# Blaise On-the-Go: Using Blaise IS With Mobile Devices

*Alerk Amin (CentERdata, Tilburg, The Netherlands)*
*Arnaud Wijnant (CentERdata, Tilburg, The Netherlands)*

## 1 Introduction

Blaise IS provides a mechanism for respondents to complete their questionnaires over the web. The interface is designed for respondents with a mouse, using a web browser running on full-sized screen. However, people are increasingly accessing the internet with touchscreen mobile devices, such as smartphones and tablets. For mobile respondents, answering Blaise questionnaires should be as natural and comfortable as any other app on their device. The unique characteristics of mobile devices require a different type of user interface, which the standard Blaise IS style sheets do not offer.

Figure 1 shows a questionnaire on a mobile screen, using the default Blaise style sheets. The text and buttons are sized correctly for a desktop screen, but they clearly are too small for the mobile screen. Respondents must zoom in to read the question text, and then either zoom out or scroll the page down to reach the next button. This makes the questionnaire difficult to use with a mobile device, and greatly increases the respondent burden.

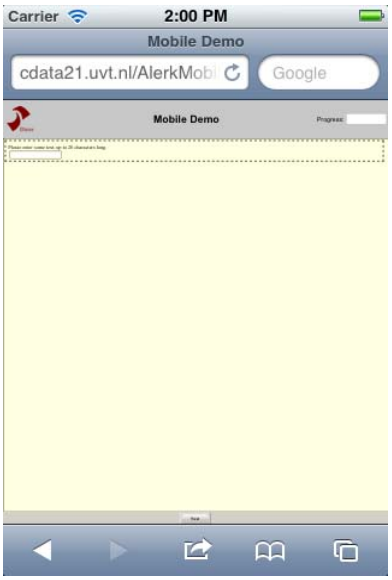


*Figure 1*

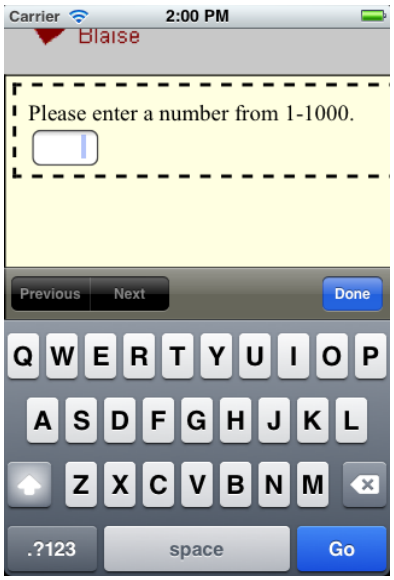


*Figure 2*

Another issue concerns on-screen keyboards. Most touchscreen mobile devices utilize on-screen keyboards. Because the size of the screen is limited, the on-screen keyboards do not display all of the keys at one time. Figure 2 shows a problematic case. The question asks the respondent to enter a number, but the on-screen keyboard shows letters. The respondent must take the additional step to click the number button in the bottom left corner, to switch the keyboard to the numeric keyboard.

To address these issues and others, a new style sheet was created. This mobile style sheet generates questionnaire pages that are adapted for mobile devices. Addressing issues such as screen size, on-screen keyboards, touch gestures, network bandwidth, and more, it allows respondents to more easily complete a Blaise IS questionnaire on a mobile device.

# 2 C-Moto Mobile Style Sheet

The new style sheet and associated files constitute the CentERdata Mobile Touchscreen Style Sheet, or C-Moto. The following sections describe the design principles behind the development of C-Moto, and how it fits into the Blaise IS architecture.

## 2.1 Separation of Presentation and Content

A principle of good web design is the separation of presentation and content. For questionnaires, Blaise accomplishes this through the separation between the BLA file (content) and the BML ModeLib file (presentation). While the BLA file contains the actual questions, the BML defines the various styles and the presentation of the questions. When the system is run with Blaise IS, web pages are generated that contain the combine the question and style, so they are displayed properly in a web browser.

For many organizations, this separation works well. But for organizations with experience in building websites, this mechanism creates problems.

