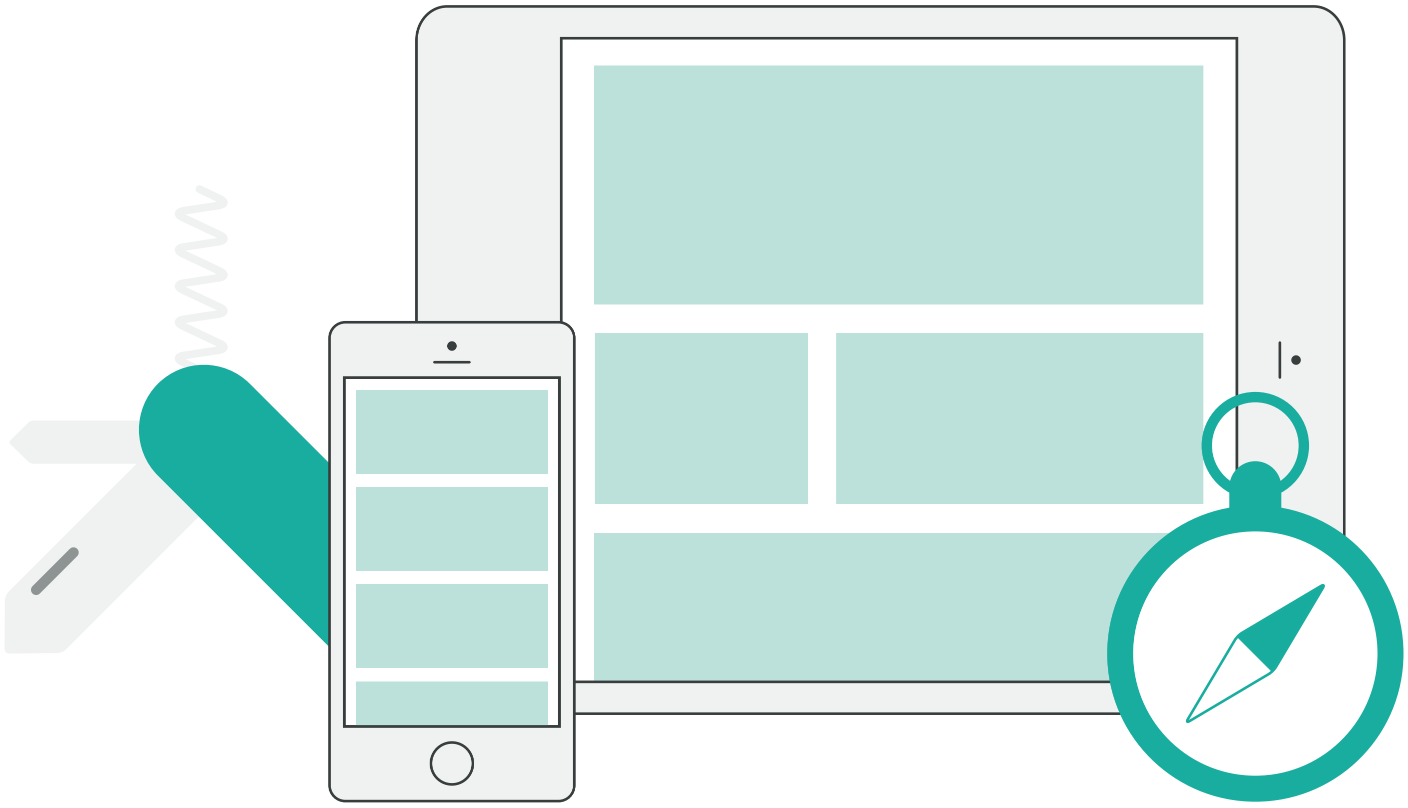
DeviceAtlas™

A CONTENT ADAPTATION SURVIVAL GUIDE

HOW TO ADDRESS MOBILE USERS IN A FUNDAMENTALLY DIFFERENT WAY



CONTENTS

INTRODUCTION

While the average size of a typical PC screen hasn't changed much during the last decade, mobile devices have been released in a myriad of sizes and shapes. As a result, users can now access websites on a variety of web-enabled devices, from low-end phones, to smartphones, tablets and even wearable devices.

