УДК 004.9

ПРОГРЕССИВНЫЕ ВЕБ-ПРИЛОЖЕНИЯ

К.С. ГОЛОВАН
(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Раскрываются понятия нативных приложений, прогрессивных веб приложений, манифеста, ServiceWorkers. Приводится сравнение нативных и прогрессивных приложений.

Интерес к разработке приложений для мобильных операционных систем в сообществе разработчиков стабильно растет, количество инструментов и подходов позволяющих создавать приложения становится все больше. Примерно с 2005 года веб-технологии перешли от использования статических страниц на динамические, т.е. генерируемые сервером (PHP, ASP.NET) и клиентской стороной (AJAX), а также пришли к понятию респонсивный дизайн. Однако с выходом первой линейки телефонов iPhone данные технологии отошли на второй план, так как нативные приложения предлагали более быструю работу приложения и удобный интерфейс для пользователя. В середине 2010 года улучшения в HTML5, CSS3 и JavaScript наравне с развитием браузерных движков сделали так называемые гибридные приложения настоящей альтернативой нативным приложениям. Гибридные приложения «воспроизводят» нативный интерфейс и требуют установки из специального хранилища приложений. Комбинация скриптов, стилей и разметки дает возможность создавать интерфейсы без использования технологии Flash. Также гибридные приложения не имеют доступа по URL, поддерживают богатый пользовательский интерфейс и имеют доступ к операционной системе. В 2015 году компанией Google был представлен один из новых подходов к разработке веб-приложений, а именно прогрессивные веб-приложения (ProgressiveWebApps, прогрессивные веб-приложения).

Прогрессивные веб-приложения используют преимущества новых технологий, чтобы принести пользователю все самое лучшее из мобильных сайтов и нативных приложений. Они надежные, быстрые и привлекательные. Они доступны из безопасного источника и загружаются независимо от состояния сети.

PWA – это название группы приложений, которые используют стек Web технологий (JS + HTML + CSS) и позволяют соединить простоту использования Web сайта со специфичными для нативных приложений операционной системы UX и техническими возможностями.

Основное предназначение PWA увеличить конверсию, количество пользователей и удобство использования веб-приложений на мобильных устройствах.

ProgressiveWebApps является логическим продолжением AcceleratedMobilePages. Таким образом, если вы ранее создавали AMP приложения, то вам однозначно стоит обновить свое приложение к нормам PWA приложений. Если вы до этого ничего не слышали о AMP, то это не станет для вас проблемой во время изучения PWA.

PWA приложению необходимо соответствовать следующим тебованиям:

- быть прогрессивным – работать с каждым пользователем вне зависимости от окружения, используя метод постепенного улучшения как основной принцип работы;
- адаптивным – подстраиваться под любое устройство: десктоп, смартфоны, планшеты или что либо другое;
- независимым от соединения, используя ServiceWorker приложение должно работать в оффлайн режиме при прерывании или отсутствии соединения;
- выглядеть нативно – приложение должно соответствовать привычным для пользователя способам взаимодействия и навигации;
- самообновляемым – приложение должно контролировать процесс автоматического обновления посредством ServiceWorker API;
- безопасным – посредством использования HTTPS предотвращать перехват и подмену данных;
- определяемым – посредством W3C манифеста и регистрации через ServiceWorker приложение идентифицируется как «приложение» поисковыми системами;
- удерживающим – используя технические возможности, мотивируем пользователя еще раз использовать приложение, например, посредством push уведомлений;
- легким в установке – позволяет «сохранить» приложение на устройстве пользователя посредством добавления PWA приложения в список установленных приложений без использования магазина приложений;
- легким в использовании – для начала использования приложения достаточно открыть URL. Установка приложения не обязательна.

Для создания прогрессивных веб-приложений используются следующие основные технолгии: манифест и ServiceWorkers [1].

Манифест веб-приложения предоставляет информацию о приложении (такую как имя, авторство, иконку и описание) в формате JSON-файла. Цель манифеста – установить веб-приложение на домашний экран устройства, предоставляя пользователю более быстрый доступ и больше возможностей.

