MAML Web Translation

Simplifying web pages for mobile users in developing regions of the world

Syed Faizan Haider Zaidi (Student)
New York University Abu Dhabi
Yasir Zaki (Supervisor)
New York University Abu Dhabi

Abstract

In the developing regions of the world, the internet is faced with many challenges: high page load times due to complex web pages, lack of local content, lack of infrastructure, and an apathy by content providers to tailor their content for mobile use [1][13][20]. This project is aimed at resolving some of the problems that exist at the application level on the client's side i.e. the web page. In order to reduce the page load time and hence improve the web experience of users, we propose the MAML Web Translation project. MAML, Mobile Application Markup Language, is a newly proposed web specification language, primarily for mobile users in developing regions [29]. We translate existing HTML web pages into MAML pages, and allow content providers to recreate their content in a way that is simple and quick to render, and caters to the needs and preferences of each user. We have created the MAML Auto-Translation engine that translates web pages from HTML to MAML, reducing the page size, the number of objects needed to create the web page and the page load time, while maintaining and preserving the aesthetics and look of the page. The MAML Editor Tool is a web application, designed for the purpose of content creation using MAML, allowing content providers to create and publish web pages using simple drag-and-drop features. We have managed to effectively translate web pages, reducing their size, number of objects and load times, while maintaining their look and interface.

1 Introduction

The internet, over the last few years, has become more complex and popular [1][2]. Data is being created and consumed unlike ever before, and our desire to connect and interact with the world grows stronger. As of 2019, the number of web pages available on the public internet is 5.5 billion, and the number of people with access to those pages has grown to 4.4 billion [31]. Being online however, is an experience, one that users want to be seamless, quick and efficient. If a web page takes longer than 3 seconds to load, more than half of users will stop visiting that page [32]. We would therefore expect page load times (PLT) of web pages to be close to that benchmark. The reality however is much different. In 2018, it took on average 9 seconds for a web page to load on desktop, while on mobile, that time was 22 seconds [33]. This is the situation in developed countries, where users have access to smartphones with powerful processors and sufficient memory, as well as good network connectivity. In the case of developing countries, where access to the internet is primarily through low-end mobile phones, page load times are as high as 60 seconds [5][15][36]. Despite the importance of the web in our lives today, a significant fraction of the world's population does not have proper access to it.

1.1 Background

The poor performance of the web in developing regions has been a subject of extensive and continuous study by scholars. A variety of angles have been studied, ranging from core infrastructural issues to geographical locations [4][5]. For example, recent studies found that the "lack of good caching infrastructure and DNS servers are the primary causes of poor performance" [6][5]. Another related study has shown CDN server placements and routing protocols to be primarily responsible for performance issues [4]. This is especially problematic in the context of modern web pages. Modern web pages have evolved significantly over the past couple of years into a very complex ecosystem with large number of objects that span several servers across the globe [30]. Butkiewicz et. al. has shown that more than 60 percent of web pages request data from at least 5 different non-origin sources, and that these contribute to more than 35 percent of the overall page size [1]. Loading one web page can involve downloading hundreds of objects [1], creating dozens of network connections [11][12], issuing several DNS requests [12] and handling multiple recursive calls by JavaScript and HTTP redirections [1]. These objects essentially form a complex object dependency graph. As these objects are downloaded and evaluated, they trigger the download of other objects. This causes unpredictability in the object dependency graph of a web page, causing delays in rendering the page [13].

This problem is further amplified for mobile phone users in these regions, as Shoaib Ahmad et al. suggests. They analyzed the characteristics of cell phones based on different features (e.g., CPU, memory, and cellular interface) to identify potential device-level bottlenecks for internet access [7]. Their research into mobile phones was motivated by the immense popularity of mobile phones in developing regions. The International Telecommunication Union (ITU) reported that mobile-broadband subscriptions were expected to reach 4.6 billion by the end of 2017, with 60 percent of them in developing countries [3]. Mobile devices in developing regions are therefore very popular and the primary source of accessing the Internet, being used for a variety of services including agricultural information dissemination, education, and health care delivery [8][9]. While the Internet infrastructure has been steadily improving in developing regions and several efforts (e.g., Google's Project Loon and Facebook's Connectivity Lab) focus on providing Internet connectivity via satellites, balloons, drones, and planes, anecdotal evidence suggests that the common use of low end devices with a slow Internet connection can lead to poor user experience in developing regions [10]. Research by Shoaib at al. into the characteristics of mobile phones in developing regions offers important insights; mostly low end phones, and not the ones tested on by developers in developed regions, are in use in these regions [7]. These mobile phones are not capable of handling the increasing complexity of web pages and can often lead to poor performance for web browsing and web-based multimedia applications [4][20]. Furthermore, a significant fraction of phones (49.5 percent) support only wireless application protocol (WAP) browsers. Unlike modern web browsers (e.g. Safari or Chrome), a WAP browser only supports a stripped down version of XHTML and uses the WAP protocol for accessing the Internet. However, it does not support JavaScript, the primary language of the web and a significant source of web page complexity and loading delays [7].

1.2 Existing implementation of Web Pages

1.2.1 Understanding the DOM

Let us examine the processes a browser (a client) undergoes when loading a web page, from the time the request is sent to the server to the time the page becomes interactive. There are essentially three main stages in this stack. First, a request is sent from the browser to the server, through cellular towers (in the case of mobile phones), DNS servers, data centers and much more, after which the server responds with an HTML file (index.html). The browser then examines and parses the HTML file, creating the document object model (DOM) structure, followed by requesting the critical assets needed to render the page. This mainly includes any CSS and JavaScript files, and graphic content such as images and videos. Once the critical assets have been imported into the browser, the browser then needs to parse and execute the JavaScript files. The execution of JavaScript files can happen synchronously and asynchronously. Mostly synchronous, the execution of JavaScript causes the browser to pause its parsing of HTML, as the JavaScript may cause changes to the DOM upon its execution [20]. Moreover, the JavaScript files recursively trigger the download and execution of more such files, leading to the formation of a complicated and deep object dependency graph [21][22]. This computation is a significant factor that makes up as much as 35 percent of the critical path, the deepest subtree in the DOM. Synchronous JavaScript therefore, plays a significant role in page load time by blocking HTML parsing [20]. Consequently, this entire process consumes a significant amount of time. The time required to load a web page therefore, will only increase as the complexity of web pages also increases.

