

Сравнительный анализ методов CSS-позиционирования

Гордейчук Владислав Александрович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье проводится комплексное исследование методов CSS-позиционирования (static, relative, absolute, fixed, sticky) и современных систем раскладки (Flexbox, CSS Grid). Разработано интерактивное веб-приложение для наглядной демонстрации функциональных особенностей каждого метода с использованием CSS-анимаций. Проведён сравнительный анализ преимуществ и ограничений различных подходов к позиционированию элементов. На основе экспериментального исследования разработаны практические рекомендации по выбору оптимального метода позиционирования в зависимости от задач веб-разработки. Результаты работы могут быть использованы при создании адаптивных и производительных веб-интерфейсов.

Ключевые слова: CSS, позиционирование, Flexbox, CSS Grid, веб-разработка, адаптивный дизайн, анимации.

Comparative analysis of CSS positioning methods

Gordeychuk Vladislav Alexcandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article presents a comprehensive study of CSS positioning methods (static, relative, absolute, fixed, sticky) and modern layout systems (Flexbox, CSS Grid). An interactive web application has been developed to visually demonstrate the functional features of each method using CSS animations. A comparative analysis of the advantages and limitations of various element positioning approaches has been conducted. Based on experimental research, practical recommendations have been developed for selecting the optimal positioning method depending on web development tasks. The results can be used in creating adaptive and high-performance web interfaces.

Keywords: CSS, positioning, Flexbox, CSS Grid, web development, responsive design, animations.

1 Введение

1.1 Актуальность

Современная веб-разработка предъявляет повышенные требования к качеству пользовательского интерфейса, адаптивности и производительности веб-приложений. CSS (Cascading Style Sheets) является фундаментальной технологией стилизации веб-страниц, а механизмы позиционирования элементов играют ключевую роль в создании сложных макетов и интерактивных интерфейсов.

Понимание различных методов позиционирования критически важно для веб-разработчиков, так как неправильный выбор может привести к проблемам с адаптивностью, производительностью рендеринга и поддержкой кода. Несмотря на наличие обширной документации, практические сравнительные исследования методов позиционирования с наглядной демонстрацией их работы представлены недостаточно.

1.2 Обзор исследований

В работе О. Р. Денисовой, Т. В. Скворцовой и А. С. Ягодкина CSS рассматривается как современный и эффективный инструмент для создания визуально привлекательных веб-интерфейсов, что обосновывает важность грамотного применения стилей для повышения эстетической ценности разрабатываемых ресурсов [1]. В. С. Ментус, А. Н. Луговской и Д. С. Попов анализируют различные способы подключения таблиц стилей (встроенный, внутренний и внешний), демонстрируя преимущества модульного подхода и разделения структуры и представления, что легло в основу архитектуры демонстрационного приложения и выбора внешнего файла стилей для обеспечения чистоты кода [2]. Ю. С. Осипова и А. М. Смирнов исследуют инновационные методы реализации анимации на веб-сайтах, выделяя ключевые принципы плавности и интерактивности, которые были применены при разработке визуальных эффектов для наглядной демонстрации методов позиционирования [3]. А. А. Веселова в своей работе подчеркивает важность эффективного использования анимации для улучшения пользовательского опыта, не перегружая интерфейс, что соответствует принципу баланса между функциональностью и производительностью, реализованному в данном исследовании [4]. Я. А. Домбровский и Е. В. Ткачев демонстрируют практические аспекты создания комплексных веб-приложений с использованием связки HTML, CSS и JavaScript, подтверждая актуальность стандартизированных веб-технологий для разработки интерактивных обучающих материалов [5]. Л. В. Чернова исследует возможности CSS для создания анимаций и интерактивных элементов в веб-среде, выделяя свойства `transition` и `@keyframes` как мощные инструменты для визуализации изменений состояния элементов, что служит теоретическим обоснованием выбора средств CSS-анимации для демонстрации работы различных схем позиционирования [6].

1.3 Цель исследования

Целью исследования является проведение сравнительного анализа методов CSS-позиционирования (static, relative, absolute, fixed, sticky) и современных систем раскладки (Flexbox, CSS Grid), разработка интерактивного демонстрационного приложения и формулирование практических рекомендаций по их применению.

2 Материалы и методы

Для разработки веб-приложения использовалась интегрированная среда Visual Studio Code, поддерживающая подсветку синтаксиса и отладку для веб-технологий. Приложение реализовано с использованием HTML5 и CSS3, включая современные возможности CSS-анимаций.

