Научно-исследовательский журнал «Modern Economy Success»
https://mes-journal.ru
2025, № 6 / 2025, Iss. 6 https://mes-journal.ru/archives/category/publications
Научная статья / Original article
Шифр научной специальности: 5.2.3. Региональная и отраслевая экономика (экономические науки)
УДК 338.24;004.4

[1] Коновалик Е.А.,
[1] EPAM Systems

Сравнительный технико-экономический анализ методов переноса веб-сайтов в экосистему Connected TV средствами Javascript (к вопросу об использовании Lightning JS и альтернатив)

Аннотация: *целью* исследования является проведение сравнительного технико-экономического анализа методов переноса веб-сайтов в экосистему Connected TV с использованием инструментов JavaScript (Lightning JS, нативные SDK и кроссплатформенные фреймворки).

Методы: в качестве методов в представленном исследовании применяются технико-экономический анализ, сравнительный анализ архитектурных решений, структурно-функциональный подход. Проведено сопоставление технических параметров (FPS, RAM, latency) и экономических показателей (затраты на разработку, QA и поддержку, продление жизненного цикла оборудования).

Результаты (Findings): в исследовании представлены сравнительные модели, отражающие различия между нативными SDK, веб-подходами и Lightning JS. Установлено, что использование Lightning JS позволяет повысить производительность интерфейса на 40-50%, сократить время отклика до 90-110 мс, снизить совокупную стоимость владения приложением примерно на 35%.

Выводы: результаты анализа показали, что Lightning JS занимает оптимальное положение между нативными и веб-решениями по соотношению цены и производительности. Применение Lightning JS в CTV-разработке позволяет минимизировать затраты, повысить UX-качество и ускорить масштабирование сервисов, что делает данный фреймворк перспективным инструментом в стратегии цифровой трансформации медиарынка.

Ключевые слова: Connected TV, Smart TV, Lightning JS, JavaScript, кроссплатформенные фреймворки, нативные SDK, технико-экономический анализ, производительность UI, совокупная стоимость владения (TCO)

Для цитирования: Коновалик Е.А. Сравнительный технико-экономический анализ методов переноса веб-сайтов в экосистему Connected TV средствами Javascript (к вопросу об использовании Lightning JS и альтернатив) // Modern Economy Success. 2025. № 6. С. 316 – 326.

Поступила в редакцию: 22 августа 2025 г.; Одобрена после рецензирования: 20 октября 2025 г.; Принята к публикации: 24 ноября 2025 г.

[1] Konovalik E.A.,
[1] EPAM Systems

Comparative techno-economic analysis of methods for porting websites to the Connected TV ecosystem using JavaScript (on the use of Lightning JS and alternatives)

Abstract: the purpose of the study is to conduct a comparative techno-economic analysis of methods for porting websites to the Connected TV ecosystem using JavaScript tools (Lightning JS, native SDKs, and cross-platform frameworks).

Methods: the research employs techno-economic analysis, comparative analysis of architectural solutions, and a structural-functional approach. A comparison was carried out between technical parameters (FPS, RAM, latency) and economic indicators (development, QA and maintenance costs, as well as equipment life-cycle extension).

Findings: the study presents comparative models that highlight the differences between native SDKs, web-based approaches, and Lightning JS. It was established that the use of Lightning JS improves interface performance by 40-50%, reduces response time to 90-110 ms, decreases the total cost of ownership (TCO) of the application by approximately 35%, and extends the lifespan of devices by up to three years.

Conclusions: the results show that Lightning JS occupies an optimal position between native and web-based solutions in terms of cost-to-performance ratio. The use of Lightning JS in CTV development helps minimize costs, enhance UX quality, and accelerate service scalability, making this framework a promising tool within the strategy of digital transformation of the media market.

Keywords: Connected TV, Smart TV, Lightning JS, JavaScript, cross-platform frameworks, native SDKs, techno-economic analysis, UI performance, total cost of ownership (TCO)

For citation: Konovalik E.A.. Comparative techno-economic analysis of methods for porting websites to the Con-nected TV ecosystem using JavaScript (on the use of Lightning JS and alternatives).. Modern Economy Success. 2025. 6. P. 316 – 326.

The article was submitted: August 22, 2025; Approved after reviewing: October 20, 2025; Accepted for publication: November 24, 2025.

Введение

Современная экосистема цифрового телевидения находится на стадии активной технологической трансформации, связанной с конвергенцией медиа, интернета и интерактивных сервисов. В отличие от традиционного телевизионного вещания, в котором пользователь являлся пассивным потребителем контента, в эпоху Smart TV пользователь становится непосредственным и активным участником медиапроцесса, который взаимодействует с контентом через интерфейсы, приложения и персонализированные сервисы. По данным аналитического отчета GrandView Research, мировой рынок Connected TV оценивается в 267,7 млрд. долл. США в 2024 году и, по прогнозам, достигнет 530,9 млрд. долл. США к 2030 году при среднем ежегодном темпе роста 12,8 % (рис. 1). Существующие тенденции и перспективы роста рынка доказывают потенциал CTV-сегмента как одного из основополагающих направлений развития цифрового телевидения.