1.2.2 Problem with HTML and CSS

HTML, or Hypertext Markup Language, invented in 1991, was intended as a simple format to create and share enriched static documents, which could refer to one another through hyperlinks.

The original design was fairly simple, containing only 18 elements. As of today, the number of HTML elements has grown to 108, while the HTML specification document is more than 1200 pages long [35]. It is therefore not a surprise that HTML is growing fast. In fact, too fast. Developers want HTML to support their latest features, given the competitive nature of the internet, and as a result, there are complexities and redundancies. It becomes difficult for stable browsers to follow the rapid pace of development, leading to different HTML implementations and support on different browsers.

The situation further complicates when we take into account CSS. CSS, or Cascading Style Sheets, is used to generate styling for HTML. CSS however, has also increased in size, complexity and contains many redundancies. Animations, for example, are now supported in CSS, which can easily and effectively be implemented in JavaScript. Moreover, CSS blocks rendering and loading of the page, and tends to get bloated. Every time the CSS file needs to be accessed, one or more HTTP requests need to be made. This is especially problematic if such requests are made in the rendering of the critical path.

As a result, the DOM becomes unnecessarily complex with a large critical path.

1.2.3 Cost of JavaScript

Over the last few years, JavaScript has become very popular among web developers due to the advent of frameworks. Instead of writing web pages in HTML, developers use JavaScript frameworks to create and manipulate HTML and the DOM. The manipulation of DOM by JavaScript however, is computationally very expensive. This becomes exacerbated, and therefore more evident, on low-end phones with weak connectivity. Loading and executing JavaScript can take anywhere between 50 to 85% of the entire PLT [36]. When compared to high-end phones, low-end phones take 400% the time to load web pages, resulting in PLT as high as a minute.

Suggestions have been made to render HTML on server side, and limit the use of client-side frameworks to pages that absolutely require them [36].

1.3 Suggested Solutions

We can therefore safely assume that in order to enhance the experience and performance of web pages in developing regions, we need to focus on simplifying the DOM and minimize our dependence on HTML and JavaScript. Before we begin discussing the need for, and features of, our solution, let us take a look at some of the solutions proposed in academia and the web development community to tackle the challenge of increasing page load times. We will then see how this project builds upon and compares to those solutions.

Several solutions have been proposed, and some implemented, to overcome this challenge. A system named Klotski suggests re-organizing contents of a web page relevant to a users preferences, leading to fast selection and load-time estimation [13]. Another system, Shandian, proposes restructuring the web page load process and exercising control over what portions of the web page get communicated and in what order [23]. Polaris is another suggestion, which suggests a way to enable fast loading of the page by using fine-grained object dependency graphs to load the objects in the optimal way [21]. Google introduced AMP HTML, an extended version of the existing HTML, which includes tags and elements that are used to manage resource loading [24]. Facebook Instant Articles is a platform within Facebook's mobile application, for publishers to create fast and interactive articles. Facebook claims Instant Articles makes the loading of articles ten times faster than loading of standard mobile articles by preloading the data[25]. Opera Mobile browser and its lower-end sibling, Opera Mini browser, request web pages through a proxy server called Opera Turbo, compressing web pages by about 90 percent rendering them faster by two to three times[26]. What most of these proposals have in common is the need to rewrite and reform the contents of a web page, while preloading or precompiling it. The problem however with these suggestions is their reliance on HTML and JavaScript, which this paper intends to avoid.

1.4 Aims

In this paper, we take a closer look into the structure and design of web pages, especially from the perspective of mobile phone usage. We aim to build upon some of the proposed solutions discussed, and offer a solution of our own, one that aims at minimizing the complexity of web pages in the form of MAML, a proposed web specification language, thereby reducing the PLT. We will demonstrate how MAML can be used to simplify web pages by translating existing HTML pages to reduce their size and number of objects. We will also demonstrate how it can be used by content providers in developing countries to generate content that caters to the needs and concerns of users in their regions, while at the same time, maintaining the aesthetics of their websites. To do so, we provide a tool to derive a significantly lighter version of the original page that loads much faster for bandwidth constrained users in the developing region. We do this by pre-compiling web pages using the MAML specification. The MAML version of the page includes most of the properties of the page objects including the relationships among objects, and their relative layout structure. MAML aims at overcoming the complex object dependency tree that exists in HTML. It reduces the number of objects that the browser needs to download from the network. MAML derives a simpler, leaner and more comprehensible version of a web page that reduces the object dependency graph of a page to a flattened tree structure. As a result, most of the computations are performed on server-side, reducing the load on the client, and there is no blocking by, and waiting for, JavaScript. As a result, we hope to reduce the size of the page, as well as the PLT.

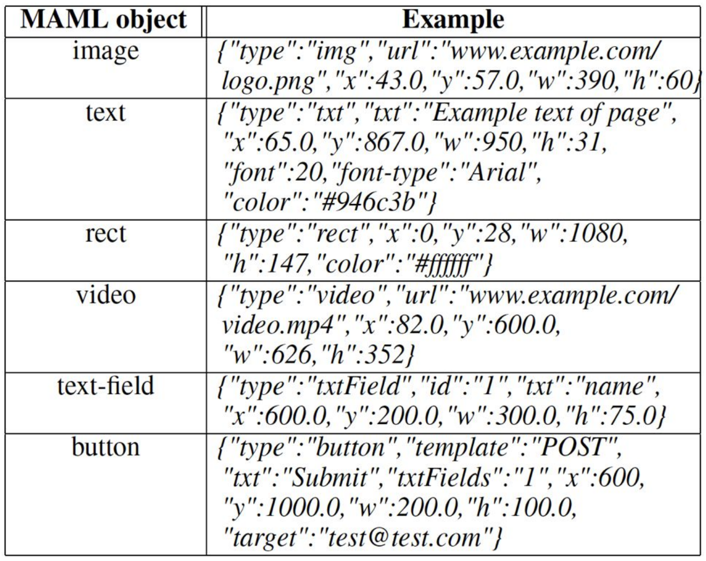
2 Methods and Approach

This project is essentially divided into two parts: MAML Auto-Translation and MAML Content Creation. The Auto-Translation translates existing HTML pages to MAML format, and then renders them in a way that is easily viewable. The purpose of Auto-Translation is to show the effectiveness of MAML in reducing page size and PLT, when compared to HTML. In the future however, we want content providers, especially in developing regions, to create web pages using MAML. To promote that, we have worked on creating a content creation tool in MAML, called the MAML Editor Tool.

