A WEB FOR EVERYONE

Designing Accessible User Experiences

by Sarah Horton and Whitney Quesenbery

Foreword by Aaron Gustafson

Rosenfeld

A WEB FOR EVERYONE

DESIGNING ACCESSIBLE USER EXPERIENCES

Sarah Horton and Whitney Quesenbery

Rosenfeld Media
Brooklyn, New York

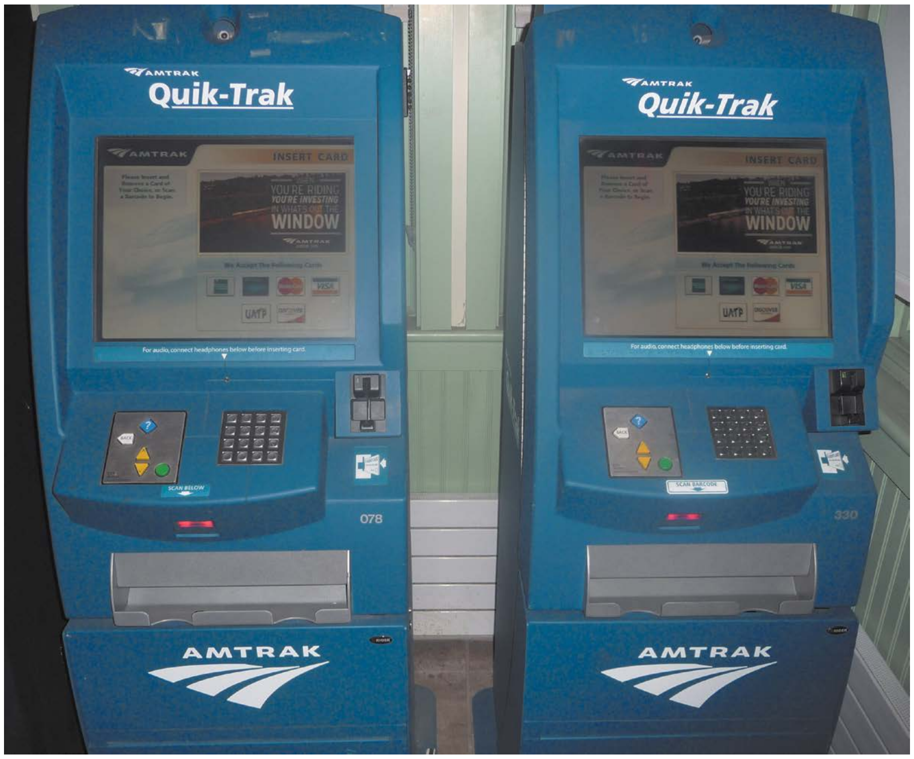
CONTENTS

CHAPTER 5

Easy Interaction: Everything Works

A REAL WORLD EXAMPLE

EZ Access on Amtrak Ticket Machine



Many people use these Amtrak ticketing machines and never notice the EZ Access options, but they are there for those who need them. Source: http://trace.wisc.edu/ez/

There's very little as fundamental to independence as being able to get around on your own, without needing assistance. For many people, getting around means using kiosks to purchase tickets. For trains, buses, subways ... touchscreen kiosks are everywhere. But how do you use one if you are blind?

Enter EZ Access, designed at the Trace Center. By adding tactile buttons and an audio interface, EZ Access makes touchscreens easy to operate for everyone, including people who can't see.

Even better, the EZ Access features are unobtrusive—they are available for those who need them, but do not get in the way of those who don't.

The ultimate design "fail" is a website or app that people can't get to work. By contrast, everyone loves sites that are easy to operate and whose interaction feels intuitive. They can use them successfully with any device or assistive technology, and the site makes them feel confident and capable.

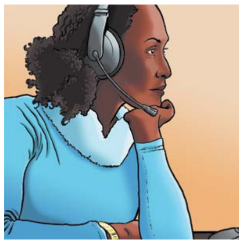
One way to think of interaction failures is as "I can't" moments: "I can't figure out how to submit this form." "I can't select this menu item." "I can't figure out how to enter my phone number in the right format." As users, we want—and hope for—interfaces that are self-explanatory, guiding us as effortlessly as possible. The controls should be easy to understand and use, which leads to "I can" moments.

