

MDN's new design is in Beta! A sneak peek: <https://blog.mozilla.org/opendesign/mdns-new-design-beta/>

Learn web development

Mobile accessibility

[← Previous](#)[↑ Overview: Accessibility](#)[Next →](#)

With web access on mobile devices being so popular, and popular platforms such as iOS and Android having fully-fledged accessibility tools, it is important to consider the accessibility of your web content on these platforms. This article looks at mobile-specific accessibility considerations.

Prerequisites:	Basic computer literacy, a basic understanding of HTML, CSS, and JavaScript, an understanding of the previous articles in the course .
Objective:	To understand what problems exist with accessibility on mobile devices, and how to overcome them.

Accessibility on mobile devices

The state of accessibility — and support for web standards in general — is good in modern mobile devices. Long gone are the days when mobile devices ran completely different web technologies to desktop browsers, forcing developers to use browser sniffing and serve them completely separate sites (although quite a few companies still detect usage of mobile devices and serve them a separate mobile domain).

These days, mobile devices in general can handle "full fat" websites, and the main platforms even have screenreaders built in, to enable blind users to use them successfully. Modern mobile browsers tend to have good support for [WAI-ARIA](#) too.

To make a website accessible and usable on mobile, you mainly just need to follow general good web design and accessibility best practices.

There are some exceptions that need special consideration for mobile; the main ones are:

- Control mechanisms — Make sure that interface controls such as buttons are accessible on mobiles (i.e. mainly touch screen), as well as desktop/laptop (mainly mouse/keyboard).
- User input — Make user input requirements as painless as possible on mobile (e.g. in forms, keep typing to a minimum).
- Responsive design — Make sure layouts work on mobile, conserve image download sizes, and think about provision of images for high resolution screens.

Summary of screenreader testing on android and iOS


The most common mobile platforms have fully-functional screenreaders. These function in much the same way as desktop screenreaders, except that they are largely operated using touch gestures rather than key combinations.

Let's have a look at the main two — TalkBack on Android and VoiceOver on iOS.

Android TalkBack

The TalkBack screenreader is built in to the Android operating system.

To turn it on, select *Settings > Accessibility > TalkBack*, then press the slider switch to turn it on. Follow any additional on-screen prompts that you are presented with.


 **Note:** Older versions of TalkBack are turned on in [slightly different ways](#).

When TalkBack is turned on, your Android device's basic controls will be a bit different. For example:

1. Single tapping an app will select it, and the device will read out what the app is.
2. Swiping left and right will move between apps, or buttons/controls if you are in a control bar. The device will read out each option.
3. Double-tapping anywhere will open the app/select the option.
4. You can also "explore by touch" — hold your finger down on the screen and drag it around, and your device will read out the different apps/items you move across.

If you want to turn TalkBack off:

1. Navigate to your *Settings* app using the above gestures.
2. Navigate to *Accessibility > TalkBack*.
3. Navigate to the slider switch and activate it to turn it off.

 **Note:** You can get to your homescreen at any time by swiping up and left in a smooth motion. If you have more than one homescreen, you can move between them by swiping two fingers left and right.

 **Note:** For a more complete list of TalkBack gestures, see [Use TalkBack gestures](#).

Unlocking the phone

When TalkBack is turned on, unlocking the phone is a bit different.

You can do a two-finger swipe up from the bottom of the lock screen. If you've set a passcode or pattern for unlocking your device, you will then be taken to the relevant entry screen to enter it.

You can also explore by touch to find the *Unlock* button at the bottom middle of the screen, then double-tap.

Global and local menus

TalkBack allows you to access global and local context menus, wherever you have navigated to on the device. The former provides global options relating to the device as a whole, and the latter provides options relating just to the current app/screen you are in.

To get to these menus:

1. Access the global menu by quickly swiping down then right.
2. Access the local menu by quickly swiping up then right.
3. Swipe left and right to cycle between the different options.
4. Once you've selected the option you want, double click to choose that option.

For details of all the options available under the global and local context menus, see [☞ Use global and local context menus](#).

Browsing web pages

You can use the local context menu while in a web browser to find options to navigate web pages using just the headings, or form controls, or links, navigate line by line, etc.

For example, with TalkBack turned on:

1. Open your web browser
2. Activate the URL bar
3. Enter a web page that has a bunch of headings on it, like the front page of bbc.co.uk. To enter the text of the URL:
 1. Select the URL bar by swiping left/right till you get to it then double tapping.
 2. Hold your finger down on the virtual keyboard until you get the character you want, then release your finger to type it. Repeat for each character.
 3. Once you've finished, find the Enter key and press it.
4. Swipe left and right to move between different items on the page.
5. Swipe up and right with a smooth motion to enter the local content menu
6. Swipe right until you find the "Headings and Landmarks" option.
7. Double tap to select it. Now you'll be able to swipe left and right to move between headings and ARIA landmarks.

