intelliware
software development

Putting Mobile Development With HTML5 Into Perspective

Best practices, pitfalls and how to ensure your mobile app succeeds

Introduction

Mobile development using HTML5 has gone mainstream.

It's not hard to understand the motivations driving many enterprises to embrace HTML5-based mobile apps. The promise of HTML5 is that it solves a number of different problems currently being experienced by enterprises attempting to offer mobile solutions.

First, mobile developers are a scarce resource. The demand currently exceeds the supply, and many of those developers gravitate toward start-ups and other companies that work with leading-edge technology.

Second, the continuing battle for market share in the smartphone space means that most organizations care about supporting at least two, possibly more mobile application platforms. It's not enough to have an iPhone app; consumers expect an Android version of the app, and possibly a BlackBerry version as well. Developing native apps on these platforms allows for very little in the way of code reuse – the platforms are too different. As a result, supporting multiple platforms can often mean three times the application development cost.

Third, many organizations involved in transactional data have already invested in large data centres for high-volume user traffic – these organizations want to leverage that investment for mobile development.

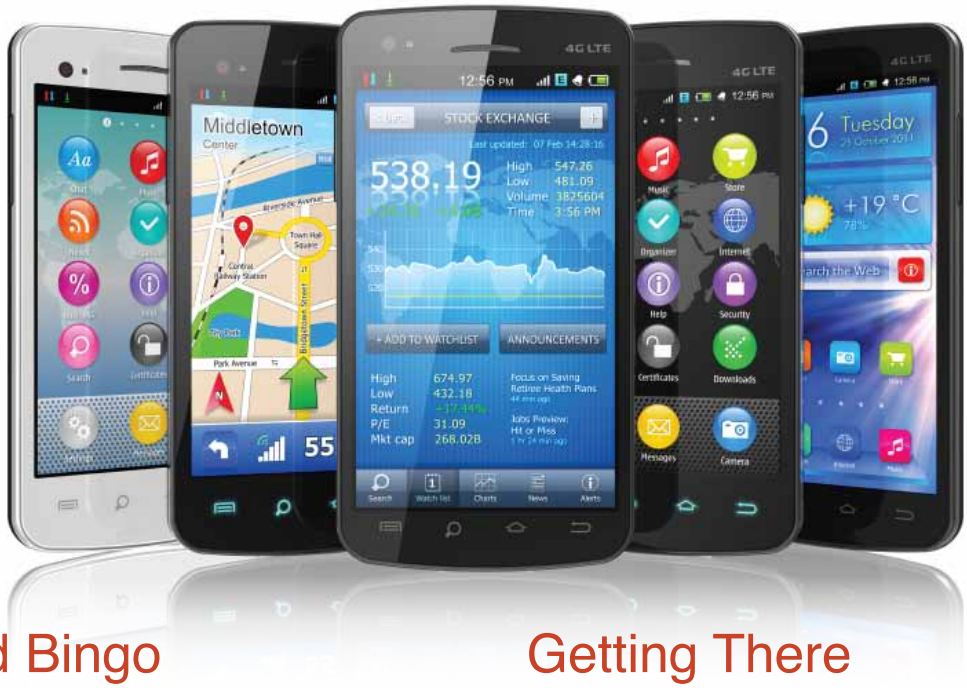
HTML5 (and the related technologies that make up this kind of solution, including CSS and JavaScript) can be used to build applications that assuage these concerns.

The web application development industry has ensured a large pool of HTML/web technology developers.

Because all three major mobile platforms allow for embedded web components, HTML5 components (or perhaps entire applications) can be shared across those platforms.

Although a basic, self-contained HTML5 app can be built with HTML, CSS and JavaScript, many enterprises want some kind of host connectivity which makes use of their existing servers.

In the IT world, there is considerable debate about the merits of HTML5 versus native app development. This paper is not intended to advocate for one approach over another, as we believe both approaches have their merits. Rather, the thesis of this paper is as follows: if you plan to build an HTML5 mobile app, you should view it as very different from a traditional web application. Failure to recognize this difference will lead to a mobile application that has poor performance, lacklustre user experience, and may well be unstable.



Buzzword Bingo

Like "Web 2.0", the term "HTML5" is as much a marketing term as a description of a technology. Speaking as narrowly as possible, HTML5 refers to the latest edition of the HTML standard, which includes some considerations for HTML rendering on mobile devices.

