ОСНОВЫ WEB ПРОГРАММИРОВАНИЯ

02.03.03 -Математическое обеспечение и администрирование информационных систем, направленность (профиль) -разработка и администрирование информационных систем

http//vikchas.ru

Лекция 3. «Глобальные сети. Языки разметки гипертекста HTML и CSS. Скриптовый язык программирования JavaScript Язык программирования PHP»

Часовских Виктор Петрович
доктор технических наук, профессор кафедры
ШИиКМ, ФГБОУ ВО «Уральский государственный
экономический университет

Екатеринбург 2026

1. Глобальные сети

1.1. Понятие глобальной сети

Глобальная сеть — это компьютерная сеть, объединяющая устройства, локальные сети, серверы и пользователей на больших расстояниях: между городами, странами и континентами.

Главный пример глобальной сети — **Интернет**.

Интернет объединяет миллионы сетей и позволяет пользователям обмениваться данными, использовать веб-сайты, электронную почту, облачные сервисы, мессенджеры, видеосвязь и другие цифровые услуги.

1.2. Основные элементы глобальных сетей

К основным элементам глобальной сети относятся:

- **клиенты** — устройства пользователей: компьютеры, смартфоны, планшеты;
- **серверы** — компьютеры, предоставляющие ресурсы и сервисы;
- **маршрутизаторы** — устройства, направляющие сетевой трафик;
- **каналы связи** — кабельные, оптоволоконные, спутниковые, мобильные сети;
- **провайдеры** — организации, обеспечивающие доступ к сети;
- **сетевые протоколы** — правила передачи данных.

1.3. Клиент-серверная модель

Большинство веб-сервисов работает по модели **клиент — сервер**.

- **Клиент** отправляет запрос.
- **Сервер** обрабатывает запрос.
- **Сервер** возвращает ответ.

Пример:

1. Пользователь вводит адрес сайта в браузере.
2. Браузер отправляет запрос на сервер.

3. Сервер возвращает HTML-страницу, стили CSS, JavaScript-файлы и другие данные.
4. Браузер отображает сайт пользователю.

1.4. Основные сетевые протоколы

В глобальных сетях применяются разные протоколы:

- **IP** — отвечает за адресацию устройств в сети;
- **TCP** — обеспечивает надёжную передачу данных;
- **UDP** — используется для быстрой передачи данных без гарантии доставки;
- **HTTP** — протокол передачи веб-страниц;
- **HTTPS** — защищённая версия HTTP с шифрованием;
- **DNS** — преобразует доменные имена в IP-адреса;
- **FTP/SFTP** — передача файлов;
- **SMTP, POP3, IMAP** — работа электронной почты.

1.5. URL, домены и DNS

Чтобы открыть сайт, пользователь вводит адрес, например:

https://example.com/page

Этот адрес называется **URL**.

Он может включать:

- протокол: https;
- доменное имя: example.com;
- путь к ресурсу: /page.

DNS переводит понятное человеку доменное имя в IP-адрес сервера.

Например:

example.com → 93.184.216.34

2. HTML и CSS

HTML и CSS — базовые технологии создания веб-страниц.

Если представить сайт как здание:

- **HTML** — это каркас и структура;
- **CSS** — внешний вид и оформление;
- **JavaScript** — поведение и интерактивность;
- **PHP** — серверная логика и работа с данными.

2.1. HTML

2.1.1. Понятие HTML

HTML расшифровывается как **Hyper Markup Language**, то есть язык гипертекстовой разметки.

HTML используется для описания структуры веб-страницы:

- заголовков;
- абзацев;
- списков;
- таблиц;
- ссылок;
- изображений;
- форм;
- блоков;
- мультимедиа.

HTML не является языком программирования. Он не выполняет вычисления и не содержит алгоритмов. Это язык разметки.

2.1.2. Гипертекст

Гипертекст — это текст, содержащий ссылки на другие документы или ресурсы.

Именно гиперссылки позволяют переходить между страницами, сайтами и документами в интернете.

Пример ссылки:

HTML

```
<a href="https://example.com">Перейти на сайт</a>
```

2.1.3. HTML-документ

Типичный HTML-документ имеет структуру:

HTML

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>Моя страница</title>
</head>
<body>
   <h1>Заголовок страницы</h1>
   <p>Это абзац текста.</p>
</body>
</html>
```

Основные части:

- — указывает тип документа;
- — корневой элемент страницы;
- — служебная информация;
- — видимое содержимое страницы.

2.1.4. HTML-теги

HTML состоит из **тегов**.

Примеры:

HTML

```
<h1>Заголовок</h1>

<p>Абзац</p>

<img src="image.jpg" alt="Описание изображения">

<a href="page.html">Ссылка</a>
```

2.1.5. Семантическая разметка

Современный HTML использует семантические элементы, которые описывают смысл содержимого:

HTML

