A Client-side Approach to Improving One-handed Web Surfing on a Smartphone

Karsten Seipp and Kate Devlin

Department of Computing, Goldsmiths College, University of London,
Lewisham Way, New Cross, London SE14 6NW, U.K.
{k.seipp,k.devlin}@gold.ac.uk
http://www.gold.ac.uk/computing/

Abstract. Our work examines the impact of natural movements on usability and efficiency of one-handed website operation on touchscreen smartphones. We present and evaluate a JavaScript framework which transforms the controls of all page elements into an interface that corresponds to the natural arcing swipe of the thumb. This can be operated via swipe and tap, without having to stretch the thumb or change the grip on the phone. The interface can be toggled on or off as required and keeps the original presentation of the website intact. Two user studies show that this choice of interface and its simplified interaction model can be applied to a diverse range of interactive elements and websites and provides a high degree of usability and efficiency.

Keywords: One-handed Operation, Mobile, Web, Wheel Menu, Interface Adaptation, Curved Interfaces, CSS3, JavaScript, HTML.

1 Introduction

The majority of smartphones sold today use modern operating systems such as Android and iOS, both of which have a powerful and largely standard-compliant browser paired with a touchscreen interface and a large screen. With the help of established adaptation techniques for websites on mobile devices [20, 2] as well as responsive themes [6] employing CSS3 media queries, device-independent websites are becoming the norm. In addition, further approaches exist to adapt websites to mobile device constraints, although these are either proprietary [1], dependent on a proxy server [10] or bound to a specific browser [16, 26]. For non-adapted pages, built-in actions such as pinching and tapping to zoom can improve matters. Although these approaches result in an adapted and improved display on a range of mobile devices, they do not provide an adapted interaction model for thumb-based use, which has been identified as a preferred mode of operation by many users [13]. The limited mobility and reach of the thumb represent completely different challenges to the designer regarding the layout and operation of the website; simply crafting it to ensure a correct display of the page elements does not suffice.

While some browsers [16] offer improvements for one-handed operation as part of their interface (using simple gestures, for example), we explore whether one-handed operation can be reliably improved regardless of the browser or plugin used. We do this by improving the display and control of elements operated via direct touch, without the need for gestures. As devices and browsers become increasingly powerful, the question arises as to whether such improvements can be made directly at runtime in the browser, and how efficient and usable these improvements are in comparison to non-enhanced websites.

In this paper we present a JavaScript-based framework which we have named the One Hand Wonder (OHW). The OHW prototype provides an on-demand, thumb-optimised interface for all interactive elements on a web page, facilitating operation and navigation across a wide range of websites. It gives quick access to the most common elements and augments the interaction model of the browser and the standard HTML elements of a web page with additional one-handed UI features that can easily be toggled on and off. These augmentations are temporary and do not change the design of the website. The OHW is built using solely client-side technologies and is implemented by simply embedding the code into the the web page. Initial user testing together with informal feedback during a demo session has confirmed acceptance and learnability [18]. This paper is based on a previous publication [19], but provides more insight into the design rationale as well as a more detailed description of the various interface modules. In particular, this paper focuses on the following:

1. Detailed description of design and functionality of the framework and its modules.
2. Head-to-head comparison of the OHW's performance against normal, non-enhanced operation.
3. A performance test of the system based on its implementation into popular websites.
4. Overall discussion of the OHW as a tool for one-handed website operation on touchscreen smartphones.

2 PREVIOUS RESEARCH

When designing thumb-friendly interfaces, Wobbrock et al. [23] suggest supporting and evoking horizontal thumb movements as much as possible, as vertical movements were found to be overly challenging. On this basis they suggest a horizontal layout of interactive elements on the screen to accommodate the economic peculiarities of the thumb and improve usability. Katre [14] shows that a curved arrangement of elements on a touchscreen is perceived as comfortable and easy, as it supports a more natural circular motion of the thumb. An application of this is found in interfaces such as ArchMenu or ThumbMenu [12], where the user moves their thumb over an arch of elements placed in the bottom right corner of the screen.

Other researchers have explored the use of concentric menus such as the Wavelet menu [8] and SAM [5] to enhance thumb-based interaction, similar to the first generation Apple iPod. In the case of the Wavelet menu, research has shown that this approach with its consistent interaction model is easy to learn and efficient to use. Lü and Li [15] present Gesture Avatar where the user can highlight a GUI element on screen to gain better control of it via an enlarged avatar. While this is an innovative way of improving one-handed device operation, it requires the user to draw their interface first, depends on a proprietary application and cannot be customised by the webmaster.

In terms of general adaptation of websites for mobile devices, one approach is web page segmentation [11, 10], using a proxy server to re-render the page into new logical units which are subsequently served to the device. A proprietary solution is the Read4Me browser [26] where the browser offers to optimise the page for mobile display via a proxy-server that then serves it to the user. Bandelloni et al. suggest a different system [3] where the developer creates an abstract XML-based description of the layout and a proxy server renders the information for the respective access device. In addition to these techniques, various services, themes and frameworks exist to adapt websites for mobile devices and CSS3 media queries offer a flexible approach for display adaptation.