What HTML5 has come to mean, more colloquially, is the use of advanced features of HTML, CSS and JavaScript to create a number of effects (including, for example, animations) that emulate the types of behaviours that are typical of iPhone and other modern mobile applications. These effects include binding actions to mobile-type interactions including touching and swiping, transitions between screens such as sliding or popping, and animated ways of communicating common concepts like "the item has been deleted."

Part of the marketing message of HTML5 is that it allows developers to use their existing web skills to create mobile apps. It's not that that message is untrue. But the message doesn't fully communicate that to do HTML5 correctly, you need above-average skill with HTML, JavaScript and CSS.

Getting There

Although it's possible to create an HTML5 mobile application from scratch, in truth, few people do this. The majority of HTML5 mobile apps are created using one or more mobile development frameworks. Here are some of the most common ones:

jQuery Mobile
jQuery Mobile is an extension of the popular jQuery JavaScript library – it uses small amounts of non-standard HTML mark-up to indicate that certain HTML constructs should be rendered in ways that resemble common mobile application widgets such as lists and toggles. For example, a standard HTML unordered list ("<ul>") can be annotated to indicate that it should resemble an iPhone UITableView.

Sencha Touch
Sencha Touch is a JavaScript library that creates mobile user interface controls. Sencha Touch uses less HTML and CSS, favouring JavaScript and a non-standard (but not uncommon) CSS-like styling language called SASS.

PhoneGap
PhoneGap's focus is different than jQuery Mobile and Sencha Touch. It provides a container in which HTML5 assets can reside. While it can be used in many different ways, it's ideal for apps that are wholly contained on the device. It is not uncommon to combine PhoneGap with one of the other mobile frameworks, since PhoneGap provides the container, and the other framework provides the controls and/or themes.

Dojo Mobile
Similar to jQueryMobile being a JQuery extension that emulates mobile UI components, Dojo Mobile is a Dojo extension that emulates mobile UI components.

A Case Study

Recently, Intelliware worked with a financial services client. The client had previously engaged a development partner to build new functions on their existing mobile banking app. Unlike the existing functions, the new features were to be developed using HTML5 with content served up from a .NET web infrastructure.

When Intelliware was engaged, the client was dealing with a project that was failing from a technical perspective. What follows in this case study are specific reasons about why the project was in trouble, how to avoid the pitfalls this client faced, and what was required to get the development back on track. The key learning for organizations from this story is they need to recognize that Mobile development, even using the well-understood web technologies such as HTML, JavaScript and CSS, requires a different approach to development.

HTML5 Development Considerations

Let's consider a few examples of typical technical design considerations. Each of these examples attempts to show that the patterns and ideas that are considered acceptable for browser-based web applications can lead to mobile applications that are slow, unstable, or which download and transmit in a way that yields a sub-optimal user experience.

Transitions

Understanding how jQuery transitions work is a good way to understand why building HTML5 mobile applications are more complex than traditional web applications. The first thing to understand is that jQuery-Mobile uses custom attributes on HTML documents to indicate what parts of the document can be considered a "page".
Consider the following example:

```
<html>
      <body>
            <div id="page1" data-role="page">
                  ...
            </div>
            <div id="page2" data-role="page">
                  ...
            </div>
      </body>
</html>
```

jQuery-Mobile will process this HTML document by rendering "page1" on the device. "page2" will remain hidden. A transition between those two pages can be easily implemented by, for example, clicking on an anchor such as this:

```
<a data-role="button" data-transition="slide" href="#page2">Next</a>
```

jQuery-Mobile will respond to this click by sliding the new "page" ( ) on top of the original "page", and then marking the new page as the "active page". Part of the selling feature of jQuery-Mobile is that such page transitions are easy to implement, and require essentially no special JavaScript knowledge for the implementer.

Transitions between pages in different documents, on the other hand, become slightly more complicated. The slide effect – popularized by the iPhone – is very different than the traditional HTML transition behaviour when two different documents are concerned. And yet, it is precisely this kind of "native-like feel" on which users will judge the acceptability of the transitions.

Under normal circumstances, standard browsers transition from one HTML document to another HTML document by clearing the screen and loading the new document. Most mobile HTML5 apps want to avoid this kind of transition, because it broadcasts to the user that the application is web application-like rather than a mobile application-like.

