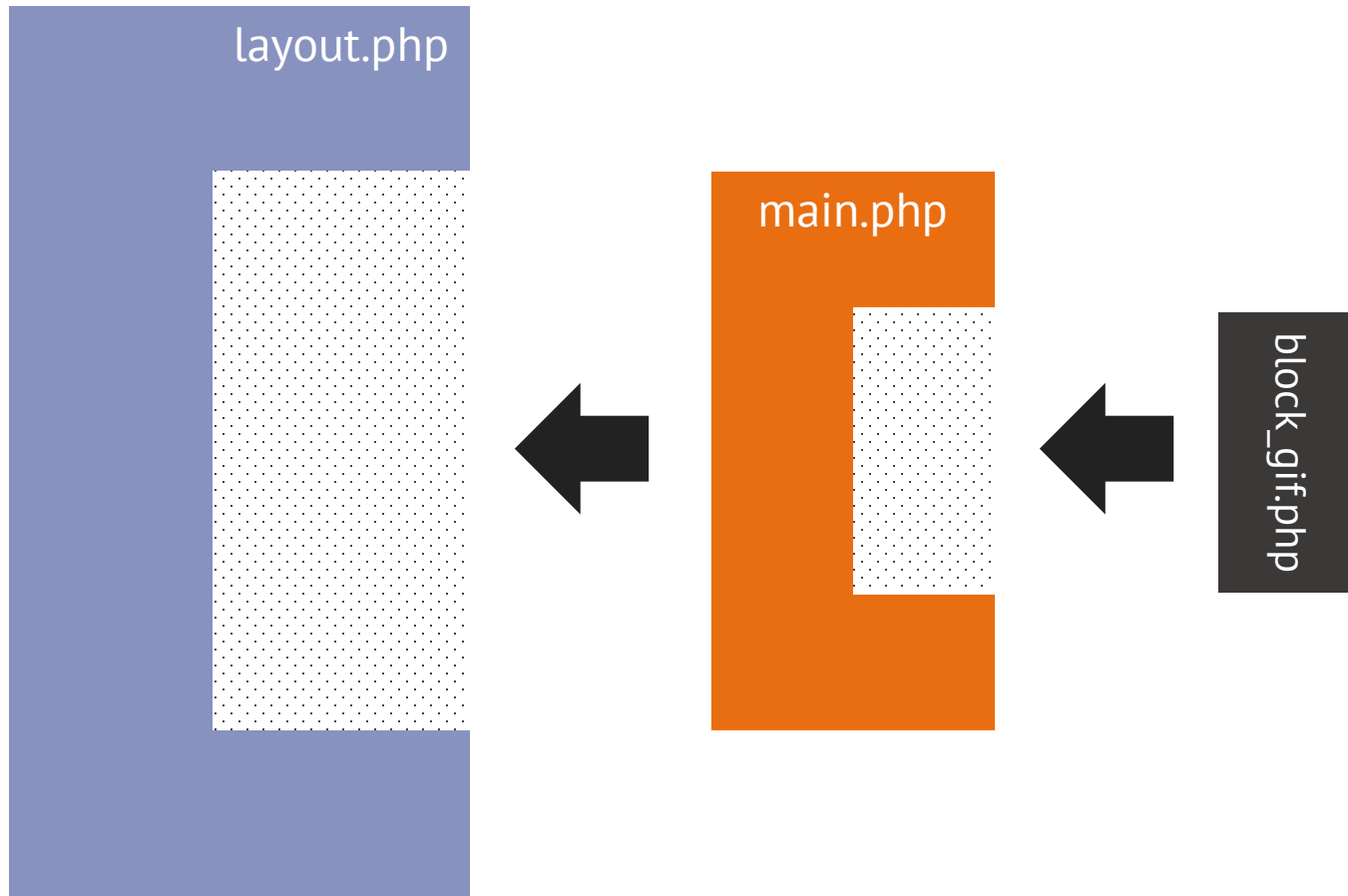
index.php

Собираем из шаблонов страницу



index.php

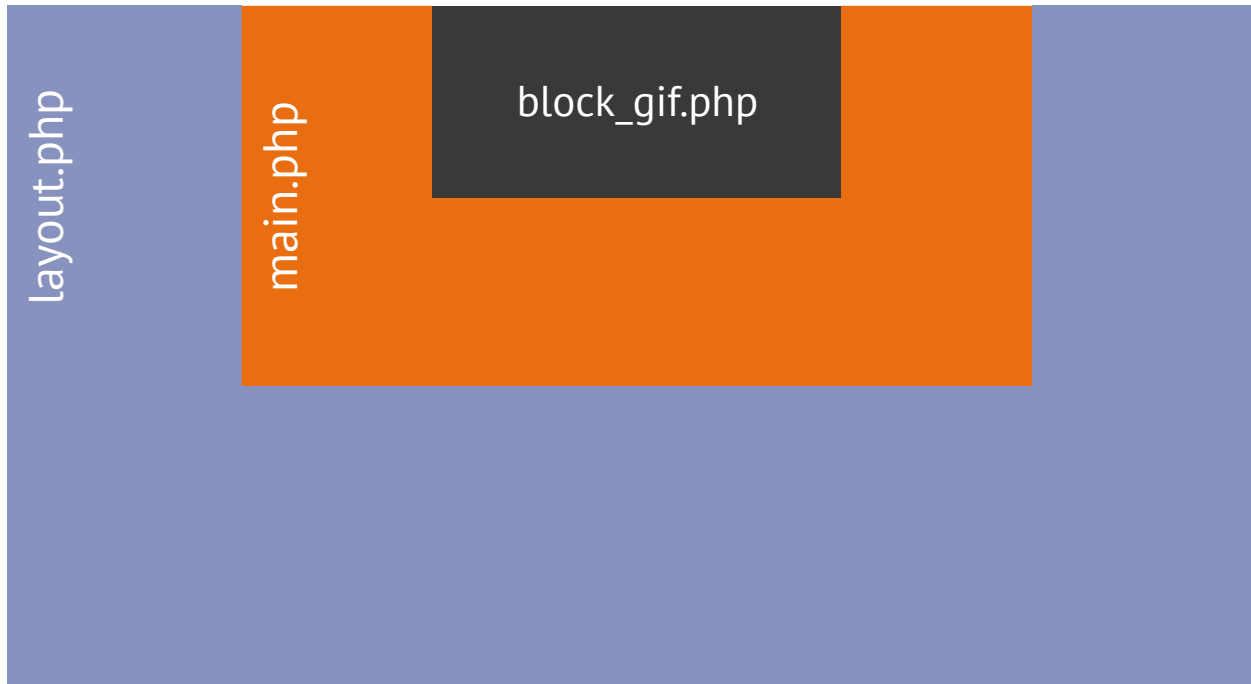
Собираем из шаблонов страницу



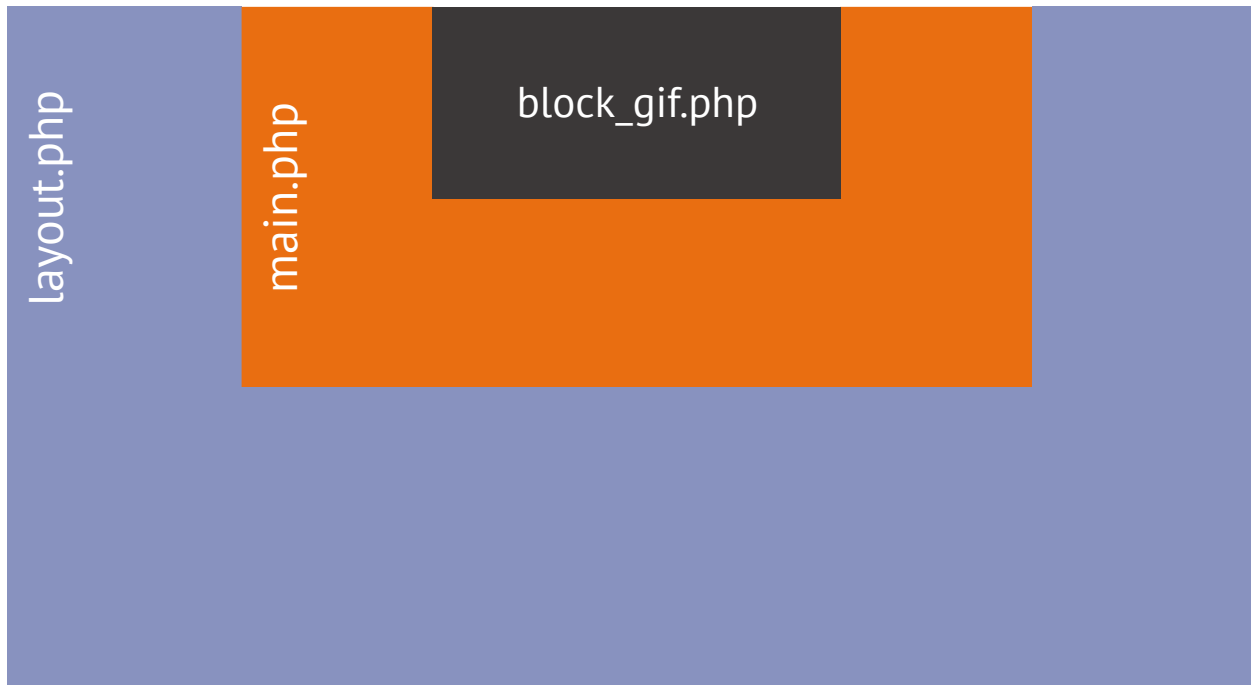
index.php

Собираем из шаблонов страницу

Собираем из шаблонов страницу



Собираем из шаблонов страницу



index.php



Поиск гифки...

НАЙТИ

Мой GIFTUBE

Регистрация

Вход для своих

КАТЕГОРИИ

Видеоигры

Животные

Люди

Наука

Приколы

Спорт

Фейлы

Фильмы и анимация

Видеоигры

НАЗАД



[Unstoppable](#)

@frexin

4



[Battlefield 1](#)

@frexin

0

Видеоигры

НАЗАД



[Unstoppable](#)

@frexin

4 ❤️



[Battlefield 1](#)

@frexin

0 ❤️

layout.php

GIFUBE



Поиск гифки...

НАЙТИ

YOUTUBE

Регистрация

Вход для своих

КАТЕГОРИИ

Видеоигры