While existing approaches are all concerned with the display of a website on mobile devices, the OHW addresses a so-far neglected aspect: the specific support of one-handed operation of the web page. By implementing the OHW into a website, improvements are made to the operation – rather than the presentation – of the site, as this remains problematic even on well-adapted sites when operating it with one hand. Most importantly, the enhancement is done at runtime and on the client, can be fully configured by the webmaster and is dependent only on the browser itself and the user (who can choose to switch the enhancements on or off at any time).

3 Development

To verify the findings of previous researchers [14, 23] promoting a curved input control for thumb-based GUIs, we conducted a user study with 7 participants (3 F, mean age 31.43 years, SD 4.65), all of who declared to be frequent users of touchscreen mobile devices. Participants were asked to swipe 10 times using their right and left thumb without looking at the device. They were instructed to swipe in a way that was most natural and comfortable to them, avoiding bending and stretching of the joints. Traces of these swipes were recorded on a hidden layer and saved. Stacking the resulting images on top of each other shows the curved movement created by a horizontal swipe, supporting the findings of previous researchers and informing our design of the interface (Fig. 1). In a separate study with 27 participants (8 F, mean age 22.33 years, SD. 2.94), we explored the impact of button position on grip stability. For this we measured the gyroscope amplitude around the X, Y and Z axis when the user tries to reach an item with their thumb in 60 positions. Visualising the data shows that the least amount of movement on the Z axis is in the area highlighted by our swipe test (Fig. 1). By positioning interactive elements in this area, stretching and bending of the thumb

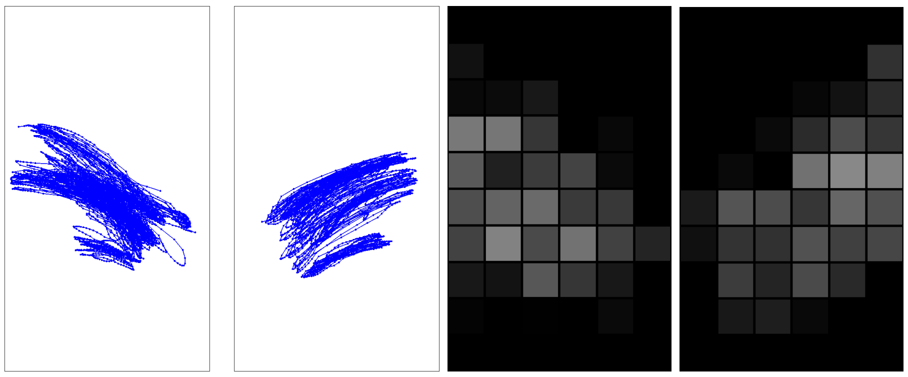


Fig. 1. Far left and left: Visualisation of the swipe data created by the horizontal movement of the left and right thumb. **Right and far right:** Visualisation of the Gyroscope amplitude around the Z-axis. Darker areas present a high amplitude (15793 Hz max.) and strong device movement for reaching a button in these positions. Lighter areas represent element positions with low amplitudes (3273 Hz max.) and little movement of the device. Left image taken from [19] with permission from WEBIST.

and the resulting shifting of the phone in the user's hand is minimised, providing a more secure grip and further supporting our design decision. To develop and verify our design, we iteratively tested paper prototypes with users to transform the wheel menu metaphor into a comprehensive website interface, supporting a more natural operation and minimal strain. Building on discussions with web developers, we made the OHW as non-intrusive and supportive as possible in the form of an easily accessible, half-circle-shaped interface that can be added to a page by simply dropping the code into the website.

4 Functionality

The interface consists of three main parts:

- A *display zone* (Fig. 2, red) showing menu content or the currently active element, such as a video or a form element. When in scrolling mode or when selecting an item from the wheel menu, this zone displays the web page.
- An *interaction zone* (Fig. 2, green) on which the user swipes and taps to manipulate items in the *display zone.*
- A *Start/Back button* (Fig. 2, yellow) which is used for moving back through different states of the interface and for showing and hiding the wheel menu.