Creating websites and web applications that people can use might seem like a no-brainer, but it's not an easy task. With the myriad technologies that can be used for building, different devices and software for accessing, different modes of interaction, different user needs and preferences, it's a complex environment for creating sites and apps that don't break. And then there's the pace of change!

How Easy Interaction Supports Accessibility

In Chapter 4, "Solid Structure," we talked about the importance of producing websites and web applications that software, including assistive technology, can read. Now, we'll look at how people interact with websites, and what is needed for accessibility.

Interfaces can be difficult to operate using standard tools, never mind specialized technologies. For web accessibility, easy interaction must take into account different needs.


Lea

Don't make me work so hard.

I love my keyboard. I tried dozens until I found one that fits my hands perfectly, so I hardly have to move to type. Maybe you think I'm a bit over the top, but it makes a difference for me by the end of the day. Using a mouse takes more energy than you think, and I have to conserve mine if I'm going to make it through the day. So do me a favor and let me use my keyboard for everything. OK?

With easy interaction, people can use the product across all modes of interaction and operating with a broad range of devices.

- **Sites don't create barriers, or sites make barriers easy to overcome.** The worst barriers keep someone from using a site at all. A few years ago, there was a debate about whether interactive worlds like *Second Life* were accessible. Many people with disabilities used them, both for the general enjoyment and because the interactive environment made their disability invisible. Even for blind users, it turned out that the most inaccessible part of the interface—the one that kept them from using *Second Life* at all—was the sign-in screen! (*Second Life* has since fixed this problem, marking up the sign-in form correctly.) Even small barriers can drive a user away. These are the sites that "sort of work" if you can just figure out how to get around the speed bumps of things like confusing markup or navigation. As Clayton Lewis put it, "Many barriers to cognitive *accessibility* are the same as *usability* problems for a general user audience ... *but more severe.*"
- **Designs work for people.** When things work well, we hardly notice them. That's why bad designs are so annoying and frustrating. CAPTCHA[1] is an example of a perfect storm of bad interaction—hard for everyone to use, not just people with disabilities. CAPTCHAs try to make sure that a real person (rather than software) is filling out a form, so they display a code word in distorted letters. The idea is that robots or other software can't read the letters, even using character recognition. To make them accessible, CAPTCHAs can include a distorted audio version that is also supposed to be undecipherable by software. Unfortunately, both the visual and audio distortions are difficult for many people, even with good eyesight or hearing. People may have difficulty separating sounds from background noise or spelling words with difficult letter combinations. Ironically, CAPTCHAs aren't even very effective—a spammer can pay for keys to break them for under a penny each.
- **People can choose their own way to interact with a site.** Some disabilities impact dexterity, making it difficult to respond quickly or to operate some kinds of controls. For example, some people work best with tactile controls—buttons and other controls they can feel—while others work best with pointing devices. All of this adds up to

1 CAPTCHA stands for Completely Automated Public Turing test to Tell Computers and Humans Apart, but they can be so frustrating that Jared Smith from WebAIM suggests it should mean Completely Automated Patience Test to Confuse the Hell out of your Audience.

giving people the ability and means to control their own environment, the time and space to work at their own pace in their own way, and the software and hardware that works best for them.

Designing for Easy Interaction

Success in interaction design is largely a matter of following established patterns, so people can apply what they already know to new contexts. Using known and well-established interactive controls goes a long way in designing for easy interaction. There are specific considerations that will help make controls more usable for people using assistive technologies. And there are design considerations that make interaction more usable and enjoyable for everyone, including people with disabilities.

Identify and describe interactive elements

Interactive elements should be easy to distinguish from other elements on the page. For example, links and buttons can be identified in the following ways:

- **Visually.** Links are often colored and underlined, and buttons are identifiable by shape.
- **In code.** Link and button markup codes distinguish these elements, making it possible for browsers to identify them.
- **Through interaction.** Links and buttons can show their state, such as when they are active, through changes in their appearance. They can also be accessed through the keyboard or in lists of interactive elements constructed by assistive technology.

Between HTML, WAI-ARIA, and features of the technology platform, there are many options for providing accessible interactivity by using code to identify and describe interactive elements.

Use basic HTML codes correctly

In addition to coding interactive components, you can also describe their function programmatically. HTML has codes that help software communicate information about components to users.

