УДК 004.942:004.422.83
DOI: http://dx.doi.org/10.21686/1818-4243-2024-3-56-64

Д.С. Филиппов
Российский университет транспорта (МИИТ), Москва, Россия

Методы адаптивной трансформации контента в формате гипертекста на корпоративном ТВ и информационных киосках ВУЗа

Цель исследования: *рассмотреть и реализовать на практике систему вывода контента на экраны разного разрешения и ориентаций применительно к образовательному учреждению, используя как чистый Smart TV (встроенный браузер современного ТВ), так и подключённый к экранам Digital Signage плеер. В проектировании системы основной упор (и как следствие ключевая задача) делается на отсутствие дублирования контента для разных типов экранов вывода (включая интерактивные сенсорные киоски), что особенно актуально, т.к. реализованная система будет построена на свободных технологиях с открытым исходным кодом и интегрирована в текущий web-сайт ВУЗа, чего не сможет предложить ни одна другая проприетарная программная система Digital Signage.*

Материалы и методы: *в исследовании используются разные методы трансформации и вывода контента, но преимущественно задействованы методы адаптации, основанные на каскадных таблицах стилей (CSS). Изучен опыт зарубежных программистов, работающих с адаптивной вёрсткой для web и ведущих свои блоги (в разделе «Литература»). Рассмотрены и предложены компромиссные методы вывода графических изображений и видео материалов на основе полученного опыта во время апробации прототипа системы.*

Результаты: *разработка рабочей системы вывода контента на разноформатные экраны, требующая последующего минимального сопровождения и вмешательства администратора, отвечающего за наполнение контентом. Проведён анализ некоторых потенциальных решений и их совместимости с учётом поддержки браузерами стандартов CSS3 и HTML5, отобраны рабочие решения, которые могут быть использованы в проектировании индивидуальных шаблонов вывода контента в формате гипертекста и CSS конструкции (правил), работающих в указанных выше целях исследования. В статье сравниваются подходы к проектированию адаптивных шаблонов вывода web контента в формате HTML (гипертекста). Рассмотрены основные принципы, при которых контент, предназначенный в первую очередь для сайта ВУЗа, может без стороннего администратора (а также, в некоторых случаях, с минимальным вмешательством) отображаться на корпоративных ТВ и сенсорных интерактивных киосках ВУЗов благодаря применению каскадных таблиц стилей, относительных единиц измерения (vh, vw, vmin, vmax) и тегов, введённых в HTML5. В статье также рассмотрено как можно нивелировать нюансы, связанные с разным разрешением графических изображений и ориентацией экранов.*

Заключение: *в заключительных абзацах статьи делается вывод о нюансах функционирования разработанной системы, эффективность которой напрямую зависит от выбранных и рассмотренных в статье методах реализации и используемых технологиях. Внедрение системы позволит повысить информированность студентов образовательного учреждения о событиях (анонсированных или проведённых), а администратору системы позволит повысить эффективность работы, публикуя и администрируя контент только в «одном окне» (например, на сайте ВУЗа), а разработанный шаблон вывода и/или CSS правила отобразят его на внутренние экраны, сохранив удобочитаемость и восприятие информации. Выработанные и апробированные методы могут быть использованы программистами, работающими в образовательных учреждениях.*

Ключевые слова: *каскадные таблицы стилей, CSS, CSS3, HTML, HTML5, гипертекст, Digital Signage, адаптивная вёрстка, адаптивные изображения.*

Dmitry S. Filippov
Russian University of Transport (MIIT), Moscow, Russia

Methods for Adaptive Transformation of Content in Hypertext Format on Corporate TV and Information Kiosks of Educational Institutions

Purpose of the study: *to consider and implement in practice a system for displaying content on screens of different resolutions and orientations in relation to an educational institution, using both a pure Smart TV (built-in browser of modern TV) and a player connected to Digital Signage screens. In system engineering, the main emphasis (and, as a result, the key task) is on the absence of duplication of content for different types of output screens (including interactive touchscreen kiosks), which is especially important because the implemented system will be built on free, open-source technologies and integrated into the current university website, which no other proprietary Digital Signage software system can offer.*

Materials and methods: *the study uses different methods of transformation and output of content, but adaptation methods based on cascading style sheets (CSS) are mainly used. The experience of foreign programmers working with adaptive layouts for the web and operating their own blogs was studied (in the "Reference" section). Compromise methods for displaying graphic images and video materials are considered and proposed based on the experience gained during testing of the prototype system.*

Results: *development of a system for displaying content on multi-format screens, requiring minimal subsequent support and intervention from the administrator responsible for filling it with content. An*

analysis of some potential solutions and their compatibility was carried out, taking into account browser support for CSS3 and HTML5 standards, and working solutions were selected that can be used in the design of individual templates for displaying content in hypertext format and CSS designs (rules) that work for the purpose of the study. The article compares approaches to designing adaptive templates for displaying web content in HTML (hypertext) format. The basic principles are considered under which content intended primarily for a university website can, without a third-party administrator (and also, in some cases, with minimal intervention) be displayed on corporate TV and touchscreen kiosks of universities thanks to the use of cascading style sheets (CSS) and relative units of measurement (vh, vw, vmin, vmax) and tags introduced in HTML5. The article also discusses how to level out the nuances associated with different resolutions of graphic images and screen orientation.