The OHW facilitates one-handed web browsing by assembling all interactive elements on request in a region easily accessible by the user's thumb and restricts the range of interactions to just two – swipe and tap. The layout of the page stays untouched and users can decide whether or not to use the interface at any

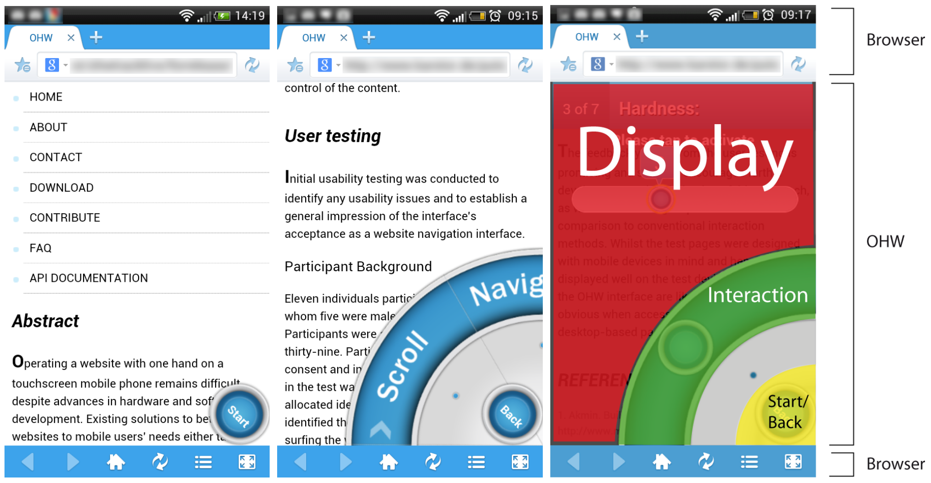


Fig. 2. Left: The OHW as it appears on start up. **Middle:** The OHW opened for right-handed use. **Right:** The three zones of the OHW. Colours were added for easier differentiation. **Red:** The *display zone*. **Green:** The *interaction zone*. **Yellow:** The *Start/Back* button. Images 1 and 2 taken from [19] with permission from WEBIST.

time by switching it on or off (Fig. 2). Thus, the OHW is not an interface for mobile optimisation, which can be achieved using the techniques outlined above. Rather, the OHW's purpose is to enhance one-handed operation of a website regardless of its degree of adaptation, without spoiling the design. It augments the interaction model, not the display. To function, the OHW requires a browser with CSS3 support together with the jQuery JavaScript library – the most popular JavaScript library to date [17] – present on the website. Other than this, there are no minimum standards required and the OHW can be implemented into pages that already include libraries such as MooTools, for example. It has been trialled on a range of Android and iOS devices with HTML4 and HTML5 mark-up in Standards and Quirks mode.

The OHW interface consists of a variety of modules whose availability depends on the content of the website and the interface's configuration. Each module is represented as a wedge and together they form a wheel-type interface, either at the right-hand or left-hand bottom corner of the screen, depending upon the user's choice (Fig. 2). Only the modules that correspond to elements found on the page are loaded, but additional modules can be added at runtime by listening to updates of the Document Object Model (DOM).

To implement the OHW, the webmaster only needs to ensure that the jQuery JavaScript library is available on the website before linking to the OHW's code using a basic <script> tag. The webmaster can optionally edit the configuration file, which is a JavaScript object, and adjust themes, selectors and custom functionality. As each and every aspect of the interface can be adjusted via CSS and HTML, the OHW can fit the look and content of a wide variety of websites. Once implemented, the code scans the website for certain tags from which to build the interface. By default these are basic HTML elements, such as <nav>, <video>, <audio>, <form>, <h2>, and <a>, and from these the standard names of the wedges are derived. This can easily be extended by using CSS selectors and custom wedges declared in the configuration file. The OHW contains several methods to cope with incorrect mark-up and can report any encountered problems to the webmaster.

The OHW's use is optional for the user and the interface can be hidden and brought back at any time. The interface is launched by tapping the Start button on either side of the screen to make it visible (Fig. 2). It can be spun by swiping over it to reveal all available functions. The Start button then becomes a Back button and can be used to either hide the interface completely or to go back one level. For example, if the user was standing and only had one hand free, they could tap the Start button and operate the site one-handedly with the help of the interface. As they sit down and free their other hand, they could hide the interface by tapping the Back button and continue to operate the website with both hands, without a change in the website's design.

4.1 Views

The OHW offers improved presentation and one-handed operation for all interactive page elements. This can be achieved either by accessing them via the respective wedge in the wheel or by double-tapping them on the page. When the interface is launched, swiping and tapping actions in the *interaction zone* are used to manipulate the views in the *display zone*. A variety of views are employed by default to visualise different element types, but can be altered and combined by the webmaster to extend the OHW's functionality:

List view This is employed whenever items are presented in a list and can hold text or images with text, depending on the content (Fig. 3). It is used for the *Headlines*, *Navigation*, *Links* and *Media* menu, as well as drop-down lists. By sliding their thumb over the *interaction zone*, the user can move the list content up or down. The circular button inside the *interaction zone* indicates the scroll position and confirms the selection within the blue *highlight zone* at the top of the screen upon tapping. Tapping on the interface (and not on the interaction button) will scroll the list view to the respective position.

Media player If a natively supported media element is double-tapped on the page or selected via a list view, it is played back in the media player. Swiping over the *interaction zone* controls the playback position (Fig. 3).

Form view All elements of a form are displayed in a horizontal arrangement which the user can navigate using the semi-circular scroll pane in the *interaction zone* (Fig. 3). Tapping the interaction button will activate the current element.