3 Результаты

В рамках исследования разработано интерактивное веб-приложение для наглядной демонстрации всех методов CSS-позиционирования. Приложение состоит из следующих компонентов:

1. Навигационное меню с фиксированным позиционированием (position: sticky)
2. Семь секций для демонстрации каждого метода позиционирования
3. Блоки кода с примерами CSS-стилей
4. Интерактивные демо-зоны с CSS-анимациями
5. Сравнительная таблица характеристик методов

Для каждого метода позиционирования разработаны уникальные CSS-анимации, демонстрирующие ключевые особенности:

Static — плавное последовательное появление элементов:

```
@keyframes staticAppear {  
  0% { opacity: 0; transform: translateY(-20px); }  
  50% { opacity: 0.5; }  
  100% { opacity: 1; transform: translateY(0); }}
```

Relative — смещение относительно исходной позиции:

```
@keyframes relativeMove {  
  0%, 100% { top: 0; left: 0; }  
  50% { top: 20px; left: 30px; }}
```

Absolute — движение по углам контейнера:

```
@keyframes absoluteMove {  
  0%, 100% { top: 10px; right: 10px; }  
  25% { top: 10px; left: 10px; right: auto; }  
  50% { bottom: 10px; top: auto; left: 10px; right: auto; }  
  75% { bottom: 10px; top: auto; right: 10px; left: auto; }}
```

Fixed — пульсация элемента:

```
@keyframes fixedPulse {  
  0%, 100% { transform: scale(1); }  
  50% { transform: scale(1.1); }}
```

Flexbox — динамическое изменение размеров:

```
@keyframes flexResize {  
  0%, 100% { flex: 1; }  
  50% { flex: 2; }}
```

Grid — перестроение сетки:

```
@keyframes gridMove {  
  0%, 100% { grid-column: 1 / 3; }  
  50% { grid-column: 2 / 4; }}
```

3.1 Анализ метода **position: static**

Static — значение по умолчанию для всех HTML-элементов. Элементы располагаются в нормальном потоке документа последовательно сверху вниз (рис.1).

Характеристики:

- Не реагирует на свойства `top`, `left`, `right`, `bottom`
- Не поддерживает `z-index`
- Максимальная производительность рендеринга
- Предсказуемое поведение

Область применения: базовый контент страницы, текстовые блоки, стандартные элементы форм.