Животные

Люди

Наука

Приколы

Спорт

Фейлы

Фильмы и анимация

main.php

Видеоигры

НАЗАД



[Unstoppable](#)

@frexin

4 ❤️



[Battlefield 1](#)

@frexin

0 ❤️

layout.php



Поиск гифки...

НАЙТИ

- РЕГИСТРАЦИЯ
- Регистрация
- Вход для своих

КАТЕГОРИИ

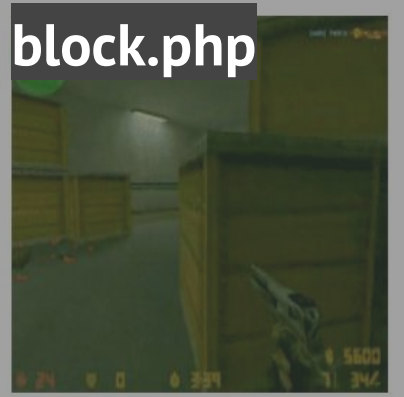
- Видеоигры
- Животные
- Люди
- Наука
- Приколы
- Спорт
- Фейлы
- Фильмы и анимация

main.php

Видеоигры

НАЗАД

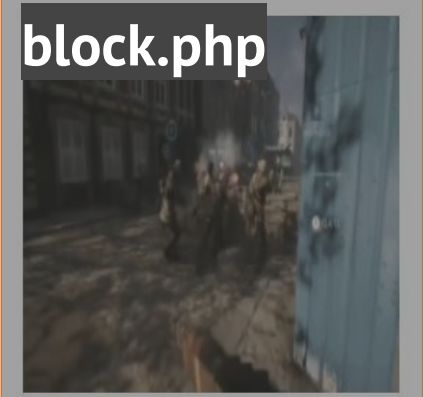
block.php



[Unstoppable](#)

@frexin 4

block.php



[Battlefield 1](#)

@frexin 0



Перерыв 10 минут



Терминология шаблонизации

Терминология шаблонизации

Лейаут (layout)

PHP-сценарий, содержащий HTML код общих частей страницы: стили, метатеги, шапка, подвал. Содержит область для вставки контента страницы.