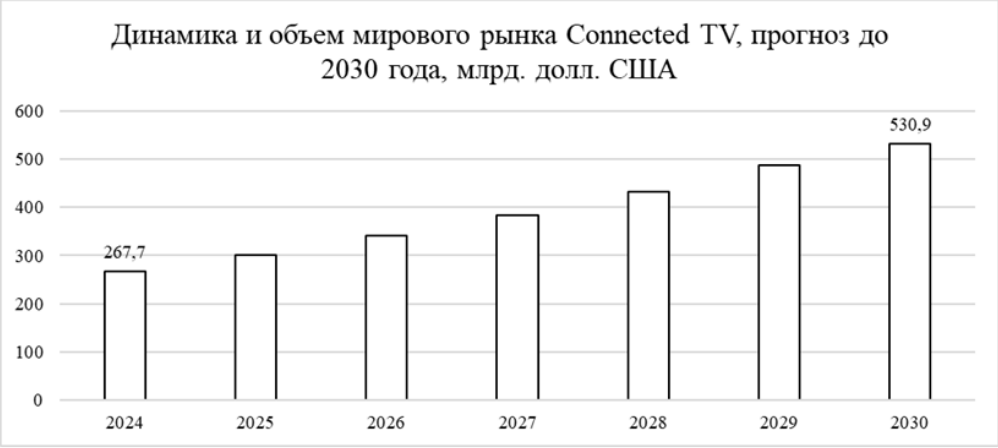

Рис. 1. Динамика и объем мирового рынка Connected TV, прогноз до 2030 года, млрд. долл. США [7].
Fig. 1. Dynamics and volume of the global Connected TV market, forecast until 2030 (in billion USD).

Стоит отметить, что в целом цифровизация телевещания является одной из предпосылок формирования новых моделей дистрибуции и монетизации контента – переход к цифровому формату, как справедливо указывал В.Л. Скобелев, сопровождается созданием мультимедийных платформ, которые обеспечивают диверсификацию каналов взаимодействия с аудиторией. В результате телевидение становится одной из сервисных функций, на базе которых аккумулируются множественные продукты и услуги; закономерно, приложения и интерактивные интерфейсы в цифровой среде телевещания определяют пользовательский опыт и его качество [2]. В целом на глобальном уровне переходные процессы приобретают форму устойчивого технологического тренда, который можно

определить как медиа-интеграцию и переход от телевизора как устройства к телевизору как платформе. И.А. Полуэхтова справедливо рассматривает «умный телевизор» не просто как новый бытовой прибор, а как часть социокультурной среды, влияющей на ритмы повседневности и поведение человека [1]. Экстраполируя тезисы автора, Smart TV становится не только инструментом потребления контента, но и медиатором между цифровыми сервисами и домохозяйством, который формирует новую модель досуга и информационного обмена. Таким образом, переход бизнеса в CTV-среду обусловлен не только техническими факторами, но и изменением самой структуры медиапотребления, что формирует ряд экономических предпосылок его рассмотрения.

С другой стороны, интенсивное развитие рынка сопровождается повышением проблемы фрагментарности архитектурных решений. Производители телевизоров внедряют собственные программные стеки (например, Tizen, webOS, Roku OS, Fire TV, Android TV и др.), что создает барьеры для унификации и существенно увеличивает издержки разработчиков. В результате у производителей контента и провайдеров OTT-сервисов возникает объективная потребность в инструментах, оперируя которыми удастся переносить существующие вебплатформы в CTV-среду без полного переписывания кода и дублирования команд разработки. Данная проблема является одной из центральных для современного медиабизнеса, поскольку именно стоимость и скорость переноса определяют конкурентоспособность сервисов на многоплатформенном рынке, а также, в конечном счете, и их доступность для потребителей.

В этой связи особый интерес приобретают кроссплатформенные фреймворки и инструменты, которые позволяют использовать единый стек технологий, прежде всего JavaScript, для создания интерфейсов, которые будут совместимы с различными устройствами. Опираясь на исследование компании Research2Guidance, отметим, что использование кроссплатформенных фреймворков и инструментов позволяет сократить сроки разработки более чем на 30%, а в некоторых случаях и до 50%, при сохранении функционального и визуального соответствия оригинальным приложениям [8]. По существу, тенденция к унификации кода и интерфейсов становится экономически оправданной стратегией развития бизнеса, одновременно с которой возрастает интерес к специализированным JavaScript-решениям, ориентированным именно на телевизионные интерфейсы.

Одним из наиболее обсуждаемых в отрасли инструментов стал Lightning JS, разработанный компанией Metrological, появление которого связано с практической задачей повышения производительности UI в CTV-приложениях при ограниченных аппаратных ресурсах. В отличие от классического стека HTML/CSS/DOM, Lightning JS использует прямой доступ к GPU через WebGL, что позволяет повысить частоту кадров и обеспечить гладкость, отсутствие прерываний в анимации. По данным отраслевого отчета 3SS, использование Lightning JS позволяет повысить производительность интерфейса до 50% и продлить жизненный цикл устройств на три года за счет программного обновления без замены аппаратной базы [6], что делает его исследование, а также технико-экономическое обоснование и сравнительный анализ с другими решениями особенно актуальными.