"Part of the marketing message of HTML5 is that it allows developers to use their existing web skills to create mobile apps. It's not that that message is untrue. But the message doesn't fully communicate that to do HTML5 correctly, you need above-average skill with HTML, JavaScript and CSS."

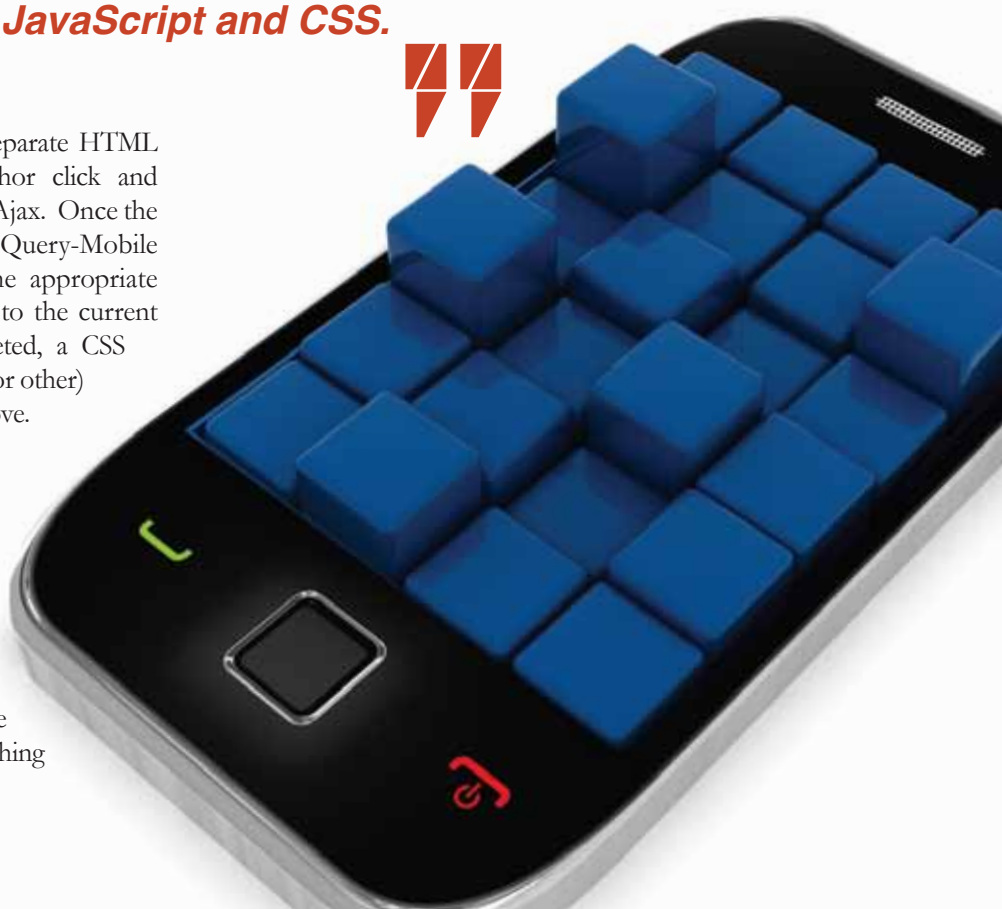
So, to support native-like transitions between separate HTML documents, jQuery-Mobile intercepts the anchor click and fetches the new document asynchronously using Ajax. Once the new document has been downloaded, the jQuery-Mobile framework scans the new document, finds the appropriate "data-role='page'" item and appends that item to the current document's DOM model. Once that's completed, a CSS animation can be used to enable a slide or popup (or other) transition between one "page" and another, as above.

This is a powerful feature of jQuery-Mobile. jQuery-Mobile makes it easy for developers to create native-like transitions from one HTML document to another. The example used above, relates to simple clicking from one page to the next. But the topic extends to many types of page navigation, including FORM entry and POSTing: not only should the POST-redirect-GET patterns be eschewed, anything that causes a full-page reload should be avoided.

Some of the consequences of the power of JQM transitions, however, warrant discussion. The first, most important consequence relates to the management of resources.