***Conclusion:** the final paragraphs of the article conclude on the nuances of the functioning of the developed system, the effectiveness of which directly depends on methods of implementation and technologies used, chosen and discussed in the article. The implementation of the system will increase the awareness of students of an educational institution about events (announced or conducted), and will allow the system administrator to increase work efficiency by publishing and administering content only in "one window" (for example, on the university website), and the developed output template and/or CSS rules will display it on the internal screens, maintaining readability and perception of information. The developed and tested methods can be used by programmers, working in educational institutions.*

***Keywords:** cascading style sheets, CSS, CSS3, HTML, HTML5, hypertext, Digital Signage, adaptive layout, adaptive images.*

Применительно к системам, построенным на web-технологиях, каскадные таблицы стилей (CSS) являются базовым методом, на котором функционируют адаптивные шаблоны для вывода контента в системах Digital Signage, применяемых на корпоративных ТВ экранах. Спецификация CSS определяет стилевое оформление, т.е. то, как будет выглядеть разнородный контент в web формате (в формате гипертекста) на экранах.

Прежде всего, адаптивный дизайн в контексте цифрового ТВ необходим для вывода визуального контента из единой базы данных (БД) на экраны разного разрешения, соотношения сторон и ориентации. Адаптивный дизайн позволяет существенно экономить время и не создавать новый шаблон (дизайн) для каждого разрешения и ориентации экранов, а менять размеры и расположение отдельных элементов или скрывать их вовсе при необходимости. Смысл не только в масштабировании, но и в переориентации ключевых блоков шаблона вывода [Рисунок 1]. Ключевое требование – избавиться от необходимости в прокрутке (скроллинга) и дублирования контента (вручную администратором или автоматически) для разных шаблонов вывода. Да, можно сделать скрипт медленной прокрутки экрана, например, через Javascript методом window.scrollBy(); или использовать сенсорный киоск, но приём прокрутки более характерен для индивидуальных пользовательских устройств, чем для публичных цифровых экранов. Что же касается дублирования контента, чтобы замостить его под тот или иной экран, то подобная идея противоречит принципу DRY (Do not repeat yourself – не повторяй себя), которая заключается в том, что информация любого вида не должна дублироваться. Это же касается и цифрового ТВ.

Отдельно о том, почему используется гипертекст в качестве контента, отображаемого на цифровых экранах, будет раскрыто ниже в тексте. Современные ВУЗы имеют свои интернет-сайты и профили в социальных сетях, а иногда и по несколько для каждого вида образования, начиная со среднего профессионального, заканчивая дополнительным профессиональным образованием. Объём контента на таких сайтах прямо пропорционален количеству обучающихся. В коридорах таких образовательных учреждений часто установлены корпоративные экраны, и автоматический вывод разнородного контента в подавляющем большинстве подразумевает вывод контента, который уже размещён на сайтах или в социальных сетях. Именно поэтому важно грамотно использовать уже большое

Рис. 1. Единая БД с контентом, но разные шаблоны вывода для разных экранов (реализация в Юридическом институте РУТ; кодовое имя проекта: «Информационный TV хаб Юридического института»)

Fig. 1. A single database with content, but different output templates for different screens (implementation at the RUT Law Institute; project code name: "Information TV Hub of the Law Institute").

количество готового, целевого и обновляемого web-контента с сайта на экранах, не дублируя его, не привлекая новых специалистов и не задействуя для этого стороннее специализированное программное обеспечение для развёртывания сетей Digital Signage. В нашем примере браузер, запущенный в полноэкранном режиме, является полноценным клиентом сети Digital Signage.

Используя каскадные таблицы стилей, контент можно «заставить» подстраиваться под разное разрешение экрана [1], так, чтобы его было удобно просматривать или взаимодействовать с ним. Смысл в изменении единиц измерения с абсолютных (пикселей) на относительные (например, проценты), скрытия малозначимых блоков и части контента, а также наборе CSS Media Queries правил, например, таких как “@media screen and (720px <= width <= 1080px) {...}” с ограничениями на минимальную и максимальную ширину экрана, относительно установленных на корпоративных экранах. Разрабатывая набор CSS правил, мы точно знаем все технические характеристики ТВ экранов, установленных в ВУЗе, что исключает необходимость написания правил для других устройств. Так можно скрыть блоки свойством “display:none;”, которые гарантированно не поместятся с приемлемым масштабом на экранах с низким разрешением или иной ориентацией [рис. 2].