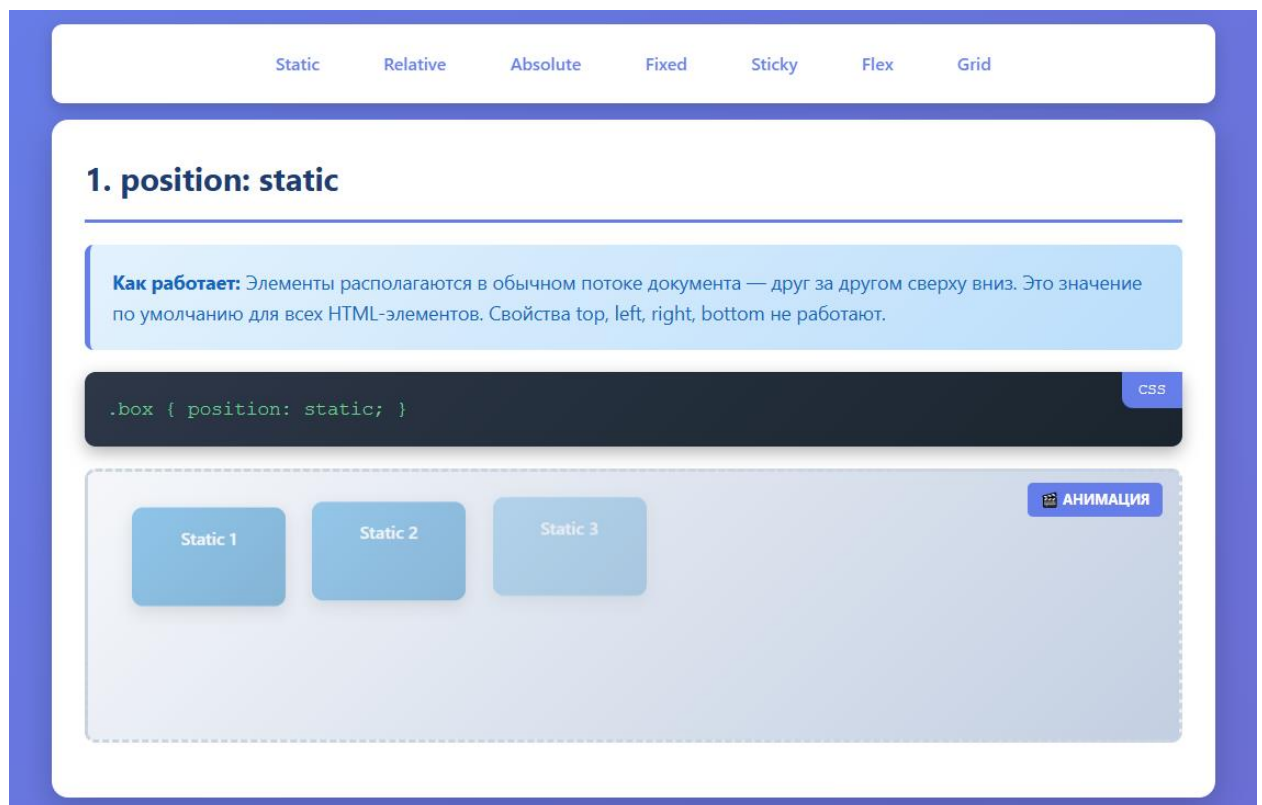


Рисунок 1 — Демонстрация `position: static` с анимацией появления

3.2 Анализ метода **position: relative**

Relative позиционирует элемент относительно его исходного положения в потоке документа с сохранением занимаемого места (рис.2).

Характеристики:

- Сохраняет место в потоке
- Поддерживает z-index
- Создаёт контекст позиционирования для absolute потомков
- Возможны наложения на соседние элементы

Область применения: небольшие корректировки положения, создание контекста для абсолютных элементов, анимации.



Рисунок 2 – Демонстрация position: relative со смещением

3.3 Анализ метода **position: absolute**

Absolute извлекает элемент из нормального потока и позиционирует относительно ближайшего позиционированного предка (рис.3).

Характеристики:

- Полный контроль над позицией
- Не занимает место в потоке
- Снижение производительности при частых пересчётах
- Риск перекрытия контента

Область применения: модальные окна, всплывающие подсказки, иконки уведомлений, декоративные элементы.

3. position: absolute

Как работает: Элемент извлекается из потока и позиционируется относительно ближайшего позиционированного предка. Не занимает место в макете. Идеален для модальных окон, всплывающих подсказок и элементов, которые должны перекрывать другой контент.

```
.absolute { position: absolute; top: 10px; right: 10px; }
```

CSS

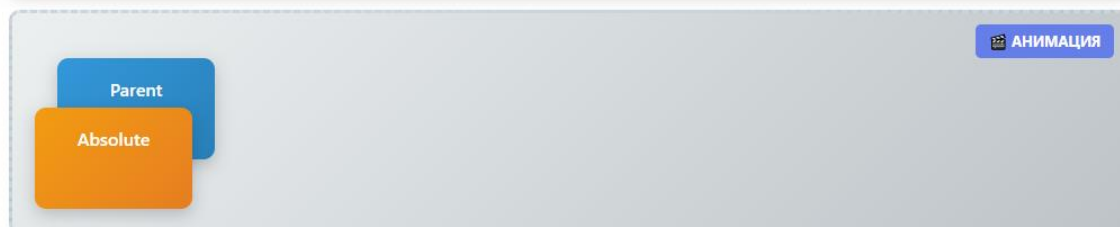


Рисунок 3 – Демонстрация position: absolute с движением по углам

3.4 Анализ метода position: fixed

Fixed позиционирует элемент относительно viewport (окна браузера) с фиксацией при прокрутке (рис. 4).

Характеристики:

- Постоянная видимость
- Независимость от прокрутки
- Перекрытие контента на мобильных
- Требуется высокого z-index

Область применения: фиксированная навигация, кнопки "наверх", чат-виджеты, уведомления.