Imagine there are two HTML documents: firstPage.html and secondPage.html. When jQuery-Mobile is used to implement a transition from one page to another, jQM extracts a portion of secondPage.html and includes that in firstPage.html. Now, if secondPage.html needs a particular CSS stylesheet or JavaScript dependency in order to function properly, then firstPage.html must also reference those styles and JavaScript utilities (whether or not it needs them for itself). In the worst case, all pages in a jQuery-Mobile application must refer to all header assets so that the pages will function properly after a jQM transition. And in some cases, that can be a lot of code to download.

Now, obviously, there are limitations to this transition model. Cross-site scripting prevention features built into browsers make jQM transitions impossible between pages hosted on different domains. If, for example, you're building an RRSP investment app, and you want to provide access to RRSP prospectuses



hosted on another domain, jQM transitions aren't going to work out of the box.

The next implication is more concerning: it involves the complexities of writing JavaScript in a jQM environment. Here's an aspect of the implication that's relatively easy to describe: managing the JavaScript namespace.

Mobile developers working on HTML5 applications that serve up content from the web cannot afford to be slapdash about performance optimization.

Although global variables are frowned upon by leading JavaScript developers, many, many web applications continue to use "global variables" because they're not really global in most web applications. Most web applications have "global variables" that are really limited to the page's scope. Using the example, above, if firstPage.html has a global variable called "status" and secondPage.html has a global variable called "status", in a traditional web application, these two variables never collide because their "global scope" is still limited to the page. When firstPage is unloaded from the browser, the "status" variable disappears, and when secondPage is loaded, a new "status" variable is in scope.

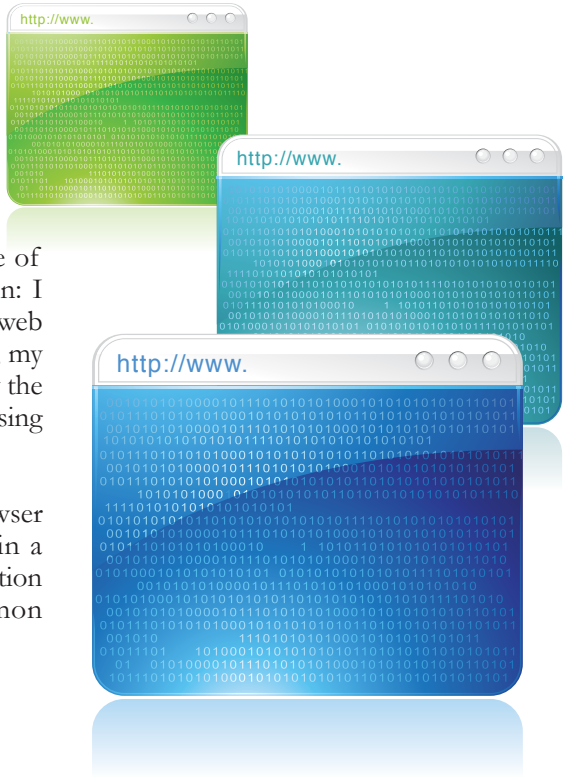
In an application using jQM transitions, HTML documents aren't loaded and unloaded in the same way. A global variable used by firstPage.html continues to be in scope after jQM transitions to secondPage.html. This is not an insurmountable concern. But even recognizing that the concern exists requires a certain degree of JavaScript aptitude and attention to detail that most web application developers can get by without.

This concern about global variables can be generalized to many different aspects of the JavaScript environment. jQuery and jQuery-Mobile applications usually involve binding a number of handler functions that will be invoked on certain events, such as when a user taps on a control. It requires skill and discipline to ensure that these events are registered at the right time, and not registered multiple times if the same page is transitioned to more than once. Not only might multiple registrations have undesired consequences, but too many event registrations can easily bring a mobile HTML5 app to a crawl.

The crux of this issue is this: the behaviours and practices that typical developers employ to build standard web applications are not good enough to build HTML5 mobile applications. To do this right, you need better-than-average skill with the HTML5 toolset.

intelliware
software development

Up In the Air



One of the examples that we find ourselves frequently citing is the example of the HTML5 web application operating in an airport. Imagine this situation: I have a mobile app that serves up HTML5 content from the app-provider's web servers. If I launch this app immediately after connecting to the airport wifi, my app's request for the first HTML screen will be redirected to a page hosted by the airport wifi provider asking me to accept their terms and conditions for using Internet access. After confirming, I'm redirected to an airport home page.