Научная и практическая значимость рассматриваемой в исследовании проблематики заключается в оценке преимуществ и ограничений существующих методов переноса веб-проектов в экосистему Connected TV с позиции технических и организационно-экономических факторов; необходимо конкретизировать возможности фреймворка Lightning JS как инструмента снижения совокупной стоимости владения приложением и повышения эффективности процессов его разработки.

Целью исследования является проведение сравнительного технико-экономического анализа методов переноса веб-сайтов в CTV-экосистему средствами JavaScript (на примере Lightning JS по сравнению с нативными SDK и кросс-платформенными фреймворками).

Материалы и методы исследований

Методологическая основа исследования строится на принципах технико-экономического анализа, который предполагал одновременное рассмотрение технологических, архитектурных, организационных и стоимостных аспектов переноса веб-решений в среду Connected TV. С технической точки зрения были выделены различия в архитектуре и основные барьеры, возникающие при миграции приложений; экономические аспекты позволили сравнительным путем сопоставить совокупную стоимости разработки и поддержки с использованием различных инструментальных подходов. Сравнительная характеристика особенностей разработки приложений для Smart TV представлена в табл. 1.

Таблица 1

Сравнительная характеристика особенностей разработки приложений для Smart TV.

Table 1

Comparative characteristics of Smart TV application development features.

№	Критерий сравнения	Особенности	Технические и UX-следствия
1	Управление и пользовательский ввод	Управление пультом ДУ (5-7 кнопок); отсутствие сенсорного и мышиного ввода; необходимость фокус-менеджмента; ввод текста через экранную клавиатуру или голосом	Ограниченные возможности взаимодействия; повышенные требования к навигации; снижение удобства авторизации и поиска
2	Рендеринг и графическая подсистема	Отсутствие DOM и HTML/CSS; использование низкоуровневых API (WebGL, OpenGL ES); наличие декларативных языков описания (SceneGraph, др.)	Высокая производительность, но повышенная сложность разработки; требуется отдельная экспертиза; затруднен перенос веб-приложений
3	Системные API и доступ к ресурсам	Ограниченный доступ к файловой системе; контролируемая интеграция платежных систем; встроенные DRM (Widevine, PlayReady и др.) с сертификацией	Снижение гибкости интеграции сторонних решений; усложнение работы с мультимедиа и безопасностью контента
4	Показатели производительности	Cold start ≤ 2 сек.; потребление памяти ≤ 150–200 МБ; задержка ввода ≤ 150 мс	Несоблюдение метрик приводит к снижению удержания пользователей и риску аварийного завершения процессов (выключение)

Экономический анализ основывается на сопоставлении совокупных затрат жизненного цикла приложения, для оценки применены показатели стоимости разработки, времени вывода продукта на рынок, стоимости QA и поддержки, общей стоимости владения.

Результаты и обсуждения

Современные телевизионные платформы характеризуются высокой степенью технологической дивергенции, поскольку, как было указано нами ранее, каждая экосистема использует собственный набор API, драйверов, библиотек рендеринга и моделей взаимодействия с пользовательским вводом. Отсутствие единого SDK создает ограничения при разработке и тестировании приложений, поскольку одно и то же приложение должно соответствовать нескольким несовместимым средам.

Так, как указывается в исследовании B.S. Ahmed, A. Gargantini, и M. Bures, даже при близости интерфейсных компонентов структура переходов между состояниями и логика событий в Smart TV-приложениях существенно отличается от мобильных и настольных аналогов. Авторы разработали модель-ориентированный фреймворк тестирования на основе разделения подмоделей (Model Separation), использование которого позволяет снижать трудоемкость QA-процедур. Универсальное тестирование в условиях множественности GUI-состояний определяет экспоненциальное повышение сложности разработки и актуализирует необходимость создания специализированных инструментов для Smart TV-среды [5]. Схожие выводы представлены и в более раннем исследовании B.S. Ahmed, M. Bures, в котором авторами была предложена стратегия EvoCreeper – автоматизированная система обратного моделирования интерфейсов Smart TV. Алгоритм EvoCreeper показал, что навигация пультом дистанционного управления требует иной событийной модели: пользователь переходит между состояниями интерфейса не напрямую, а через серию промежуточных состояний, в связи с чем в пересмотре нуждаются принципы DOM-структурирования и оптимизации графических переходов, что в свою очередь делает перенос обычных веб-страниц на платформы Smart TV неэффективным [3]. Иначе говоря, существует выраженный технологический разрыв между веб-средой и CTV-платформами, который подтверждает целесообразность поиска инвариантных решений, позволяющих свести к минимуму затраты на адаптацию к каждой конкретной операционной системе.

