УДК 004.738.5

Я.А. Игнаткова, ст.преп;
Н.П. Шутько, доц., канд. техн. наук
(БГТУ, г. Минск)

НОВЫЕ ВОЗМОЖНОСТИ CSS ДЛЯ АДАПТИВНОЙ ВЁРСТКИ

Адаптивная вёрстка, долгое время базировавшаяся на медиазапросах, подошла к критическому рубежу. Классический подход, привязывающий логику отображения исключительно к размерам окна браузера, демонстрирует свою недостаточность в эпоху компонентных дизайн-систем и сложных, модульных интерфейсов. Современные веб-приложения требуют более гибкой, контекстно-зависимой и предсказуемой адаптивности на уровне отдельных компонентов, а не всей страницы. Это вынуждает искать новые методологии и инструменты в рамках самого CSS, развитие которого в последние годы предоставило разработчикам мощные и нативные решения для этих задач.

Целью данного исследования является комплексный анализ ключевых современных возможностей спецификации CSS, предназначенных для создания адаптивных интерфейсов нового поколения.

В рамках работы ставится задача детально рассмотреть четыре фундаментальные технологии:

1. Контейнерные запросы (Container Queries).
2. Селектор :has().
3. Каскадные слои (@layer) .
4. Система современных логических и viewport-единиц.

Исследование направлено на выявление их практического потенциала для замены или дополнения устоявшихся подходов, а также на демонстрацию синергетического эффекта при их совместном использовании для построения устойчивых, переиспользуемых и истинно адаптивных компонентов.

Актуальность темы обусловлена завершением этапа экспериментальной поддержки рассматриваемых технологий ведущими браузерами. Их повсеместная имплементация переводит данные возможности из разряда экспериментальных в категорию промышленных стандартов, готовых к немедленному внедрению в реальные проекты. Игнорирование этих инструментов сегодня создает риск отставания в методологии разработки и увеличении стоимости поддержки кодовой базы.

Суть современного подхода заключается в переходе от адаптивности макета к адаптивности самих компонентов. Компонент должен

быть интеллектуальным: он должен адаптироваться не к размеру экрана, а к конкретному контексту своего расположения и внутреннему состоянию, оставаясь при этом независимым и переиспользуемым. Это достигается за счет набора новых CSS-возможностей, которые позволяют описывать такое поведение декларативно, без привлечения JavaScript.

Новые способы кардинально упрощают создание адаптивных интерфейсов. Они представляют собой целостную систему взаимодополняющих решений, охватывающих различные аспекты разработки. Ключевыми элементами этой системы являются: контейнерные запросы, переносящие логику адаптивности с уровня страницы на уровень родительского контейнера; селектор :has(), позволяющий компоненту реагировать на свое внутреннее содержимое; каскадные слои, решающие извечную проблему управления специфичностью и конфликтами стилей; современные единицы измерения и функции, обеспечивающие точный и гибкий контроль над размерами и типографикой в условиях разнообразных устройств и динамических интерфейсов.

Контейнерные запросы решают проблему, когда один и тот же компонент должен выглядеть по-разному в разных частях макета. Раньше для этого приходилось писать сложные медиазапросы, учитывающие вложенность, или использовать JavaScript, отслеживая размеры и применяя модификаторы вроде .card--compact. Теперь логика инкапсулирована в сам компонент. Родительский контейнер объявляется как container-type: inline-size;. Далее, для самой карточки достаточно написать правило @container (max-width: 400px) { .card { flex-direction: column; } }, чтобы она автоматически становилась вертикальной, как только ширина её контейнера становится меньше 400px, независимо от размера всего окна. Это делает компоненты самодостаточными и переиспользуемыми в любой сетке [1].

Селектор :has() открывает эру контекстной стилизации, устраняя множество мелких, но раздражающих задач, решавшихся скриптами. Например, отслеживать наличие активного пункта внутри выпадающего меню для подсветки его заголовка теперь реализуется одним CSS-правилом: .dropdown:has(.menu-item.active) { color: #2563eb; font-weight: bold; }. Стиль применяется к элементу с классом .dropdown только при условии, что внутри него существует элемент с классами .menu-item.active. Это мощнейший инструмент для создания интерактивных форм, где поле может подсвечиваться при наличии внутри невалидного ввода (.field:has(:invalid)), или навигаций, динамически меняющих своё состояние. Кроме того, применение селектора :has() облегчит работу верстальщикам, а также повысить скорость загрузки

сайтов в случае, когда, например, необходимо стилизовать особым образом карточки, внутри которых есть иконки (рис. 1) [2].