```
<header>Шапка сайта</header>

<nav>Навигация</nav>

<main>Основной контент</main>

<section>Раздел</section>

<article>Статья</article>

<footer>Подвал сайта</footer>
```

Семантика важна для:

- поисковых систем;
- доступности для людей с ограничениями;
- читаемости кода;
- правильной структуры документа.

2.2. CSS

2.2.1. Понятие CSS

CSS расшифровывается как **Cascading Style Sheets**, то есть каскадные таблицы стилей.

CSS отвечает за внешний вид веб-страницы:

- цвета;
- шрифты;
- размеры;

- отступы;
- расположение блоков;
- анимации;
- адаптивность;
- оформление кнопок, форм, меню.

HTML задаёт структуру, а CSS — оформление.

2.2.2. Пример CSS

CSS

```
body {
   font-family: Arial, sans-serif;
   background-color: #f5f5f5;
}

h1 {
   color: blue;
     -align: center;
}

p {
   font-size: 16px;
   line-height: 1.5;
}
```

2.2.3. Подключение CSS

CSS можно подключать тремя способами.

Внешний файл

HTML

```
<link rel="stylesheet" href="style.css">
```

Это наиболее правильный и распространённый способ.

Внутри HTML-документа

HTML

```
<style>
   p {
      color: red;
   }
</style>
```

В атрибуте style

HTML

```
<p style="color: red;">Текст</p>
```

Этот способ нежелателен для больших проектов, так как усложняет поддержку.

2.2.4. Каскадность CSS

CSS называется каскадным, потому что стили могут наследоваться и переопределяться.

На итоговый стиль влияют:

- порядок правил;
- специфичность селекторов;
- наследование;
- важность правила !important.

2.2.5. Адаптивный дизайн

CSS позволяет создавать сайты, которые корректно отображаются на разных устройствах:

- компьютерах;
- планшетах;
- смартфонах.

Для этого используются:

- гибкие сетки;
- относительные единицы измерения;
- медиазапросы.

Пример:

CSS

```
@media (max-width: 600px) {
  body {
    font-size: 14px;
  }
}
```

3. JavaScript

3.1. Понятие JavaScript

JavaScript — это скриптовый язык программирования, который чаще всего используется для создания интерактивности на веб-страницах.

Он выполняется в браузере пользователя и позволяет изменять страницу без полной перезагрузки.

JavaScript делает сайты динамическими.

3.2. Для чего используется JavaScript

JavaScript применяется для:

- обработки нажатий на кнопки;
- проверки форм;
- создания выпадающих меню;
- слайдеров и галерей;
- всплывающих окон;
- загрузки данных без перезагрузки страницы;
- работы с API;
- создания одностраничных приложений;

- игр в браузере;
- интерактивных карт;
- визуализации данных.

3.3. Пример JavaScript

HTML

```
<button onclick="showMessage()">Нажми меня</button>


<script>
function showMessage() {
   alert("Привет!");
}
</script>
```

При нажатии на кнопку браузер выполнит функцию и покажет сообщение.

3.4. DOM

Одно из ключевых понятий JavaScript в браузере — **DOM**.

DOM — это объектная модель документа, представление HTML-страницы в виде дерева объектов.

JavaScript может:

- находить элементы на странице;
- изменять текст;
- менять стили;
- добавлять новые элементы;
- удалять элементы;
- реагировать на события.

Пример:

JAVASCRIPT

```
document.querySelector("h1").  Content = "Новый заголовок";
```

3.5. События

JavaScript активно работает с событиями:

- клик мыши;
- ввод текста;
- отправка формы;
- загрузка страницы;
- наведение курсора;
- нажатие клавиши.

Пример:

JAVASCRIPT

```
document.querySelector("button").addEventListener("click", function() {
   alert("Кнопка нажата");
});
```

3.6. JavaScript и асинхронность

JavaScript может выполнять асинхронные запросы к серверу. Это позволяет получать данные без перезагрузки страницы.

Пример:

JAVASCRIPT

```
fetch("https://api.example.com/data")
   .then(response => response.json())
   .then(data => console.log(data));
```

Такая технология лежит в основе современных веб-приложений.

4. PHP

4.1. Понятие PHP

PHP — это серверный язык программирования, предназначенный для создания динамических веб-сайтов и веб-приложений.

Если JavaScript чаще выполняется в браузере пользователя, то PHP выполняется на сервере.

Пользователь не видит PHP-код. Он получает результат его выполнения — обычно HTML-страницу или данные.

4.2. Для чего используется PHP

PHP применяется для:

- обработки форм;
- регистрации и авторизации пользователей;
- работы с базами данных;
- генерации HTML-страниц;
- загрузки файлов;
- создания интернет-магазинов;
- управления контентом;
- создания API;
- отправки писем;
- обработки заказов и платежей.

4.3. Пример PHP

PHP

```
<?php

$name = "Иван";

echo "Привет, " . $name;

?>
```

Сервер выполнит код и отправит пользователю результат:

```
Привет, Иван
```

4.4. PHP и HTML

PHP может встраиваться в HTML:

PHP

```
<!DOCTYPE html>
<html>
<body>
   <h1>Добро пожаловать!</h1>

   <?php
      echo "<p>Сегодня: " . date("d.m.Y") . "</p>";
   ?>
</body>
</html>
```

На сервере PHP-код выполняется, а браузер получает готовый HTML.

4.5. PHP и базы данных

Одно из главных применений PHP — работа с базами данных, например MySQL.

Типичная схема:

1. Пользователь отправляет форму.
2. PHP получает данные.
3. PHP проверяет их.
4. PHP записывает данные в базу.
5. PHP возвращает пользователю результат.

Пример использования:

- регистрация пользователя;
- добавление товара;
- сохранение комментария;
- поиск информации;
- оформление заказа.

5. Как эти технологии работают вместе

Рассмотрим типичный процесс открытия сайта.

1. Пользователь вводит адрес сайта в браузере.
2. Браузер через интернет обращается к серверу.
3. Сервер получает запрос.
4. PHP обрабатывает запрос, при необходимости обращается к базе данных.
5. Сервер формирует HTML-страницу.
6. Браузер получает HTML, CSS и JavaScript.
7. HTML задаёт структуру страницы.
8. CSS оформляет страницу.
9. JavaScript добавляет интерактивность.
10. Пользователь взаимодействует с сайтом.

Упрощённая схема

Пользователь

↓ запрос

Браузер

↓ HTTP/HTTPS

Интернет

↓

Веб-сервер

↓

PHP + база данных

↓

HTML + CSS + JavaScript

↓

Браузер отображает страницу

6. Различие между HTML, CSS, JavaScript и PHP

Технология	**Основная роль**	**Где выполняется**
HTML	Структура страницы	В браузере
CSS	Внешний вид	В браузере
JavaScript	Интерактивность и логика на клиенте	Обычно в браузере
PHP	Серверная логика	На сервере

7. Пример взаимодействия

Допустим, есть форма входа на сайт.

HTML создаёт форму

HTML

```
<form method="post" action="login.php">
    <input type="  " name="login" placeholder="Логин">
    <input type="password" name="password" placeholder="Пароль">
    <button type="submit">Войти</button>
</form>
```

CSS оформляет форму

CSS

```
form {
    width: 300px;
    margin: 50px auto;
}


input {
```

```
    display: block;
    margin-bottom: 10px;
    padding: 8px;
}
```

JavaScript может проверить ввод

JAVASCRIPT

```
document.querySelector("form").addEventListener("submit", function(event) {
    const login = document.querySelector("[name='login']").value;

    if (login === "") {
        event.preventDefault();
        alert("Введите логин");
    }
});
```

PHP обрабатывает данные

PHP

```
<?php
$login = $_POST["login"];
$password = $_POST["password"];

// Проверка пользователя в базе данных
echo "Попытка входа пользователя: " . htmlspecialchars($login);
?>
```

8. Общая концепция веб-разработки

Современная веб-разработка строится на разделении ролей:

- **HTML** отвечает за содержание и структуру;
- **CSS** отвечает за визуальное оформление;

- **JavaScript** отвечает за поведение страницы;
- **PHP** отвечает за серверную обработку;
- **глобальная сеть** обеспечивает передачу данных между клиентом и сервером.

Вместе эти технологии формируют основу большинства сайтов и веб-приложений.

Краткий итог

Глобальные сети обеспечивают связь между устройствами и серверами по всему миру.

HTML описывает структуру веб-страницы.

CSS задаёт внешний вид и оформление.

JavaScript делает страницу интерактивной и динамической.

PHP выполняется на сервере, обрабатывает данные, работает с базами данных и формирует ответ пользователю.

В совокупности эти технологии позволяют создавать современные сайты: от простых информационных страниц до сложных веб-приложений, интернет-магазинов, порталов и онлайн-сервисов.

Оценочные материалы для проведения текущего контроля по дисциплине «Современные ВЕБ-технологии»

Открытые вопросы

1. Как связаны между собой HTML, CSS, JavaScript и PHP?
2. Что происходит, когда пользователь вводит адрес сайта в браузере?
3. Как браузер получает HTML-страницу?
4. Что делает сервер при получении запроса?
5. Как PHP может сформировать HTML-страницу?
6. Почему важно разделять структуру, оформление и поведение сайта?
7. Что такое фронтенд?

8. Что такое бэкенд?
9. Какие технологии относятся к фронтенду?
10. Какие технологии относятся к бэкенду?
11. Что такое веб-приложение?
12. Чем статический сайт отличается от динамического?
13. Как работает форма обратной связи на сайте?
14. Как происходит регистрация пользователя на сайте?
15. Какую роль играет база данных в веб-приложении?
16. Почему для современных сайтов важна безопасность?

Закрытые вопросы

1. **Какая технология отвечает за структуру веб-страницы?**
   - **HTML**
   - **CSS**
   - **JavaScript**
   - **PHP**
2. **Какая технология отвечает за оформление веб-страницы?**
   - **HTML**
   - **CSS**
   - **JavaScript**
   - **PHP**
3. **Какая технология отвечает за интерактивность страницы?**
   - **HTML**
   - **CSS**
   - **JavaScript**
   - **PHP**
4. **Какая технология выполняется на сервере?**
   - **HTML**
   - **CSS**
   - **JavaScript**
   - **PHP**
5. **Динамический сайт:**
   - Показывает заранее готовые страницы
   - Формирует страницы в зависимости от данных и действий пользователя