Так, в отличие от классического браузера, Smart TV не предназначен для обработки тяжелых DOM-деревьев и CSS-анимаций; при этом даже высокопроизводительные телевизоры уступают мобильным устройствам по объему оперативной памяти и скорости процессора. Согласно исследованию Y. Liu и соавторов, при сравнительном анализе 3445 пар приложений для Android-смартфонов и Android TV выяснилось, что до 50% кода и 43% ресурсов могут быть использованы повторно, однако именно модуль взаимодействия

с пользователем нуждается в полной переработке. ТВ-версии тех же приложений используют меньше разрешений, меньшее количество интерактивных элементов и обладают отличной от мобильной модели системой разрешений и прав доступа [9]. Особое внимание стоит уделять влиянию архитектуры рендеринга на производительность. Основная проблема Smart TV-браузеров связана с ограниченным доступом к GPU и неэффективным использованием кэширования при анимации, что создает необходимый базис для внедрения альтернативных JavaScript-фреймворков, основанных на WebGL-рендеринге и отказе от тяжелого DOM. Lightning JS, широко применяемый в современных CTV-платформах, как раз реализует этот принцип, что и делает возможным перенос сложных вебсцен без потери скорости и стабильности.

Технологические преимущества CTV сопровождаются повышением рисков безопасности, на что указывают в своем исследовании Y. Zhang и соавторы. В их работе представлено эмпирическое исследование уязвимостей Smart TV, связанных с мультиканальными атаками через пульт дистанционного управления. Так, при отсутствии четкой изоляции каналов возможно перехватывание управляющих сигналов и компрометация пользовательских данных [12]. Следовательно, при выборе архитектуры для переноса веб-сайта в CTV-экосистему следует учитывать не только производительность, но и подверженность потенциальным угрозам, которая обеспечивается путем использования асинхронной модели взаимодействия с аппаратными API, характерной для Lightning JS. Схожим образом J. Varmarken и соавторы исследовали скрытые механизмы трекинга и рекламной аналитики в Smart TV-экосистеме, на основании чего установили, что значительная часть CTV-приложений интегрирована с сетями рекламных трекеров и SDK-аналитики, что закономерно предъявляет повышенные требования к безопасности и приватности данных. В контексте разработки предполагается, что современные фреймворки должны не только обеспечивать производительность, но и изолировать потоки данных, что нужно для предотвращения утечек данных [11].

Итак, результаты проведенного анализа (табл. 2) позволяют указать, что развитие Connected TV-приложений находится на пересечении двух противоположных тенденций, поскольку операторы и разработчики стремятся как к снижению затрат и ускорению выпуска новых сервисов, так и вынуждены адаптировать интерфейсы под аппаратно-ограниченные устройства и несогласованные между собой стандарты нескольких платформ. По этой же причине основополагающим принципом выбора методологии становится соотношение технической производительности и совокупных экономических издержек.

Таблица 2

Сравнительная характеристика подходов к разработке приложений для Smart TV.

Table 2

Comparative characteristics of Smart TV application development approaches.

№	Подход	Основные платформы и технологии	Преимущества	Недостатки
1	Нативные SDK	Tizen (Samsung), webOS (LG), Roku SceneGraph (BrightScript)	Полный доступ к системным API и DRM, высокое качество и производительность, соответствие требованиям сертификации производителей	фрагментация экосистемы, невозможность прямого переноса кода, высокая стоимость и сложность поддержки, необходимость специфической экспертизы
2	Кроссплатформенные решения	Lightning JS, React Native TV, Cordova / Capacitor for TV	Снижение затрат за счет повторного использования кода, быстрый вход для веб- и мобильных разработчиков, возможность задействования единого CI/CD	Ограниченный доступ к системным API, снижение производительности на слабых устройствах, необходимость ручной адаптации под фокус и DRM
3	Универсальные браузерные приложения (PWA)	Android TV, Fire TV (через браузер / WebView)	минимальные затраты и простота обновления, универсальность кода, возможность быстрого запуска существующего веб-приложения	отсутствие поддержки на ряде платформ (Tizen, webOS, Roku), ограниченный доступ к API, низкая производительность при мультимедиа-нагрузке

Сравнение архитектур показало, что нативные SDK (к которым относятся Tizen Studio, WebOS TV SDK, Android TV SDK) обеспечивают наивысшую оптимизацию под конкретные устройства и доступ к системным API (GPU, DRM и датчики ввода). Однако производительность достигается ценой высокой фрагментации, т.к. каждая операционная система зависит от своей базы коды, системы тестирования и сертификации. В среднем поддержка одного релиза на нескольких платформах увеличивает длительность QA-цикла на 45-60% и приводит к удвоению затрат на обновления.

В свою очередь, кроссплатформенные фреймворки (React Native TV, Flutter TV, Xamarin) отличает более рациональный баланс между скоростью и издержками. Использование единой кодовой базы и универсальных библиотек позволяет сократить время разработки до 30-50 % по сравнению с нативными SDK, а степень переиспользования кода достигает 90%. Тем не менее инструменты ограничены при обращении к низкоуровневым функциям телевизора (аппаратное декодирование, HDCP, CEC). Визуально интерфейс уступает нативным решениям по плавности и FPS, что особенно выражено на устройствах с малым объемом оперативной памяти [Fullestop].