The separation of presentation and content can be extended to web pages. Good web design involves using HTML to describe the content of a page, and CSS (Cascading Style Sheet) to describe the presentation of the content. The web pages generated by Blaise IS have presentation information mixed into the HTML, making it difficult to adjust the layout via CSS.

The C-Moto solution consists of a new style sheet, which replaces the biHTMLWebPage.xsl and biSimpleHTMLWebPage.xsl that are included with Blaise IS. The new style sheet creates “clean” HTML, without any presentation information. This HTML is combined with JavaScript and CSS files to make the web browser render the page properly.

## 2.2 Mobile Framework

With C-Moto, the HTML generated by Blaise IS can be completely controlled. This opens up the possibility to leverage a mobile framework to generate pages that work well on mobile devices.

There are many mobile frameworks available to aid the development of mobile apps and websites. Based on the requirements of mobile questionnaires, and to create an environment that works well with Blaise IS, the C-Moto was developed with jQuery Mobile.

jQuery Mobile is very easy to use and implement. Most functionality is accomplished via HTML markup, simply by adding extra attributes to the HTML elements. As the new style sheet gives total control over the generated HTML, it is straightforward to integrate the appropriate jQuery Mobile markup. When the page is loaded in the browser, the jQuery Mobile JavaScript transforms the HTML elements into appropriately styled elements for mobile devices. jQuery Mobile also has a very broad platform support, including all major desktop and mobile browsers. It also degrades gracefully, in that some features may not work in all browsers, but the pages in those browsers will still look nice and be functional for respondents.

## 2.3 Mobile Style Sheet Architecture

The architecture of C-Moto works seamlessly with the normal Blaise IS architecture. The key to the solution is the new mobile style sheet, which replaces the style sheets provided with Blaise IS. The new mobile style sheet can easily be added in the .BIS file, along with the supporting JavaScript, CSS and image files.

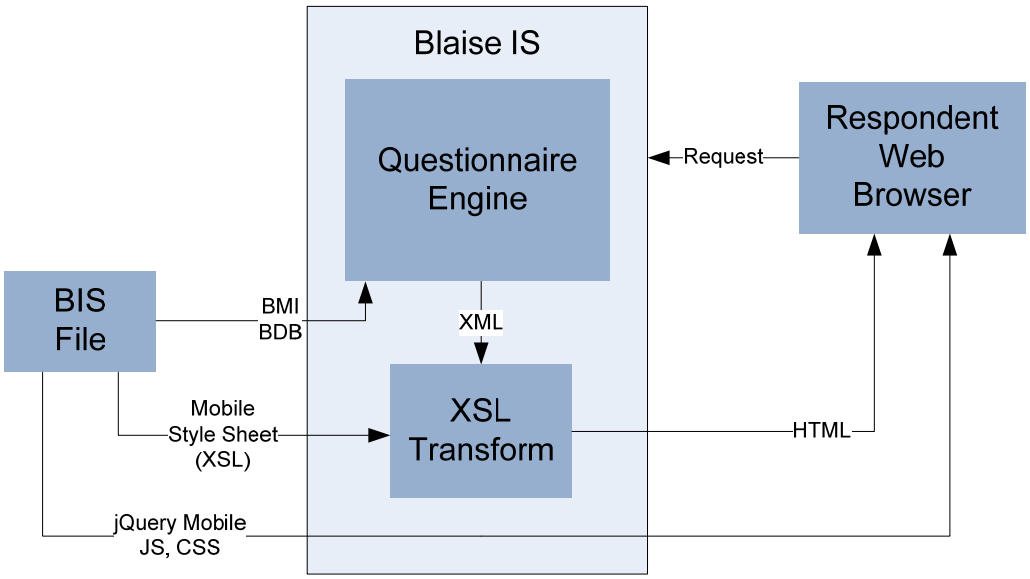


*Figure 3*

Figure 3 shows the process flow of a web questionnaire that uses C-Moto. When the respondent requests a page, the normal Blaise questionnaire engine (a combination of the web server, rules server and data server) creates an XML page. The mobile style sheet is used to convert this XML into an HTML page. The HTML page uses the JavaScript and CSS files required for jQuery Mobile, instead of those provided with Blaise IS.