Before we discuss the implementation of these two sub-projects, it is imperative that we gain a more concrete understanding of MAML itself. We shall take a look at the MAML specification and some of the objects described by it.

2.1 Structure and Specification of MAML

To realize MAML, we need to have a mapping from HTML objects to MAML, as HTML is still widely used as the language of the web. Therefore, to translate HTML pages into MAML, we need to have a concrete understanding of the MAML specification. As shown in Table 1, the current MAML specification supports six different object kinds: image, text, video, rectangle, text-field, and button. Let us take the image object as a case study. In MAML, the image object represents the mapping from the HTML "img" tag. The URL, which mirrors the "src" attribute in HTML, is the source of the image i.e. where the image is hosted. It also has x and y coordinates, width, height, and an optional hyperlink - in case the image is clickable - as attributes.



MAML object	Example
image	*{"type":"img","url":"www.example.com/logo.png","x":43.0,"y":57.0,"w":390,"h":60}*
text	*{"type":"txt","txt":"Example text of page", "x":65.0,"y":867.0,"w":950,"h":31, "font":20,"font-type":"Arial", "color":"#946c3b"}*
rect	*{"type":"rect","x":0,"y":28,"w":1080, "h":147,"color":"#ffffff"}*
video	*{"type":"video","url":"www.example.com/video.mp4","x":82.0,"y":600.0, "w":626,"h":352}*
text-field	*{"type":"txtField","id":"1","txt":"name", "x":600.0,"y":200.0,"w":300.0,"h":75.0}*
button	*{"type":"button","template":"POST", "txt":"Submit","txtFields":"1","x":600, "y":1000.0,"w":200.0,"h":100.0, "target":"test@test.com"}*

Table 1: An example of the specification of MAML objects

In MAML, positioning of web objects or elements is absolute, unlike relative layout in HTML. The two-dimensional Cartesian coordinate system is used for positioning them on the web page, with the top-left corner of the document being the origin.

Having gained an understanding of MAML objects and their specification, we can now attempt to translate HTML pages to MAML.

2.2 MAML Auto-Translation

The Auto-Translation is essentially a method of translating HTML pages to MAML. To do so, we have set up our own LAMP stack i.e. Linux (operating system), Apache (server), MySQL (database) and Python (instead of PHP) as programming and scripting language to execute actions on server-side and communicate with the client. The translation itself is done using the Selenium Web-Driver API [27] and the BrowserMob Proxy [28], where the regular HTML page will be downloaded first and then rendered using the PhantomJS and Firefox webDriver interface. The Apache server that we have set up is where the translation occurs, and essentially acts as a proxy between content providers and the user.

2.2.1 Translation of existing HTML pages to MAML

There are essentially four steps in the translation: requesting, scraping, converting and rendering.

To access a MAML page, the user has to request the web page through our server. The url of the requested web page is sent as an argument to our server, which then initializes a headless browser (Firefox and PhantomJS webDriver) using Selenium. Once Selenium has completely loaded the web page, it scrapes the page and fetches the elements of interest as per the MAML specification. We obtain objects containing text, colors, images, videos and other necessary features, using the Selenium API. Selenium provides a variety of ways of searching for and accessing these objects - referred to as web elements by Selenium - such as using their HTML tag names, their "id" attribute, CSS selectors, XPath and more. For example, to obtain a list of all elements with the "div" tag, we can use this method (implemented in Python): webDriver.findElementsByTagName("div"). Scraping elements of a web page takes O(n) time, as the DOM has to be completely traversed to ensure that all necessary elements are obtained. Therefore, time to scrape would increase linearly with the number of objects on the page.

Once the scraping is complete, the web elements need to be converted to MAML format. Each element is stored in a suitable data structure, as per the MAML specification. We have opted to use a dictionary because retrieval of information is quick, O(1) time on average. To better understand this, let us use a *link* object as a case study. A *link* object in MAML is the "a" tag in HTML, which is used to navigate to a certain url. Figure 1 shows a MAML *link* object with its attributes.

```
{"type": "link", "url": https://www.ourserver.com/index.maml?url=https://www.nytimes.com,
"text": "New York Times", "x": 500, "y": 1000, "w": 50, "h": 50, "font": "Arial"}
```

Figure 1: An example of a MAML *link* object

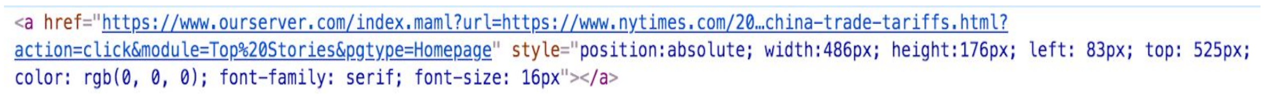
For *link* objects, we also add the path to our proxy server as a prefix to the url attribute. This is done to provide users with the ease of navigating to a MAML version of the referenced web page instead of retyping the url in the browser. Once all the web elements have been converted as per the MAML specification, they are written to MAML file with a .maml extension and saved on our server. As a result, when another user requests the same MAML page, we do not have to perform the translation and can simply return the already existing MAML file.

The last step is to render the MAML file as a web page in the user's browser.

2.2.2 Rendering and displaying of MAML web pages

To display content in a browser, HTML has to be used. That is because HTML is the language of the internet and therefore, web browsers. For a browser to understand any other language, a new rendering engine for that browser would have to be created. That rendering engine would have to be designed in a way that adheres to the specification of that new language. The creation of a rendering engine however, would take a significant amount of time and is also outside the scope of this project.

