ДИСЦИПЛИНА	Фронтенд и бэкенд разработка
ИНСТИТУТ	Институт перспективных технологий и индустриального программирования
КАФЕДРА	Кафедра индустриального программирования
ВИД УЧЕБНОГО МАТЕРИАЛА	Практические занятия
ПРЕПОДАВАТЕЛЬ	Загородних Николай Анатольевич
СЕМЕСТР	1 семестр, 2024-2025 гг.

Практическое занятие 5. Работа с классами в CSS

Краткая теория

CSS классы

Классы в CSS позволяют применять стили к элементам HTML, которые имеют одинаковые характеристики. Классы обозначаются точкой перед именем класса и могут быть добавлены к любому элементу. Это позволяет легко изменять внешний вид элементов на странице, не затрагивая другие. Использование классов помогает организовать код и делает его более читаемым.

Пример:

```
<div class="card">...</div>
.card {
    border: 1px solid #ccc;
    padding: 10px;
    margin: 10px;
}
```

Эффективное написание CSS

Эффективное написание CSS — это ключ к созданию чистых, поддерживаемых и производительных веб-страниц. Вот несколько советов, которые помогут вам улучшить ваши навыки в CSS:

1. **Структурируйте свой код**. Разделяйте стили по компонентам или страницам. Используйте комментарии для обозначения различных секций и упрощения навигации по коду.

2. **Используйте семантические классы**. Названия классов должны отражать их назначение, а не визуальный стиль. Например, вместо .red-button используйте .primary-button, что делает код более понятным и гибким.

3. **Избегайте дублирования**. Старайтесь не повторять одни и те же стили. Используйте общие классы или переменные для повторяющихся значений.

4. **Организуйте селекторы**. Начинайте с более общих селекторов и переходите к более специфичным. Это поможет избежать конфликтов и повысит производительность.

5. **Минификация и сжатие**. Перед публикацией сжимайте CSS-файлы, чтобы уменьшить их размер и ускорить загрузку страницы.

6. **Проверяйте производительность**: Используйте инструменты разработчика в браузере для анализа времени загрузки стилей и выявления узких мест.

Использование одного CSS-файла для всех страниц или отдельных CSS-файлов для каждой страницы

Выбор между использованием одного CSS-файла для всех страниц или отдельными CSS-файлами для каждой страницы зависит от нескольких факторов, включая размер проекта, его структуру, производительность и удобство поддержки. Рассмотрим плюсы и минусы каждого подхода.

Один CSS на все страницы

Плюсы:

1. **Упрощение управления**. Один файл проще поддерживать и редактировать, так как все стили находятся в одном месте.

2. **Меньше HTTP-запросов**. При загрузке страницы браузер делает один запрос для получения CSS, что может ускорить загрузку.

3. **Кэширование**. Если файл не изменяется, он может быть кэширован браузером, что снижает время загрузки при повторных посещениях.

Минусы:

1. **Размер файла**. Если у вас много страниц с различным содержимым, общий CSS-файл может стать слишком большим, загружая стили, которые не используются на каждой странице.

2. **Производительность**. Большой файл может замедлить загрузку страницы, особенно на мобильных устройствах с медленным интернет-соединением.

Разные CSS для каждой страницы

Плюсы:

1. **Оптимизация загрузки**. Каждая страница загружает только те стили, которые ей нужны, что может снизить объем передаваемых данных и ускорить загрузку.

2. **Четкость и организация**. Легче управлять стилями, если они разделены по страницам, особенно в больших проектах.

3. **Изоляция стилей**. Меньше шансов столкнуться с конфликтами стилей между разными страницами.

Минусы:

1. **Больше HTTP-запросов**. Каждый CSS-файл требует отдельного запроса, что может замедлить загрузку страниц, особенно если их много.

2. **Сложность управления**. Если проект разрастается, управление множеством CSS-файлов может стать трудоемким.

Рекомендации

Комбинированный подход. В некоторых случаях можно использовать комбинацию подходов. Например, иметь общий CSS для общих стилей (шрифты, базовые стили) и отдельные файлы для специфичных стилей каждой страницы.

Использование CSS-препроцессоров. Препроцессоры, такие как SASS или LESS, могут помочь организовать стили и создавать модульные компоненты, которые затем можно компилировать в один файл или несколько.