With basic HTML, interaction is limited to links and form controls. The codes you use to provide interaction include the attributes needed to make the elements accessible. Implementing those elements fully, according to specification, goes far in providing accessible interactivity. Take, for example, a label for a text input field, as shown in Figure 5.1.

Visually, the label is related by proximity, usually appearing right before the field. In code, the label is related using the `<label for>` element and attribute, which programmatically connects the label with the input field. That way software can tell the user which type of information to enter into the field.

FIGURE 5.1
Labels are visually associated by proximity with text input fields. In code, labels are programmatically connected using the `<label>` element, the "for" attribute, and the input field's "id" attribute making the connection.

```
<label for="firstname">First name:</label>

<input type="text" id="firstname" />

<label for="lastname">Last name:</label>

<input type="text" id="lastname" />
```

Use WAI-ARIA for complex elements

Until recently, there was not the same built-in accessible support for complex, page-level interaction as there is for links and forms. But that is finally changing. With WAI-ARIA, you can identify and describe interactive elements in a way that software can read, so it is accessible to users of assistive technology.

For example, one interaction pattern is the "accordion" widget, which is a link that, when clicked, expands to show hidden content (see Figure 5.2). Clicking a second time collapses the content back to its hidden state. This pattern is helpful for content that may not be relevant to all users. It saves precious screen real estate, and also provides a way to learn more in context, without jumping to different pages or scrolling through one long page.

ARIA provides codes you can use to identify and describe interactive components like an accordion widget programmatically so that assistive technology can communicate information about the component to users. For example, in the case of an accordion widget, the "aria-expanded" attribute can be set programmatically to "true" or "false," depending on the state of the component.

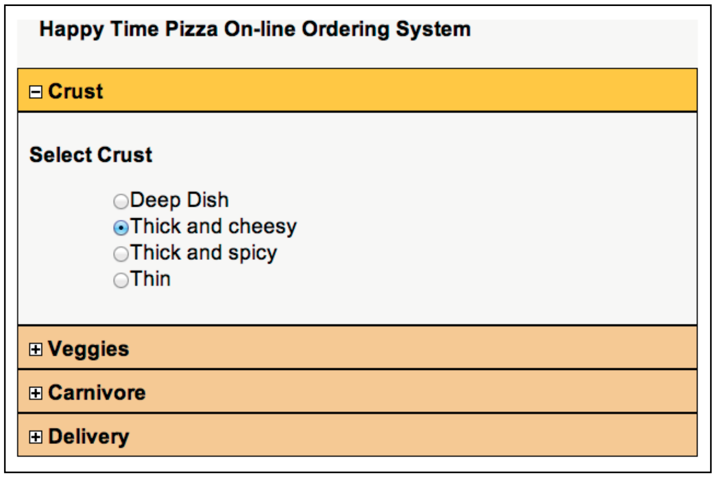
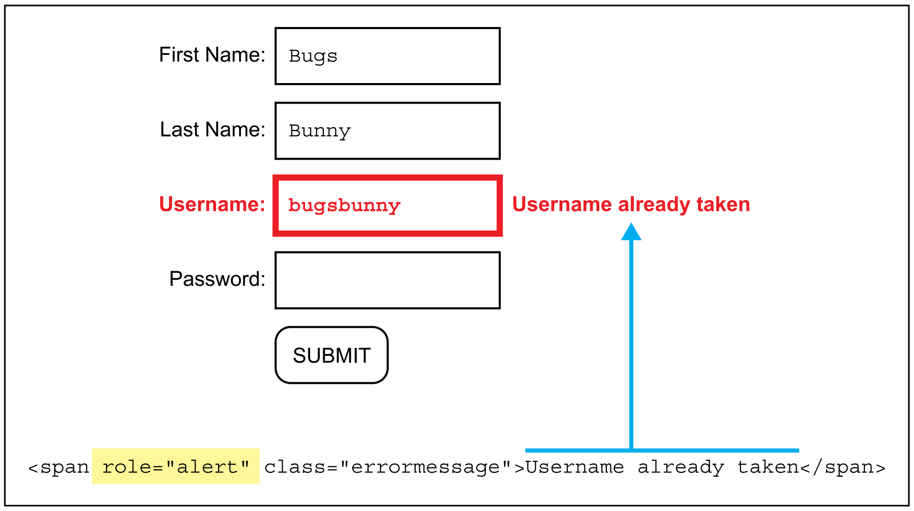


FIGURE 5.2
The OpenAjax Alliance (www.oaa-accessibility.org) provides examples and downloadable code for many common design patterns, including an accessible accordion widget. (www.oaa-accessibility.org/examplep/accordian1/)

ARIA is helpful for other interactions as well. For example, in the sample sign-up form, shown in Figure 5.3, error messages are coded with the attribute `role="alert"` so that the helpful in-line error messages can be announced to screen reader users.