Если разрешения экранов корпоративных ТВ и, например, интерактивного киоска одинаковые (1920x1080 и 1080x1920), то для последнего лучшим решением станет загрузка отдельного CSS файла “@import url(“kiosk.css”) screen and (orientation: portrait) {...}” с набором правил под отдельный экран, что позволит оставить не тронутой основную каскадную таблицу стилей, а править только один файл. В целом такая практика позволяет легче следить за наследованием и каскадностью CSS правил и поддерживать чистоту кода.

Что касается разрешения экранов с одинаковым соотношением сторон (но разным разрешением экранов, например, 3840x2160, 1920x1080 и 1280x720), то допускается HTML шаблон вывода и CSS правила сделать едиными для всей линейки разрешения экранов с тем лишь исключением, что в зависимости от ширины экрана, использовать свойство zoom, например “zoom:150%;” для увеличения в полтора раза от исходных параметров шаблона. В этом случае масштабирование шрифтов и векторных элементов, используемых в шаблоне, пройдёт гладко, однако стоит учитывать, что растровые графические элементы при увеличении их масштаба хотя бы на 1% от оригинала потеряют в качестве. И чем выше увеличение, тем заметнее это будет видно на больших экранах. Однако никто не мешает сделать шаблон вывода и адаптировать графические элементы и фотографии под экран с максимальным разрешением, уменьшая затем число в свойстве zoom. При таком подходе размытие растровых элементов удастся избежать. Применяя один шаблон вывода на разные устройства и масштабирование через CSS под разные ТВ и киоски, следует избегать любых «рекомендаций» браузеру по масштабированию в самом шаблоне вывода, таких как мета-тэг viewport со значением initial-scale=1, чтобы исключить ситуации, когда приоритет по масштабу отдаётся ему, а не указанным в нём CSS правилам. Также в шаблонах вывода крайне не рекомендуется применять inline CSS стили (правила, прописанные через атрибут style=”...” самого тэга), т.к. их возможно переопределить только используя правило (модификатор) «!important» в CSS файлах и больше никак. Только в этом случае они будут иметь наивысший приоритет по сравнению с другими правилами [4]. В контексте этой статьи это важно, т.к. над адаптацией контента для корпоративного ТВ и интерактивных киосков могут работать разные команды, а подход с inline стилями очень сильно усложнит сопровождение и поддержку.

Рассмотрим имеющиеся варианты вывода контента [Рисунок 3]. За ключевой источник контента принимаем сайт ВУЗа. Благодаря тому, что источник контента один, а разными являются только правила адаптации контента (на базе CSS), изменив информацию на сайте, у нас нет необходимости как-то взаимодействовать с контентом в других системах (ТВ и киоски). Предположим, на сайте ВУЗа имеется три новостных материала и их необходимо вывести на ТВ экраны, установленные в коридорах образовательного учреждения.

В первом случае все три материала помещаются в шаблон вывода, который масштабируется простым CSS свойством “zoom”. Если шаблон вывода единый и для горизонтальной ориентации экрана и для вертикальной, то часть экрана

Расположение блоков на классическом ТВ экране с горизонтальной ориентацией

Вариант адаптации блоков на ТВ экране (интерактивном киоске) с вертикальной ориентацией

Рис. 2. Вывод контента при адаптивном дизайне шаблона.
Fig. 2. Content output with adaptive template design.