Терминология шаблонизации

Лейаут (layout)

PHP-сценарий, содержащий HTML код общих частей страницы: стили, метатеги, шапка, подвал. Содержит область для вставки контента страницы.

Шаблон страницы

PHP-сценарий, содержащий HTML код, уникальный для одной страницы.

Например: стр. регистрации, главная, список пользователей. Может включать в себя блоки.

Терминология шаблонизации

Лейаут (layout)

PHP-сценарий, содержащий HTML код общих частей страницы: стили, метатеги, шапка, подвал. Содержит область для вставки контента страницы.

Шаблон страницы

PHP-сценарий, содержащий HTML код, уникальный для одной страницы.

Например: стр. регистрации, главная, список пользователей.
Может включать в себя блоки.

Блок

PHP-сценарий, с HTML кодом небольшого блока страницы.

Например: превью гифки, меню.

Один блок может использоваться в разных шаблонах страниц.

Изоляция данных

Изоляция данных

Любой компонент шаблонизации (лейаут, шаблон страницы, блок) должны иметь доступ только к данным, которые были явно ему переданы

Изоляция данных

Любой компонент шаблонизации (лейаут, шаблон страницы, блок) должны иметь доступ только к данным, которые были явно ему переданы

Для обеспечения такой изоляции, файл шаблона можно подключать внутри функции.

Сама функция должна принимать данные для шаблона.

Шаблонизатор

Шаблонизатор – функция, которая подключает файл шаблона, передаёт ему данные и возвращает сгенерированный HTML

Шаблонизатор

Шаблонизатор – функция, которая подключает файл шаблона, передаёт ему данные и возвращает сгенерированный HTML

*Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut
labore et*

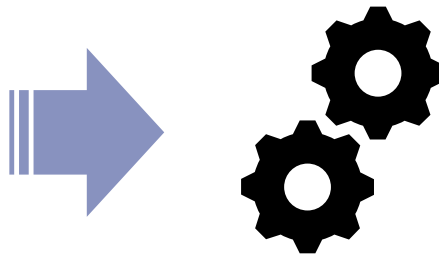
Данные для шаблона

Шаблонизатор

Шаблонизатор – функция, которая подключает файл шаблона, передаёт ему данные и возвращает сгенерированный HTML

*Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut
labore et*

Данные для шаблона



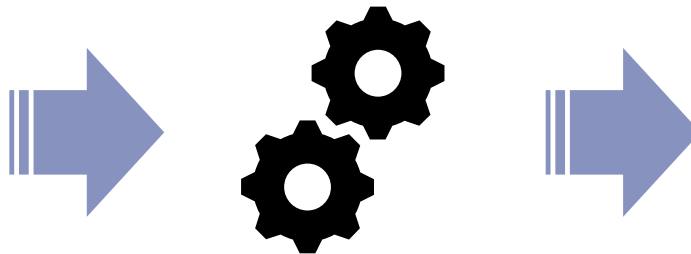
Шаблонизатор

Шаблонизатор

Шаблонизатор – функция, которая подключает файл шаблона, передаёт ему данные и возвращает сгенерированный HTML

*Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut
labore et*

Данные для шаблона



Шаблонизатор

<HTML>

```
<html>  
<title>HTML</title>  
<body>  
This is HTML!  
</body>  
</html>
```

Готовый HTML

Список действий

Что потребуется сделать для шаблонизации сайта:

Список действий

Что потребуется сделать для шаблонизации сайта:

1. Создать основной PHP-сценарий для этой страницы

Список действий

Что потребуется сделать для шаблонизации сайта:

1. Создать основной PHP-сценарий для этой страницы
2. Создать layout, где будет только HTML код, общий для всех страниц сайта

Список действий

Что потребуется сделать для шаблонизации сайта:

1. Создать основной PHP-сценарий для этой страницы
2. Создать layout, где будет только HTML код, общий для всех страниц сайта
3. Создать шаблон страницы, состоящий из её уникального HTML кода с включениями PHP-переменных

Список действий

Что потребуется сделать для шаблонизации сайта:

1. Создать основной PHP-сценарий для этой страницы
2. Создать layout, где будет только HTML код, общий для всех страниц сайта
3. Создать шаблон страницы, состоящий из её уникального HTML кода с включениями PHP-переменных
4. Выделить повторяющиеся части страницы в отдельные блоки

Список действий

Что потребуется сделать для шаблонизации сайта:

1. Создать основной PHP-сценарий для этой страницы
2. Создать layout, где будет только HTML код, общий для всех страниц сайта
3. Создать шаблон страницы, состоящий из её уникального HTML кода с включениями PHP-переменных
4. Выделить повторяющиеся части страницы в отдельные блоки
5. В шаблоне страницы вызвать необходимый блок

Список действий

Что потребуется сделать для шаблонизации сайта:

1. Создать основной PHP-сценарий для этой страницы
2. Создать layout, где будет только HTML код, общий для всех страниц сайта
3. Создать шаблон страницы, состоящий из её уникального HTML кода с включениями PHP-переменных
4. Выделить повторяющиеся части страницы в отдельные блоки
5. В шаблоне страницы вызвать необходимый блок
6. В основном сценарии вызвать шаблон страницы с передачей данных

Список действий

Что потребуется сделать для шаблонизации сайта:

1. Создать основной PHP-сценарий для этой страницы
2. Создать layout, где будет только HTML код, общий для всех страниц сайта
3. Создать шаблон страницы, состоящий из её уникального HTML кода с включениями PHP-переменных
4. Выделить повторяющиеся части страницы в отдельные блоки
5. В шаблоне страницы вызвать необходимый блок
6. В основном сценарии вызвать шаблон страницы с передачей данных
7. В основном сценарии вызвать лейаут с передачей шаблона страницы

Список действий

Что потребуется сделать для шаблонизации сайта:

1. Создать основной PHP-сценарий для этой страницы
2. Создать layout, где будет только HTML код, общий для всех страниц сайта
3. Создать шаблон страницы, состоящий из её уникального HTML кода с включениями PHP-переменных
4. Выделить повторяющиеся части страницы в отдельные блоки
5. В шаблоне страницы вызвать необходимый блок
6. В основном сценарии вызвать шаблон страницы с передачей данных
7. В основном сценарии вызвать лейаут с передачей шаблона страницы
8. Вывести на экран результат рендера лейаута

Из чего состоит шаблон?