Most marketers are acutely aware that they need to address the needs of mobile users. However, it is hard to come up with a solution that exploits mobile to its fullest potential.

While performance issues with RWD are being dealt with, addressing the mobile context solely with RWD can be inefficient in some cases. **Perhaps you have looked at Responsive Web Design (RWD), but decided that it doesn't give the flexibility required to treat mobile in a fundamentally different way.**

What we are specifically referring to in this e-book, is the ability to change content according to the visitor's device characteristics. If you are thinking of following this approach and designing differently for mobile, then you've come to the right place. This e-book will help you understand the techniques of content redirection and adaptation.

Here's what you'll learn in this e-book

- Why catering to a wide range of devices is crucial today.
- What are the advantages of content adaptation over 'classic' RWD?
- What techniques are available for content adaptation?

DEVICE FRAGMENTATION

While a top-notch mobile experience was once the preserve of high-end devices on generous data plans, it is now available to low and mid-range devices. Ever-growing device fragmentation has opened up whole new markets to mobile marketers and developers.

Check out a few notable fragmentation figures

- 1.2 billion smartphone shipped worldwide in 2013 (IDC)
- 16,000 distinct web-capable devices recognized by DeviceAtlas device detection solution
- 19 million wearable devices will be shipped in 2014 and 111.9 million will be shipped in 2018 (predicted, IDC research)

Device fragmentation doesn't only refer to users on-the-go. The results from Adobe's 2013 Mobile Consumer Survey showed 80% tablet users interact with their devices in the comfort of their homes. Whilst according to a Nielsen study, 84% of smartphone and tablet owners use their gadgets when watching TV.

Moreover, new innovative devices emerge on a regular basis. Though initially intended for tech-savvy early-adopters, the likes of smartwatches, smartglasses, and head-mounted displays may become mainstream products in the near future.

The growth of mobile device usage is generally great news for businesses because it opens new possibilities for conveying brand messaging, but it is also a big challenge to get it right.

A BAD MOBILE EXPERIENCE CAN HAVE LONG-LASTING EFFECTS

With powerful smartphones that allow you to view desktop websites are constantly hitting the market, some may be inclined not to adapt content at all. Such a decision may end up damaging your brand's online presence.

Providing mobile users with only a desktop website usually leads to a bad experience, given that these websites aren't built for mobile devices' screen size and resolution. They also require a lot of processing power and bandwidth.

Users expect websites to load on mobile devices as quickly as they do via desktop browsers. They also expect the experience to be as good as the desktop. Numerous statistics prove that the consequences of a poor mobile experience can be far reaching.

- 85% of users expect pages to load on mobile devices as fast or faster than they load on desktops.
- 46% of mobile web users are unlikely to return to a website they had trouble accessing in the past.
- 33% of tablet users are less likely to purchase online from a company if they experience poor website performance.

Source: mobiForge.com

One aspect of the online landscape today is that desktop websites generally load slower and weigh more in terms of downloadable assets. The median loading speed for the top 500 e-commerce websites was 9.3 seconds in 2014 (7.7 seconds in 2013), while the median page size of these top 500 sites was 1,436KB in 2014, as against 1,094KB in 2013.

While Responsive Web Design is a great approach for providing users with a better mobile experience, it has its limitations. RWD websites generally require more processing power, which can make them even slower than their non-responsive counterparts.

You can read more on RWD and other content adaptation techniques later in this e-book.

WHY UTILIZE CONTENT ADAPTATION?

A website's loading speed impacts the conversion rate and this rule applies to mobile visitors as well. Strangeloop Networks conducted an experiment in which 4 groups of users were displayed 4 versions of the same mobile website, of which only one was optimized, while the three others had varying levels of delay. The study found that with only 1 second delay the bounce rate was 8.3% higher than for the fully optimised site, the conversion rate was 3.5% lower, and there was 9.4% fewer page views.