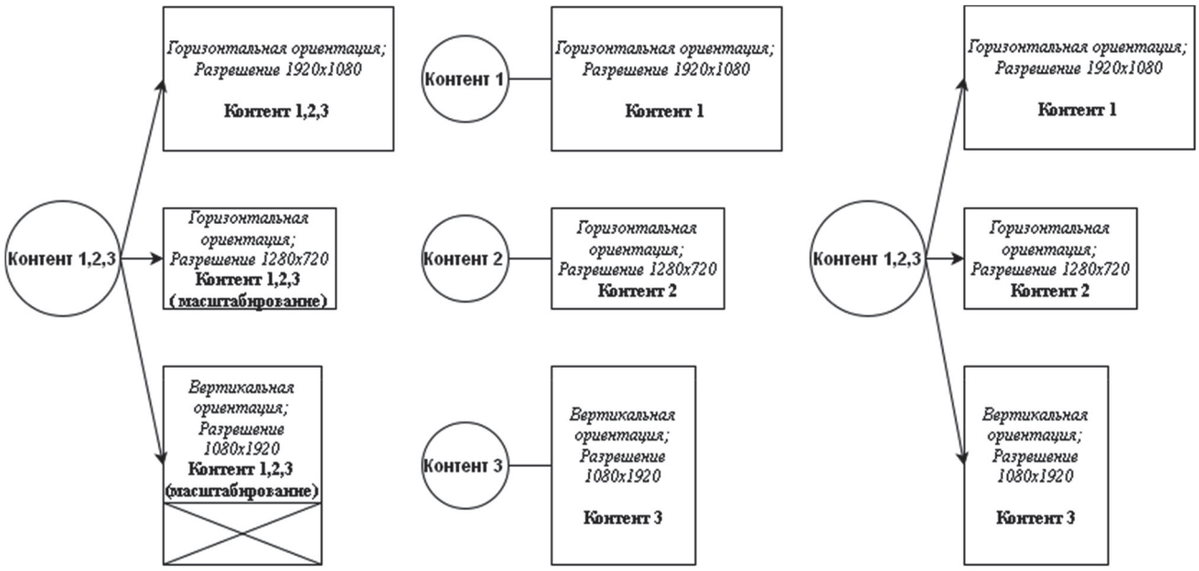

Рисунок 3. Подходы к выводу контента в формате гипертекста на ТВ
Figure 3. Approaches to displaying content in hypertext format on TV

может оказаться незаполненной. На экранах с низким разрешением контент может оказаться нечитаемым, т.к. может получиться слишком мелким. Дополнительно может быть задействован Javascript для ротации контента на всех экранах.

Во втором случае для каждого типа экрана с одинаковыми характеристиками выводится свой контент. Какая новость и для какого экрана – решает администратор корпоративного ТВ или администратор сайта. Недостаток такого подхода в том, что экран не может красиво и гармонично отображать контент другого типа, не предназначенного для этого экрана. Для каждого из них готовится свой контент или своя версия контента, если изначально он для него не предназначался, что требует от администратора дополнительного времени.

Третий подход позволяет экрану отображать любой тип контента (включая все три), адаптировав его с помощью CSS “@media” «под себя», а не используя масштаб (свойство “zoom”), как в первом случае. В конкретный промежуток времени можно вывести конкретную новость, которая будет качественно смотреться на любом ТВ экране, причём участие администраторов не потребуется. Логика может быть прописана в шаблоне вывода и/или CSS правилах.

Слово контент подразумевает не только текст, но и фото, видео и другой встраиваемый материал. И если с текстом никаких проблем не возникает, то с использованием растровых изображений есть некоторые нюансы [2] (помимо описанного выше, связанного потерей качества при с масштабировании). Адаптивность графических элементов можно реализовать разными способами. Когда сотрудник учебного заведения публикует материал на сайте института, к которому прикрепляет сопроводительные фотографии, то эти фотографии всегда имеют конкретно заданное физическое разрешение. Задача заключается в том, чтобы вывести это изображение с текстом на внутреннее ТВ так, чтобы оно гармонично смотрелось на всех экранах, использующихся внутри учебного заведения. С помощью CSS можно ограничить ширину (или высоту в зависимости от дизайнерского решения шаблона вывода), задать обтекание текстом и ограничить количество фотографий в шаблоне вывода по сравнению с версией на полноценном сайте.

Отбросив нюансы, можно выделить три подхода «адаптации» изображений: с помощью атрибута “max-width:100%;” для тега <img>, с помощью атрибута srcset=”...” этого же тега и с помощью тегов <source>, прописанных внутри контейнера <picture>. Первый подход позволяет адаптировать изображение под максимальную ширину родительского HTML блока и не превышать её. Соответственно при изменении ширины родительского блока (при загрузке на другом экране) изображение будет автоматически подстраиваться под нее и не выйдет за пределы ширины родительского контейнера. К слову, помимо изображений, подобный подход можно применять и для видео, также других объектов, встраиваемых в браузер: “img, video, embed, object {max-width:100%; height:auto;}” (установка автоматической высоты нужна для сохранения пропорций, если физическая высота изображения прописана в качестве атрибута тега). Такой подход не идеален, т.к.

Рис. 4. Позиционирование портретной фотографии на горизонтальных экранах
Fig. 4. Portrait photography positioning on horizontal screens

при любом масштабировании изображений возникает проблема художественного оформления [рис. 4]. Например, если на сайте ВУЗа отображается большой портретный снимок со студентом посередине и при просмотре в браузере на настольном компьютере никаких проблем не возникнет, то при отображении этого же изображения на экране ТВ параллельно с сопроводительным текстом снимок уменьшается. Такое фото будет выглядеть плохо, так как студент будет очень меленьким и его будет тяжело разглядеть. Вероятно, будет лучше показать меньшую портретную картинку, на которой человек отображается в увеличении (в приближении).