8. To go back to the default mode, enter the local context menu again by swiping up and right, select "Default", then double tap to activate.

 **Note:** See [Get started on Android with TalkBack](#) for more complete documentation.

iOS VoiceOver

A mobile version of VoiceOver is built into the iOS operating system.

To turn it on, go to Your *Settings* app and select *General > Accessibility > VoiceOver*. Press the *VoiceOver* slider to enable it (you'll also see a number of other options related to VoiceOver on this page).

Once VoiceOver is enabled, the iOS basic control gestures will be a bit different:

1. A single tap will cause the item you tap on to be selected; your device will speak the item you've tapped on.
2. You can also navigate the items on the screen by swiping left and right to move between them, or by sliding your finger around on the screen to move around items (when you find the item you want, you can remove your finger to select it).
3. To activate the selected item (e.g. open a selected app), double tap anywhere on the screen.
4. Swipe with three fingers to scroll through a page.
5. Tap with two fingers to perform a context-relevant action, for example taking a photo while in the camera app.

To turn it off again, you'll have to navigate back to *Settings > General > Accessibility > VoiceOver* using the above gestures, and toggle the *VoiceOver* slider back to off.

Unlock phone

To unlock the phone, you need to press the home button (or swipe) as normal. If you have a passcode set, you can select each number by swiping/sliding (as explained above) and then double tapping to enter each number when you've found the right one.

Using the Rotor

When VoiceOver is turned on, you have a navigation feature called the Rotor available to you, which allows you to quickly choose from a number of common useful options. To use it:


1. Twist two fingers around on the screen like you are turning a dial. Each option will be read aloud as you twist further around. You can go back and forth to cycle through the options.
2. Once you've found the option you want:
 1. Release your fingers to select it.
 2. If it is an option you can iterate the value of (such as Volume or Speaking Rate), you can do a swipe up or down to increase or decrease the value of the selected item.

The options available under the Rotor are context-sensitive — they will differ depending on what app or view you are in (see below for an example).

Browsing web pages

Let's have a go at web browsing with VoiceOver:

1. Open your web browser
2. Activate the URL bar
3. Enter a web page that has a bunch of headings on it, like the front page of bbc.co.uk. To enter the text of the URL:
 1. Select the URL bar by swiping left/right till you get to it then double tapping.
 2. For each character, hold your finger down on the virtual keyboard until you get the character you want, then release your finger to select it. Double-tap to type it.
 3. Once you've finished, find the Enter key and press it.
4. Swipe left and right to move between items on the page. You can double tap an item to select it (e.g. follow a link).
5. By default, the selected rotor option will be Speaking Rate — you can currently swipe up and down to increase or decrease the speaking rate.
6. Now turn two fingers around the screen like a dial, to show the rotor and move between its options. Here are a few examples of the options available:
 1. *Speaking Rate*: change speaking rate.
 2. *Containers*: Move between different semantic container on the page.
 3. *Headings*: Move between headings on the page.
 4. *Links*: Move between links on the page.
 5. *Form Controls*: Move between form controls on the page.
 6. *Language*: Move between different translations, if they are available.
7. Select *Headings*. Now you'll be able to swipe up and down to move between headings on the page.

 **Note:** For a more complete reference covering the VoiceOver gestures available and other hints on accessibility testing on iOS, see [Test Accessibility on Your Device with VoiceOver](#).

Control mechanisms

In our CSS and JavaScript accessibility article, we looked at the idea of events that are specific to a certain type of control mechanism (see [Mouse-specific events](#)). To recap, these cause accessibility issues because other control mechanisms can't activate the associated functionality.

As an example, the [click](#) event is good in terms of accessibility — an associated event handler can be invoked by clicking the element the handler is set on, tabbing to it and pressing Enter/Return, or tapping it on a touchscreen device. Try our [simple-button-example.html](#) example ([see it running live](#)) to see what we mean.

On the other hand, mouse-specific events like [mousedown](#) and [mouseup](#) create problems — their event handlers cannot be invoked using non-mouse controls.

If you try to control our [simple-box-drag.html](#) ([see example live](#)) example with keyboard or touch, you'll see the problem. This occurs because we are using code like the following:

```
1 | div.onmousedown = function() {  
2 |     initialBoxX = div.offsetLeft;  
3 |     initialBoxY = div.offsetTop;  
4 |     movePanel();  
5 | }  
6 |  
7 |  
   document.onmouseup = stopMove;
```

To enable other forms of control, you need to use other, equivalent events, for example touch events work on touch screen devices:

```
1 | div.ontouchstart = function(e) {  
2 |     initialBoxX = div.offsetLeft;  
3 |     initialBoxY = div.offsetTop;  
4 |     positionHandler(e);  
5 |     movePanel();  
6 | }  
7 |  
8 | panel.ontouchend = stopMove;
```

We've provided a simple example that shows how make use of mouse and touch events together — see [multi-control-box-drag.html](#) ([see the example live](#) also).