Speed and performance are important aspects for the largest online retailers utilizing various levels of content adaptation. According to mobiForge's findings from 2012, 82% of the Alexa 100 top sites used some form of server-side device detection to serve optimised content on their main website entry point. 74% displayed 3 different versions tailored for a variety of devices. Every website in the Alexa top 10 used some form of content adaptation.

These stats are on par with 2014 Radware study which found that 81% of the top 100 retail sites as rated by Alexa.com use content adaptation by sending a mobile specific website to smartphones.

Here are a few notable examples of content adaptation.

TripAdvisor

- Different versions of website sent to different devices.
- A GPS-based list of restaurants, hotels and attractions nearby.
- A super-light (40KB) version for low-end phones.

Home Depot

- An m-dot website using location sharing to detect a customer's nearest store.
- Access to real-time store inventory and details on the aisle location of products.

AccuWeather

- An m-dot website for mobile devices.
- A GPS-based 'use current location' option for checking the weather forecast.

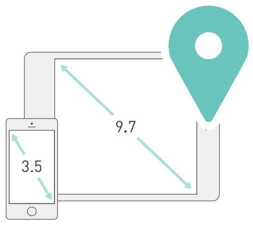
DESIGN CONSIDERATIONS BEFORE APPROACHING CONTENT ADAPTATION

There are a number of important considerations before designing a website that offers a consistent context-aware experience for different types of users.

NAVIGATION STYLE

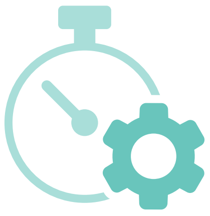
An important aspect to consider refers to available input methods, such as touchscreen, gesture, voice control, remote control, controllers, etc. Input methods define the range of possible interactions.

DISPLAY AND DEVICE CAPABILITIES



Web-enabled devices can vary significantly and have different capabilities from raw processor speed, to location capabilities, to screen size, to page rendering abilities.

PAGE SIZE



Processing power is not the only factor affecting websites' loading speed on mobile devices. There's also bandwidth availability and the user's data plan. A hefty website may take a long time to download, even accessed on devices with 4G support and multi-core CPUs. Also many users get charged for data, so a large page can be considerably expensive to access.

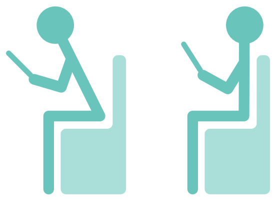
See connectivity analysis for more info. https://deviceatlas.com/resources/dynamic-data#connectionSpeed

CONTEXT

How and why a user is accessing a website is important to understand, so the most suitable content can be served. If a mobile user is on WiFi, they may be more accepting of larger pages and increased site depth, whereas if the user is out and about on a mobile carrier then they may want just specific information.

USER POSTURE

Mobile devices are often used on-the-go, but a more stationary home use is also very common. According to some media theorists there are two engagement styles for stationary use, including ‘lean back’ and ‘lean forward’. ‘Lean forward’ implies paying attention to the interaction with the medium, while ‘lean back’ refers to a less focused use. User posture allows you to make decisions about the type of content displayed on mobile devices.



Did you know...?
Mobile search increasingly important

- 25% of overall search queries are now on mobile devices.
- 58.7% of smartphone users and 73.9% of tablet users access search.
- 42% of calls to businesses are driven by mobile search.
- 80% of local searches on mobile devices turn into purchases.

Source: mobiForge, Search Engine Land

SERVER-SIDE CONTENT ADAPTATION

Content adaptation is all about providing the user with a first-class experience specifically tailored for the user's device and context. You can achieve that by applying a number of techniques.

Server-side content adaptation combines web design with server-side device techniques. These do not exclude responsive design (RWD) which is briefly explained later in this e-book.

DEVICE TYPE IDENTIFICATION

In order to serve the best content to each user, you need to analyse their devices. To do this you can use web traffic logs and a device detection platform such as DeviceAtlas. The recognition is based on HTTP header parsing to return all the important information about the device, such as the device type, OS, type of browser, device name and make, etc.

Then you can classify the users into a number of 'buckets' to reduce the amount of custom design work that needs to be done. Device type is often used as the criteria for bucket definition and typically there are buckets for smartphones, tablets and desktop users. There could be also a basic experience for low-end devices.