Второй подход (“srcset”), наряду с атрибутом “sizes”, позволяет добавить дополнительные изображения, чтобы браузер выбрал подходящее для того или иного ТВ экрана. Атрибут “srcset” включает названия изображений и физические размеры видимой зоны, среди которых браузер выберет нужное, а “sizes” определяет перечень “@media” выражений (например, ширину экрана) и указывает предпочтительную ширину изображения (для примера ниже при ширине экрана от 720 пикселей изображение должно занимать 720 пикселей, при ширине экрана от 1080 – 1080 пикселей, и на экранах шире 1920 пикселей – 1920 пикселей.): <img srcset=”small-image.jpg 720w, medium-image.jpg 1080w, large-image.jpg 1920w” sizes=”(min-width: 720px) 720px, (min-width: 1080px) 1080px, (min-width: 1920px) 1920px” src=”image.jpg”> (нужно вручную прописать дескриптор ширины каждого изображения, т.е. для какой видимой области оно предназначено: в примере 720w, 1080w и 1920w пикселей по ширине; изображение, указанное в атрибуте src=”...” будет выведено на ТВ экран, если параметры экрана не соответствуют ни одному из правил). Этот подход не идеален, хотя и довольно интересен. Так, например, на экраны с вертикальной ориентацией возможно автоматически помещать только заранее подготовленные изображения, т.е. отредактированные физически в графическом редакторе и отличающиеся от оригинального, размещаемого на сайте-источнике. Но при таком подходе мы частично нарушаем принцип отсутствия в необходимости дублирования информации, т.к. администратору придётся готовить несколько изображений, которые браузер автоматически выберет на нужных экранах.

Третий подход с контейнером <picture> очень похож по идеологии на предыдущий вариант. Это, безусловно, более современный и гибкий инструмент для целей Digital Signage. Этот контейнер содержит некоторое количество элементов <source>, которые предоставляют браузеру выбор нескольких разных источников. Элемент <source> принимает атрибут “media”, который содержит @media условие. При помощи этих условий определяется, какое изображение будет выведено, а атрибут “srcset” содержит путь изображения, которое будет выведено на ТВ экран:

```
<picture>
<source media=”(orientation: landscape) and (min-width: 1920px)” srcset=”...”>
<source media=”(orientation: landscape) and (min-width: 1080px)” srcset=”...”>
<source media=”(orientation: portrait) and (min-width: 720px)” srcset=”...”>
<img src=”image.jpg”>
</picture>
```

Принципиальное отличие от второго подхода в том, что там мы указываем условия и целевой размер изображения, а «подбором» наиболее подходящего изображения из доступных занимается браузер, а в <source> мы явно указываем изображение, которое должно быть выведено на ТВ экран при выполнении условия. Т.е. второй подход в отличие от третьего идеологически и синтаксически предполагает, что мы отображаем одно и тоже изображение, но в разных размерах и разрешениях. Недостаток третьего подхода ровно в том же, что и во втором подходе – подготовка нескольких изображений для разных экранов. Если углубиться немного глубже, то всплывает ещё один недостаток, которого нет у <img> и srcset, – это как минимум трёхкратное увеличение узлов в дереве DOM шаблона вывода (по каждому графическому изображению). Это может быть актуально, если используется маломощный компьютер (плеер) или вовсе Smart TV без подключения внешних видео источников, т.к. каждый узел дерева DOM увеличивает расход памяти и замедляет работу скриптов. Преимущество же в более понятном и удобном синтаксисе по сравнению с атрибутами sizes и srcset в <img>, а также в возможности указать в контейнере <picture> более современные форматы изображений, например, *.avif, *.webp или *.svg.

Возвращаясь к шаблонам вывода, отдельно стоит выделить CSS фреймворки, такие как Bootstrap или Foundation, а также непосредственно сами модули Flexbox и CSS Grid с «колоночным» подходом к вёрстке [Рисунок 5]. Все они могут выступать отправной точкой при разработке адаптивного шаблона вывода, создавая его на основе наборов горизонтальных и вертикальных линий (основной и поперечной осей) [3]. Фреймворки включают в себя основные функции для создания прототипов, но принципы написания CSS кода и работа с фреймворками выходят за рамки данной статьи.

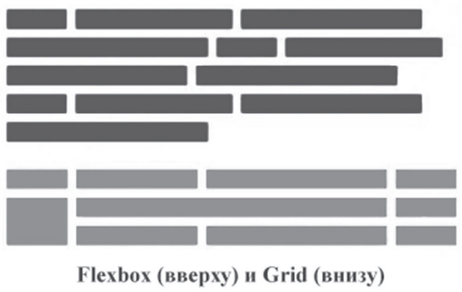

Рис. 5. Адаптивное расположение блоков с помощью Flex контейнеров (Flexbox) и колоночного/сеточного принципа (Grid)

Fig. 5. Adaptive layout of blocks using Flex containers (Flexbox) and column/grid principle (Grid)