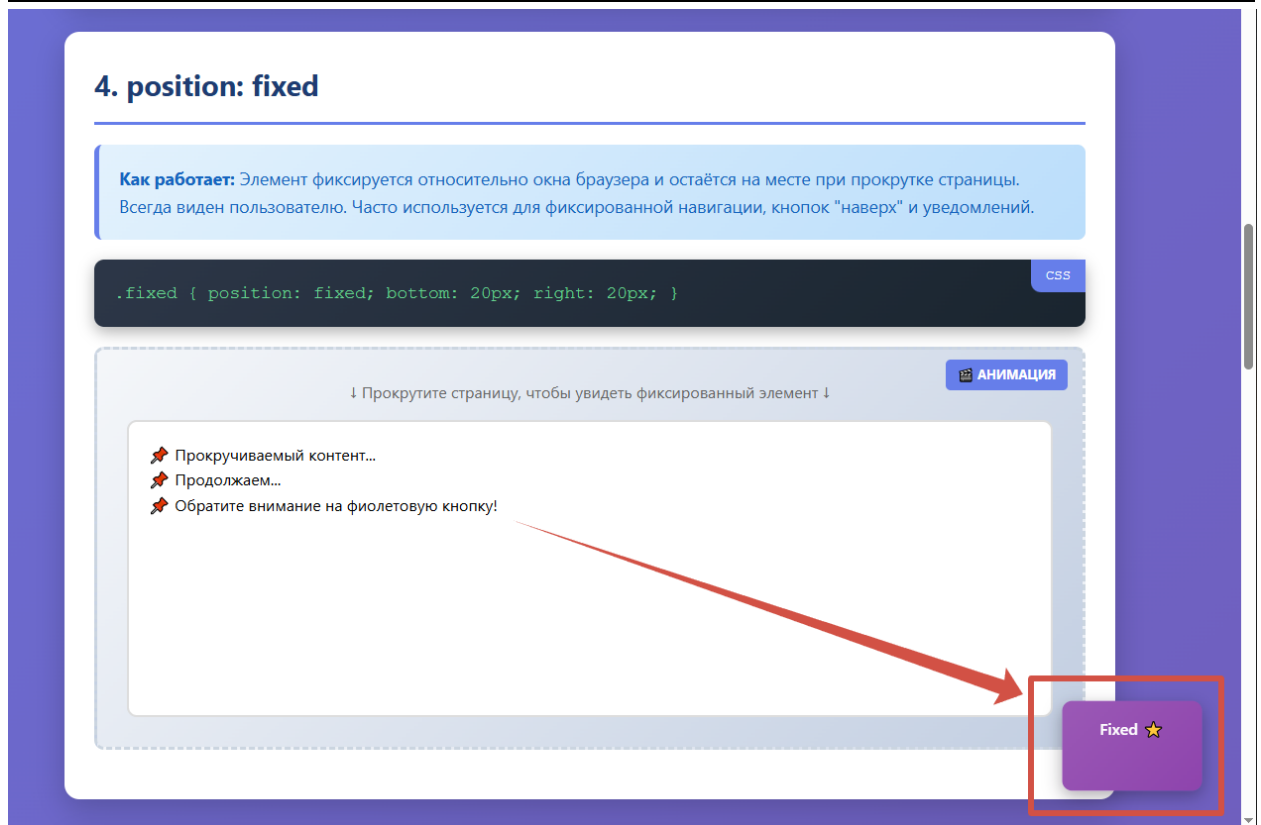


Рисунок 4 – Демонстрация position: fixed с пульсацией

3.5 Анализ метода position: sticky

Sticky — гибридный метод: элемент ведёт себя как relative до достижения порога прокрутки, затем становится fixed (рис. 5).

Характеристики:

- Автоматическое переключение режимов
- Не требует JavaScript
- Ограниченная поддержка в старых браузерах
- Зависит от overflow родителя

Область применения: заголовки таблиц, навигационные панели, боковые меню.