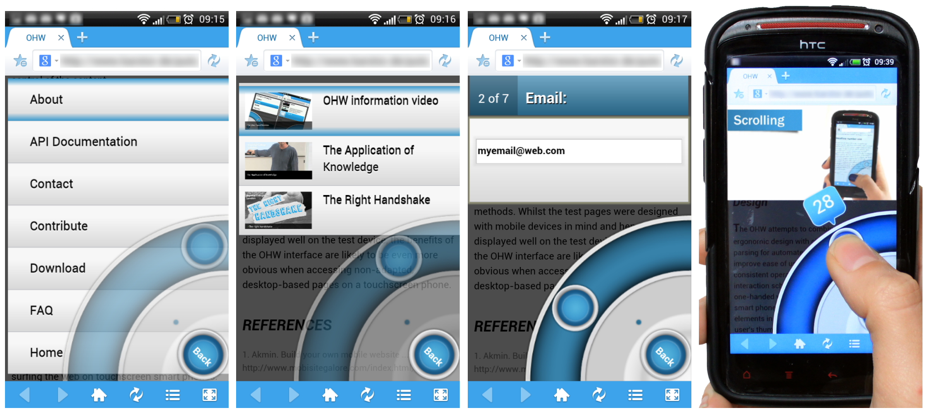


Fig. 3. Far left: The standard list view. The blue *highlight zone* at the top of the screen indicates the currently selected element. **Left:** The list view with images. **Right:** A text input filed in the form overview. **Far right:** The media player view being operated by a user. Images taken from [19] with permission from WEBIST.

Text Input For text input, the basic OS interface is used, as users preferred the standard system keyboard over an adapted semi-circular JS/CSS keyboard (Fig. 4).

Slider An input field of the type `range` is transformed into a slider: Swiping in the *interaction zone* moves the slider head in the *display zone* (Fig. 4).

Date selector Activating an element of the type `datetime` will transform the content area in the form element overview into three lists which can be operated like a normal list view (Fig. 4).

On/Off switches Checkboxes and Radio buttons are displayed as on-off switches that can be operated by tapping the interaction button in the *interaction zone* (Fig. 5).

Buttons Buttons are represented by large buttons operated via the *interaction zone* or direct tap (Fig. 5.

Scrolling The OHW also provides scroll functionality similar to that found in the Opera Mini browser. Tapping the Scroll wedge allows the user to scroll the page by swiping their thumb over the *interaction zone*. While scrolling, interactive elements closest to the current scroll position are outlined one at a time and can be activated by tapping the interaction button of the interface without the user having to stretch their thumb to reach these distant elements (Fig. 5).

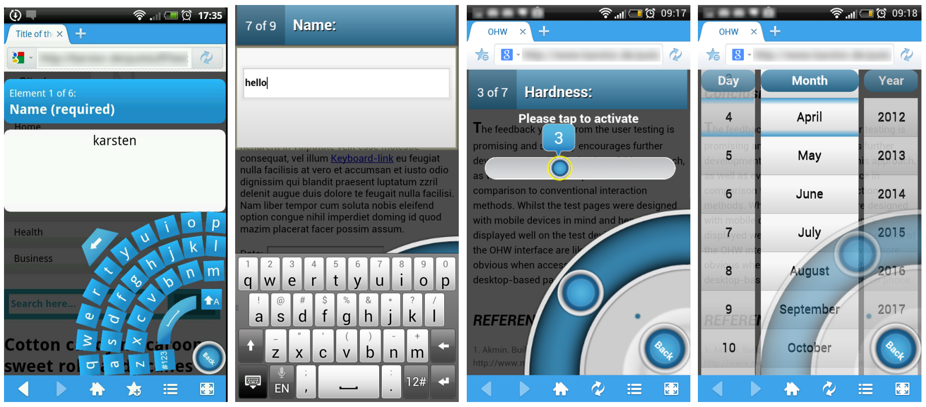


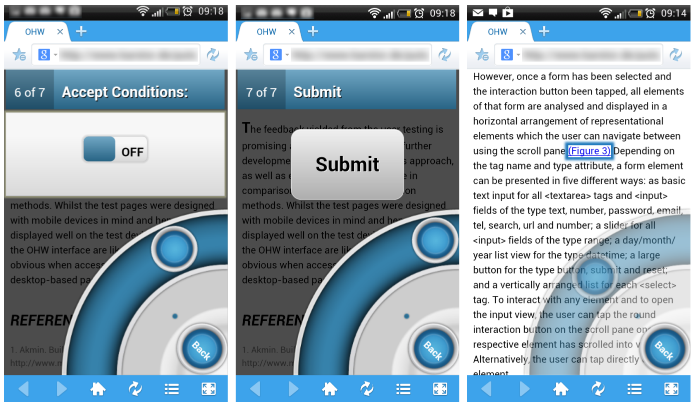
Fig. 4. Far left: Text input via the deprecated OHW keyboard. **Left:** Text input using the system standard keyboard. **Right:** The slider input view. **Far right:** The date input view. Right and far right image taken from [19] with permission from WEBIST.