Как было сказано в начале статьи, основной принцип построения систем Digital Signage на web-технологиях заключается динамическом масштабировании элементов, – уходе с фиксированной ширины измерения блоков в пикселях на гибкую, указываемую в процентном соотношении от ширины и/или высоты экрана. Но помимо процентных величин при выводе контента на ТВ экраны и киоски в спецификации CSS3 вводятся новые единицы измерения, связанные с областью просмотра: vw, vh, vmin и vmax. По факту их больше, например, модификаторы с префиксом s, l, d, но нам они не интересны, т.к. они завязаны на учёт панели навигации, адресной строки браузера разных мобильных устройств, а в интерактивных кисках и ТВ экранах эти элементы не задействованы, т.к. всегда используется полноэкранный режим отображения. Единица измерения vw (viewport width) представляет собой процент ширины видимой части экрана (области просмотра/отображения), а vh (viewport height) – процент высоты. vmin это значение минимальной ширины и высоты

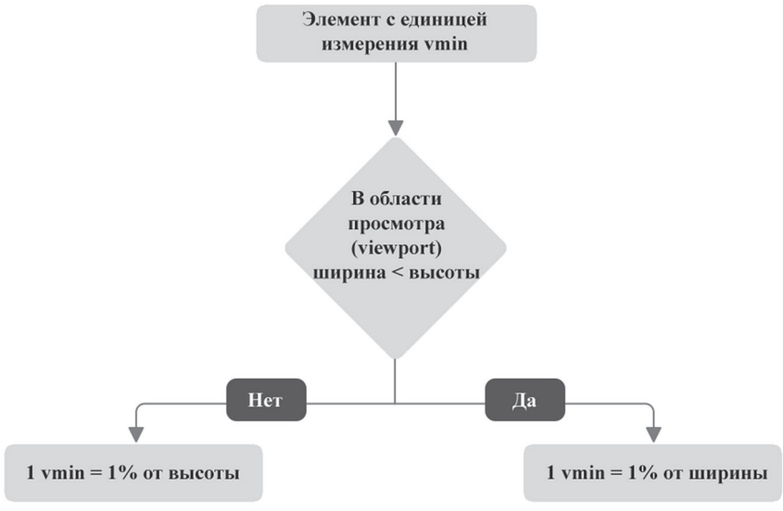

Рис. 6. Определение минимальной отображаемой в данный момент ширины или высоты экрана просмотра.

Fig. 6. Defining the minimum currently displayed width or height of the viewing screen.

видимой области просмотра. Если ширина области просмотра больше высоты (горизонтальный режим, характерный для ТВ), то значение будет рассчитываться на основе высоты и наоборот, когда высота больше ширины (вертикальный режим, характерный для киосков) [рис. 6]. В противоположность vmin есть vmax. Значение рассчитывается на основе максимальной ширины и высоты.

Рассмотренные единицы измерения, связанные с видимой областью просмотра, зависят от корневого элемента web контента (по сути от физического экрана, ведь браузер всегда будет запущен в полноэкранном режиме), а проценты зависят от контейнера/селектора, внутри которого происходит определение (в этом разница между 100vh/100vw и 100%).

Сами по себе единицы vw, vh, vmin, vmax, хоть и используются в системах Digital Signage, построенных на web технологиях (например, для задания размера шрифта, межстрочного или межблочного интервалов при адаптивном расположение блоков) [рис. 5], не так интересны, как в комбинации с функцией calc(), позволяющей применять базовые математические операции. Используя её в независимости от ориентации экрана и его разрешения можно оставить небольшую контролируемую высоту под классическую бегущую строку, отняв от видимой высоты экрана 80 пикселей: “height: calc(100vh - 80px);”. Или оставив задел для меню навигации в киоске: “width: calc(100vw - 250px);” [рис. 1].

Одну из особенностей, которую будет полезно рассмотреть в данной статье – скрытие ссылок и полноценных предложений/абзацев с текстом «перейти по ссылке», «скачать здесь», т.к. они будут смотреться неуместно на ТВ экранах, размещённых в образовательном учреждении, с которыми исключено взаимодействие студентов (исключение – интерактивные киоски, но даже на них «скачивание» чего-либо не имеет смысла, как и переходы по ссылкам за пределы домена).