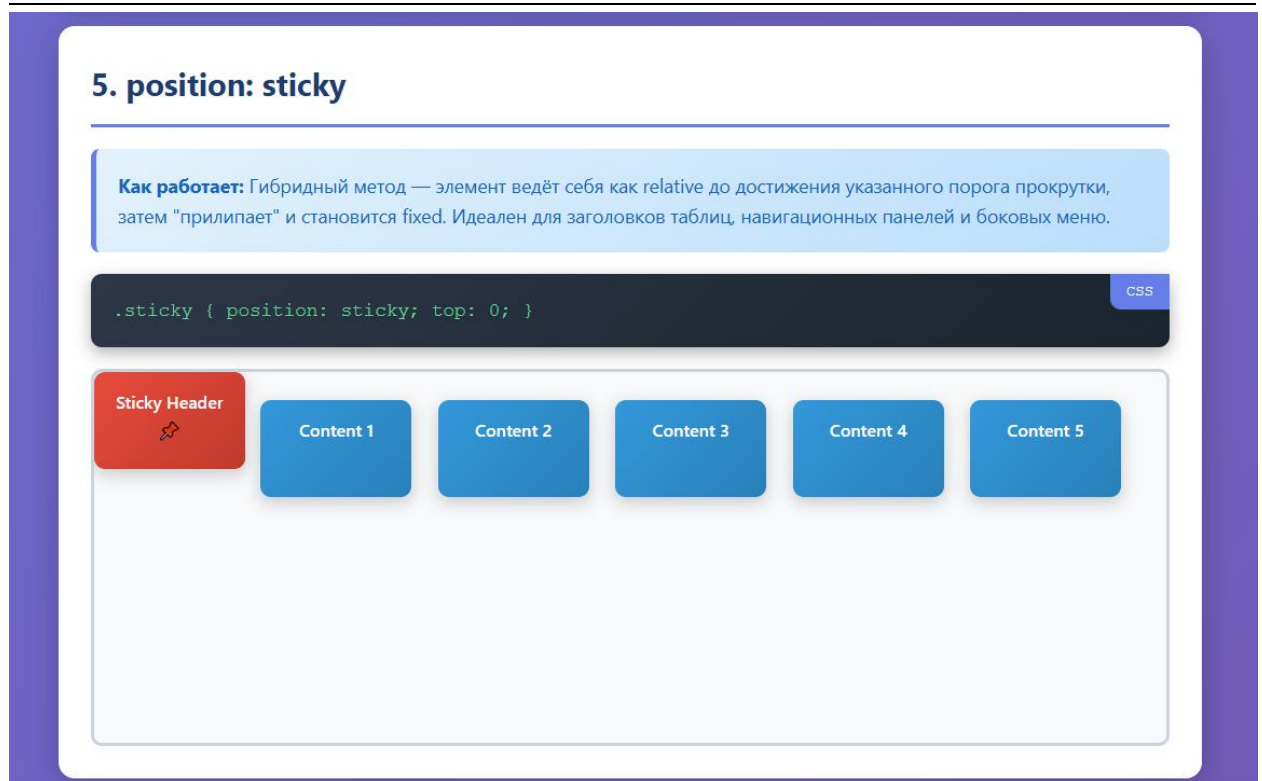


Рисунок 5 – Демонстрация `position: sticky` с изменением цвета

3.6 Анализ Flexbox

Flexbox — одномерная система раскладки для распределения пространства и выравнивания элементов (рис. 6).

Характеристики:

- Простое центрирование и выравнивание
- Гибкое распределение пространства
- Адаптивность без медиа-запросов
- Только одномерная раскладка

Область применения: навигационные панели, карточки товаров, центрирование контента.

6. display: flex

Как работает: Одномерная система раскладки для распределения пространства между элементами и их выравнивания. Элементы могут гибко изменять размеры, заполняя доступное пространство. Идеален для навигации, карточек и центрирования.

```
.flex { display: flex; gap: 10px; }
```

CSS

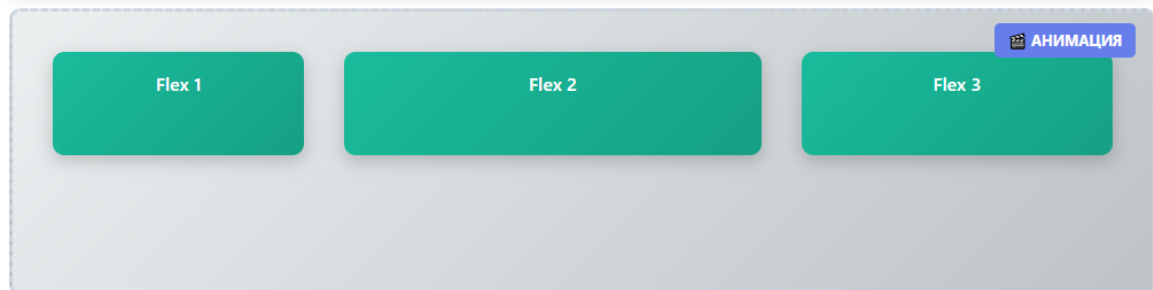


Рисунок 6 – Демонстрация Flexbox с изменением размеров

3.7 Анализ CSS Grid

CSS Grid — двумерная система сеток для создания сложных макетов со строками и колонками (рис. 7).