5 First User Study

After iterative paper prototyping the interface was built and a pilot user test with 11 participants (6 F, all frequent smartphone users) was conducted to identify usability issues and to establish acceptance. Users were given a set of tasks one might perform on a page consisting of headlines, forms, videos, navigation and links, which they completed using the OHW without assistance. The page was designed to be device-independent using CSS3 media queries. All actions were recorded in video and audio, and feedback was given in a questionnaire on a five-point Likert scale. Feedback was predominantly positive and in response to the collected data we created an improved version of the interface. During a demo session at a conference [18] we presented this interface to visitors, implementing it on a range of popular websites to be experienced "hands-on". Informal feedback from users during these sessions was consistently positive, complimenting on the ease-of-use, usefulness of the approach and quality of the implementation, thus supporting the validity of our approach.

6 Second User Study

With the imroved interface we conducted a second user study to determine the efficiency and speed of the OHW in comparison to the normal operation (with one hand without the OHW) of a mobile-optimised website. Altogether, 22 participants (7 F) aged 20 to 34 took part in the study, 19 of whom were final year undergraduate Computing students and the remainder were young professionals. All of them were right-handed, regular users of touchscreen mobile devices, such as phones and media players. The 19 Computing students were briefly introduced



methods. Whilst the test pages were designed with mobile devices in mind and he

However, once a form has been selected and the interaction button been tapped, all elements of that form are analysed and displayed in a horizontal arrangement of representational elements which the user can navigate between using the scroll pane (Figure 3) Depending on the tag name and type attribute, a form element can be presented in five different ways: as basic text input for all <textarea> tags and <input> fields of the type text, number, password, email, tel, search, url and number; a slider for all <input> fields of the type range; a day/month/ year list view for the type datetime; a large button for the type button, submit and reset; and a vertically arranged list for each <select> tag. To interact with any element and to open the input view, the user can tap the round interaction button on the scroll pane on respective element has scrolled into v Alternatively, the user can tap directly element

Fig. 5. Left: Checkbox and radio button view. **Middle:** Button view. **Right:** Media player view. Images taken from [19] with permission from WEBIST.

to the OHW during class and asked to do some self-directed exploring on a test page. On the day of the study, all users were given a 5-minute explanation of the usage of the OHW to ensure its operation was fully understood.

Using a within-subjects design, participants carried out the study one-handedly both with and without the use of the OHW. The study was counterbalanced by altering the mode in which the tasks were first performed. The first part of the usability study comprised 10 separate standard tasks a user might perform on a website and cover the whole spectrum of the OHW:

1. Finding a menu item in the navigation
2. Finding a video and forwarding it to a certain time
3. Finding a form and activating a checkbox
4. Finding another video and starting it
5. Finding a form and filling in a date
6. Finding a link in the body text
7. Finding a form and filling in a range value
8. Finding a headline
9. Finding a form and pressing a button
10. Scrolling and clicking on a link

The study featured a website presenting the OHW. It was coded in HTML5 and CSS3 and contained a page navigation (<nav>), headlines (<h1>, <h2>, <h3>), a form with various elements of input types (“text”, “range”, “date-time”, “submit”, “checkbox”), paragraphs of text (<p>), three video files with poster images (<video>), links (<a>), images (<img>) as well as various <div> elements for the layout. These elements are very common and can be considered as representative for many websites. CSS3 media queries and relative measures were used to make the website’s presentation device-independent. To conduct the study we used a HTC Sensation XE with Android 4.03 and the Maxthon Mobile Browser.

Tasks were performed directly on the website and were preceded by an instruction screen. Each task commenced from the top of the page. The number of interactions and time needed to accomplish the task were recorded using JavaScript. Recording began when users pressed OK on the instruction screen and stopped when a task was completed. For example, recording was only stopped once the target link was clicked or a certain value was entered into a field.

While the above is well-suited for determining efficiency on discrete tasks, it is less suitable for predicting the OHW's "real-life" performance on a page containing any number of these elements, where spatial proximity could affect performance. To address this we also measured the performance in 10 additional, consecutive tasks (1c to 10c), mimicking a set of coherent actions. After each part of this use case, recording was paused to show the instructions for the next part, but the current state of the website (scroll position, opened menus etc.) remained unchanged:

1c. Navigating to a headline in the text
2c. Scrolling and clicking on a link
3c. Finding a menu item in the navigation
4c. Finding a video and forwarding it to a certain time
5c. Navigating to another headline in the text
6c. Finding a link in the body text by scrolling
7c. Finding a form and entering a word into a text field
8c. Activating a checkbox in the same form
9c. Filling in a date in the same form
10c. Pressing a button in the same form

7 RESULTS

The data was evaluated using a Wilcoxon signed-rank test. Due to the varying, skewed results and small sample size we chose a series of non-parametric tests over the ANOVA. As tasks were not comparable in their results because of their different nature, they had to be treated as separate. Comparison of the one-handed task performance of users with the OHW against the normal, non-enhanced way draws a clear picture of the benefits the OHW offers to one-handed website operation. The effect of the OHW on efficiency can be derived from the median number of interactions required to perform a task (Tab. 1) as well as from the time needed to complete it (Tab. 2). Note: The values of the use case (shown in the tables as C) are based on the time and interactions needed to complete the whole use case, consisting of the 10 additional parts 1c to 10c, forming one large task.