Веб-подход (HTML5 или PWA) традиционно рассматривается в качестве наиболее доступного способа миграции, который особенно распространен среди владельцев существующих сайтов и веб-порталов. Веб-подход позволяет быстро масштабировать решение и достаточно просто интегрируется с CDN-сетями; в тоже время, тяжеловесный DOM, каскадные CSS и многоуровневая обработка событий увеличивают нагрузку на слабые CPU телевизоров. Даже при минимизации ресурсов и асинхронной загрузке данные показывают падение FPS до 30 кадров в секунду при сложной анимации, а также увеличение времени отклика интерфейса. Веб-приложения часто используют до 70% доступной памяти браузера, в связи с чем возможны «зависания» и может потребоваться перезапуск устройства (что негативно сказывается на пользовательском опыте).

На фоне представленных решений Lightning JS выступает как компромиссное, т.е. альтернативное, но перспективное в технологическом плане решение. Концепция Lightning JS заключается в отказе от DOM-рендеринга и прямом обращении к GPU средствами WebGL. Интерфейс создаётся в виде сценографа (scene graph), в котором каждый элемент визуализируется как отдельный слой, что снижает задержку рендера и экономит RAM. Результаты пилотных внедрений показывают прирост производительности пользовательского интерфейса на 40-50% при тех же аппаратных параметрах, что особенно эффективно при работе на базе старых STB-устройств. Дополнительное преимущество заключается в возможности использовать одну и ту же кодовую базу для Smart TV, OTT-платформ и наследуемых Linux-приставок, что позволяет синхронизировать UX между поколениями устройств. В качестве примера, сопоставим код инициализации приложения в ligning.js и Roku SceneGraph / BrightScript (рис. 2, 3).

```
// src/App.js
import { Lightning, Utils } from '@lightningjs/sdk'

export default class App extends Lightning.Component {
  static _template() {
    return {
      // корневой контейнер
      Wrapper: {
        w: 1920, h: 1080, rect: true, color: 0xff101014,
        // узел-текст (рендерится как текстура в WebGL)
        Title: { x: 80, y: 60, text: { text: 'Smart TV Demo', fontFace: 'Regular',
fontSize: 64 } },
        // карточки как дочерние ноды
        Cards: { x: 80, y: 160, flex: { direction: 'row', wrap: true, padding: 0,
spacing: 24 } }
      }
    }
  }

  _init() {
    // динамическая генерация карточек (текстуры + прямоугольники; без DOM)
    const items = Array.from({ length: 12 }, (_, i) => ({
      type: Card,
      w: 380, h: 220,
      label: `Item ${i + 1}`
    }))
    this.tag('Cards').children = items
    this._index = 0
    this._patchFocus()
  }

  // фокус/навигация: управление через пульт
  _handleLeft()  { this._move(-1) }
  _handleRight() { this._move(+1) }
  _handleUp()    { this._move(-3) }
  _handleDown()  { this._move(+3) }
  _handleEnter() { this.tag('Cards').children[this._index].enter() }

  _move(delta) {
    const cards = this.tag('Cards').children
    this._index = Math.max(0, Math.min(cards.length - 1, this._index + delta))
    this._patchFocus()
  }

  _patchFocus() {
    this.tag('Cards').children.forEach((c, i) => c.patch({ focused: i === this._index
}))
  }
}

class Card extends Lightning.Component {
  static _template() {
    return {
      rtt: true, // Render-To-Texture для эффекта отблеска/тени
      w: 380, h: 220, rect: true, color: 0xff24242a, shader: { type:
Lightning.shaders.RoundedRectangle, radius: 18 },
      Label: { x: 24, y: 24, text: { text: '', fontSize: 32 } },
      Focus: { alpha: 0, rect: true, color: 0x33ffffff, w: w => w, h: h => h }
    }
  }

  set label(v) { this.tag('Label').text.text = v }
  set focused(v) { this.tag('Focus').setSmooth('alpha', v ? 1 : 0) }

  enter() {
    // действие по OK/Enter
    this.signal('select', { label: this.tag('Label').text.text })
  }
}
```

Рис. 2. Архитектура и принципы построения интерфейса Smart TV-приложения на Lightning JS.
Fig. 2. Architecture and design principles of a Smart TV application interface built with Lightning JS.

```
sub init()
  m.top.observeField("visible", "onVisibleChange")
  ' Загружаем всё сразу (блокирующе)
  list = getJson("https://api.example.com/feed") ' синхронно, без таймаута
  items = []
  for each it in list  ' создаём 500+ нод сразу
    item = CreateObject("roSGNode", "ContentNode")
    item.title = it.title
    item.hdPosterUrl = it.image ' 2-5MB картинки, без ресайза
    items.push(item)
  end for
  m.grid = m.top.findNode("RowList")
  m.grid.content = items ' один гигантский массив → OOM на дешёвых приставках
  m.grid.setFocus(true)  ' без явного focus mgmt между элементами
end sub

function getJson(url as string) as object
  xfer = CreateObject("roUrlTransfer")
  xfer.setUrl(url)
  return ParseJson(xfer.GetToString()) ' блокирует UI; нет retry/backoff
end function
```

Рис. 3. Пример синхронной архитектуры интерфейса Smart TV-приложения на BrightScript (нативный SDK Roku).
Fig. 3. Example of a synchronous interface architecture of a Smart TV application developed in BrightScript (native Roku SDK).