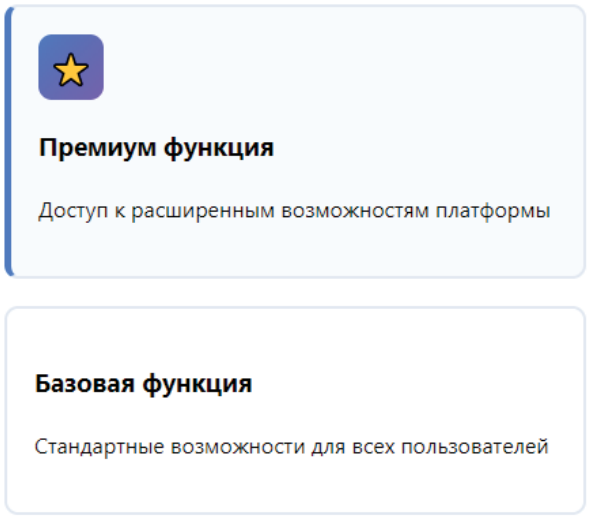

Рисунок 1 – Демонстрация работы селектора :has()

Каскадные слои (@layer) решают фундаментальную проблему архитектуры больших проектов − управление специфичностью. Вместо борьбы с агрессивными стилями сторонних библиотек через ещё более специфичные селекторы или !important, разработчик теперь может явно определить порядок слоёв: @layer reset, lib, components, utilities;. Оборачивая стили библиотеки в @layer lib { ... }, а свои компоненты − в @layer components { ... }, можно получить гарантию, что стили из более позднего слоя (components) будут иметь приоритет над стилями из более раннего (lib). Это создаёт предсказуемую и чистую CSS-архитектуру, легко масштабируемую в больших командах (рис. 2) [3].

Современные единицы измерения и функции делают адаптивную типографику и компоновку простыми и математически точными. Вместо громоздких медиазапросов с несколькими контрольными точками для масштабирования заголовка теперь достаточно одной строки с функцией clamp(): font-size: clamp(1.2rem, 4vw, 2rem);. Это задаёт плавное изменение размера между минимальным (1.2rem) и максимальным (2rem) значением. Для корректной работы с мобильными браузерами, где высота viewport меняется при скролле, вместо проблематичного 100vh используется height: 100dvh, что гарантирует точное заполнение именно видимой в данный момент области. Единицы lh и rlh позволяют привязывать отступы к высоте строки, создавая идеальный вертикальный ритм в тексте [4].

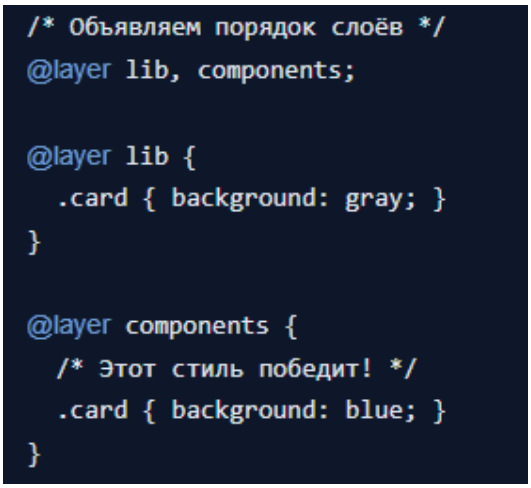

Рисунок 2 – Пример работы с правилом @layer

Таким образом, новые возможности CSS представляют собой не просто отдельные инструменты, а целостную экосистему для создания адаптивных интерфейсов следующего поколения. Они позволяют перенести логику адаптивности и интерактивности из JavaScript в декларативную область стилей, что повышает производительность, упрощает поддержку и делает компоненты по-настоящему независимыми и переиспользуемыми. Контейнерные запросы, селектор :has(), каскадные слои и современные единицы − это новая базовая реальность фронтенд-разработки. Их освоение и внедрение является не опциональным трендом, а необходимым шагом для создания современных, устойчивых и эффективных веб-приложений, отвечающих требованиям сложных пользовательских интерфейсов.

ЛИТЕРАТУРА

1. CSS container queries: [Электронный ресурс] // MDN Web Docs. − 2025. − URL: https://developer.mozilla.org/en-US/docs/Web/CSS/Guides/Containment/Container_queries (дата обращения: 19.02.2026).

2. Mastering CSS Container Queries: Adaptive Layouts for Modern Web Design: [Электронный ресурс] / Superflex.ai. − 2025. − URL: https://www.superflex.ai/blog/mastering-css-container-queries (дата обращения: 19.02.2026).

3. Container Queries: [Электронный ресурс] // CSS-Tricks. − 2024. − URL: https://css-tricks.com/css-container-queries/ (дата обращения: 19.02.2026).

4. Modern CSS for 2024: Nesting, Layers, and Container Queries: [Электронный ресурс] // Builder.io. − 2023. − URL: https://www.builder.io/blog/css-2024-nesting-layers-container-queries (дата обращения: 19.02.2026).