Любой шаблон (шаблон страницы, лейаут, блок)
может состоять только из:

Из чего состоит шаблон?

Любой шаблон (шаблон страницы, лейаут, блок) может состоять только из:

- HTML кода

Из чего состоит шаблон?

Любой шаблон (шаблон страницы, лейаут, блок) может состоять только из:

- HTML кода
- Переменных и вызова функций

Из чего состоит шаблон?

Любой шаблон (шаблон страницы, лейаут, блок) может состоять только из:

- HTML кода
- Переменных и вызова функций
- Простых языковых конструкций (условия и циклы)

Из чего состоит шаблон?

Любой шаблон (шаблон страницы, лейаут, блок) может состоять только из:

- HTML кода
- Переменных и вызова функций
- Простых языковых конструкций (условия и циклы)
- Включения других шаблонов

Практика

соберём все вместе

План работы

План работы

1. Создать базовый лейаут

План работы

1. Создать базовый лейаут
2. Создать блок для превью гифки

План работы

1. Создать базовый лейаут
2. Создать блок для превью гифки
3. Создать шаблон для главной страницы

План работы

1. Создать базовый лейаут
2. Создать блок для превью гифки
3. Создать шаблон для главной страницы
4. Специфицировать функцию шаблонизации

План работы

1. Создать базовый лейаут
2. Создать блок для превью гифки
3. Создать шаблон для главной страницы
4. Специфицировать функцию шаблонизации
5. Написать сценарий, который подключит шаблоны и покажет итоговый результат

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <title><?=$title;?></title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
<div class="container">
  <header class="main-header"><h1 class="visually-hidden">Giftube</h1></header>
  <div class="main-content">
    <section class="navigation">
      <h3 class="navigation__title ">Категории</h3>
      <nav class="navigation__links">
        <?php foreach ($categories as $cat):?>
          <a href="/category?id=<?=$cat['id']; ?>"><?=$cat['name'];?></a>
        <?php endforeach; ?>
      </nav>
    </section>
    <main class="content">
      <?=$content;?>
    </main>
  </div>
  <footer class="main-footer">
    <div class="main-footer__col">
      Giftube: <a href="mailto:info@giftube.com">info@giftube.com</a>.
    </div>
  </footer>
</div>
</body>
</html>
```

Динамическое содержимое лейаута

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <title><?=$title;?></title>
  <link rel="stylesheet" href="css/main.css">
</head>
<body>
<div class="container">
  <header class="main-header"><h1 class="visually-hidden">Giftube</h1></header>
  <div class="main-content">
    <section class="navigation">
      <h3 class="navigation__title">Категории</h3>
      <nav class="navigation__links">
        <?php foreach ($categories as $cat):?>
          <a href="/category?id=<?=$cat['id']; ?>"><?=$cat['name'];?></a>
        <?php endforeach; ?>
      </nav>
    </section>
    <main class="content">
      <?=$content;?>
    </main>
  </div>
  <footer class="main-footer">
    <div class="main-footer__col">
      Giftube: <a href="mailto:info@giftube.com">info@giftube.com</a>.
    </div>
  </footer>
</div>
</body>
</html>
```



```
<!DOCTYPE html>
<html lang="ru">
<head>
  <title><?=$title;?></title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
<div class="container">
  <header class="main-header"><h1 class="visually-hidden">Giftube</h1></header>
  <div class="main-content">
    <section class="navigation">
      <h3 class="navigation__title">Категории</h3>
      <nav class="navigation__links">
        <?php foreach ($categories as $cat):?>
          <a href="/category?id=<?=$cat['id']; ?>"><?=$cat['name'];?></a>
        <?php endforeach; ?>
      </nav>
    </section>
    <main class="content">
      <?=$content;?>
    </main>
  </div>
  <footer class="main-footer">
    <div class="main-footer__col">
      Giftube: <a href="mailto:info@giftube.com">info@giftube.com</a>.
    </div>
  </footer>
</div>
</body>
</html>
```

Динамическое
содержимое лейаута

Сюда будет вставлено
содержимое страницы

Когда нужны разные лейауты?

Иногда требуется сделать больше одной лейаута для особых страниц сайта:

Когда нужны разные лейауты?

Иногда требуется сделать больше одной лейаута для особых страниц сайта:

- страница с ошибкой (404)

Когда нужны разные лейауты?

Иногда требуется сделать больше одной лейаута для особых страниц сайта:

- страница с ошибкой (404)
- страница входа

Когда нужны разные лейауты?

Иногда требуется сделать больше одной лейаута для особых страниц сайта:

- страница с ошибкой (404)
- страница входа
- лендинг

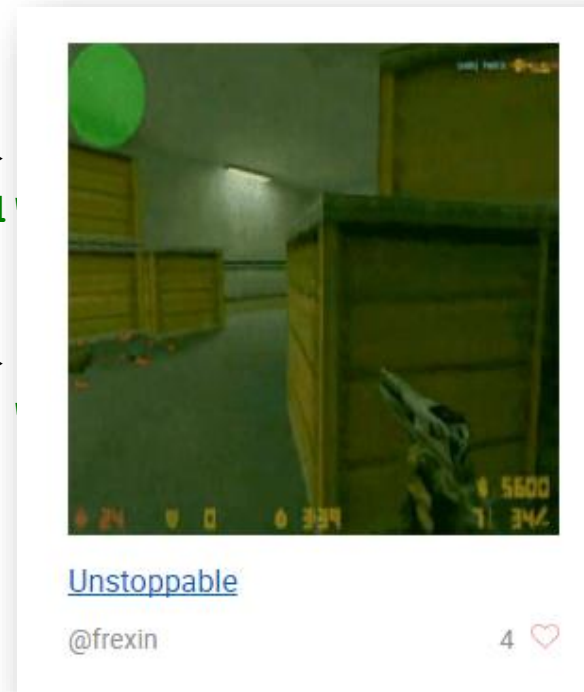
Блок для превью гифки

```
<li class="gif gif-list__item">
  <div class="gif__picture">
    <a href="/gif/view?id=<?=$gif['id'];?>" class="gif__preview">
      
    </a>
  </div>
  <div class="gif__desctiption">
    <h3 class="gif__desctiption-title">
      <a href="/gif/view?id=<?=$gif['id'];?>"><?=$gif['title'];?></a>
    </h3>

    <div class="gif__description-data">
      <span class="gif__likes"><?=$gif['like_count'];?></span>
    </div>
  </div>
</li>
```