When this kind of situation happens in a browser, I have access to the browser URL bar, and I can call up a different URL. But when this happens in a server-based HTML5 mobile app, the user usually has no control over navigation and can't get back to the app's functionality. This is a painfully common circumstance, and yet most HTML5 mobile apps get tripped up on it.

Performance

Web sites that are visited by millions of users every day are heavily tuned for performance. But for many sites on the web, performance can be considered "good enough" without much work.

The web hasn't always been like this. In the earliest days of the Internet, web designers were expected to track a lot of information about how to optimize each page for web display.

In his presentation, "Going Fast on the Mobile Web", Jason Grigsby argues that we've left optimization behind, and instead have become gluttons for bandwidth. Because of increases in download speeds and processing power, the early fastidiousness about site optimization has become a lost art. He argues that the average modern web page is now 20 times the size of the average web page of 1995.

This is salient because the mobile web more closely resembles the early days of the Internet than it resembles the modern web. Even though we're becoming amazed at the power of smart phones, it cannot be underscored enough that they have slower CPUs, slower JavaScript engines and slower download speeds.

That being said, mobile developers working on HTML5 applications that serve up content from the web cannot afford to be slapdash about performance optimization.

Now, it's not like the art of website performance is esoteric. As mentioned earlier, the most popular websites need to focus on performance optimization in order to satisfy millions of requests per day. Grigsby points to Yahoo's key performance tuning recommendations.

Some of Yahoo's recommendations are more applicable to mobile than others, but a good mobile HTML5 developer needs to be fluent with these optimization techniques, and needs to consider many of those techniques as the application is being developed.

From the list of 35 techniques Grigsby advocates, there are eight key recommendations for mobile developers:

- Turn on Gzip Compression
- Fewer files
- Aggressively encourage caching
- Minimize file sizes
- Simple, valid markup
- Reduce DNS lookups
- Avoid redirects
- Limit cookies

Applications with a non-trivial native shell can additionally serve up certain assets directly from the app. Mobile application developers need to be able to speak about each of these optimization strategies fluently.

Scrolling

In many ways, scrolling can be seen as a specific sub-category of performance, but it's an especially crucial behaviour to talk about because users' expectations of a native-like experience really come to the fore in scrolling. In many ways, scrolling is the behaviour that makes or breaks a user's acceptance of your application as an acceptable mobile experience.

There are a number of "scrolling libraries" available on the web – scrollability, touch-scroll, iscroll and iscroll-lite among others. The essential function these libraries offer is an easy ability to construct HTML5 pages that have the three common "regions" of most mobile screens: a fixed header, an optional fixed footer and a scrollable "middle area." (The importance of these libraries may lessen as the need to support iOS 3 and 4 disappears).

But "easy ability to construct" is clearly a relative notion. Returning to our case study, we were engaged by our client to work with their original development partner. That partner had already chosen a scrolling library – touch-scroll. When we came on board, the application was plagued by numerous scrolling bugs: the scroll area wouldn't scroll; its height wouldn't change as new content was added to the middle area. Again, this was an example of a development team with only average skills being unable to create a good mobile user experience.