Table 1. Median interactions per task and the use case (C) w/out the OHW including Z and p values as well as % of interactions (I) needed with OHW (Normal = 100%). Table taken from [19] with permission from WEBIST.

Task	**OHW**	**Normal**	**Z**	p	**%I OHW**
1	6	19	3.49	$< .001$	32%
2	14	13.5	0.63	.526	104%
3	14	22.5	2.93	.003	62%
4	9	12.5	2.1	.036	72%
5	26.5	27	0.15	.884	98%
6	7.5	16	3.9	$< .001$	47%
7	12	12.5	0.06	.952	96%
8	14	19	1.97	.049	74%
9	13.5	9.5	2.95	.003	142%
10	51.5	26	3.98	$< .001$	198%
C	104	127	2.18	.029	82%

7.1 Results: Number of Interactions Needed

In five out of ten tasks, the OHW allows users to complete the task with fewer interactions (Tab. 1). This is most visible in Tasks 1, 3 and 6 where the same task could be accomplished with only 32%, 62% and 47% of the interactions required without the interface. This highlights the OHW's enhancement of tasks such as finding an item in the navigation, operating a checkbox and retrieving a link from the body text. Other tasks that took fewer interactions to perform with the OHW than without were locating a video (Task 4, 72%) and finding a headline (Task 8, 74%). However, the results also show areas where more interactions are required with the OHW than without. This includes finding and pressing a submit button (Task 9, 142%) and lengthy scrolling to find a link (Task 10, 198%).

7.2 Results: Amount of Time Needed

Evaluating the performance based on the actual time needed to complete a task draws an even clearer picture of the OHW's effectiveness (Tab. 2). Using the OHW in all tasks but one is significantly faster than performing the same tasks without the OHW. The most striking differences can be observed in Task 1 (33% of the time needed), Task 3 (47%), Task 4 (36%), Task 6 (33%) and Task 8 (46%). However, scrolling through the document (Task 10) takes more time with the OHW (147% of time needed).

Table 2. Median time (T) in seconds needed per task (1 to 10) and the use case (C) w/out the OHW including Z and p values as well as % of time needed with the OHW (Normal = 100%). Table taken from [19] with permission from WEBIST.

Task	**OHW**	**Normal**	**Z**	***p***	**%T OHW**
1	11.40	34.30	4.11	<.001	33%
2	25.90	45.10	4.07	<.001	57%
3	20.50	43.90	4.11	<.001	47%
4	10.20	28.60	4.11	<.001	36%
5	33.90	46.60	3.98	<.001	73%
6	9.50	29.10	4.11	<.001	33%
7	18.80	26.30	3.85	<.001	72%
8	14.80	31.90	4.07	<.001	46%
9	15.50	22.10	3.17	.002	70%
10	65.40	44.50	3.56	<.001	147%
C	153.20	255.10	4.11	<.001	60%

7.3 Results of the Use Case

The results of the use case show that in a real-life application the impact of the OHW on efficiency is significant, as overall it took participants only 60% of the time and 82% of the interactions when using the OHW as opposed to operating the website normally (Tab. 2 and Tab. 1).

7.4 Implementation into Popular Websites and General Performance

To determine the OHW's versatility and suitability for different types of websites, we implemented it on several popular websites via a proxy script that injected the OHW code into the loaded page: Wikipedia [22], BBC News [4], W3C [21], Google [9], WordPress [24] and YouTube [25], but failed to implement the OHW on Flickr [7] as our script could not successfully retrieve the site. For the implementation, the names of the wedges had to be adapted to better match the content of each page. Startup time was short (Tab. 3) and the interface felt very responsive on all pages. However, a performance decrease was observed on the Wikipedia implementation, where the links menu held 538 items and on the desktop version of the BBC website, where concurrent scripts and fading animations occasionally impacted the interface.

The results (Tab. 3) show an overall acceptable to good performance and start-up time of the OHW on an Android device and an iPhone. Whereas the time to create a view increases with the amount of elements to display on the

Table 3. Mean time in seconds needed by the HTC Sensation XE (S. XE) and iPhone 3GS (3GS) to create a list view (Fig. 3, right) after tapping a wedge in the wheel using each device's standard browser. For this we chose the WordPress blog [24] as a base and injected additional elements into the DOM when fetching the page using the proxy script. Second part shows mean start-up time of the system (SU) for various websites. For this we measured the start up time of the system by loading and initialising the OHW once the page had finished loading. All measurements were performed three times on each device with a cleared browser cache. Table taken from [19] with permission from WEBIST.