PROPERTY CONSIDERATIONS

The device properties to target depend on the type of adaptation and how fine-grained it should be. At the simplest level the property isMobilePhone might be used to redirect a user to the mobile version of a site. However, typically a finer-grained control is more effective.The following table outlines some of the properties that could be used with DeviceAtlas's recognition.

Properties	Use case
isMobilePhone	Used for coarse grained targeting mobile phones.
primaryHardwareType	Used for finer redirection or template selection.
touchScreen	Used in conjunction with other properties to decide if a device is a smartphone. It is also an important property to ensure the interaction with the website is optimal.
displayWidth & displayHeight	Used to dynamically resize images to suit different screen sizes.
devicePixelRatio	Used to identify "retina" screens and when used with the displayWidth/Height properties can help determine the optimal image size to serve.

DeviceAtlas provides you with a list of 157 device properties. For a full list see the properties page on deviceatlas.com.

3 APPROACHES TO SERVER-SIDE ADAPTATION

In this section you'll find 3 examples of server-side content adaptation.

The following approaches will be discussed with code samples:

1. Redirection - The user is redirected to another website suitable for their device.

2. Templates - A different output template is used depending on the device.

3. Dynamic Adaptation - The page content is modified to suit the device.

Note

The adaptation examples here are based on Java and Servlets. The usage under other languages, especially dotNet, would be very similar.

These techniques are based on DeviceAtlas solution, however they may refer to other device recognition solutions as well.

Creating a DeviceAtlas instance

Before showing examples for Redirection, Templates and Dynamic Adaptation, we explain how to correctly create a DeviceAtlas API instance which would be common between all web-application use cases.

web.xml

```
<web-app ...>
    <listener>
        <listener-class>TheListenerClass
        </listener-class>
    </listener>

    ...

</web-app>
```

TheListenerClass.java

```
import java.servlet.*;

public class TheListenerClass implements
ServletContextListener {

    public void contextInitialized(ServletContextEvent
    e) {
     /* load the datafile in a try/catch block as several
       exceptions may be thrown */
     try {
        // create a DeviceApiWeb instance
        DeviceApiWeb deviceApi = new DeviceApiWeb();
        // load the device atlas JSON data file
        deviceApi.loadDataFromFile("/PATH/TO/
        DEVICEATLAS/DATA-FILE.json");

        /* store the DeviceApiWeb instance in the
          servlet context */
        ServletContext context = e.getServletContext();
        context.setAttribute("DeviceApiWeb",
        deviceApi);

     } catch  java.io.FileNotFoundException ex) {
        // data file was not found
        /*  you would need to become aware of this
         exception */
     } catch (Exception ex) {
        // other errors (you can log the errors)
     }
   }

}
```

1. Redirection

This is the simplest technique to handle different user types. It simply redirects the user to a different site or subdomain depending on the user's device type.

Let's assume that DESKTOP_URL, TABLET_URL, LOWEND_MOBILE_URL, HIGHEND_MOBILE_URL and DESKTOP_URL are the addresses of our specialized sites.

```
// get the API instance from servlet context
DeviceApiWeb deviceApi = (DeviceApiWeb)context.
getAttribute("DeviceApiWeb");

// get the properties from the current request object
Properties properties = deviceApi.
getProperties(request);

/* Based on device properties redirect the user to the
 proper site or subdomain. In this example the
 redirection in based on device type assuming we
 already have specialized sites for each device type. */

/* it's a desktop browser or a device that is
 masquerading itself as desktop browser */
if (properties.contains("isBrowser", true) ||
properties contains("isMasqueradingAsDesktop",
true)) {
     /* redirects to the URL/website designed to handle
     desktop browser requests */
     response.sendRedirect(DESKTOP_URL);
     return;
}

// it's a tablet device
if (properties.contains("isTablet", true)) {
     /* redirects to the URL/website designed to handle
     tablet requests */
     response.sendRedirect(TABLET_URL);
     return;
}

// it's a mobile device
if (properties.contains("mobileDevice", true)) {
     /* you can create conditions on various properties
     to understand if the device is low-end or
     high-end or even get more specific details on what
     it supports */

     /* redirects low-end devices which support WML
     but  not basic XHTML to a URL/website which
     provides contents wrapped in WML */
     if (properties.contains("markup.wml1", true) &&
     properties.contains("markup.xhtmlBasic10",
     false))
     {
          response.sendRedirect(LOWEND_MOBILE_
         URL);
          return;
     }
```

```
        // continued from previous page

        /* redirects high-end devices to the URL/
        website designed to handle their requests */
        response.sendRedirect(request.
        getContextPath() + HIGHEND_MOBILE_URL);
        return;
    }

    // it's a robot
    if (properties.contains("isRobot", true)) {
        out.println("output something like the site-map for
        the robots");
    } else {
      /* anything not handled is redirected to a default
     URL (desktop experience) */
        response.sendRedirect(DESKTOP_URL);
    }
```