Как можно заметить, рендер и сеть в случае Roku SceneGraph / BrightScript находятся в одном потоке, без observeField, без пагинации/рецикла.

Сопоставляя значения, технико-экономическая характеристика подходов к портированию веб-проектов в CTV-среду приобретает следующий вид (табл. 1):

Таблица 3

Сравнительная технико-экономическая характеристика подходов к портированию веб-проектов в CTV-среду.

Table 3

Comparative techno-economic characteristics of web project porting approaches to the CTV environment.

Показатель	Нативные SDK	Кросс-платформа (React Native TV / Flutter TV)	Веб (PWA / HTML5)	Lightning JS
Переиспользование кода, %	до 10	70-90	более 95	80
Средняя скорость разработки, % от базового значения*	100	70	50	60
Производительность UI, FPS	60	45-55	25-35	55-60
Время отклика интерфейса, ms	менее 80	100-130	более 250	90-110
Поддержка низкоуровневых API	Полная	Ограниченная	Минимальная	Достаточная
Средняя стоимость разработки, %	100	65-70	40–50	60–65
Затраты на QA и поддержку, %	100	60	50	55
Энергопотребление и ресурсы RAM, %	100	85	120	80
Продление жизненного цикла устройств	Отсутствует	Дополнительный 1 год	Отсутствует	До 3 дополнительных лет
Ориентировочный эффект по TCO	Базовый	-30%	-40% (при пониженном качестве)	-35% (при сохранении качества)

** показатели приведены относительно нативного SDK = 100%. Составлено на основании значений [8, 10 и др.]*
** The figures are given relative to the native SDK = 100%. Compiled based on the values of [8, 10, etc.].*

Опираясь на представленные в табл. 1 данные, стоит указать, что Lightning JS занимает устойчивое положение в сегменте «оптимальное соотношение цены и производительности». При небольшом снижении TCO (на 35%) он обеспечивает качество UX, сравнимое с нативным уровнем, и позволяет уменьшить затраты на поддержку. Экономический эффект усиливается за счет продления срока эксплуатации старых приставок на дополнительные три года, что снижает амортизационные издержки операторов.

Так, опираясь на исследования B.S. Ahmed, M. Bures [3, 4, 5], можно выделить ряд закономерностей, характерных для CTV-среды (рис. 4):

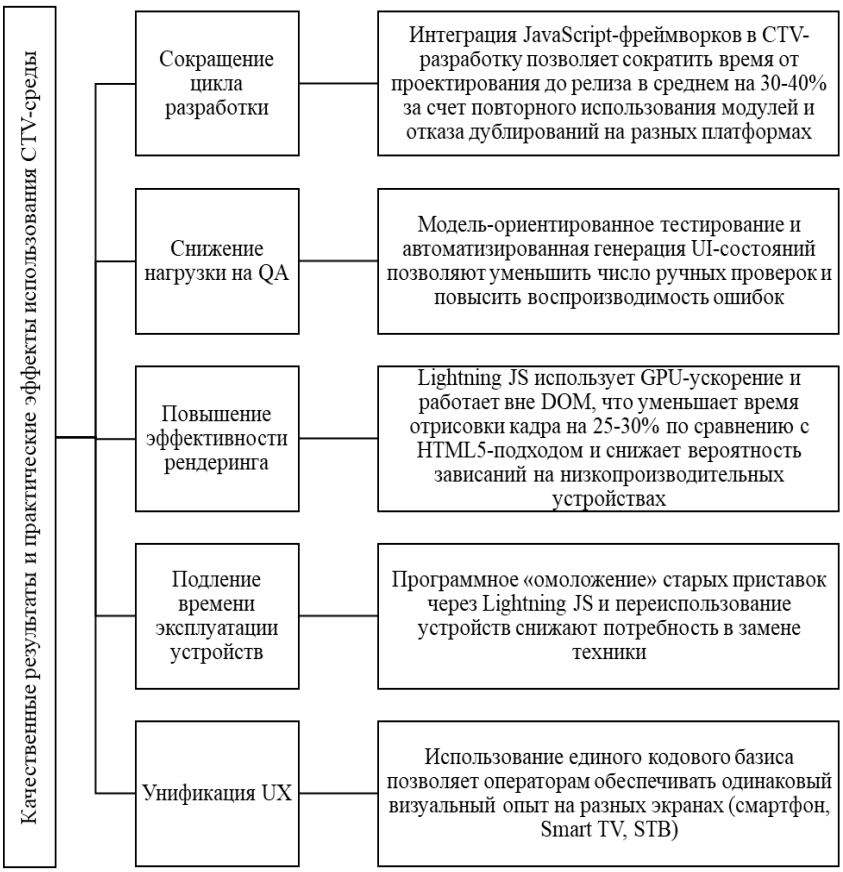

Рис. 4. Качественные результаты и практические эффекты использования CTV-среды.
Fig. 4. Qualitative outcomes and practical effects of using the CTV environment.