Task	**S. XE**	**3GS**
List view, 30 items	1.1	0.6
List view, 60 items	1.5	0.7
List view, 120 items	1.9	0.7
List view, 480 items	2.4	0.6
SU Wikipedia	1.4	0.9
SU BBC News	2.3	1.4
SU W3C	0.3	0.8
SU Google	0.3	1.0
SU WordPress	1.5	1.4
SU YouTube	0.9	0.9

Android device, the rendering time on the iPhone was not impacted by an increase in items. On comparatively complex websites, such as the BBC page and Wikipedia, start-up time on the iPhone is 39% and 36% faster than on the Android device, whereas the comparatively simple sites, such as Google.com and W3c.org, were parsed more rapidly on the Android device. The performance on the YouTube page and WordPress.com was similar on both devices.

8 Discussion

First we discuss usability and efficiency from a user perspective. Next we discuss the performance of the framework to determine the boundaries of its deployment.

8.1 Usability and Efficiency

The two user studies show that the controls for different elements can be successfully reduced to a curved interface with a consistent interaction model, completely operable within the thumb's comfort zone. In the majority of cases, using the OHW requires less interactions to complete a task and in 90% of the examined cases, a task is completed significantly faster when using the OHW.

However, the results also highlight a weak point of the OHW. When the user has to scroll a large section of the page, the performance of the OHW is significantly weaker than the normal mode of operation (147% of time needed). It shows that scrolling with one hand is already very efficient and that the OHW's approach cannot compete with the existing solution. This has since been addressed by combining native scrolling and OHW scrolling so that the user can scroll the website as usual, but can make more precise selections by moving their thumb over the interface at the same time.

8.2 Versatility

Customisation of the OHW is easy: the webmaster can adapt the text of the wedges to reflect the website content using the supplied templates. Custom functionality is achieved by using the OHW's plugin model to accommodate a website's own set of interactions, such as the accordion-like blocks on Wikipedia [22] with custom callback functions. Thus configured, the OHW is suited to pages with categorised text and images that stretch over many screens and would otherwise need scrolling to access, as found on news websites, wikis, forums, blogs and search engines. Benefits for all types of websites include quick access and operation of forms, navigation, and control of audio or video items if supported natively by the browser.

The OHW performs well on mobile-adapted pages if the content per menu is not excessive. Operation is smooth even with 120 items to be displayed and scrolled. Beyond that the list scrolling performance decreases on the HTC Sensation XE with the amount of data to be presented, whereas it stays the same on the iPhone 3GS with up to 480 list items. While this decrease is likely to only happen in rare cases on mobile websites – as observed in our Wikipedia test – it is more likely to occur on desktop-oriented pages due to the larger page load and other resource-depleting processes. Therefore the webmaster has to be considerate when implementing the OHW: a site loaded with badly coded animations that already struggles being displayed on a mobile device will not necessarily be improved by the OHW. This highlights the main problem of using an integrated client-side approach: the interface has to share the resources with the content of the website, which can directly influence performance. Luckily, this is in the hands of the webmaster implementing the OHW and thus straightforward to address. However, it also shows that the OHW is not a magical one-size-fits-all solution for making any website easier to interact with when operating the device with one hand. What it does, though, is significantly improve operation and efficiency on already mobile-adapted websites (Tab. 2) together with a short start-up time and high responsiveness (Tab. 3).

9 CONCLUSION

Our research shows that applying the wheel-menu metaphor as the basis for thumb-based website interaction and offering a curved input control based solely

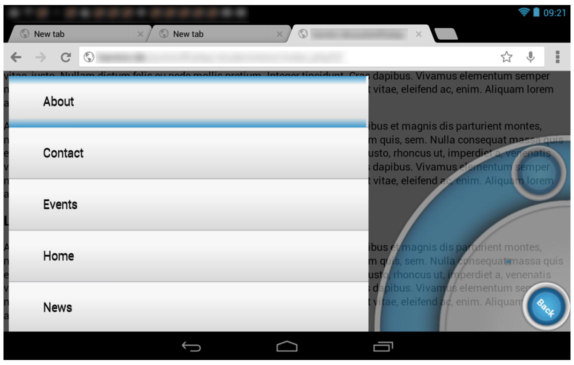


Fig. 6. The OHW on a Nexus 7 in horizontal orientation. Image taken from [19] with permission from WEBIST.

on swipe and tap for all interactive elements clearly improves one-handed website operation: it allows users to complete their goals more quickly and comfortably as they do not have to loosen their grip on the device when trying to reach elements outside the arc of their thumb. Given the demand for a simple, one-handed way to access websites on a mobile device [13], the OHW is a practical and highly effective solution from both a web developer and end-user perspective, if the technical requirements are met, and allows immediate and flexible use without prior user configuration. By dividing the display into a *display zone* and and *interaction zone* (Fig. 2), interface occlusion is reduced when compared to non-enhanced interaction, but is not completely solved due to the fact that the interface is still touch-operated. However, the OHW promotes a free and inclusive way of improving user experience on the mobile web and is a promising approach for enhancing one-handed web browsing on a wide range of mobile touchscreen devices (Fig. 6). The use of standard web technologies allows it to easily adapt to new challenges and ensures its longevity and ease-of-use for the webmaster. Future work will address performance optimisations for the operation of large lists and the development of a SVG and core JavaScript implementation. We plan to evaluate the OHW's applicability as an interface for HTML5-based smartphone apps and the development of an extended plugin model to allow more advanced custom functionality.