Манифест веб-приложения внедряется в HTML-страницу, с помощью тега ссылки в заголовке документа:

```
<link rel="manifest" href="/manifest.webmanifest">
```

Пример манифеста:

```
{
  "name": "HackerWeb",
  "short_name": "HackerWeb",
  "start_url": ".",
  "display": "standalone",
  "background_color": "#fff",
  "description": "A simply readable Hacker News app.",
  "icons": [{
    "src": "images/touch/homescreen48.png",
    "sizes": "48x48",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen72.png",
    "sizes": "72x72",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen96.png",
    "sizes": "96x96",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen144.png",
    "sizes": "144x144",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen168.png",
    "sizes": "168x168",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen192.png",
    "sizes": "192x192",
    "type": "image/png"
  }],
  "related_applications": [{
    "platform": "web"
  }, {
    "platform": "play",
    "url": "https://play.google.com/store/apps/details?id=cheeaun.hackerweb"
  }]
}
```

Манифест типового прогрессивного веб-приложения содержит параметры, которые определяют ожидаемый цвет фона для веб-приложения; описание того, что делает приложение; предпочтительный для разработчика режим отображения приложения; массив объектов изображений, которые могут быть

использованы как иконки приложения в различных контекстах, например в разных ОС; удобочитаемое имя, ориентацию [2].

На рисунке 1 представлен пример различного отображения приложения для разработчика.

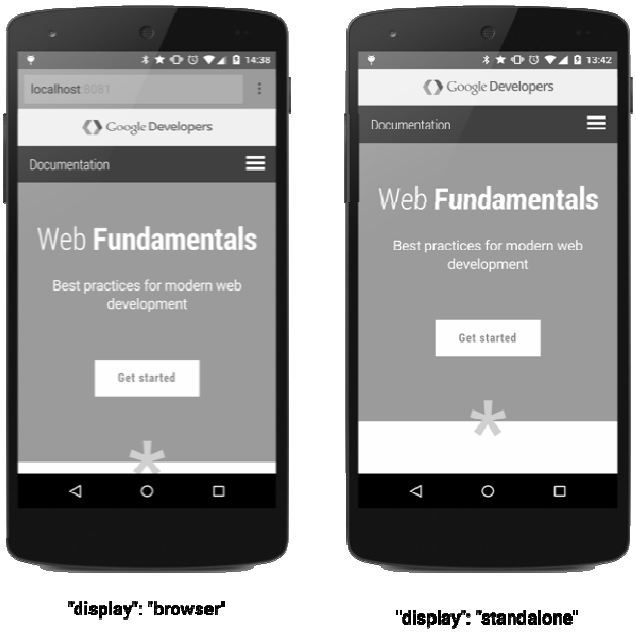

Рисунок 1. – Пример различного отображения прогрессивного веб-приложения при изменении параметра display в манифесте

Serviceworker – это событийно-управляемый worker, регистрируемый на уровне источника и пути. Он представляет собой JavaScript-файл, который может контролировать веб-страницу/сайт, с которым он ассоциируется, перехватывать и модифицировать запросы навигации и ресурсов, очень гибко кешировать ресурсы для того, чтобы предоставить вам полный контроль над тем, как приложение ведет себя в определенных ситуациях (например, когда сеть не доступна).

Serviceworker запускается в контексте worker'а, поэтому он не имеет доступа к DOM и работает в потоке отдельном от основного потока JavaScript, управляющего вашим приложением, а следовательно не блокирует его. Он призван быть полностью асинхронным; как следствие, синхронные API, такие как XHR и localStorage, в serviceworker'е использовать нельзя.

Из соображений безопасности serviceworker'ы работают только по HTTPS [3].