Note: You can also see fully-functional examples showing how to implement different control mechanisms at [Implementing game control mechanisms](#).

Responsive design


Responsive design is the practice of making your layouts and other features of your apps dynamically change depending on factors such as screen size and resolution, so they are usable and accessible to users of different device types.

In particular, the most common problems that need to be addressed for mobile are:

- Suitability of layouts for mobile devices. A multi-column layout won't work as well on a narrow screen, for example, and the text size may need to be increased so it is legible. Such issues can be solved by creating a responsive layout using technologies like [media queries](#), [viewport](#), and [flexbox](#).
- Conserving image sizes downloaded. In general, small screen devices won't need images that are as large as their desktop counterparts, and they will be more likely to be on slow network

connections. It is therefore wise to serve smaller images to narrow screen devices as appropriate. You can handle this using [responsive image techniques](#).

- Thinking about high resolutions. Many mobile devices have high resolution screens, and therefore need higher resolution images so that the display can continue to look crisp and sharp. Again, you can serve images as appropriate using responsive image techniques. In addition, many image requirements can be fulfilled using the SVG vector images format, which is well-supported across browsers these days. SVG has a small file size and will stay sharp whatever size it is being displayed at (see [Adding vector graphics to the web](#) for some more details).

 **Note:** We won't provide a full discussion of responsive design techniques here, as they are covered in other places around MDN (see above links).

Specific mobile considerations

There are other important considerations to think about when making sites more accessible on mobile. We have listed a couple here, but will add more when we think of them.

Not disabling zoom

Using [viewport](#), it is possible to disable zoom, using code like this in your `<head>`:

```
1 | <meta name="viewport" content="user-scalable=no">
```

You should never do this if at all possible — many people will rely on zoom to be able to see the content of your website, so taking this functionality away from them is a really bad idea. There are certain situations where zooming might break the UI; in such cases, if you feel that you absolutely need to disable zoom, you should provide some other kind of equivalent, such as a control for increasing the text size in a way that doesn't break your UI.

Keeping menus accessible

Because the screen is so much narrower on mobile devices, it is very common to use media queries and other technologies to make the navigation menu shrink down to a tiny icon at the top of the display when the site is viewed on mobile, which can be pressed to reveal the menu only when needed. This is commonly represented by a "three horizontal lines" icon, and the design pattern is consequently known as a "hamburger menu".

When implementing such a menu, you need to make sure that the control to reveal it is accessible by appropriate control mechanisms (normally touch for mobile), as discussed in [Control mechanisms](#) above, and that the rest of the page is moved out of the way or hidden in some way while the menu is being accessed, to avoid confusion with navigating it.

See here for a [good hamburger menu example](#).

User input

On mobile devices, inputting data tends to be more annoying for users than the equivalent experience on desktop computers. It is more convenient to type text into form inputs using a desktop or laptop keyboard, than a touchscreen virtual keyboard or a tiny mobile physical keyboard.

For this reason, it is worth trying to minimize the amount of typing needed. As an example, instead of getting the user to fill out their job title each time using a regular text input, you could instead offer a `<select>` menu containing the most common options (which also helps with consistency in data entry), and offer an "Other" option that displays a text field to type any outliers into. You can see an simple example of this idea in action in [common-job-types.html](#) (see the [common jobs example live](#)).

It is also worth considering the use of HTML5 form input types like `date` on mobile platforms as they handle them well — both Android and iOS for example display usable widgets that fit well with the device experience. See [html5-form-examples.html](#) for some examples (see the [HTML5 form examples live](#)) — try loading these and manipulating them on mobile devices. For example:

- Types `number`, `tel`, and `email` display suitable virtual keyboards for entering numbers/telephone numbers
- Types `time` and `date` display suitable pickers for selecting times and dates.

If you want to provide a different solution for desktop, you could always serve different markup to your mobile devices using feature detection. See [input types](#) for raw information on detecting different input types, and also check out our [feature detection article](#) for much more information.

Summary

In this article we have provided you with some details of common mobile accessibility-specific issues and how to overcome them. We also took you through usage of the most common screenreaders, to aid you in accessibility testing.

See also

- [Guidelines For Mobile Web Development](#) — A list of articles on Smashing Magazine covering different techniques for mobile web design.
- [Make your site work on touch devices](#) — Useful article about using touch events to get interactions working on mobile devices.

[← Previous](#)[↑ Overview: Accessibility](#)[Next →](#)

Was this article helpful?



Learn the best of web development



Get the latest and greatest from MDN delivered straight to your inbox.

SIGN UP NOW