Ленивая загрузка (lazy loading). Для страниц с тяжелыми стилями можно использовать технику ленивой загрузки, загружая дополнительные CSS-файлы только при необходимости.

В конечном итоге выбор подхода зависит от специфики вашего проекта, его масштаба и требований к производительности.

Сжатие CSS

Сжатие CSS — это процесс уменьшения размера файла CSS без потери функциональности. Это важно для ускорения загрузки страниц и улучшения производительности.

Вот несколько способов сжать CSS:

1. **Минификация**

Удалите все пробелы, переносы строк и комментарии. Минификация делает код компактнее, сохраняя его функциональность.

Для этого можно использовать онлайн-инструменты, такие как:

- CSS Minifier
- Minify CSS

Также существуют инструменты командной строки, такие как cssnano или clean-css, которые можно интегрировать в ваш процесс сборки.

2. **Использование сборщиков**

Инструменты сборки, такие как Webpack, Gulp или Grunt, могут автоматически минимизировать CSS во время процесса сборки вашего проекта.

Например, в Gulp можно использовать плагин gulp-clean-css:

```
const gulp = require('gulp');
const cleanCSS = require('gulp-clean-css');

gulp.task('minify-css', () => {
  return gulp.src('src/*.css')
    .pipe(cleanCSS({ compatibility: 'ie8' }))
    .pipe(gulp.dest('dist'));
});
```

3. **Сжатие на сервере**

Настройте сервер для сжатия CSS-файлов при их передаче клиенту. Используйте Gzip или Brotli для сжатия файлов на уровне сервера.

В случае использования Apache добавьте в файл .htaccess следующие строки:

```
AddOutputFilterByType DEFLATE text/css
```

4. **Использование препроцессоров**

Если вы используете CSS-препроцессоры (например, SASS или LESS), они часто имеют встроенные функции минификации при компиляции.

5. **Удаление неиспользуемого CSS**

Используйте инструменты вроде PurgeCSS или UnCSS, чтобы удалить неиспользуемые стили из вашего CSS-файла.

Полезные ссылки

- Исчерпывающий справочник по CSS
https://developer.mozilla.org/ru/docs/Web/CSS/Reference
- Пошаговое руководство по изучению CSS
https://developer.mozilla.org/ru/docs/Learn/CSS/First_steps
- Все о CSS (англ.)
https://www.w3schools.com/css/default.asp
- Учим CSS (структурированный справочник с примерами) есть аудиоматериал
https://hcdev.ru/learn/css3

Тренажеры

- CSS htmlacademy

https://htmlacademy.ru/courses/297/run/5

Задание

1. Создайте новые классы:

- Добавьте классы к элементам в вашей HTML-странице, чтобы стилизовать их. Например, создайте классы для заголовков, параграфов и карточек товаров.

- Убедитесь, что у вас есть классы для:

- Заголовков (например, .main-heading, .sub-heading)

- Параграфов (например, .text-paragraph)

- Карточек товаров (например, .product-card)

2. Стилизация с использованием классов:

- В файле styles.css создайте стили для созданных классов.

- Для заголовков измените цвет и размер шрифта.

- Для параграфов добавьте отступы и измените цвет текста.

- Для карточек товаров добавьте тень, границы и эффекты при наведении.

3. Применение нескольких классов:

- Попробуйте применить несколько классов к одному элементу. Например, у карточки товара могут быть классы .product-card и .highlight.

4. Использование псевдоклассов и псевдоэлементов:

- Используйте псевдоклассы (:hover, :active) для создания эффектов при наведении на элементы с классами.

- Используйте псевдоэлементы (::before, ::after, ::first-line, ::first-letter и ::selection) чтобы сократить ваш CSS.

5. Оптимизируйте (сожмите) итоговый css.

Критерии оценивания

- Правильность использования классов. Классы применены корректно, соответствуют заданию.

- Качество стилей. Стили выглядят эстетично, элементы хорошо оформлены.

- Использование псевдоклассов. Эффекты при наведении и клике работают корректно.

Форма сдачи

Студенты должны представить свой код в виде файлов .html .css, который можно открыть в браузере. Также важно прокомментировать код, чтобы объяснить, что делает каждый элемент.