First Name:	Bugs
Last Name:	Bunny
Username:	bugsbunny
Password:	

```
<span role="alert" class="errormessage">Username already taken</span>
```

FIGURE 5.3
In this sample account sign-up form, alerts are identified programmatically using the ARIA role attribute.

Use features of the technology platform

When you are coding elements using programming other than HTML, you should use the features of the technology to fully identify and describe interactive elements. For the most part, technologies like Flash and Java have the necessary hooks for accessibility. Those who develop

Guidelines for Rich Internet Applications in Flex, Flash, and Silverlight

The websites for financial service companies often include complex financial information, including real-time data, price charts, and information to help investors manage their money. The guidelines below are excerpted from Ann Chadwick-Dias and Marguerite Bergel's work with product development teams to make sure that their applications are accessible, especially for older customers managing their retirement funds.

1. Clearly indicate and manage focus.
    - Ensure users can visually track their focus when keyboard navigating.
    - Shift focus into layers as they open; return focus to the originating link once closed.
    - Keep users' focus on controls as they operate them (expand/collapse toggle).
2. Help users know where to begin and reinforce where they are.
    - Use style changes for selected menu items, buttons, and visited links in content pages.
    - Add a button that simulates the browser's Back button, if needed.
    - Mirror the page's information hierarchy in the code using proper semantic markup.
3. Clearly indicate what is interactive and *how* to interact with it.
    - Offer clear instructions or demos, where appropriate.
    - Change cursors and styles for interactive controls on hover.
    - Use active terminology on interactive elements.
    - Use ARIA to communicate custom controls' role and state to assistive technologies.

using these tools just need to use them correctly and to design so it is possible to code the interaction accessibly.

But you should consider the ramifications carefully before moving away from standard technologies. Is the interaction necessary to the purpose and goals of the product? If so, can you accomplish what's needed using standard coding? Exhaust the possibility of using standard web technologies before you make a commitment to a non-standard, and therefore less stable and accessible format.

Provide accessible instructions and feedback

In Chapter 4, *Organize code for clarity and flow*, we discussed how some modes of interaction rely on linearized access, and the code order

4. Progressively reveal content.
    - Draw attention to changes (fading colored backgrounds, data loading indicators).
    - Change content downstream of users' focus.
    - Use ARIA live regions to announce updated content to assistive technologies.
5. Design forgiving controls and inputs.
    - Use large click, grab, tap, drop targets. Wrap supporting icons in adjacent text links.
    - Don't make controls too sensitive, requiring fine motor control.
    - Add redundant text entry fields next to sliders.
6. Let users control movement on the page.
    - Don't auto-play A/V content or loop animations users can't control.
    - Warn users in advance if links launch A/V content.
7. Respect users' settings.
    - Make fonts responsive to browsers' text size controls (e.g., IE's View>Text Size).
    - Ensure that content wraps and containers adjust as font sizes scale.
    - Don't override color display settings users set in their browser or operating system.

matters because it affects the order in which elements are presented. For example, the audio mode of a screen reader cannot present more than one piece of information or interactive option at a time. Interactive elements that are not sequenced correctly can create barriers for everyone, but especially for people relying on a linear presentation.

It's not the details of the interaction itself that create the barrier, but how it is structured in the code and presented to users. A simple rule of thumb is to design the page so that any changes made after it loads the first time happen "downstream" of the cursor.

The location in the code makes a real difference to the accessibility of forms and error messages. For example, as a user fills in a form, the code checks each entry to be sure it is valid. It might check to see if a username is available. But, if feedback is displayed *above* the field, the assistive technology doesn't see the change, and it simply proceeds to the next field. Even worse, some forms display error messages in a "modal" pop-up window that requires the user to close the window before correcting the errors. With the error messages no longer displayed, the user must try to remember the list of problems while looking for the fields to correct.