Therefore, in order to render MAML pages in existing browsers, we translate MAML to a simplified form of HTML, which then gets parsed and rendered by the browser. Figure 2 shows an example of a MAML *link* object translated to HTML (refer to Figure 1 for specification of a MAML *link* object). We use Python as a scripting language to read from the MAML file, and write to the browser window in HTML.



```
<a href="https://www.ourserver.com/index.maml?url=https://www.nytimes.com/20…china-trade-tariffs.html?
action=click&module=Top%20Stories&pgtype=Homepage" style="position:absolute; width:486px; height:176px; left: 83px; top: 525px;
color: rgb(0, 0, 0); font-family: serif; font-size: 16px"></a>
```

Figure 2: Example of a MAML *link* object converted to simplified HTML

Another advantage of using simplified HTML for rendering of MAML pages is that it allows MAML to become more easily acceptable by users. No transitioning or additional installation is needed on client-side to use MAML pages, as existing browsers can be used.

2.2.3 Challenges Faced

We faced a number of challenges when attempting to translate pages from HTML to MAML. We needed to conceive a methodology that considered every kind of web page and could translate it completely and effectively.

The biggest challenge with regards to that was the fact that different approaches are adopted by developers when designing their web pages. There is no protocol or set way of designing web pages. In fact even the most simplest of actions can be performed through many different ways. Consider the example of adding text to a web page. It can be added through at least 10 different HTML tags - a, p, li, span, h1, h2, h3, h4, h5, h6, div, strong (and more) - and even CSS and

JavaScript. Images can be added through a variety of ways as well, using HTML through the "img" tag or as a child of an "a" tag, using JavaScript, and using CSS - as background or background image.

Therefore, a general approach had to be designed that took into consideration all these possibilities and ways. We would come up with an approach that would work well for a certain web page but yield no result for another. Our approach therefore went through multiple iterations and expansions until it achieved satisfactory results.

With the increasing popularity of JavaScript, dynamic web pages are also becoming increasingly common. These web pages respond to and alter themselves because of various user actions, such as scrolling or hovering over certain objects on the web page. These interactions had to be taken into account and replicated during the translation.

2.3 Content Creation in MAML

We have created a tool that content providers can use to create web pages according to the MAML specification. This is a simple web application, mainly containing drag and drop features [reason why drag-and-drop: ease of content creation in an environment that may not be that familiar with existing technologies. Perhaps talk about popularity of websites like wix and squarespace]. The MAML Editor has been designed using HTML, CSS and JavaScript for front-end, Python for back-end and MySQL for database. In order to access the tool, a user will have to register first, and login every time after that. This allows us to maintain and effectively separate user sessions, preferences and data. Users are then taken to the homepage of their profile, where they have various options: view an existing page or create a new one. This would take them to the editing tool. If an existing webpage is to be viewed, it would be displayed in the editor, otherwise, the editor would display a blank canvas, as shown in Figure 3. The canvas is the area within which users can create or edit their page. The editor allows a great degree of options to the user. Users can add many objects to the canvas, such as shapes, text, buttons, images etc. Moreover, all of the objects are interactable, meaning they are draggable and resizable. There are also special actions that a user can perform on certain objects, for example changing the color of a rectangle or the font type and size of text etc. If the user wishes to delete a certain object, or clear the entire canvas, that is also possible. Once a page has been designed, the user has the option of saving, previewing or publishing it. Saving the page would allow the page and all its contents to be saved on our server, the path to which would be saved in the database. Previewing the page would allow the user to view the page as it would look on a phone screen, the dimensions of which can be adjusted by the user. Publishing a page would make the page public and viewable to anyone who has the url of the page. The page would of course be published in MAML.

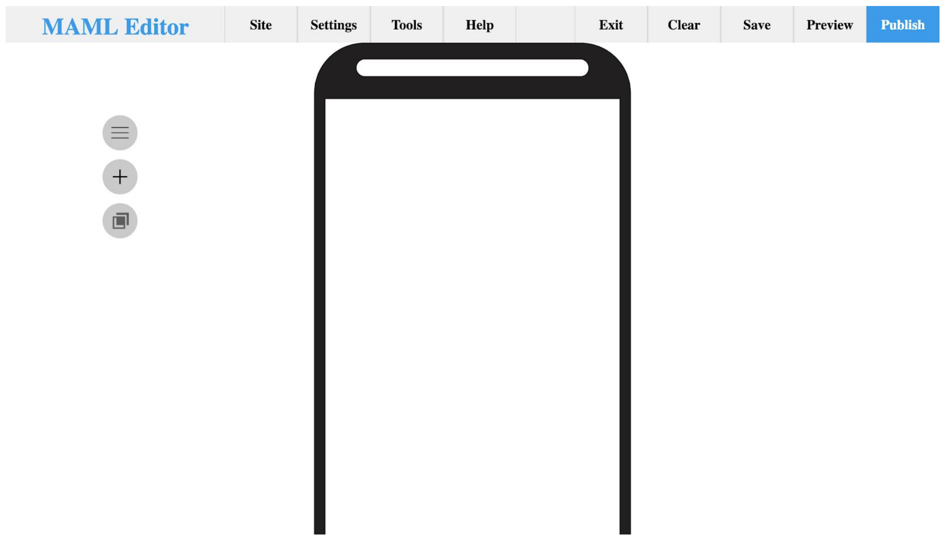


Figure 3: A blank canvas in the MAML Editor tool

Our goal is to encourage content creation in MAML, not simply translate HTML pages to MAML. We want MAML to become the web-specification language in developing regions and this tool would be a great opportunity and way to do so. We hope that through this tool, we can encourage content generation in developing regions, in MAML. With more local content, users will be able to gain access to content much faster, increasing the popularity of MAML and the use of this tool.