Итак, сопоставление показателей показывает, что выделяется не просто разница в стоимости или времени разработки, а именно формируется новая модель управления разработкой CTV-продуктов. Если раньше решения для Smart TV создавались в закрытых средах и требовали аппаратной интеграции, то сегодня успешность проекта определяется способностью компании оперативно адаптировать существующий веб-контент. С этой точки зрения Lightning JS становится элементом стратегии «software first», при которой инновации в программном слое позволяют продлить жизнь устройству и повысить окупаемость активов.

Экономика портирования веб-сайтов сводится к ряду проявлений:

1) устраняются затраты на улучшение архитектуры (синхронные вызовы (Roku GetToString(), XHR в Tizen); на этапе портирования нужно переписать сетевой слой на асинхронный, что требует дополнительной разработки (от 20 до 80 человеко-часов), а также введения retry/backoff-стратегий); отсутствие визуализации определяет, что рефакторинг с рециклом/ленивой загрузкой может занять до 25-30% времени всей миграции; если навигация строится через O(n) перебор DOM или хаки через tabIndex, необходимо будет реализовывать собственную матрицу переходов, что в среднем увеличит время разработки и отладки на 2-3 недели;

2) затраты на QA и тестирование в среде Smart TV крайне фрагментированы (для сертификации на Tizen, webOS, Roku OS, Fire TV, Android TV потребуется апробация на 5-7

реальных устройствах; один и тот же код может работать стабильно на LG OLED 2023 и вызывать OOM на Samsung 2018; по оценкам операторов, тестирование одного релиза на парке из 10 устройств может занимать до 120-160 человеко-часов);

3) поддержка и TCO – ошибки требуют повторной сертификации, что прямо сказывает на стоимости поддержки; если не внедрить деградацию качества (ресайз постеров, адаптивное видео), растут расходы на CDN и увеличивается количество обращений в поддержку, а также поддержка legacy-устройств и изменение под новые версии SDK приводит к увеличению в диапазон 15-20% от годового бюджета проекта.

Экономический эффект усиливается благодаря сокращению инфраструктурных расходов и унификации команд разработчиков. Переход к единой архитектуре JavaScript уменьшает потребность в специалистах по конкретным платформам и снижает барьер входа для новых участников рынка. С точки зрения операторов и провайдеров появляется возможность осуществлять более быстрое масштабирование сервисов и увеличивать долю повторных подписок. С учетом глобальной тенденции роста CTV-рынка, оптимизация позволяет получать прирост дохода при сокращении капитальных вложений.

Технологическая эффективность Lightning JS, в свою очередь, проявляется в его экосистемной гибкости. Фреймворк применяется в различных моделях устройств, поддерживает рендеринг через WebGL и Canvas, что делает возможным портирование на платформы с минимальными ресурсами. Таким образом, развитие CTV-интерфейсов с использованием JavaScript-средств становится не только экономически оправданным, но и технологически целесообразным.

Выводы

Таким образом, проведенный сравнительный анализ нативных SDK, кросс-платформенных фреймворков, веб-подходов и решений на основе Lightning JS позволил установить наличие прямой взаимосвязи между архитектурной моделью и экономическими показателями жизненного цикла приложения. Нативные SDK-инструменты утрачивают эффективность в условиях фрагментированного рынка, т.к. их использование приводит к повышению трудоемкости и удорожанию QA-процессов, поскольку каждая платформа требует разработки собственного кода и инициации специфического цикла тестирования. Кросс-платформенные фреймворки отличает снижение издержек при умеренной потере производительности; веб-подход (PWA или HTML5) остается перспективным с точки зрения простоты миграции и экономии ресурсов, однако не обеспечивает требуемого качества пользовательского опыта.

Наиболее сбалансированным решением по итогам сравнительного технико-экономического анализа оказался фреймворк Lightning JS, применение которого позволяет повысить производительность интерфейса на 40-50%, сократить время отклика до 90-110 мс и снизить совокупную стоимости владения приложением примерно на 35%. Помимо экономии времени и бюджета разработки, Lightning JS обладает возможностью продления жизненного цикла оборудования на дополнительные три года без физической замены устройств.

Список источников

1. Полуэхтова И.А. «Умный телевизор» в контексте повседневности // Знание. Понимание. Умение. 2013. № 3. С. 169 – 174.

2. Скобелев В.Л. Стратегическое развитие телевещания в России // Петербургский экономический журнал. 2016. № 2. С. 44 – 52.