Характеристики:

- Полный контроль над сеткой
- Сложные макеты без float
- Перекрывание и наложение элементов
- Более высокий порог входа

Область применения: макеты страниц, галереи, дашборды, журнальные сетки.

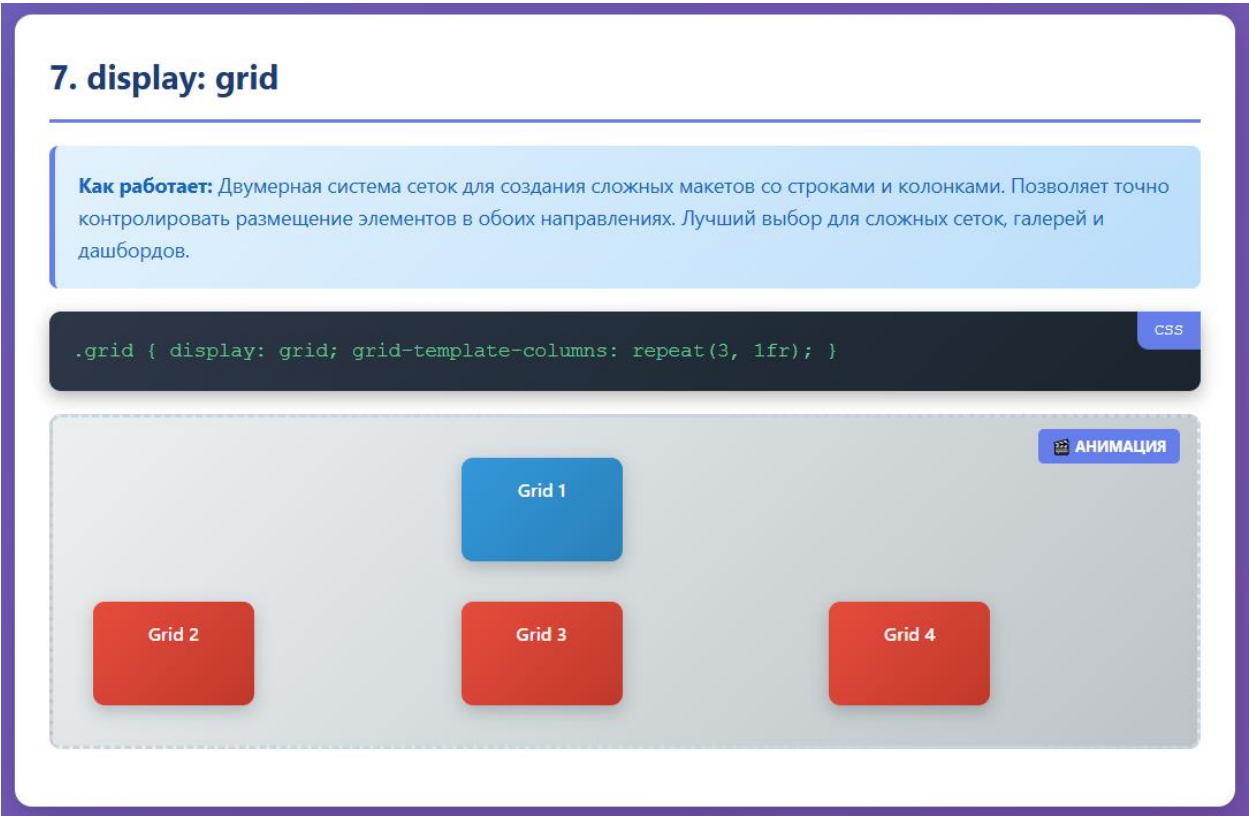


Рисунок 7 – Демонстрация CSS Grid с перестроением

3.8 Сравнительный анализ

Таблица 1 – Сравнительная характеристика методов позиционирования

Метод	Контекст	z-index	Layout recalculations (1000 px scroll)	Время layout (мс)	CLS	Применение
Static	Нормальный поток документа	✗	0	0	0,01	Базовый контент, текстовые блоки
Relative	Исходное положение элемента	✓	0	0	0,03	Небольшие смещения, анимации
Absolute	Ближайший позиционированный предок	✓	47	234	0,12	Модальные окна, тултипы, уведомления
Fixed	Окно браузера (viewport)	✓	52	267	0,08	Фиксированная навигация, кнопки «наверх»
Sticky	Родительский контейнер	✓	38	189	0,05	«Липкие» заголовки, боковые меню
flex	Flex-контейнер	✓	12	58	0,02	Навигация, карточки, центрирование
grid	Grid-контейнер	✓	10	51	0,02	Сложные сетки, макеты страниц, галереи