2.4 Advantages of MAML over HTML

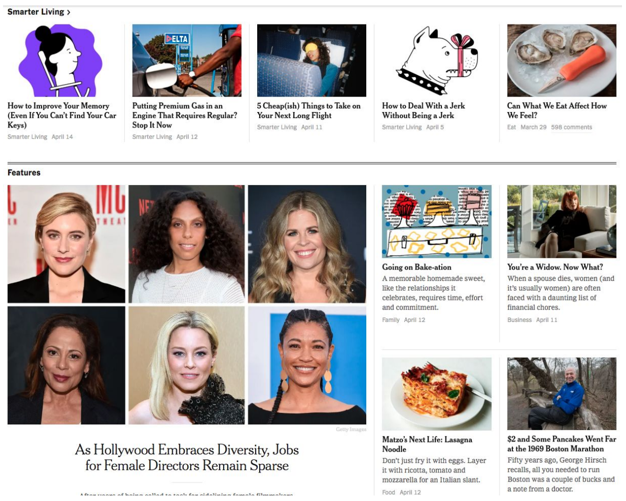
The MAML representation of a page provides several important benefits to dramatically reduce page load times and size for bandwidth constrained users:

1. Web page pre-compilation allows the Web browser to eliminate recursive DNS queries and JavaScript execution, thus cutting down the size of the page and reducing the page load time.
2. Pre-specification of objects in a page significantly simplifies the page evaluation process for a browser and enables scheduling and prioritizing downloads upfront (including dynamic support for parallel downloads).
3. The flattened tree structure layout also simplifies the object loading, object parsing, and the object rendering, thus enabling faster page rendering.

3 Results

In order to confirm our observations, we carried out a series of tests that evaluated MAML under various criterions. We gathered two kinds of data: qualitative and quantitative. The qualitative data was obtained by conducting participant research to study the 'look and feel' of the translated MAML web page. The quantitative data comprises of measurements related to performance testing of MAML pages and their effectiveness in reducing page size and PLT.

3.1 Some examples of translated web pages



Smarter Living ›

How to Improve Your Memory (Even If You Can't Find Your Car Keys)
Smarter Living April 14

Putting Premium Gas in an Engine That Requires Regular? Stop It Now
Smarter Living April 12

5 Cheap(ish) Things to Take on Your Next Long Flight
Smarter Living April 11

How to Deal With a Jerk Without Being a Jerk
Smarter Living April 5

Can What We Eat Affect How We Feel?
Eat March 29 598 comments

Features

Getty Images

As Hollywood Embraces Diversity, Jobs for Female Directors Remain Sparse

After years of being called to task for sidelining female filmmakers

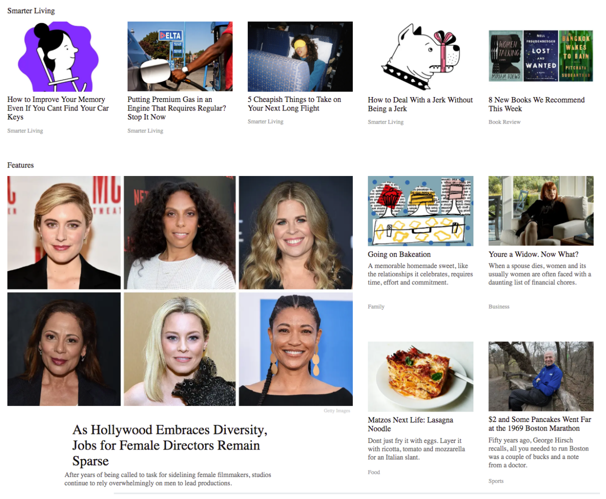
Going on Bake-ation
A memorable homemade sweet, like the relationships it celebrates, requires time, effort and commitment.
Family April 12

You're a Widow. Now What?
When a spouse dies, women (and it's usually women) are often faced with a daunting list of financial chores.
Business April 11

Matzo's Next Life: Lasagna Noodle
Don't just fry it with eggs. Layer it with ricotta, tomato and mozzarella for an Italian slant.
Food April 12

$2 and Some Pancakes Went Far at the 1969 Boston Marathon
Fifty years ago, George Hirsch recalls, all you needed to run Boston was a couple of bucks and a note from a doctor.

Figure 4(a): An original HTML web page (Source: www.nytimes.com)



Smarter Living

How to Improve Your Memory Even If You Cant Find Your Car Keys
Smarter Living

Putting Premium Gas in an Engine That Requires Regular? Stop It Now
Smarter Living

5 Cheapish Things to Take on Your Next Long Flight
Smarter Living

How to Deal With a Jerk Without Being a Jerk
Smarter Living

8 New Books We Recommend This Week
Book Review

Features

Getty Images

As Hollywood Embraces Diversity, Jobs for Female Directors Remain Sparse
After years of being called to task for sidelining female filmmakers, studios continue to rely overwhelmingly on men to lead productions.

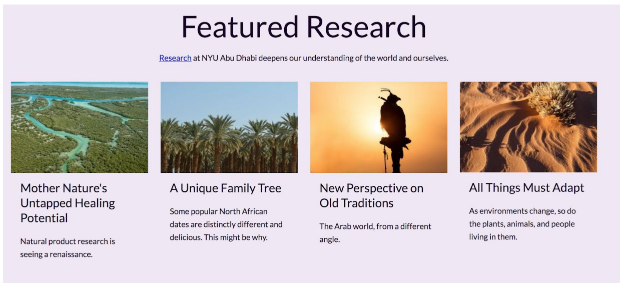
Going on Bakeation
A memorable homemade sweet, like the relationships it celebrates, requires time, effort and commitment.
Family

Youre a Widow. Now What?
When a spouse dies, women and its usually women are often faced with a daunting list of financial chores.
Business

Matzos Next Life: Lasagna Noodle
Dont just fry it with eggs. Layer it with ricotta, tomato and mozzarella for an Italian slant.
Food

$2 and Some Pancakes Went Far at the 1969 Boston Marathon
Fifty years ago, George Hirsch recalls, all you needed to run Boston was a couple of bucks and a note from a doctor.
Sports

Figure 4(b): The MAML translated version of web page in Figure 4(a)



Featured Research

Research at NYU Abu Dhabi deepens our understanding of the world and ourselves.

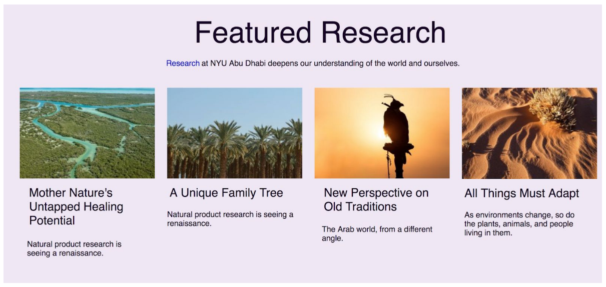
Mother Nature's Untapped Healing Potential
Natural product research is seeing a renaissance.