3. Ahmed B.S., Bures M. EvoCreeper: Automated Black-Box Model Generation for Smart TV Applications // IEEE Transactions on Consumer Electronics. 2019. DOI: https://doi.org/10.48550/arXiv.1904.02956

4. Ahmed B. S., Bures M. Testing of Smart TV Applications: Key Ingredients, Challenges and Proposed Solutions // Advances in Intelligent Systems and Computing. 2019. Vol. 1. P. 241 – 256. DOI: 10.1007/978-3-030-02686-8_20.

5. Ahmed B.S., Gargantini A., Bures M. An Automated Testing Framework For Smart TV Apps Based on Model Separation // ICST2020 and INTUITESTBEDS2020 Workshop. DOI: https://doi.org/10.48550/arXiv.2002.00404

6. Breathing new life into old set-tops is a win-win-win. URL: https://www.3ss.tv/blog/legacy-set-top-ux-upgrade-lightningjs

7. Connected TV Market (2025-2030). URL: https://www.grandviewresearch.com/industry-analysis/connected-tv-market-report

8. Cross-platform tools can save more than 30% of app development time. URL: https://research2guidance.com/cross-platform-tools-can-save-more-than-30-of-app-development-time/

9. Liu Y., Chen X., Liu Y., Kong P., Bissyande T. F., Klein J., Sun X., Chen C., Grundy J. A Comparative Study of Smartphone and Smart TV Apps. DOI: https://doi.org/10.48550/arXiv.2211.01752

10. Native vs Cross-Platform Apps Development Costs in 2025. URL: https://www.fullestop.com/blog/native-vs-cross-platform-apps-development-costs

11. Varmarken J., Le H., Shuba A., Shafiq Z., Markopoulou A. The TV is Smart and Full of Trackers: Towards Understanding the Smart TV Advertising and Tracking Ecosystem. DOI: https://doi.org/10.48550/arXiv.1911.03447

12. Zhang Y., Ma S., Chen T., Li J., Deng R. H., Bertino E. EvilScreen Attack: Smart TV Hijacking via Multi-channel Remote Control Mimicry // IEEE Transactions on Dependable and Secure Computing. 2023. DOI: https://doi.org/10.1109/TDSC.2023.3286182

References

1. Poluekhtova I.A. “Smart TV” in the Context of Everyday Life. Knowledge. Understanding. Skill. 2013. No. 3. P. 169 – 174.

2. Skobelev V.L. Strategic Development of Television Broadcasting in Russia. Petersburg Economic Journal. 2016. No. 2. P. 44 – 52.

3. Ahmed B.S., Bures M. EvoCreeper: Automated Black-Box Model Generation for Smart TV Applications. IEEE Transactions on Consumer Electronics. 2019. DOI: https://doi.org/10.48550/arXiv.1904.02956

4. Ahmed B.S., Bures M. Testing of Smart TV Applications: Key Ingredients, Challenges, and Proposed Solutions. Advances in Intelligent Systems and Computing. 2019. Vol. 1. P. 241 – 256. DOI: 10.1007/978-3-030-02686-8_20.

5. Ahmed B.S., Gargantini A., Bures M. An Automated Testing Framework For Smart TV Apps Based on Model Separation. ICST2020 and INTUITESTBEDS2020 Workshop. DOI: https://doi.org/10.48550/arXiv.2002.00404

6. Breathing new life into old set-tops is a win-win-win. URL: https://www.3ss.tv/blog/legacy-set-top-ux-upgrade-lightningjs

7. Connected TV Market (2025-2030). URL: https://www.grandviewresearch.com/industry-analysis/connected-tv-market-report

8. Cross-platform tools can save more than 30% of app development time. URL: https://research2guidance.com/cross-platform-tools-can-save-more-than-30-of-app-development-time/

9. Liu Y., Chen X., Liu Y., Kong P., Bissyande T. F., Klein J., Sun X., Chen C., Grundy J. A Comparative Study of Smartphone and Smart TV Apps. DOI: https://doi.org/10.48550/arXiv.2211.01752

10. Native vs Cross-Platform Apps Development Costs in 2025. URL: https://www.fullestop.com/blog/native-vs-cross-platform-apps-development-costs

11. Varmarken J., Le H., Shuba A., Shafiq Z., Markopoulou A. The TV is Smart and Full of Trackers: Towards Understanding the Smart TV Advertising and Tracking Ecosystem. DOI: https://doi.org/10.48550/arXiv.1911.03447

12. Zhang Y., Ma S., Chen T., Li J., Deng R.H., Bertino E. EvilScreen Attack: Smart TV Hijacking via Multi-channel Remote Control Mimicry. IEEE Transactions on Dependable and Secure Computing. 2023. DOI: https://doi.org/10.1109/TDSC.2023.3286182

Информация об авторе

Коновалик Е.А., архитектор Программных Решений, EPAM Systems, ORCID ID: https://orcid.org/0009-0004-2354-4614, 2055 Gateway Pl #510, San Jose, CA 95110, yauheni.kanavalik@gmail.com