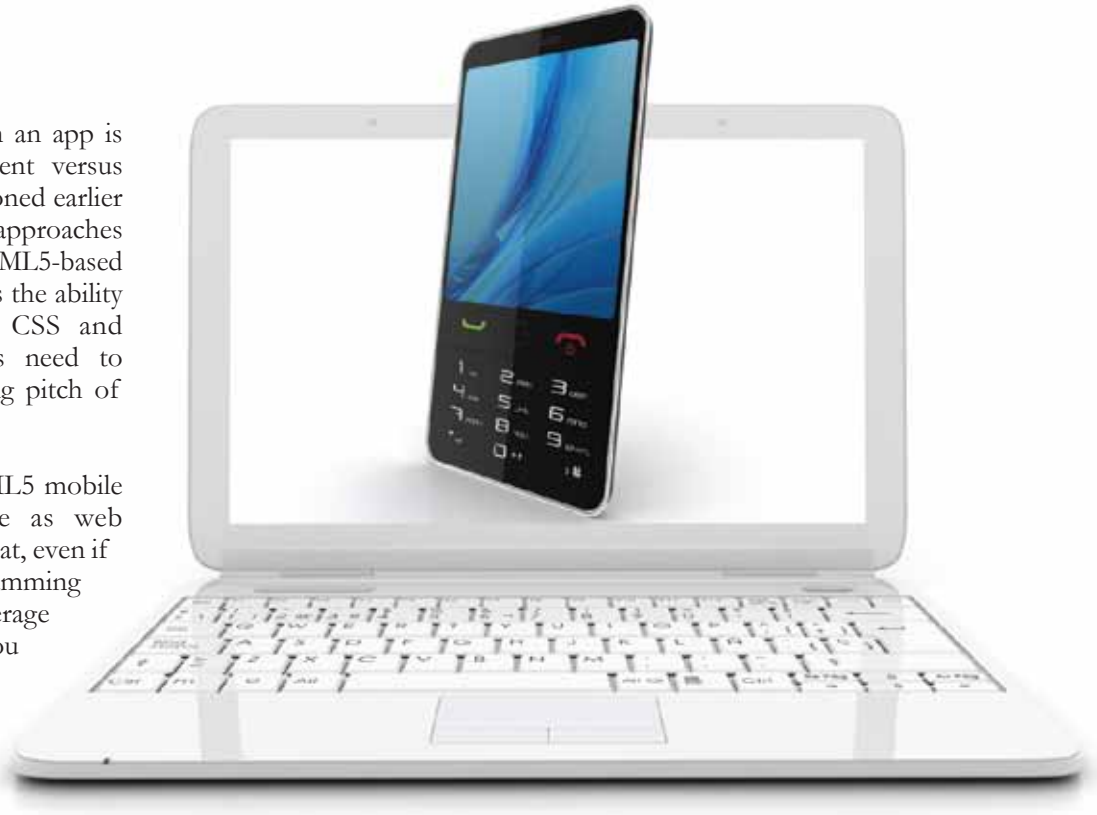
Even after we fixed the scrolling bugs, the scrolling experience wasn't completely acceptable to the client: the feel of the aggressive "flick" that causes a rapid scroll wasn't sufficiently native-like. So we spent time having a bake-off of a number of different libraries before settling on iscroll-lite. One last scrolling concern required us to really understand the way that jQM events interacted with controls as rows whizzed by the screen.

In one way, it's a bit demoralizing knowing that we had to spend so much effort getting the HTML5 scrolling to feel as native as possible. Demoralizing, especially, given that scrolling requires essentially no effort if we'd implemented in native. But the more significant point remains: we could not have come close to a native feel if we were just consumers of frameworks. We needed to be skillful JavaScript developers to make it work.

Conclusion

The debate rages on about when an app is best suited to native development versus HTML5 development. As mentioned earlier in this paper, we believe that both approaches have value. While it is true that HTML5-based mobile apps allow web developers the ability to use their existing HTML, CSS and JavaScript expertise, enterprises need to distinguish between the marketing pitch of HTML5, and the reality.

The marketing pitch is that HTML5 mobile development is just the same as web development. But the reality is that, even if you're using familiar web programming technologies, you can't rely on average web development practices if you want to make an acceptable mobile application of even moderate complexity.



intelliware
software development

About the Author

BC Holmes is an IT architect at Intelliware Development with 20 years of experience designing and building applications, often working with new technology trends. She was the architect and technical lead for the team that implemented the first web banking application in Canada, and now she works on multiple mobile applications for banking clients. She has worked in the IT Services industry and the financial services industry, and often specializes in e-Health and HL7-enabled applications. BC has spoken at many conferences and user groups. She holds a Joint Honours in Pure Mathematics and Theatre Arts.

About Intelliware Development Inc.

Intelliware is a custom software, mobile solutions and product development company headquartered in Toronto, Canada. Intelliware is a leader in Agile software development practices which ensure the delivery of timely, high quality solutions for clients. Intelliware is engaged as a technical partner by a wide range of local, national and global organizations in sectors that span Financial Services, e-Health, ICT, Retail, Manufacturing, and Government.

Intelliware placed among the Top 5 Mobile Technologies Companies in the 2012 Branham300 report, the definitive listing of Canada's Information and Communication Technology (ICT) industry leaders, as ranked by revenues.

For more information, visit www.intelliware.com.

1709 Bloor Street West
Suite 200
Toronto, Ontario
M6P 4E5, Canada

416.762.0032

200 Adelaide Street West
Suite 100
Toronto, Ontario
M5H 1W7, Canada

416.916.3457