The HTML page created with the Blaise style sheets includes a form. Blaise expects certain parameters to be submitted with this form, so the questionnaire engine can process the user input correctly. The form generated by C-Moto is designed to submit the exact same parameters as the normal Blaise form, so the Blaise questionnaire engine can process it without any other modifications.

# 3 Functionality

C-Moto contains a great deal of extra functionality for mobile and touch screen devices, compared to the standard Blaise style sheets. This section describes the various issues of questionnaires on mobile devices, and how C-Moto addresses them.

## 3.1 Screen Size

Compared to traditional web interviewing, mobile web interviewing is most often done on smaller screens. For this reason, C-Moto ensures that the text is large enough to be easily read on the screen, and buttons are large enough to be clicked with a finger.

To accomplish this, the design of the questionnaires is very compact. The aim is to minimize the unused screen area as much as possible. The screen is split into 3 areas. The header and footer areas have a fixed height and contain the basic functionality of a questionnaire (like next, back and help buttons) and one area is flexible in size and if necessary also scrollable.

*Figure 4*

Figure 4 shows the 3 areas. The two areas with a red line are the fixed areas. These areas contain more static functionality that is more questionnaire-oriented than question-oriented. The green area in the middle represents the question itself and is more flexible.

The text, input areas, and buttons on the page are all sized appropriately for a mobile device. The text is large enough to be easily read, while the input areas and buttons are large enough to be comfortable clicked with a finger.

A benefit of C-Moto is that it also works well on desktop devices. Figure 5 shows the same question on a desktop or tablet screen. The page expands to fill the available width, but otherwise most aspects are the same. This allows for both consistency and usability across a wide range of screen sizes.

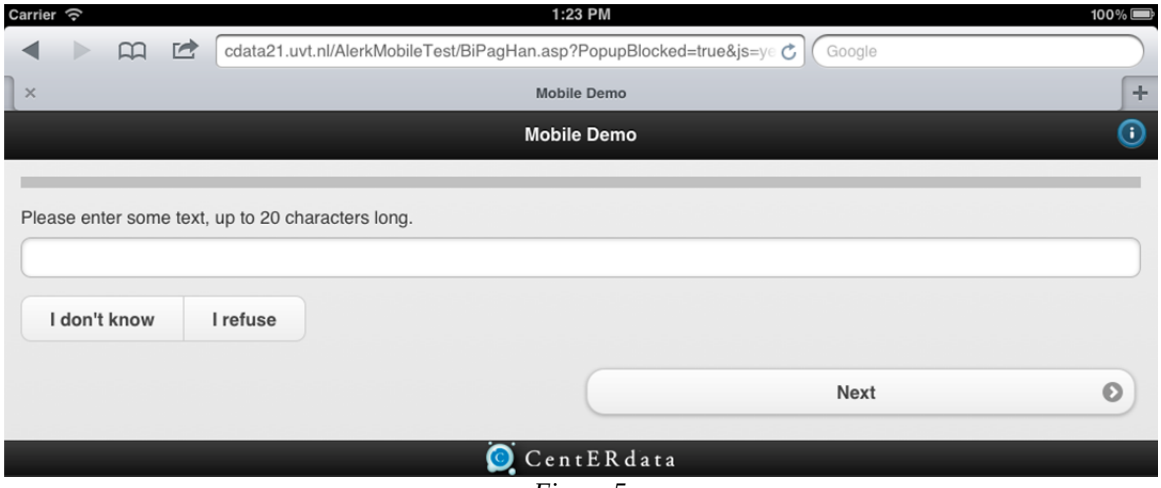

*Figure 5*

This adjustment of size is especially important for questions that have an answer type of a list or set. Figure 6 shows a question with a list using the standard Blaise IS style sheets. The list is rendered with radio buttons that are too small for respondents to click. If the question was a set instead of a list, the radio buttons would be replaced with checkboxes, but the size would be the same. Figure 7 shows the same question with C-Moto. The list is still rendered as radio buttons, but the style is much more usable for mobile respondents. The radio buttons are grouped together, in a “container box” with

rounded corners, to show that these options all go together. The box shading is slightly different than the page background, to emphasize this point. Each option is rendered with a box that contains both the radio button, as well as the label. This makes it clear to respondents that they can click anywhere on the row (in the radio button or on the label) to select an option. Increasing the size of the click target area makes it easier for respondents to click an option on a small screen.