Or it could be that, after a user submits a form, software on the server identifies a problem with the submitted data and redisplays the form so the errors can be corrected. In some cases, the program positions the cursor in the field in question. Unfortunately, the error message is displayed at the top of the page. Not only will assistive technology not see this message, but most users won't see it either.

So as you are designing forms, you should make sure that any interactive feedback appears both in the code and on-screen in a way that makes sense when linearized. Most often, this means putting the inserted feedback after the element, so it is the next thing in the tab and reading order, as well as marking it with an ARIA role, as shown in the sample sign-up form in Figure 5.3.

Sequencing also matters for instructions. Sometimes, forms are coded so that instructions and labels appear after the form fields and buttons. Users (and assistive technology) have to read ahead to determine the purpose of each field and then backtrack to fill in the field. Be sure that the elements in a form follow a logical sequence: identify and describe an element before the interaction, both visually and in the code.

Instructions and labels that appear inside the field are problematic because they disappear when the field is activated. Users who need to look at the keyboard as they type will miss the hint entirely. Others won't remember the details in the instructions and labels once they are no longer displayed.

Carol

I don't understand what the screen is saying.

I love seeing photos of my grandchildren, particularly since they live so far away. My granddaughter set up a place where she can put up pictures and notes for me. I was excited, but it took me three tries and a phone call to get me connected. I thought I filled in all the answers correctly, but the same questions kept appearing. I'm sure that program was trying to tell me what to do, but I just couldn't understand what the screen was saying.

Support keyboard interaction

The point-and-click interaction model popularized by the mouse is *not* universally usable. Nonvisual users cannot see to point the mouse. People with dexterity issues may find mouse operations awkward and cumbersome. Some alternative input devices work by activating keyboard commands instead. Also, some people find keyboard control easier, more comfortable, and more efficient than pointing.

Provide a logical tab order

In Chapter 4, we talked about how the code order affects linear access to web pages in *Organize code for clarity and flow*. Code order has a significant impact on keyboard navigation, especially when using the tab key to cycle through actionable elements (interactive controls like buttons and links) on the page. Tabbing is a common navigation approach for keyboard users, similar to how mouse users will look for and click on links and controls. Keyboard users will press the tab key repeatedly until arriving at the desired element and then press Enter to activate the control.

With standard web pages, tab order is based on the sequence of elements in the code order. Other formats use other methods—for example, Flash calculates tab order based on the location of elements on the screen. In either case, it's important to test tab order to make sure it follows a logical progression.

While it's possible to manually set tab order in code, the best approach is to sequence elements appropriately, so the natural tab order works in a logical and usable fashion.

Don't require point-and-click interaction

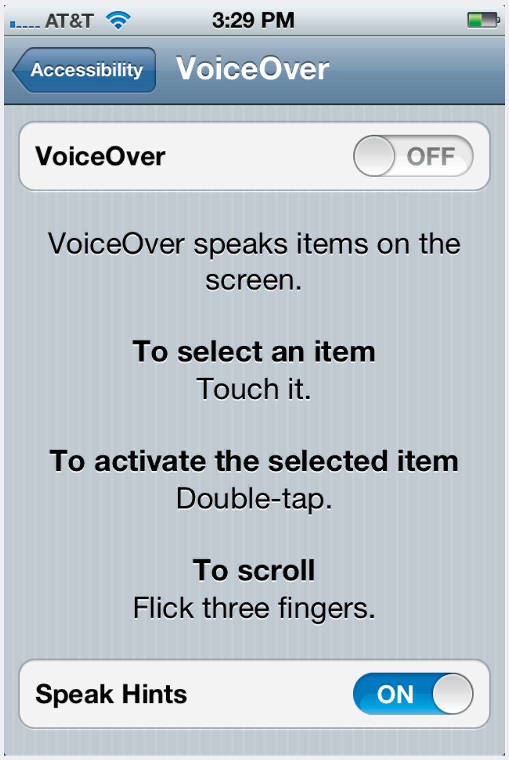
Supporting keyboard interaction doesn't mean that you can never use complex interactions like drag and drop. Just make sure that all interactions have an option that does not require pointing.

Here are some things to keep in mind when designing interactions:

- **Hover:** Some devices do not support hover, such as touchscreens—hover all you want over a touchscreen, and nothing is going to happen! Hover actions can also be annoying when they are triggered inadvertently, such as when a menu is displayed simply because the mouse pointer crossed it on the way to another part of the screen. Hovers can also be distracting to people with cognitive or attention disabilities.