A Unique Family Tree
Some popular North African dates are distinctly different and delicious. This might be why.

New Perspective on Old Traditions
The Arab world, from a different angle.

All Things Must Adapt
As environments change, so do the plants, animals, and people living in them.

Figure 4(c): An original HTML web page (Source: nyuad.nyu.edu)



Featured Research

Research at NYU Abu Dhabi deepens our understanding of the world and ourselves.

Mother Nature's Untapped Healing Potential
Natural product research is seeing a renaissance.

A Unique Family Tree
Natural product research is seeing a renaissance.

New Perspective on Old Traditions
The Arab world, from a different angle.

All Things Must Adapt
As environments change, so do the plants, animals, and people living in them.

Figure 7(d): The MAML translated version of web page in Figure 4(c)

Figures 4(b) and 4(d) are some examples of the translated web pages. The original and translated versions are presented side-by-side for a comparison and better understanding of the translation, as well as the user study.

3.2 User feedback and participation - look and feel of page

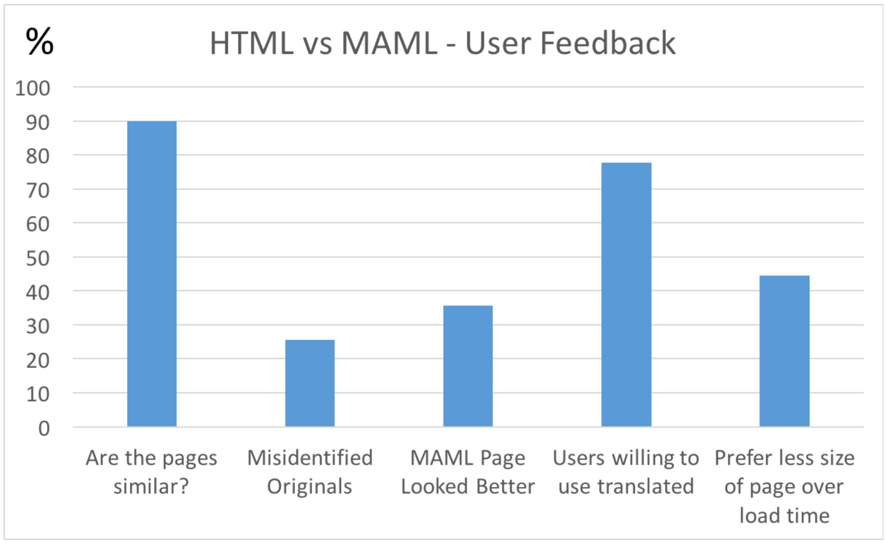


Figure 5: Comparison between original HTML web page and its MAML translation.

The user study was anonymous and the data untagged, ensuring the confidentiality of the participants. Our sample size was 15, with all participants being above the age of 18 and students at New York University Abu Dhabi.

Participants were shown images of HTML web pages and their translated MAML version side-by-side, and for each pairing, were asked a series of questions. The questions required users to compare the two pages, assess their degree of similarity (or dissimilarity) and identify the original HTML web page. The responses of the participants were then collected, tabulated and are shown in Figure 4.

These results show that almost all (90%) of the translated MAML web pages were viewed as similar to the original HTML ones. 25% of the MAML web pages were identified as the original ones, indicating the degree of accuracy in the translation. This is very significant because it indicates the effectiveness and potency of our Auto-Translation engine, which translated the page almost perfectly. Interestingly, around 35% of web pages were identified as aesthetically better, having an improved layout and design. Ultimately, a significant majority – around 80% - of users were willing to use the translated versions, if page load times could be reduced, while 45% of people found the diminished size of the web pages more appealing.

3.3 Performance Testing of MAML

In this section we present the results of the performance evaluation of a MAML page. The evaluations were performed based on comparing the MAML web page compared to the regular HTML version of the same page. We requested and compared 25 different web pages taken from Alexa Top 500 Global Sites [34]. The evaluation had three criterions: page load time, page size and number of requests made by page. The cumulative distribution functions (CDF) of these three criterions was then plotted.

For the first test, we had three scenarios: regular HTML web pages, MAML pages dynamically translated using our Auto-Translation engine and the (already) translated MAML pages. To obtain regular HTML pages, we automated the request for those pages using Selenium Webdriver, and later saved the HTTP Archive file (HAR). For the dynamically translated MAML pages, we used our Auto-Translation engine, creating a HAR file for each page. For the third scenario, we simply had the MAML file of those web pages saved on our server. These MAML pages were requested via a browser and the generated HAR file was then saved. Now, using these three different sets of HAR files, we compared the page performance of the three formats (the HTML, the dynamically translated MAML page and the already translated MAML page). The metric tested for this case was the page load time. As Figure 6(a) indicates, it takes regular HTML pages (in red) up to 60 seconds to fully load whereas a MAML page (in blue) loads in under 10 seconds - an 80% reduction in PLT. Even in the case of dynamically translating HTML to MAML (shown in orange), which includes the translation time, we get a PLT reduction of 50%.

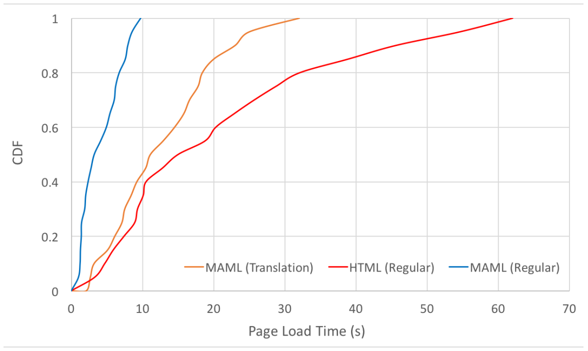


Figure 6(a): Page Load time (s)