Для обеспечения такой возможности можно создать привязку к идентификатору браузера user-agent (как правило, на ПО киосков и плагинов браузеров, работающих в полноэкранном режиме Digital Signage, есть возможность строку user-agent задать вручную, либо она уже прописана как “kiosk”, “touch” и т.п.). В качестве правил автор статьи рекомендует создать классы .tvhide, .kioskhide и .webhide (для ТВ экранов, интерактивных сенсорных киосков и сайта ВУЗа соответсвенно):

```
if(navigator.userAgent.match(/(kiosk1)|(kiosk2)/i)){
    document.write('<style>.kioskhide {display:none;}</style>');
    // или через полноценный *.css файл: document.write('<link rel="stylesheet" type="text/css" href="kiosk.css">');
    }
```

Выше пример для интерактивных киосков, однако все три класса имеют одинаковое свойство “display:none”. Отличие лишь в том, что это свойство или CSS файл, в котором описан этот класс, загружается только в шаблоне для нужного устройства или через Javascript (как в примере выше), если шаблон единый для всех устройств вывода. Таким образом, редактируя контент в одном редакторе (например, в CMS сайта), можно заранее скрыть блоки текста (обернув их в) или целые абзацы (<section>, <article>, , <p>), если они будут неуместно смотреться на том или ином устройстве. Например, можно указать класс для параграфа <p class=“tvhide”> и текст внутри этого параграфа не будет отображён только на ТВ, т.к. только для этого типа устройств будет описано свойство “display:none”; на других же устройствах текст будет виден. Аналогично эти классы можно применять для других типов устройств, либо комбинировать их, указав сразу два класса (например, одновременное использование class=“tvhide webhide” скроет соответствующие элементы на ТВ экранах и сайте ВУЗа, оставив контент на интерактивных киосках).

Альтернатива этому методу – использование стилей непосредственно в шаблоне вывода (в этом случае для каждого типа устройств он будет свой, что потребует ручной настройки). Необходимый шаблон вывода можно загружать при запросе заранее определённого URL или параметра в URL. Несмотря на то, что этот подход является более сложным, он позволит реализовать удобную автоматизацию ротации контента (например, нескольких статей или фотографий) и автообновление через Javascript, но разработка таких шаблонов для разных типов систем Digital Signage выходит за рамки данной статьи.

Один из последних методов создания адаптивного контента, который будет рассмотрен в данной статье касается видео. Поскольку разрешение и ориентация экранов у нас разная, то нам понадобится масштабировать видео под всю доступную ширину родительского контейнера, не привязываясь к физическому разрешению видео [5]. Метод состоит в том, чтобы создать блок с соответствующими пропорциями (4:3, 16:9, и т.п.), а затем поместить видео в этот блок, растянув его до размеров этого контейнера. Рассмотрим на примере вставки видео из социальной сети «ВКонтакте». Как уже обозначалось ранее в статье, inline стили – плохое решение из-за сложности дальнейшего

переназначения, но в примере ниже это сделано для удобства восприятия (конечно, для этого лучше создавать соответствующие CSS классы):

<div style=”position: relative; padding-bottom: 56.25%; overflow: hidden;”>

<iframe style=”position: absolute; top: 0; left: 0; width: 100%; height: 100%;” src=”https://vk.com/video_ext.php?oid=...”></iframe>

Установка position на relative в родительском контейнере дает возможность применить абсолютное позиционирование к самому <iframe>. Значение padding-bottom высчитывается из соотношения сторон видео файла. В примере выше соотношение составляет 16:9, что означает высоту в 56,25% от ширины (для видео с соотношением 4:3 устанавливается padding-bottom на 75%). Установив overflow на hidden, мы гарантируем, что любой контент, который вылезет за пределы этого контейнера, будет скрыт. Касательно самого <iframe> с видео использовано абсолютное позиционирование, т.к. по факту у содержащего элемента высота равна 0. Если бы <iframe> был нормально позиционирован, мы бы также установили ему высоту в 0. Свойства top и left помещают <iframe> точно в родительский контейнер. При этом свойства width и height гарантируют, что видео займет 100% доступного пространства.

В современных системах можно использовать более изящный набор CSS правил на основе указания соотношения сторон видео: iframe {width: 100%; aspect-ratio: 16/9;} без использования «обёртки» в виде как в предыдущем примере, однако в этом методе отсутствует совместимость со старыми браузерами, что особенно актуально, если ПО Digital Signage использует движок на базе Google Chrome (Chromium) ниже версии 88 [https://caniuse.com/mdn-css_properties_aspect-ratio].

Касательного любого видео файла (загруженного на сервер или встраиваемого со сторонних платформ) нужно устанавливать autoplay=1 (автостарт) и loop=1 (зацикливание) в <iframe>, если вывод осуществляется на ТВ. Однако параметр зацикливания во «ВКонтакте» не работает, а в YouTube работает только для плейлистов (т.е. от двух видео, объединённых вручную в плейлист). При загрузке видео файлов на собственный сервер таких ограничений нет.