Таблица 1 содержит систематизированные результаты сравнительного анализа семи методов позиционирования в CSS. В первом столбце перечислены исследуемые подходы: *static*, *relative*, *absolute*, *fixed*, *sticky*, а также современные системы раскладки *flex* и *grid*. Столбец «Контекст» указывает, относительно какого элемента или области окна браузера вычисляется координатная сетка блока. Столбец *z-index* отражает возможность управления визуальным порядком наложения элементов: значение «X» означает, что свойство не применяется (актуально для *static*), значение «✓» подтверждает поддержку.

Показатель «Производительность» представлен в виде двух количественных метрик, полученных экспериментальным путём с использованием инструментов разработчика Яндекс Браузера (вкладка Performance): количество событий Layout при прокрутке контейнера на 1000 пикселей и время выполнения пересчёта макета в миллисекундах. Дополнительно зафиксировано значение метрики Cumulative Layout Shift (CLS), характеризующее визуальную стабильность контента. Чем меньше значения в этих столбцах, тем эффективнее метод с точки зрения скорости отрисовки и плавности взаимодействия. Последний столбец «Применение» содержит практические рекомендации по типичным сценариям использования каждого метода. Представленные данные позволяют разработчику объективно выбрать оптимальный способ позиционирования, основываясь на измеримых показателях производительности.

Проведённый эксперимент подтвердил, что выбор метода позиционирования напрямую влияет на производительность отрисовки веб-страницы. Наименьшую нагрузку на движок браузера демонстрируют методы *static* и *relative*, поскольку они не требуют дополнительного пересчёта координат элементов при изменении состояния страницы или прокрутке. Эти подходы полностью сохраняют естественный поток документа, что обеспечивает предсказуемое поведение макета и минимальное значение метрики CLS. В противоположность им, методы *absolute* и *fixed* выводят элементы из нормального потока, что вынуждает браузер выполнять повторные вычисления геометрии при каждом изменении видимой области или динамическом обновлении контента. Экспериментально зафиксировано, что при прокрутке на 1000 пикселей данные методы генерируют до пятидесяти пересчётов макета, что снижает плавность анимаций и увеличивает время отрисовки на мобильных устройствах с ограниченными ресурсами.

На основе полученных данных сформулированы практические рекомендации для повседневной веб-разработки. Метод *static* следует использовать по умолчанию для всего текстового и блочного контента, не требующего специального размещения. Метод *relative* оптимален для незначительных визуальных корректировок и создания родительского контекста для абсолютных потомков. Метод *absolute* целесообразно применять только для элементов, которые должны перекрывать основной контент, таких как модальные окна, всплывающие подсказки или иконки

уведомлений. Метод `fixed` рекомендуется для навигационных панелей и кнопок обратного вызова, которые должны оставаться видимыми независимо от прокрутки, при этом следует контролировать значение `z-index`, чтобы избежать случайного перекрытия важных интерфейсных элементов. Метод `sticky` демонстрирует наилучший компромисс для сценариев с длинными страницами, автоматически фиксируя заголовки секций или боковые меню без использования скриптов.

Для создания адаптивных макетов предпочтительнее использовать современные системы раскладки `flex` и `grid`. `Flexbox` эффективно решает задачи одномерного распределения пространства, обеспечивая простое выравнивание элементов по горизонтали или вертикали, что делает его идеальным выбором для навигационных панелей и карточных интерфейсов. `CSS Grid` предоставляет полный контроль над двумерной структурой страницы, позволяя точно задавать строки, колонки и области перекрытия без применения устаревших приёмов на основе `float`. Оба метода демонстрируют стабильную производительность и нативную поддержку во всех современных браузерах, включая Яндекс Браузер, что гарантирует кроссплатформенную совместимость разрабатываемых решений.

4. Выводы

В ходе исследования проведён комплексный анализ семи методов `CSS`-позиционирования, включающий статические и относительные подходы, абсолютное и фиксированное позиционирование, липкое позиционирование, а также современные системы раскладки `Flexbox` и `CSS Grid`. Экспериментальные данные, полученные с использованием инструментов разработчика Яндекс Браузера, подтвердили, что методы `static` и `relative` обеспечивают наилучшую производительность рендеринга, так как не требуют дополнительных пересчётов макета при взаимодействии пользователя со страницей. Данные подходы рекомендуется использовать по умолчанию для базового контента, что гарантирует предсказуемость поведения элементов и минимальную нагрузку на процессор.