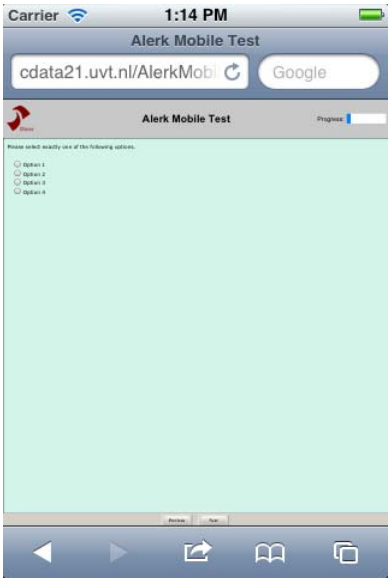


*Figure 6*

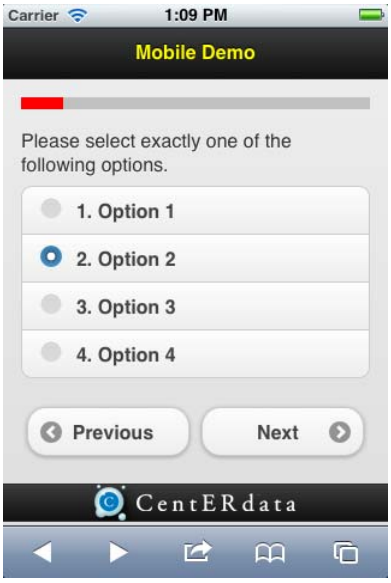


*Figure 7*

## 3.2 Tables

Blaise supports the ability to display multiple questions on a screen, either via tables or blocks. In either case, the result is a list of questions that can be converted into rows through configuration in the ModeLib and/or the Internet Workshop.

In most cases, tables are displayed with a question on each row. Each row contains the question text in the leftmost column, and then the answer box (for string or number fields) in the next column, or a set of options (for lists) in subsequent columns. There can also be columns added for error texts and other items.

On a mobile screen, this type of layout usually does not work, as the screen is just too small. In this case, it may be better to display the various questions in the table as individual questions. Each question looks just like it would without a table, but multiple questions are displayed one-on-top-of-the-other on the same page. Figure 8 shows an example of this solution.

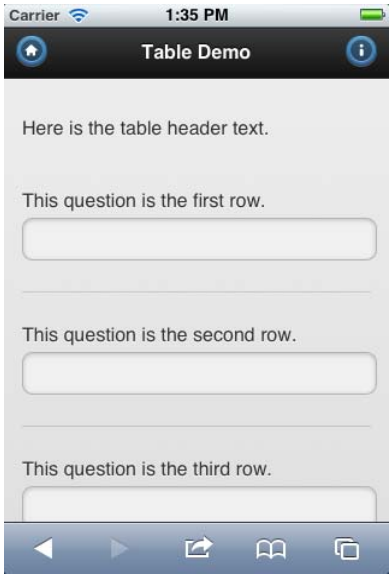


*Figure 8*

This type of layout works well for small screen, such as smartphones, but is not always the best solution for tablets. Tablets have screens that are much larger, and can support a more traditional table layout. A possible solution is to have the question text above the answers for small smartphone screens, but to the left of the answers on larger screens.

Fortunately, this can be accomplished through the use of CSS media queries. The mobile style sheets enforce the separation of content and presentation, with the presentation being controlled through CSS. CSS media queries allow for different layouts, based on characteristics of the device. The mobile style sheet adjusts tables based on a screen width of 450px, the default value in jQuery Mobile. For screens with a width smaller than 450px, tables are displayed with the question text above the answers. For screens with a width of 450px or greater, tables are displayed with the question text to the left of the answers. Figure 9 shows the same table as above, but on a larger screen.


*Figure 9*

CSS Media Queries are extremely powerful and flexible way of adjusting the same content to have a different presentation on different devices. In addition to screen sizes, presentation can be adjusted based on the type of device, or even the orientation of the device. For instance, if a respondent rotates their device from portrait to landscape mode, CSS Media Queries can adjust the style for the new layout.