Подводя итоги создания системы адаптивной трансформации контента в формате гипертекста, несмотря на описанные в начале возможности, существуют особенности, о которых также стоит упомянуть. Наборы адаптивных CSS правил не всегда (и тем более не на всех экранах) могут решить нюансы адаптации того или иного типа контента (особенно графического и мультимедийного с фиксированным физическим разрешением). В тоже время для одного ТВ экрана (например, с низким или наоборот высоким разрешением, с вертикальной ориентацией) может потребоваться написание большого количества сложных правил (в рамках только одного HTML шаблона вывода). Нюансы с адаптацией изображений могут потребовать либо подготовки отдельных изображений, либо отдельных шаблонов вывода. И в целом, не каждый шаблон вывода (структуру макета, изначально заточенную под конкретный экран Digital Signage), можно сделать адаптивным, что потребует большого количества времени web разработчика.

Из-за сложного подхода в использовании растровых изображений имеет смысл делать разные шаблоны вывода (макеты) для экранов с горизонтальной и вертикальной ориентацией. В случае использования экранов разного разрешения, но с одинаковым соотношением сторон, идеальным методом будет разработка шаблона вывода под разрешение экрана с максимальным разрешением и физическое масштабирование CSS стилем zoom с относительным уменьшающим процентом. В случае использования экранов одинаковой ориентации, но разного соотношения сторон (например, 16:10, 16:9, 4:3) HTML шаблон может быть один и тот же, но CSS (@media) правила будут отличаться разными относительными величинами для разных соотношений сторон. Несмотря на одинаковый шаблон вывода и единую базу данных с контентом (в качестве которой в подавляющем большинстве практических случаев выступает БД «движка» сайта), какими-то элементами интерфейса можно пренебречь, просто скрыв их CSS свойством “display:none”. Разные типы устройств – сенсорные киоски и простые экраны могут отличаться по структуре шаблонов вывода и дополнительным CSS правилам, например, первые могут иметь навигационное меню для взаимодействия, а вторые встроенные в шаблон вывода авторотаторы контента/слайдеры, реализованные с помощью Javascript, поскольку какое-либо взаимодействие с не сенсорными экранами невозможно.

Литература

1. Nikolaos Sagiadinos. How to create Responsive Digital Signage Content [Электрон. ресурс] // SMIL Control. 2020. Режим доступа: https://smil-control.com/responsive-digital-signage/. (Дата обращения: 18.01.2024).
2. Chris Coyier. A Guide to the Responsive Images Syntax in HTML. 2020. [Электрон. ресурс]. Режим доступа: https://css-tricks.com/a-guide-to-the-responsive-images-syntax-in-html/. (Дата обращения: 18.01.2024).
3. Anna Fitzgerald. Here's the Difference Between Flexbox, CSS Grid & Bootstrap [Электрон. ресурс]. 2022. Режим доступа: https://blog.hubspot.com/website/css-grid-vs-flexbox. (Дата обращения: 10.03.2024).
4. Nader Abd ALhamed. Unveiling the Magic: How css works behind the scene [Электрон. ресурс]. 2023. Режим доступа: https://www.linkedin.com/pulse/unveiling-magic-how-css-works-behind-scene-nader-abd-alhamed. (Дата обращения: 10.03.2024).
5. Thierry Koblentz. Creating Intrinsic Ratios for Video [Электрон. ресурс]. 2009. Режим доступа: https://alistapart.com/article/creating-intrinsic-ratios-for-video/. (Дата обращения: 24.03.2024).

References

1. Nikolaos Sagiadinos. How to create Responsive Digital Signage Content [Internet]. SMIL Control. 2020. Available from: https://smil-control.com/responsive-digital-signage/. (cited 18.01.2024).
2. Chris Coyier. A Guide to the Responsive Images Syntax in HTML. 2020. [Internet]. Available from: https://css-tricks.com/a-guide-to-the-responsive-images-syntax-in-html/. (cited 18.01.2024).
3. Anna Fitzgerald. Here's the Difference Between Flexbox, CSS Grid & Bootstrap [Internet]. 2022. Available from: https://blog.hubspot.com/website/css-grid-vs-flexbox. (cited 10.03.2024).
4. Nader Abd ALhamed. Unveiling the Magic: How css works behind the scene [Internet]. 2023. Available from: https://www.linkedin.com/pulse/unveiling-magic-how-css-works-behind-scene-nader-abd-alhamed. (cited 10.03.2024).
5. Thierry Koblentz. Creating Intrinsic Ratios for Video [Internet]. 2009. Available from: https://alistapart.com/article/creating-intrinsic-ratios-for-video/. (cited 24.03.2024).

Сведения об авторе

Дмитрий Сергеевич Филиппов
Ведущий программист, Аспирант кафедры Вычислительные системы, сети и информационная безопасность Российского университета транспорта (МИИТ), Москва, Россия
Эл. почта: dsfilippov@ui-miit.ru

Information about the author

Dmitry S. Filippov
Senior programmer, Postgraduate student of the Department of Computer Systems, Networks and Information Security at the Russian University of Transport (MIIT), Moscow, Russia
E-mail: dsfilippov@ui-miit.ru