Блок для превью гифки

```
<li class="gif gif-list__item">
  <div class="gif__picture">
    <a href="/gif/view?id=<?=$gif['id'];?>" class="gif__preview">
      
    </a>
  </div>
  <div class="gif__description">
    <h3 class="gif__description-title">
      <a href="/gif/view?id=<?=$gif['id'];?>" class="gif__preview"><?=$gif['title'];?></a>
    </h3>
    <div class="gif__description-data">
      <span class="gif__likes"><?=$gif['likes'];?></span>
    </div>
  </div>
</li>
```



Шаблон главной страницы

```
<div class="content__main-col">
  <h2 class="visually-hidden">Смешные гифки</h2>
  <a class="button" href="/gif/add">Загрузить свою</a>

  <ul class="gifs-list">
    <?php foreach ($gif_col as $gif): ?>
      <?=renderTemplate('inc/_gifs_grid', ['gif' => $gif]);?>
    <?php endforeach; ?>
  </ul>
</div>
```


Шаблон главной страницы

Подключаем и выводим
блок с превью гифки

```
<div class="col">
  <h2 class="visually-hidden">Смешные гифки</h2>
  <a class="button" href="/gif/add">Загрузить свою</a>

  <ul class="gifs-list">
    <?php foreach ($gif_col as $gif): ?>
      <?=renderTemplate('inc/_gifs_grid', ['gif' => $gif]);?>
    <?php endforeach; ?>
  </ul>
</div>
```

Шаблон главной страницы

Подключаем и выводим
блок с превью гифки

```
<div class="col">
  <h2 class="visually-hidden">Смешные гифки</h2>
  <a class="button" href="/gif/add">Загрузить свою</a>

  <ul class="gifs-list">
    <?php foreach ($gif_col as $gif): ?>
      <?=renderTemplate('inc/_gifs_grid', ['gif' => $gif]);?>
    <?php endforeach; ?>
  </ul>
</div>
```

Передаем в шаблон
гифку из массива

Топовые гифки

Свежачок

Загрузить свою



[Рыжая](#)

@frexin

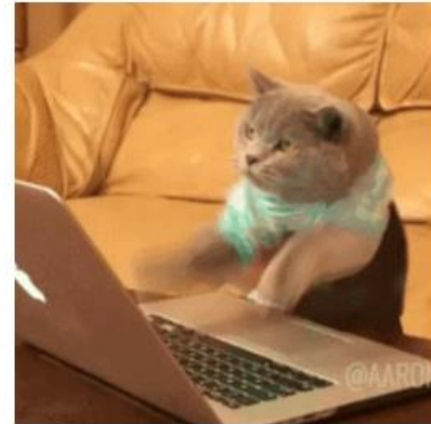
4 ❤️



[Unstoppable](#)

@frexin

4 ❤️



[Типичный юзер](#)

@frexin

1 ❤️



Функция шаблонизации

Функция шаблонизации

Аргументы функции:

- первый аргумент – путь к файлу-шаблона
- второй аргумент – массив с данными шаблона

Функция шаблонизации

Аргументы функции:

- первый аргумент – путь к файлу-шаблона
- второй аргумент – массив с данными шаблона

Что делает:

подключает переданный файл, захватывает его содержимое.

Функция шаблонизации

Аргументы функции:

- первый аргумент – путь к файлу-шаблона
- второй аргумент – массив с данными шаблона

Что делает:

подключает переданный файл, захватывает его содержимое.

Что возвращает:

тип результата – строка.

Возвращает сгенерированный HTML код шаблона

Главный сценарий

```
<?php
// двумерный массив с гифками
$gifs_col = [];


// HTML код главной страницы
$page_content = renderTemplate('inc/main.php', ['gifs' => $gifs_col]);

// окончательный HTML код
$layout_content = renderTemplate('inc/layout.php',
    ['content' => $page_content, 'title' => 'GifTube - Главная']);

print($layout_content);
```


Главный сценарий

Данные для шаблона
– в сценарии



```
<?php
// двумерный массив с гифками
$gifs_col = [];

// HTML код главной страницы
$page_content = renderTemplate('inc/main.php', ['gifs' => $gifs_col]);

// окончательный HTML код
$layout_content = renderTemplate('inc/layout.php',
    ['content' => $page_content, 'title' => 'GifTube - Главная']);

print($layout_content);
```

Главный сценарий

Данные для шаблона
– в сценарии

Получаем контент для
страницы

```
<?php
// двумерный массив с гифками
$gifs_col = [];

// HTML код главной страницы
$page_content = renderTemplate('inc/main.php', ['gifs' => $gifs_col]);

// окончательный HTML код
$layout_content = renderTemplate('inc/layout.php',
    ['content' => $page_content, 'title' => 'GifTube - Главная']);

print($layout_content);
```

Главный сценарий

Данные для шаблона
– в сценарии

Получаем контент для
страницы

```
<?php
// двумерный массив с гифками
$gifs_col = [];

// HTML код главной страницы
$page_content = renderTemplate('inc/main.php', ['gifs' => $gifs_col]);

// окончательный HTML код
$layout_content = renderTemplate('inc/layout.php',
    ['content' => $page_content, 'title' => 'GifTube - Главная']);

print($layout_content);
```

Подключаем лейаут и
передаем туда содержимое
страницы

Главный сценарий

Данные для шаблона
– в сценарии

Получаем контент для
страницы

```
<?php
// двумерный массив с гифками
$gifs_col = [];

// HTML код главной страницы
$page_content = renderTemplate('inc/main.php', ['gifs' => $gifs_col]);

// окончательный HTML код
$layout_content = renderTemplate('inc/layout.php',
    ['content' => $page_content, 'title' => 'GifTube - Главная']);

print($layout_content);
```