Serviceworkers также предназначены для использования для таких вещей, как:

- фоновая синхронизация данных;
- ответ на запросы от других источников;
- получение централизованного обновления для данных, сложных для расчетов, таких как геолокация или гироскоп, так что несколько станиц может использовать одни и те же данные;
- компиляция на клиентской стороне и управление зависимостями для CoffeeScript, less, CJS/AMD модулей и т.д. для целей разработки;
- перехват фоновых сервисов;
- кастомная шаблонизация, основанная на определенных паттернах URL;
- улучшение производительности, к примеру, предварительная загрузка ресурсов, которые понадобятся пользователю в ближайшем будущем, как несколько последующих картинок в фотоальбоме.

В будущем serviceworkers будут способны на многие другие полезные вещи для web-платформ, приблизив их к жизнеспособности нативных приложений. Примечательно, что другие спецификации могут и будут использовать контекст serviceworker, например, для следующих целей:

- фоновой синхронизации: запускать serviceworker, даже когда ни одного пользователя нет на сайте, чтобы обновить кеш;
- реакции на пуш-сообщения: запускать serviceworker для отправки сообщений пользователям, чтобы оповестить их о новом доступном контенте;
- реакции на отпределенное время и дату;
- введение гео-ограничений.

Также некоторые PWA используют архитектурный подход, который называется AppShellModel. Для мгновенной загрузки сервис воркеры хранят базовый пользовательский интерфейс или «оболочку» (shell) приложения. Данная оболочка является статическим фреймом или макетом, в который контент загружается постепенно, при этом динамически позволяя пользователям взаимодействовать с приложением вне зависимости от качества соединения. Технически оболочка представляет собой код, который хранится в кэше браузера мобильного устройства.

Таким образом, стандартный процесс работы с прогрессивным веб-приложением заключается в следующем: пользователь, используя смартфон, открывает ссылку, полученную посредством любого приложения, после загрузки web страницы пользователь получает полноценное приложение, которое он может использовать.

Тем самым начать использовать новое приложение становится намного проще, так как не нужно заходить в магазин приложений и ждать пока установится необходимое пользователю приложение.

При необходимости пользователь может добавить приложение на рабочий стол посредством пункта опций в браузере «Addtohomescreen». Или приложение может предложить пользователю сделать это вместо него посредством Webapp-install-banner. После чего на главном экране пользователя будет создана иконка, предварительно указанная в манифесте приложения. Разработчик может управлять стилем браузера и вариантом отображения приложения после установки. Также ServiceWorker, идущий в комплекте с PWA приложением, будет отвечать за кеширование, оффлайн работу и обработку пуш сообщений.

Заключение

Прогрессивные веб-приложения являются инновационным подходом к разработке, так как заключают в себе сильные стороны веб-сайтов (сами по себе ими и являются) и нативных мобильных приложений. То есть, с одной стороны, приложение имеет ссылочную структуру, небольшой размер, с другой – приложение ведет себя как настоящее нативное приложение с удобным и понятным интерфейсом. В основе PWA лежит ServiceWorker – посредник между устройством пользователя и сетью, который предоставляет контент вне зависимости от качества соединения и его состояния, при этом также предоставляет функции нативных приложений, такие как работа оффлайн, push-уведомления, экраны загрузки и другие.

На момент проведения исследований полная поддержка PWA обеспечивалась во всех браузерах, кроме Safari от Apple. С большей вероятностью данная технология не будет принята компанией Apple. Это может означать, что в ближайшем будущем может появиться новый подход в веб-разработке как конкурент прогрессивных приложений.

ЛИТЕРАТУРА

1. ProgressiveWebApps: WhoAmI[Электронный ресурс]. – Россия, 2017. – Режим доступа: https: //habrahabr.ru/post/303626/. – Дата доступа: 03.10.2017.
2. Веб-документация MDN [Электронный ресурс] / Манифест веб-приложения. – США, 2017. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/Манифест. – Дата доступа: 03.10.2017.
3. Веб-документация MDN [Электронный ресурс] / ServiceWorker API. – США, 2017. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/API/Service_Worker_API. – Дата доступа: 03.10.2017.