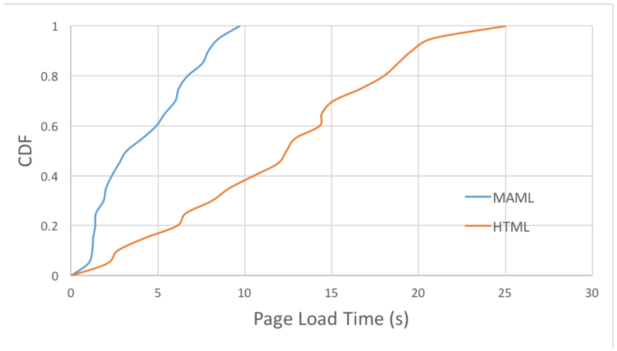


Figure 6(b): Page Load time (s)

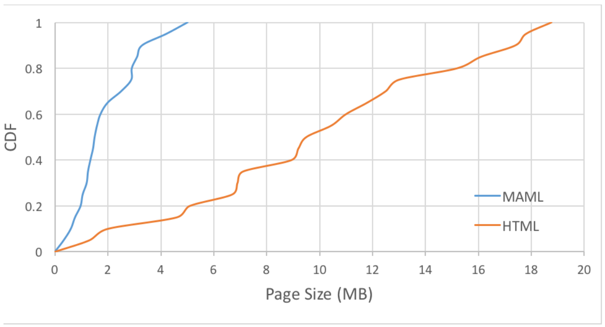


Figure 6(c): Page size (MB)

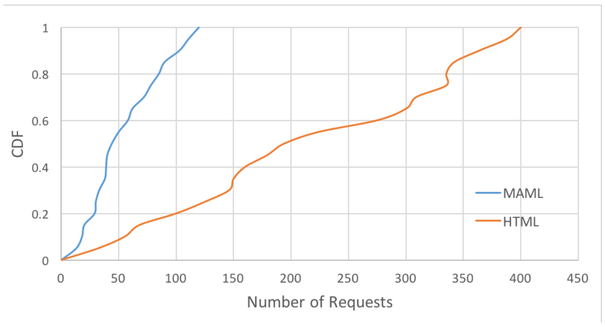


Figure 6(d): Number of requests per page

We then conducted another test, this time involving two scenarios and attempting to keep external factors constant. These factors range from network latencies and congestions to waiting at the host's server due to large web traffic. We therefore saved the HTML versions of the web pages and their MAML versions, on our server. We then fetched those web pages using the WebPageTest API [34], over slow 3G internet and a low-end mobile phone to mimic the conditions of users in the developing regions. For each case, we created the HAR file using the API. Figures 6(b) to 6(d) represent our findings, with HTML being in red and MAML in blue.

Figure 6(b) shows that despite being hosted on the same platform, and removing every other variable, MAML pages loaded significantly faster than HTML pages, achieving a median reduction in PLT of about 60%. The figure also shows that in the most extreme case the MAML PLT does not exceed more than 10 seconds, compared to more than 25 seconds PLT for the regular HTML page.

Figure 6(c) shows the CDFs of the page size for both MAML and HTML pages. The result clearly shows that a MAML page is 80% (by comparing the median value of the two curves) smaller in size than an HTML page. This drastic reduction in page size is achieved because MAML removes all the unnecessary HTML, CSS and JavaScript and only contains the necessary features (such as images, text, colors) needed to render the page. Figure 6(d) shows the CDFs comparison of the number of objects' requested by HTML and MAML. MAML is again successful, reducing the number of requests by about 60%.

These results provide clear indication and evidence of the effectiveness of MAML in reducing the page load time and page size.

4 Summary And Impact

The aim of this project is to improve the performance and delivery of web content in developing countries. Through automatic translation of HTML pages into MAML pages, content can be delivered faster to users, enhancing their web experience. This project is unique due to our focus on mobile users in developing region, rapidly growing demographic. By simplifying the structure of web pages, we are bypassing the network and infrastructural challenges plaguing developing countries, to improve page load times and user experiences. We believe MAML is necessary, as the problem lies in the manipulation of DOM by HTML, CSS and JavaScript. We hope that our research will convince users and developers to adopt MAML as the specification language of the web. In the long run, we hope that content providers will create web pages according to the MAML specifications instead. That way, users will be able to access MAML web pages directly from content providers, instead of having to translate HTML pages into MAML. Through this, we hope to simplify and improve the performance, consumption and the dissemination of web content.

5 Further Work and Research

Due to constraints of time, we were not able to obtain a large enough sample size for our user study. We hope to rectify that and perform our study with an appropriate sample size. In addition to this, there is another important area of research and further work. As of now, in our translation, we execute JavaScript (and precompute other resources) on our server, completely rendering the DOM. We gain many benefits from that, as discussed above, but as a result, the page loses its functionality and dynamicity. Therefore, user actions that require JavaScript, such as form submissions cannot be performed. MAML in its current state is ideal for information retrieval and not submission. Research needs to be carried out into the specific purpose of each JavaScript file and retain only the necessary ones.

We also want to ease the creation of content in developing regions, and encourage the use of our MAML Editor Tool. One way to do so involves creating templates that users can use to create web pages. These templates could be simple in design, such as for a blog post containing a title, a picture and a body of text.

We hope that further work goes into this project to improve it, especially for users in developing countries without proper access to it.

References

[1] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, “Understanding website complexity: Measurements, metrics, and implications,” in Proceedings of the 2011 ACM SIGCOMM Internet Measurement Conference. Scopus, 2011, pp. 313–328.

[2] M. Germonprez and I. Zigurs, “Causal factors for web site complexity,” in Working Papers on Information Environments, Systems and Organizations. Sprouts, 2005, pp. 108–121.

[3] “The world in 2017. ict facts and figures,” 2017, https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2017.pdf.

[4] A. Sharma, M. Kaur, Z. Koradia, R. Nishant, S. Pandit, A. Raman, and A. Seth, “Revisiting the state of cellular data connectivity in india,” in Proceedings of the 2015 Annual Symposium on Computing for Development - ACM DEV 15. ACM DEV, 2015, pp. 149–157.

[5] Y. Zaki, J. Chen, T. Ptsch, T. Ahmad, and L. Subramanian, “Dissecting web latency in ghana,” in Proceedings of the ACM Internet Measurement Conference (IMC). IMC, 2014.