### 3.3 Help Text

Assistance is sometimes important for respondents while filling in a questionnaire. Respondents should be able to get help during the interview. In the mobile style sheet, support for explanatory help texts is provided. If more information is available for a specific question, an “i” button pops-up in the upper right corner of the screen. If the respondent clicks this button, a status box appears that contains the help-item of this specific question. No additional requests to the server are required for this help, which makes the action as fast as a local application. Figure 10 shows how this help would pop up if the “i” button in the right upper corner was clicked.

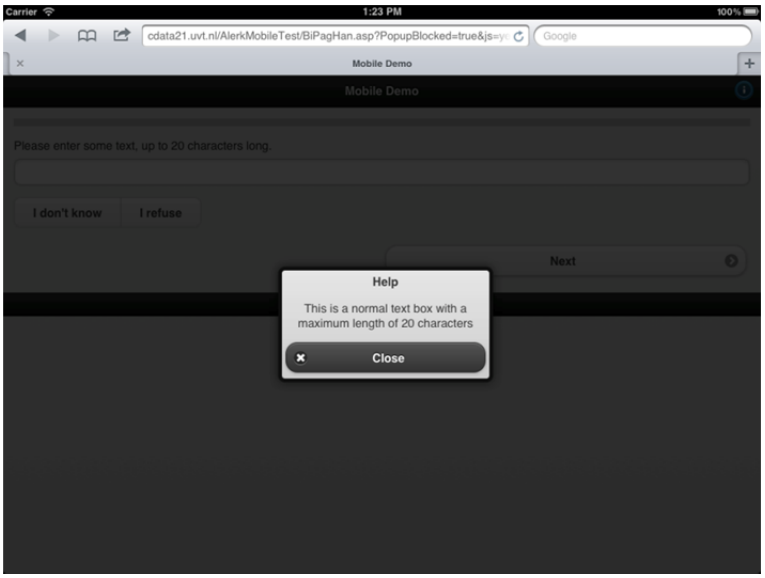


*Figure 10*

On the Blaise side, the help texts are programmed as an additional language and defined in the ModeLib, as normal. C-Moto automatically detects whether there is help text for a question. If there is help text, it shows the help icon and manages the dialog box with the help text. No additional programming or configuration is required in the ModeLib for this functionality.

### 3.4 Touch Screen Keyboards

The most common type of keyboard on a smartphone is a touch screen keyboard. Many smartphones lack a physical keyboard, and instead have a virtual keyboard that pops up on the screen when it is necessary. Since this keyboard is not fixed, it creates the possibility to adapt the keyboard layout to the situation.

In questionnaires, this is really useful as keyboard layout can match the type of question. For normal text inputs, the default (QWERTY) keyboard can be displayed. But when a question requires a number to be filled in, the keyboard can only contain the number keys, which makes it more comfortable and easy to type in the correct number. The same can be done for special data types like telephone numbers, web addresses or email addresses.


*Figure 11*


*Figure 12*


*Figure 13*

Figure 11, figure 12 and figure 13 show several examples of questions with different keyboards layouts. C-Moto uses HTML5 input types to control the keyboard. Support for HTML5 input types varies across devices and browsers. A table that shows support is available at http://wufoo.com/html5/. But even on browsers that do not support all of the input types, the functionality degrades gracefully. In these cases, the default QWERTY keyboard is displayed, giving the respondent the ability to still answer the questions.

### 3.5 Date and Time Pickers

Date and time pickers are a special case of on-screen keyboards. Many devices have native controls for date and time pickers. Figure 14 and figure 15 show two examples. However, support for these is not as extensive as for keyboards, and the look and feel of the controls vary greatly between devices. While using native controls is supported by C-Moto, it also supports an alternative method.