Показываем
сгенерированный
HTML всей страницы

Подключаем лейаут и
передаем туда содержимое
страницы

Буферизация вывода

накапливание пользовательского вывода во внутреннем «буфере» перед его действительным показом

Как работает буферизация

Как работает буферизация

```
print ( "Привет! " ) ;
```

Как работает буферизация

```
print ( "Привет! " ) ;
```



Пользовательский буфер

Как работает буферизация

```
print ( "Привет! " ) ;
```



Пользовательский буфер



Системный буфер

Как работает буферизация

```
print ( "Привет! " ) ;
```



Пользовательский буфер



Системный буфер



Вывод

Системный буфер

По умолчанию PHP накапливает весь вывод в своем буфере, а затем выводит всё вместе

Системный буфер

По умолчанию RHP накапливает весь вывод в своем буфере, а затем выводит всё вместе

Но можно заставить RHP «сбрасывать» весь буфер на экран после каждого вывода

Пример принудительного сброса

```
<?php
print("<h1>Подождите 10 секунд</h1>");

$counter = 1;

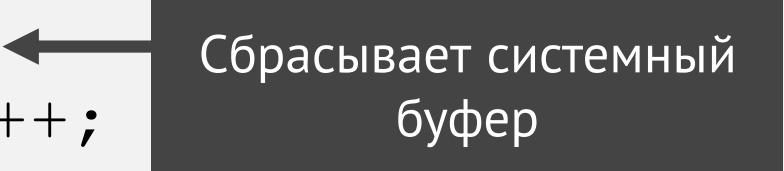
while ($counter <= 10) {
    print($counter);
    flush();
    $counter++;
    sleep(1);
}
```

Пример принудительного сброса

```
<?php
print("<h1>Подождите 10 секунд</h1>");

$counter = 1;

while ($counter <= 10) {
    print($counter);
    flush();
    $counter++;
    sleep(1);
}
```



Сбрасывает системный буфер

Пример принудительного сброса

```
<?php
print("<h1>Подождите 10 секунд</h1>");

$counter = 1;

while ($counter <= 10) {
    print($counter);
    flush();
    $counter++;
    sleep(1);
}
```

← Сбрасывает системный
буфер

Подождите 10 секунд

12345678910

Буферизация вывода

Функции, вроде *print* или *echo* выводят переданную информацию на экран.

Но PHP позволяет «захватывать» их вывод, чтобы получить это содержимое и, например, записать его в файл или использовать позже.

Буферизация вывода

Функции, вроде *print* или *echo* выводят переданную информацию на экран.

Но PHP позволяет «захватывать» их вывод, чтобы получить это содержимое и, например, записать его в файл или использовать позже.

Техника буферизации состоит из двух частей:

1. включить буферизацию вывода
2. получить содержимое буфера и очистить его

Буферизация вывода

Пример использования буферизации

```
<?php
ob_start();
print("Любой текст, вместо показа будет сохранен в буфере");
?>
<h1>some title</h1>
<?php $html = ob_get_clean();
```

Буферизация вывода

Пример использования буферизации

```
<?php
ob_start();
print("Любой текст, который будет сохранен в буфере");
?>
<h1>some title</h1>
<?php $html = ob_get_clean();
```

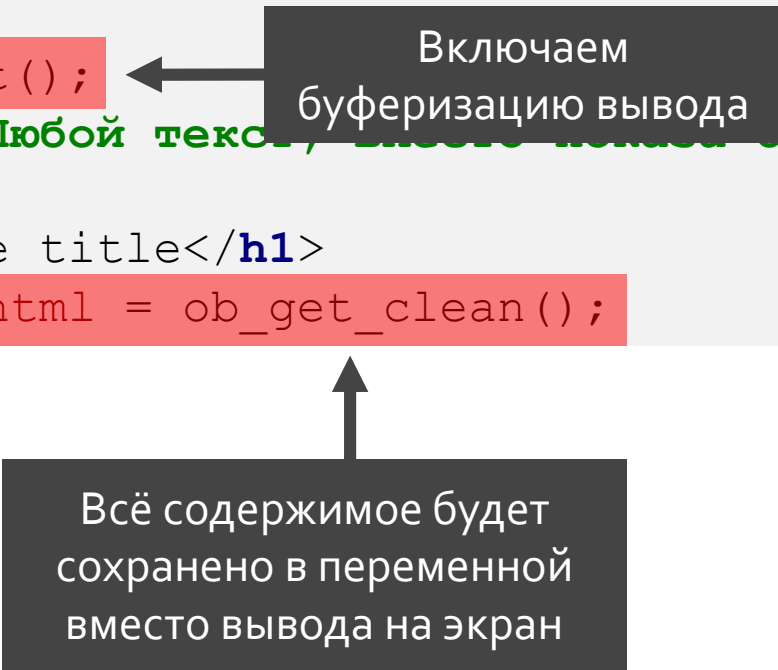
Включаем
буферизацию вывода

Буферизация вывода

Пример использования буферизации

```
<?php
ob_start();
print("Любой текст, который будет сохранен в буфере");
?>
<h1>some title</h1>
<?php $html = ob_get_clean();
```

Включаем
буферизацию вывода



Всё содержимое будет
сохранено в переменной
вместо вывода на экран

Сжатие HTML

```
<?php
ob_start('ob_gzhandler'); ?>
<!doctype html>
<html lang="ru">
<head><title>Пример PHP-сценария</title></head>
<body><h1>Информация о пользователе</h1>
<?php $browser = get_browser(null, true); ?>
<dl>
    <dt><strong>Ваш IP адрес:</strong></dt>
    <dd><em><?php print($_SERVER['REMOTE_ADDR']); ?></em></dd>
    <dt><strong>Операционная система:</strong></dt>
    <dd><em><?php print($browser['platform']); ?></em></dd>
    <dt><strong>Браузер:</strong></dt>
    <dd><em><?php print($browser['browser']); ?> версии
<?php print($browser['version']); ?></em></dd>
</dl>
</body>
</html>
<?php
ob_end_flush(); ?>
```