[6] Z. S. Bischof, J. P. Rula, and F. E. Bustamante, “In and out of cuba: Characterizing Cuba’s connectivity,” in Proceedings of the 2015 ACM Conference on Internet Measurement Conference. IMC, 2015.

[7] S. Ahmad, A. L. Haamid, Z. A. Qazi, Z. Zhou, T. Benson, and I. A. Qazi, “A view from the other side: Understanding mobile phone characteristics in the developing world,” in Proceedings of the 2016 ACM on Internet Measurement Conference - IMC 16. IMC, 2016, pp. 319–325.

[8] C. M. Danis, J. B. Ellis, W. A. Kellogg, H. V. Beijma, B. Hoefman, S. D. Daniels, and J.-W. Loggers, “Mobile phones for health education in the developing world: Sms as a user interface,” in Proceedings of the First ACM Symposium on Computing for Development - ACM DEV 10. ACM DEV, 2010.

[9] F. Velghe, “Literacy acquisition, informal learning and mobile phones in a south african town- ship,” in Proceedings of the Sixth International Conference on Information and Communication Technologies and Development - ICTD 13. ICTD, 2013.

[10] S. Wengrovitz, “Facebook advertising research around the world,” 2014, https://research.fb.com/facebook-advertising-research-around-the-world/.

[11] Y. Elkhatib, G. Tyson, and M. Welzl, “Can spdy really make the web faster?” in Networking Conference, 2014. IEEE, 2014, pp. 1–9.

[12] S. Sundaresan, N. Feamster, and R. Teixeira, “Measuring and mitigating web performance bottlenecks in broadband access networks,” in Proceedings of the 2013 Conference on Internet Measurement Conference. ACM, 2013, pp. 213–226.

[13] M. Butkiewicz, D. Wang, Z. Wu, H. V. Madhyastha, and V. Sekar, “Klotski: Reprioritizing web content to improve user experience on mobile devices,” in NSDI’15 Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation. NSDI, 2015, pp. 439–453.

[14] “The state of broadband 2017: Broadband catalyzing sustainable development,” 2017, https:

//www.itu.int/dms pub/itu-s/opb/pol/S-POL-BROADBAND.18-2017-PDF-E.pdf.

[15] J. Chen, D. Hutchful, W. Thies, and L. Subramanian, “Analyzing and accelerating web access in a school in peri-urban india,” in Proceedings of the 20th International Conference Companion on World Wide Web (WWW 11). ACM, 2011, pp. 443–452.

[16] D. M. West, “Digital divide: Improving internet access in the developing world through affordable services and diverse content.”

[17] Fontevecchia, A. (2017). India's 243 Million Internet Users And The Mobile E-Commerce Revolution. [online] FORBES. Available at: https://www.forbes.com/sites/afontevecchia/2014/07/07/indias-massive-e-commerce-opportunity-and-the-explosion-of-mobile/#65fa3625287d [Accessed 7 May 2019].

[18] L. Jing and S. Jingting, “4gs high cost has mainland buyers balking.”

[19] “Broadband availability and quality report,” 2013, department of Communications, Government of Australia.

[20] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall, “Demystifying page load performance with Wprof,” in 10th USENIX Symposium on Networked Systems Design and Implementation. NSDI, 2013, pp. 473–485.

[21] R. Netravali, A. Goyal, J. Mickens, and H. Balakrishnan, “Polaris: Faster page loads using fine-grained dependency tracking,” in NSDI’16 Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation. NSDI, 2016, pp. 123–136.

[22] “Planning for performance: Prpl (Chrome Dev Dummit 2016),” 2016, [online] Youtube. Available at: https://www.youtube. com/watch?v=RWLzUnESylc. [Accessed 23 November 2018]

[23] X. S. Wang, A. Krishnamurthy, and D. Wetherall, “Speeding up web page loads with Shandian,” in NSDI’16 Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation. NSDI, 2016, pp. 109–122.

[24] “Google amp project,” [online]. Google AMP. Available at: https://www.ampproject.org/learn/overview/. [Accessed 25 November 2018]

[25] “Facebook instant articles,”. [online] Facebook. Available at: https://instantarticles.fb.com/. [Accessed 25 November 2018]

[26] “Opera mini browser,” [online] Opera. Available at: https://www.opera.com/mobile/mini. [Accessed 25 November 2018]

[27] “Seleniumhq browser automation,” [online] Selenium. Available at: http://www.seleniumhq.org/about [Accessed 25 November 2018]

[28] P. Lightbody, “Browsermob proxy,” [online] Github. Available at: https://github.com/lightbody/browsermob-proxy. [Accessed 25 November 2018]

[29] T. Ahmad, Y. Zaki, T. Pötsh, J. Chen, A. Sathiaseelan and L. Subramanian, “GAIUS: A New Mobile Content Creation And Diffusion Ecosystem For Emerging Regions”, in ICTD 19, 2019.

[30] R. Fanou, G. Tyson, P. Francois, and A. Sathiaseelan. 2016. Pushing the Frontier: Exploring the African Web Ecosystem. In World Wide Web Conference (WWW).

[31] The State of the digital in April 2019: all the numbers you need to know. [online] We Are Social. Available at: https://wearesocial.com/blog/2019/04/the-state-of-digital-in-april-2019-all-the-numbers-you-need-to-know. [Accessed May 9 2019]

[32] “Find Out How You Stack Up to New Industry Benchmarks for Mobile Page Speed”. [online] GoogleAPIs. Available at: https://think.storage.googleapis.com/docs/mobile-page-speed-new-industry-benchmarks.pdf. [Accessed May 7 2019]

[33] Average Page Load Times for 2018 – How does yours compare? [online] Machmetrics. Available at: https://www.machmetrics.com/speed-blog/average-page-load-times-websites-2018/. [Accessed May 9 2019]

[34] “WebPageTest”. [online] WebPageTest. Available at: https://sites.google.com/a/webpagetest.org/docs/advanced-features [Accessed May 9 2019]

[35] “HTML Living Standard” [online]. Available at: https://html.spec.whatwg.org/print.pdf [Accessed May 10 2019]

[36] A.Osmani, The Cost of JavaScript. [online] Medium. Available at: https://medium.com/@addyosmani/the-cost-of-javascript-in-2018-7d8950fbb5d4. [Accessed May 10 2019]