*Figure 14*

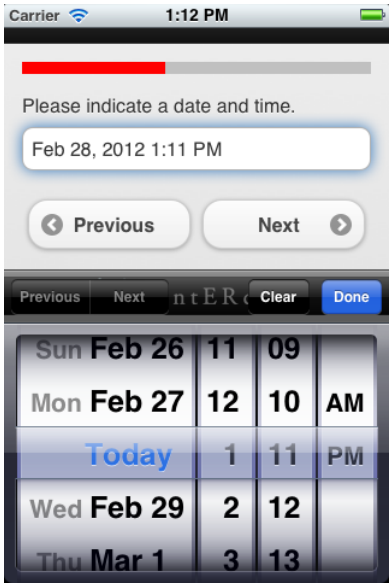


*Figure 15*

Instead of using native controls for date and time pickers, C-Moto supports the jQuery Mobile Datebox library. This library is an extension to the standard jQuery Mobile framework, and adds support for many different date and time pickers. In contrast to native controls, jQuery Mobile Datebox implements the controls using JavaScript and CSS, to create a consistent look and feel across different mobile devices.

Figure 16 shows a date picker using spinner controls, similar to those found on many Android phones, while figure 17 uses slider controls similar to those found on iPhones. By using these controls, researchers can choose what type of control respondents should use.

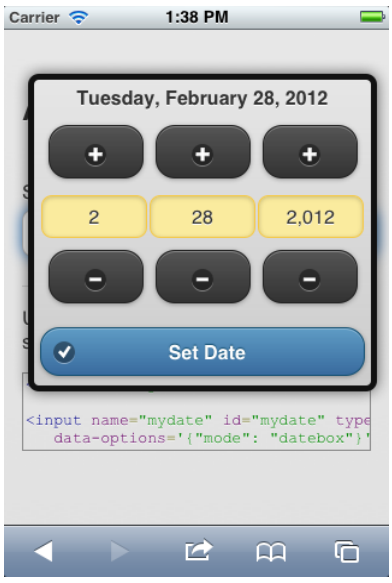


*Figure 16*


*Figure 17*


*Figure 18*

Another option would be to use a classical calendar-style date selection box, as used in many other websites. Figure 18 shows an example of a calendar control.

### 3.6 Navigation with Buttons and Gestures

The forms in the Blaise style sheet include a Next and Previous button to navigate though the questionnaire. In the HTML page, these buttons are input elements with type=submit, to submit the form. They each have a different name, so the Blaise Questionnaire engine can tell which button was pressed, to navigate in the appropriate direction.

The mobile style sheet also includes Next and Previous buttons. However, the placement of these buttons may vary. For example, on some devices, it is customary to put a Back or Previous button in the upper left corner of the screen. This puts the button in the header area of the page, outside of the form element. Additionally, many mobile users are accustomed to using gestures to navigation. Simply swiping a figure from left to right across the screen should move to the previous page, and swiping from right to left should move to the next page.

To make the layout more flexible, the mobile style sheet uses JavaScript to manage the form. The Next and Previous buttons can be placed anywhere on the screen, as input element with type=button. When they are clicked, JavaScript code manipulates the form to insert the appropriate values, so that the Blaise questionnaire engine knows whether the Next or Previous button was clicked. This JavaScript code is also use to process swipe events, to allow gesture-based navigation in the Blaise questionnaire.

### 3.7 Animation

When navigating from page to page in a mobile app, it is customary to use animation. For example, when going to the next page, the old page slides to the left, while the new page slides in from the right. When moving to the previous page, the animation is displayed in the opposite direction. Animation is an extremely powerful tool to provide visual cues to the respondent.

On a normal Blaise page, submitting a form causes a request to be sent to the server, which delivers the next page. The browser simply replaces the previous page with the new page, with no animation. The mobile style sheets make use of AJAX, a widely used technology that allows for asynchronous requests. When the user submits a form (via a button click or a gesture), the JavaScript code on the web page submits the form data, but the current page stays on the screen. When the server returns the HTML for the next page, the browser does not replace the page, as with the normal form. Instead, because the request was an AJAX request, the contents of the new HTML page are sent to the JavaScript code in the old page. This code then prepares the new page, and runs an animation. This gives the effect of the old page sliding out and the new page sliding in.

Because all of the form submission and animation are controlled by JavaScript code, more advanced options are also possible. The direction of the animation should be right-to-left when moving forward, and left-to-right when moving backward. Because the JavaScript code differentiates between a click of the Next or Previous button (or a gesture), it can display the animation in the appropriate direction. A future possibility is to improve the handling of errors. If Blaise detects an error on the page, it will not move to the next question, but rather display the same page, with error texts on this. Ideally, this type of transition would not use a right-to-left animation, to make it clear to the respondent that they have not moved to the next question, because of errors on the page.