Сжатие HTML

```
<?php
ob_start('ob_gzhandler'); ?>
<!doctype html>
<html lang="ru">
<head><title>Пример PHP-сценария</title></head>
<body><h1>Информация о пользователе</h1>
<?php $browser = get_browser(null, true); ?>
<dl>
    <dt><strong>Ваш IP адрес:</strong></dt>
    <dd><em><?php print($_SERVER['REMOTE_ADDR']); ?></em></dd>
    <dt><strong>Операционная система:</strong></dt>
    <dd><em><?php print($browser['platform']); ?></em></dd>
    <dt><strong>Браузер:</strong></dt>
    <dd><em><?php print($browser['browser']); ?> версии
<?php print($browser['version']); ?></em></dd>
</dl>
</body>
</html>
<?php
ob_end_flush(); ?>
```

Результаты сжатия

Результаты сжатия

До сжатия

Информация о пользователе

Ваш IP адрес:
127.0.0.1

Операционная система:
Win10

Браузер:
Chrome версии 60.0

Elements Console Sources Network Performance >>

View: [Icons] Group by frame Preserve log Disable cache

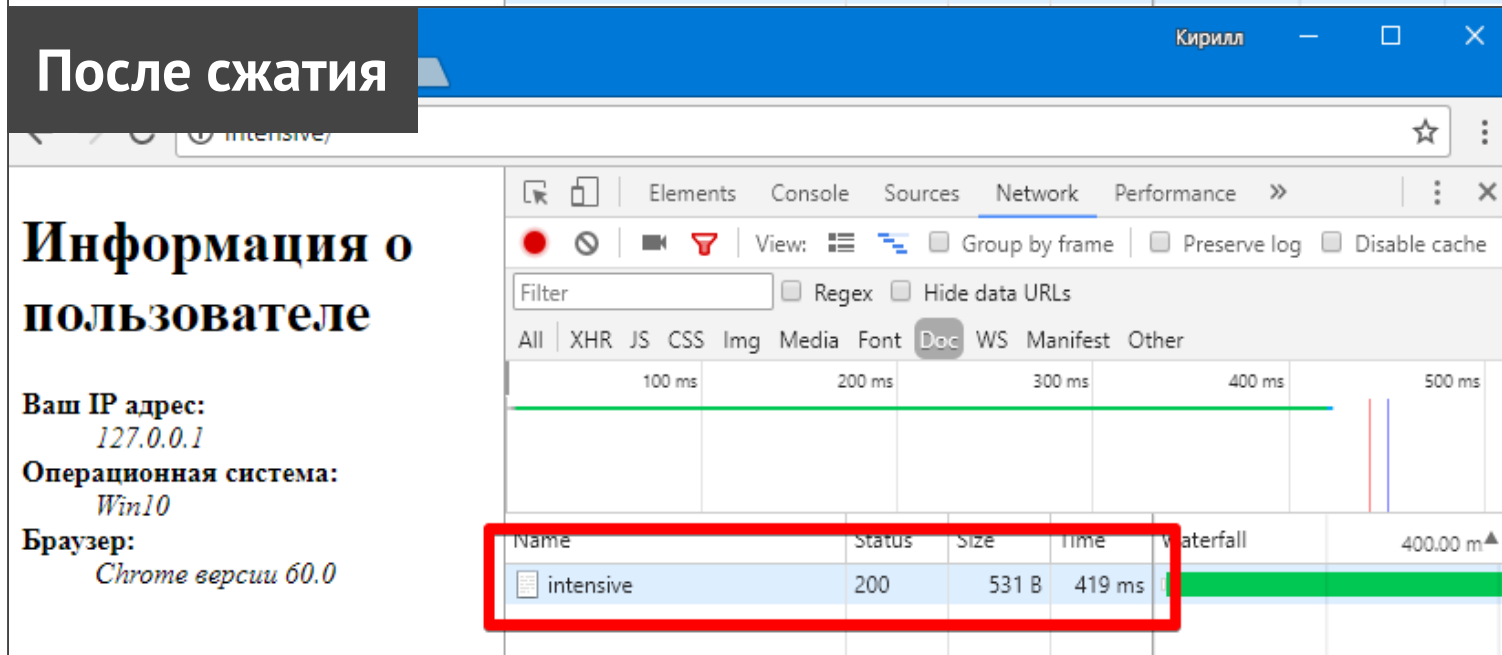
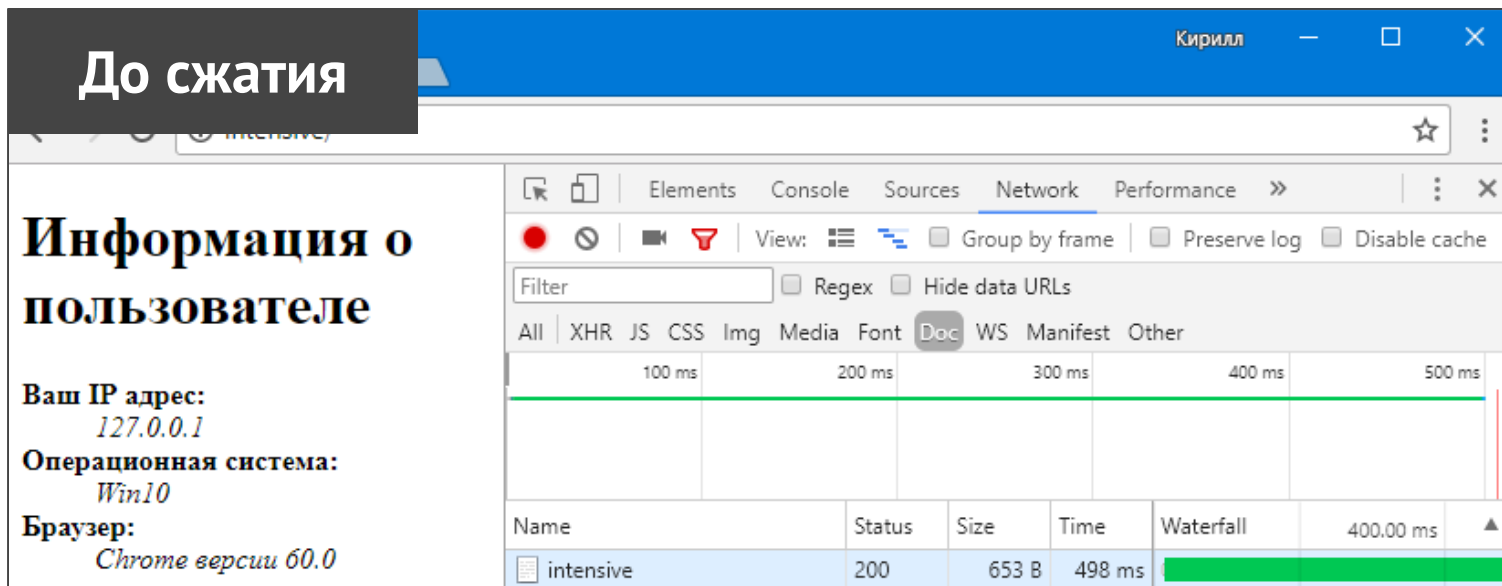
Filter [] [] Regex [] Hide data URLs