2. Templates

In this technique the website logic and content remains the same for all requests regardless of the device and browser.

However, each page can present its response based on several templates and styles, with each template designed to serve content for a specific type of device, browser or any other device property. RWD can be implemented within every template to fine-tune the experience within certain sets of resolutions.

The API is used to get device properties and select the best template available for the properties. This technique is very easy to implement and especially suitable for making existing solutions device aware.

```
    /* At this point the processes required for the page
    have been done and the output content has been
    prepared. We have to select a template to warp
    around the contents. Let's assume method
    "loadTemplate()" loads a template file. */

    // get the API instance from servlet context
    DeviceApiWeb deviceApi = (DeviceApiWeb)context.
    getAttribute("DeviceApiWeb");
```

```
// continued from previous page

// get the properties from the current request
object
Properties properties = deviceApi.
getProperties(request);

// it’s a tablet device
if (properties.contains(“isTablet”, true)) {
    loadTemplate(“templates/tablet.tpl”);
    return;
}

// it’s a mobile device
else if (properties.contains(“mobileDevice”, true)) {
    /* you can create conditions on various properties
    to distinct between low-end and high-end devices
    */

    /* low-end devices which only support WML but
    not basic XHTML receive contents wrapped in
    WML */
    if (properties.contains(“markup.wml1”, true) &&
    properties.contains(“markup.xhtmlBasic10”,
    false)) {
        loadTemplate(“templates/wml.tpl”);
    }

    // it’s a E-Reader device
    else if (properties.contains(“isEReader”, true)) {
        loadTemplate(“templates/ereader.tpl”);
    }
    // it’s a high-end mobile device
    else {
        loadTemplate(“templates/mobile.tpl”);
    }
    return;
}
    // it’s a spam
    else if (properties.contains(“isRobot”, true)) {
        out.println(“output something like the site-map
        for the robots”);
    return;
}

// the default template is set to desktop browsers
loadTemplate(“templates/desktop.tpl”);
```

A sample template code:

```
<!doctype html>
<html>
    <head>
        <title>Title</title>
        <meta http-equiv="Content-Type"
        content="text/html; charset=UTF-8" />
        <meta name="viewport"
        content="width=device-width, initial-scale=1.0"
        />
        <link type="text/css" rel="stylesheet"
        href="css/tablet.css" media="all" />
        <%
            if (properties.contains("touchScreen", true))
            {
                out.print("<link type=\"text/css\"
                rel=\"stylesheet\" href=\"css/touch.
                css\" media=\"all\" />");
            }
        %>
    </head>

    <%
        String displayWidth = "1024";
        if (properties.containsKey("displayWidth")) {
            displayWidth = properties.
            get("displayWidth").asString();
        }
    %>

    <body style="max-width:<%=displayWidth%>px">

        <h1>Tablet Device Experience</h1>
        <div id="contents" class="clearfix">
```

Contents:

```
<%
            if (properties.contains("html.video", true)) {
                out.print("<p>This browser supports the
                video HTML tag, we can use it to show a
                video here</p>");
            }
```

```
// continued from previous page

            if (properties.contains("flashCapable",
            true)) {
                out.print("<p>Flash is supported, so we
                can display flash here</p>");
            } else {
                out.print("<p>Flash is not supported
                </p>");
            }

            %>

        </div>

    </body>
</html>
```

3. Dynamic Adaptation

In this technique, a website provides a service or serves content that changes depending on the device.