### 3.8 Rating Scales

Rating scales, such as a Likert scale, are a common method of measuring how respondents rate something. For example, a 5-point scale that goes from “Completely Disagree” to “Completely Agree” is a common answer type to measure how a respondent feels about a statement. In Blaise, rating scales are usually implemented as a field of type List.

In Blaise IS, lists are converted to a set of options, which are usually displayed in the browser as a set of radio buttons. Clicking one button selects it, and deselects any other button that was previously

selected. On a desktop browser, the buttons are small, but a mouse can control the cursor, so the desired button is easy to click.

However, a mobile device has a much smaller screen, and respondents usually select items by touching them with their finger. This is much less accurate than a mouse/cursor, so the radio button areas need to be much bigger. While a mobile device may have room for 5 radio buttons arranged horizontally, a larger scale (such as a 11-point scale) may be too difficult to use.

A better option is to use a slider, as shown in Figure 19. The slider is displayed as a horizontal bar, and there is a small "handle", that shows the selected value. The key difference between a slider and a set of radio buttons is that sliders also include "drag" functionality. A respondent can touch the handle and drag it left or right, to the appropriate value. In practice, this is much more accurate than small radio buttons, and allows for larger scales on smaller screens.


***Figure 19***

The mobile style sheet contains support for sliders. This is accomplished in the Blaise source via the tag. A questionnaire can be programmed with fields of type list, as normal. However, if the field has a tag "slider", the style sheet will display a slider instead of a set of radio buttons.

## 3.9 Images and Video

Images and videos need special attention in the mobile style sheet. One aspect is that the size of the images should be optimized to the device it is being sent to. There is no need to send pictures that are larger than the actual screen size to the mobile device, as this causes unnecessary data traffic. To be able to know what image needs to be included, CSS Media Queries can be used to detect the screen size and include a picture that is suitable for the size of the screen.

For videos, most mobile devices support different video formats. Managing multiple formats to target different devices requires a great deal of effort. A possible solution is to use YouTube, as most mobile devices have support for displaying YouTube videos. The video can be "embedded" into a question, by inserting the embed code (from YouTube) into the question text. On desktop browsers, the video will play in this embedded box on the screen. But on most mobile devices, the video will play on the full screen, making better use of the screen space. Additionally, YouTube will manage the video formats, converting as necessary to the appropriate size and format for the device. This functionality works extremely well, and requires nothing from C-Moto.

### 3.10 Performance and Bandwidth

Besides the user friendliness of the C-Moto style sheet, performance is also an important issue for mobile device web pages. Pages should load fast enough over a mobile connection. For this reason, the output HTML of the C-Moto style sheet is much more compact than the traditional Blaise IS style sheets.

One part of this optimization is achieved by moving the presentation/ mark-up data from the HTML files to separate CSS files. These CSS files need to be loaded one time per questionnaire or even one time per questionnaire server (if all questionnaires use the same style). Additional optimization occurred by creating simpler, smaller HTML, which gets customized by the JavaScript and CSS files.

A comparison of the C-Moto and standard Blaise style sheet shows us that the data per page in C-Moto is half the size of the same page in the traditional Blaise web pages. The initial files that need to be loaded in both systems are comparable. In the traditional Blaise style sheet, these are mostly JavaScript files, whereas in the mobile style sheet most of the initial files are CSS files. In the table below some page sizes are mentioned.

| | **Blaise IS style sheet** | **C-Moto** |
|---|---|---|
| **Initial files (CSS, JS)** | 172 kB | 188 kB |
| **Question page (HTML)** | 25-35 kB | 12-18 kB |

## 4 Conclusion

Running questionnaires on mobile devices presents challenges, but also a unique opportunity to customize the questionnaire for mobile respondents. C-Moto helps bridge the gap between desktop and mobile devices, by ensuring that the same BLA questionnaire displays in a usable form on different devices. By separating the content and presentation, the display becomes easy to customize and conforms to modern web standards. Better usage of mobile input options, including virtual keyboards and gestures, allows for easier and more accurate data entry. Taken together, these features implement a better mobile interface, creating a better respondent experience.