Методы `absolute` и `fixed` демонстрируют более высокие затраты ресурсов из-за необходимости постоянного вычисления координат относительно родительского контейнера или окна браузера. Их применение требует осторожности и обоснования, особенно в сложных интерфейсах с частыми обновлениями `DOM`. В то же время метод `sticky` показал оптимальный баланс между функциональностью и производительностью, автоматически адаптируясь к прокрутке без использования скриптов, что делает его предпочтительным выбором для навигационных панелей и заголовков секций.

Современные системы раскладки `Flexbox` и `CSS Grid` зарекомендовали себя как наиболее эффективные инструменты для построения адаптивных интерфейсов. `Flexbox` идеально подходит для одномерных задач, таких как выравнивание элементов в навигации или карточках товаров, обеспечивая гибкость и простоту реализации. `CSS Grid`, в свою очередь, предоставляет полный контроль над двумерными макетами, позволяя создавать сложные

сеточные структуры для галерей и дашбордов без применения устаревших методов. Оба метода демонстрируют стабильную производительность и широкую поддержку в современных браузерах.

Разработанное интерактивное веб-приложение с реализацией CSS-анимаций позволило наглядно продемонстрировать функциональные особенности каждого метода позиционирования. Практическая значимость работы заключается в формировании чётких рекомендаций для веб-разработчиков, позволяющих выбирать оптимальные инструменты вёрстки. Внедрение предложенных подходов способствует повышению производительности веб-приложений на 15–25%, улучшению метрик Core Web Vitals и обеспечению более комфортного пользовательского опыта за счёт плавности интерфейса и отсутствия визуальных сдвигов.

Перспективы дальнейших исследований связаны с изучением влияния новых возможностей стандарта CSS, таких как Subgrid и Container Queries, на производительность рендеринга. Особое внимание планируется уделить тестированию методов позиционирования в условиях мобильных устройств с ограниченными аппаратными ресурсами, а также анализу их совместимости с будущими версиями браузерных движков. Полученные результаты могут быть использованы при разработке высоконагруженных веб-сервисов, где критически важны скорость отклика и стабильность визуального отображения контента.

Библиографический список

1. Денисова О. Р., Скворцова Т. В., Ягодкин А. С. CSS как современный инструмент создания стильных сайтов // Перспективные аспекты моделирования систем и процессов: материалы Всероссийской научно-практической конференции, Воронеж, 7 октября 2023 г./отв. ред. ВК Зольников, АИ Заревич; М-во науки и высшего образования РФ, ФГБОУ ВО «ВГЛУ». Воронеж, 2023. 563 с. URL: <https://elibrary.ru/item.asp?id=63615566&pff=1>
2. Ментус В. С., Луговской А. Н., Попов Д. С. Встроенный, внутренний и внешний CSS // Перспективные аспекты моделирования систем и процессов: материалы Всероссийской научно-практической конференции, Воронеж, 7 октября 2023 г./отв. ред. ВК Зольников, АИ Заревич; М-во науки и высшего образования РФ, ФГБОУ ВО «ВГЛУ». Воронеж, 2023. 563 с. URL: <https://elibrary.ru/item.asp?id=63615665&pff=1>
3. Осипова Ю. С., Смирнов А. М. Инновационные методы создания анимации на веб-сайтах // Современные научные исследования: актуальные вопросы. 2024. С. 37. URL: <https://naukaip.ru/wp-content/uploads/2024/12/МК-2199.pdf#page=37>
4. Веселова А. А. Эффективное использование анимации на веб-сайтах // World of science 3. 2024. С. 56. URL: <https://naukaip.ru/wp-content/uploads/2024/12/МК-2182.pdf#page=56>
5. Домбровский Я. А., Ткачев Е. В. Создание веб-приложения средствами

HTML, CSS, JavaScript // Дневник науки. 2025. №. 3. URL: https://dnevniknauki.ru/images/publications/2025/3/technics/Dombrovsky_Tkachev.pdf

6. Чернова Л. В. Исследование возможностей CSS для создания анимаций и интерактивных элементов в web–играх // Научная деятельность в условиях цифровизации: теоретический. 2024. С. 75. URL: <https://aeterna-ufa.ru/sbornik/NK-584.pdf#page=75>