For example, the website could feature GPS-based location specific options, but display them only on devices with GPS on board. Another example would be to display mobile app download options for different operating systems, such as Android, iOS, Windows Mobile, etc. Device properties can be used to detect the OS and then only show the specific download for that operating system.

```
/* Example: Find a suitable app download link for the
device */

// get the API instance from servlet context
DeviceApiWeb deviceApi = (DeviceApiWeb)context.
getAttribute("DeviceApiWeb");

// get the properties from the current request object
Properties properties = deviceApi.
getProperties(request);

// all available download links for our app
Map<String, String> allDownloadLinks = new
HashMap<String, String>();
```

```
// continued from previous page

allDownloadLinks.put("Android",
"#download-android-app");
allDownloadLinks.put("iOS", "#download-ios-app");
allDownloadLinks.put("RIM", "#download-rim-app");
allDownloadLinks.put("Windows Phone",
"#download-windows-phone-app");

// available links will be in downloadLinks
Map<String, String> downloadLinks = null;

// if osName is detected
if (properties.containsKey("osName")) {

    String osName = properties.get("osName").
    asString();

    // try to find desktop OS names
    if (osName.contains("android")) {
        osName = "Android";
    } else if (osName.contains("ios") || osName.
    contains("apple")) {
        osName = "iOS";
    } else if (osName.contains("win")) {
        osName = "Windows";
    }
    // get the download link for the OS
    String link = allDownloadLinks.get(osName);
    if (link != null) {
        downloadLinks = new HashMap<String,
        String>();
        downloadLinks.put(osName, link);
    }
}

// if operating system is unknown show all links
if (downloadLinks == null) {
    downloadLinks = allDownloadLinks;
}

/* Now we can display "downloadLinks" in the page
template. */
```

‘CLASSIC’ RWD OR RESS?

Responsive Web Design is a set of design principles and techniques that allow a website to be flexible enough to work well at changing resolutions. Responsive content rearranges itself according to the screen’s resolution and orientation.

In its ‘classic’ client-side version, the same HTML code is sent to every device and then CSS alters the rendering so that the website is displayed differently according to the screen size.

Responsive Web Design achieves screen width independence, but in its ‘classic’ version it actually doesn’t provide the user with an experience specifically optimized for mobile given that the rearrangement of content is based on the browser’s width.

Server-side adaptation techniques such as Templates and Dynamic Adaptation don’t preclude the use of RWD. This is a technique called RESS (Responsive Web Design with Server Side components). It offers a way to get the best of RWD and server side device recognition, which results in significant performance improvements over ‘classic’ RWD. You can find out more in our e-book on RESS.

KEY TAKEAWAYS

This content adaptation guide indicates that a 'one-size-fits-all' approach is not always the best option for your mobile strategy. It is possible to come up with a fine-tuned content adaptation approach, providing your users with an exquisite experience that is specifically optimized for mobile.

Here's 7 key takeaways from our content adaptation guide.

1. The growing capabilities of mobile devices have increased users' expectations for a top-notch mobile experience.
2. The consequences of below-par mobile optimization can damage a brand's online presence.
3. Addressing mobile users only with RWD may not be the best option for your business needs.
4. Content adaptation requires accurate server-side device recognition.
5. Device detection solutions such as DeviceAtlas provide you with a list of device properties for precise targeting.
6. Content adaptation can be based on buckets of users who are provided with targeted mobile experience based on templates and/or dynamic adaptation.
7. RESS allows you to combine RWD with server-side device detection components.

Why not add device awareness to your website?

DeviceAtlas can power your content adaptation decisions by providing you with accurate device intelligence in real time.

DeviceAtlas is the most accurate and comprehensive device data solution on the market. It is sourced from multiple industry-leading partnerships ensuring the most accurate detection rates.

Try it for free