All XHR JS CSS Img Media Font Doc WS Manifest Other

100 ms 200 ms 300 ms 400 ms 500 ms

Name	Status	Size	Time	Waterfall	400.00 ms
intensive	200	653 B	498 ms	[Green bar]	

Результаты сжатия



Результаты сжатия

До сжатия

Информация о пользователе

Ваш IP адрес:
127.0.0.1

Операционная система:
Win10

Браузер:
Chrome версии 60.0

Network					
Filter					
All XHR JS CSS Img Media Font Doc WS Manifest Other					
100 ms 200 ms 300 ms 400 ms 500 ms					
Name	Status	Size	Time	Waterfall	
intensive	200	653 B	498 ms		400.00 ms

После сжатия

Информация о пользователе

Ваш IP адрес:
127.0.0.1

Операционная система:
Win10

Браузер:
Chrome версии 60.0

Network					
Filter					
All XHR JS CSS Img Media Font Doc WS Manifest Other					
ms 500 ms					
Name	Status	Size	Time	Waterfall	
intensive	200	531 B	419 ms		400.00 m

Размер уменьшился на ~ 20%

Безопасность в шаблонах

Фильтрация данных

Любую информацию, полученную от пользователя, **обязательно** необходимо фильтровать перед выводом в шаблоне!



Принцип атаки

Принцип атаки

1. На сайте есть форма для публикации сообщения

Принцип атаки

1. На сайте есть форма для публикации сообщения
2. Пользователь вместо простого текста отправляет HTML код и JS

Принцип атаки

1. На сайте есть форма для публикации сообщения
2. Пользователь вместо простого текста отправляет HTML код и JS
3. Сообщение публикуется на сайте

Принцип атаки


1. На сайте есть форма для публикации сообщения
2. Пользователь вместо простого текста отправляет HTML код и JS
3. Сообщение публикуется на сайте
4. Страница с просмотром сообщения «заражена»: там выполняется JS код злоумышленника

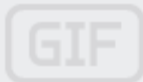
Пример атаки

Добавить гифку

НАЗАД

ГIF файл:





ВЫБРАТЬ ФАЙЛ

Категория:

Видеоигры

Название:


`<script>alert('Evil!')</script>`

Описание:

Краткое описание

ДОБАВИТЬ

Пример атаки




Поиск гифки

НАЙТИ


Подтвердите действие на giftube.academy:
Evil!
OK

```
<script>alert('Evil!')</script>
```



@frexin 143 0

Похожие гифки:



[Unstoppable](#)
@frexin 3

Что можно натворить

Что можно натворить

- испортить всю верстку страницы

Что можно натворить

- испортить всю верстку страницы
- перенаправить пользователя на свою страницу

Что можно натворить

- испортить всю верстку страницы
- перенаправить пользователя на свою страницу
- украсть куки

Что можно натворить

- испортить всю верстку страницы
- перенаправить пользователя на свою страницу
- украсть куки
- устроить DDoS-атаку

Что можно натворить

- испортить всю верстку страницы
- перенаправить пользователя на свою страницу
- украсть куки
- устроить DDoS-атаку
- множество других неприятных вещей

HTML-мнемоники

Мнемоника — это кодовое представление символа в HTML, начинающегося со знака амперсанда "&" и завершающееся точкой с запятой ";"

HTML-мнемоники

Мнемоника — это кодовое представление символа в HTML, начинающегося со знака амперсанда "&" и завершающееся точкой с запятой ";".

Result	Description	Entity Name	Entity Number
	non-breaking space	 	
<	less than	<	<
>	greater than	>	>
&	ampersand	&	&
"	double quotation mark	"	"
'	single quotation mark (apostrophe)	'	'

Как обезопасить данные

Удалить из данных любые HTML теги, либо
заменить их на HTML мнемоники

Как обезопасить данные

Удалить из данных любые HTML теги, либо заменить их на HTML мнемоники

Было

```
<script>
```

Как обезопасить данные

Удалить из данных любые HTML теги, либо заменить их на HTML мнемоники

Было

```
<script>
```

Стало

```
&lt;script&gt;
```

Как обезопасить данные

Как обезопасить данные

Преобразует все теги в мнемоники:

```
htmlspecialchars($string)
```


Как обезопасить данные

Преобразует все теги в мнемоники:

```
htmlspecialchars($string)
```

Вырезает все теги из текста:

```
strip_tags($string)
```

Пример в шаблоне

```
<li class="gif gif-list__item">
  <div class="gif__picture">
    <a href="/gif/view?id=<?=$gif['id'];?>" class="gif__preview">
      
    </a>
  </div>
  <div class="gif__description">
    <h3 class="gif__description-title">
      <a href="/gif/view?id=<?=$gif['id'];?>">
        <?=htmlspecialchars($gif['title']);?>
      </a>
    </h3>

    <div class="gif__description-data">
      <span class="gif__username">
        @<?=htmlspecialchars($gif['authorName']);?>
      </span>
      <span class="gif__likes"><?=$gif['like_count']; ?></span>
    </div>
  </div>
</li>
```