References

1. Akmin. Build your own mobile website ... in minutes. `http://www.mobisitegalore.com/index.html`, 2012.
2. Opera Software ASA. Making small devices look great. `http://dev.opera.com/articles/view/making-small-devices-look-great`, 2007.
3. Renata Bandelloni, Giulio Mori, and Fabio Paternò. Dynamic generation of web migratory interfaces. In *Proc. Mobile HCI'05*, pages 83–90, New York, 2005. ACM.
4. BBC. BBC news. `http://m.bbc.co.uk/news`, 2013.

5. David Bonnet and Caroline Appert. Sam: the swiss army menu. In *Proc. IHM 2011*, pages 5:1–5:4, New York, 2011. ACM.
6. Envato. Signum mobile — html5 & css3 and iwebapp. `http://themeforest.net/item/signum-mobile-html5-css3-and-iwebapp/1614712`, 2013.
7. Flickr. Flickr. `http://m.flickr.com`, 2013.
8. Jérémie Francone, Gilles Bailly, Eric Lecolinet, Nadine Mandran, and Laurence Nigay. Wavelet menus on handheld devices: stacking metaphor for novice mode and eyes-free selection for expert mode. In *Proc. AVI 2010*, AVI '10, pages 173–180, New York, 2010. ACM.
9. Google. Google search results. `https://www.google.co.uk/search?q=something`, 2013.
10. Aditya Gupta, Anuj Kumar, Mayank, V. N. Tripathi, and S. Tapaswi. Mobile web: web manipulation for small displays using multi-level hierarchy page segmentation. In *Proc. MC'07*, pages 599–606. ACM, 2007.
11. Gen Hattori, Keiichiro Hoashi, Kazunori Matsumoto, and Fumiaki Sugaya. Robust web page segmentation for mobile terminal using content-distances and page layout information. In *Proc. WWW'07*, pages 361–370. ACM, 2007.
12. Stéphane Huot and Eric Lecolinet. Archmenu et thumbmenu: contrôler son dispositif mobile "sur le pouce". In *Proc. IHM 2007*, pages 107–110, New York, 2007. ACM.
13. Amy K. Karlson and Benjamin B. Bederson. Studies in one-handed mobile design: Habit, desire and agility. Technical report, Computer Science Dept., Uni. of Maryland, 2006.
14. Dinesh Katre. One-handed thumb use on smart phones by semi-literate and illiterate users in india. In *HWID: Usability in Social, Cultural and Organizational Contexts*, volume 316, pages 189–208. Springer Boston, 2010.
15. Hao Lü and Yang Li. Gesture avatar: a technique for operating mobile user interfaces using gestures. In *Proc. CHI'11*, pages 207–216. ACM, 2011.
16. Mobotap. Dolphin browser. `http://dolphin-browser.com/`, 2012.
17. Pingdom. jQuerys triumphant march to success. `http://royal.pingdom.com/2010/03/26/jquery-triumphant-march-to-success/`, 2010.
18. Karsten Seipp and Kate Devlin. Enhancing one-handed website operation on touchscreen mobile phones. In *CHI EA '13*, pages 3123–3126, New York, 2013. ACM.
19. Karsten Seipp and Kate Devlin. The one hand wonder: A framework for enhancing one-handed website operation on touchscreen smartphones. In *Proc. WEBIST'14*. SCITEPRESS, 2014.
20. W3C. Mobile web best practices 1.0. `http://www.w3.org/TR/mobile-bp/`, July 2008.
21. W3C. W3C. `http://www.w3.org`, 2013.
22. Wikipedia. Deusdedit of Canterbury. `http://en.m.wikipedia.org/wiki/Deusdedit\_of\_Canterbury`, 2013.
23. Jacob O. Wobbrock, Brad A. Myers, and Htet Htet Aung. The performance of hand postures in front- and back-of-device interaction for mobile computing. *Int. J. Hum.-Comput. Stud.*, 66(12):857–875, December 2008.
24. WordPress. Just another wordpress weblog. `http://en.blog.wordpress.com/`, 2013.
25. YouTube. Youtube mobile. `http://www.youtube.com/results?client=mv-google&q=sublime`, 2013.
26. Chen-Hsiang Yu and Robert C. Miller. Enhancing mobile browsing and reading. In *E.A. CHI'11*, pages 1783–1788, New York, 2011. ACM.