From Barrier to Best-in-Class

When the Apple iPhone first came out, the disability community was dismayed: there were almost no tactile controls, making the phone impossible for the blind and people with dexterity disabilities to use. But today, Apple's touchscreen devices—iPad and iPhone—have gained a following in the blind community because the interface is now designed with voice and gesture input and a built-in screen reader, called *VoiceOver* (see Figure 5.4).



VoiceOver OFF

VoiceOver speaks items on the screen.

To select an item
Touch it.

To activate the selected item
Double-tap.

To scroll
Flick three fingers.

Speak Hints ON

FIGURE 5.4
VoiceOver speaks what is tapped, and activates whatever item currently has focus on a double-tap.

With the success of VoiceOver, other companies are also building accessibility into their devices, although none have reached the level of VoiceOver. What's new, even revolutionary, about these features is that they are built into the platform operating systems, so they can be turned on and off without any special hardware or software.

When accessibility features get this easy, they start to become universal design. Both VoiceOver and Explore-by-Touch allow eyes-free use of the device for anyone whose eyes are otherwise occupied. Siri and other voice activation features are innovative uses of voice technology that let everyone interact with their devices easily.

- **Select:** Using "select" to trigger actions is problematic for keyboard users because events are activated inadvertently as soon as they are selected. The best approach is to use a select/activate model of interaction, where elements are selected and identified, and then explicitly activated by the user. Using this model, you can build one interaction mode that works universally.
- **Drag and drop:** This style of interaction makes direct manipulation of objects easy, but typically requires a pointing device and dexterity. Instead, you can offer a keyboard-accessible alternative way to move items from one place to another (see Figure 5.5).

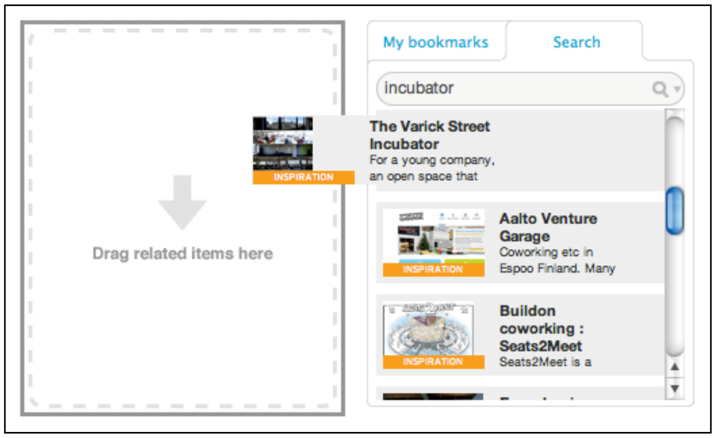

My bookmarks
Search
incubator
Drag related items here
The Varick Street Incubator
For a young company, an open space that
INSPIRATION
Aalto Venture Garage
Coworking etc in Espoo Finland. Many
INSPIRATION
Buildon coworking : Seats2Meet
Seats2Meet is a
INSPIRATION

FIGURE 5.5
This feature, collecting bookmarks for related items, requires a mouse to drag and drop items into the list. A simple Add button would make this more accessible.

Show which element has keyboard focus

Keyboard users also benefit from a clear indicator showing which element currently has focus. Browser software supplies a default focus indicator—typically a dotted outline around the element. However, keyboard usability can be improved by using CSS to provide stronger visual cues to help users make deliberate choices about which elements to interact with. Figure 5.6 shows an example of CSS code. Best practice is to provide the same visual cue as provided to indicate hover—for example, when the mouse or other pointing device is "hovering" over an element.

```
a:hover, a:active, a:focus { outline: 2px solid blue }
```

FIGURE 5.6
Use CSS to provide a visible outline around items that have keyboard focus. In this example, the same 2 pixel blue outline identifies which item is moused over (hover), active, or has keyboard focus.

The User's Finger Is the Center

We all love our smartphones, but how can someone who can't see dial the phone on a touchscreen?

The answer is the Talking Dialer, a free Android app from the Eyes-Free project (including T.V. Raman from Google). It cleverly redefines the interaction from the perspective of the user, rather than the device. Here's how it works (see Figure 5.7):

- Just touch the screen anywhere: that's where the "5" button is.
- Slide your finger up, down, left, right or diagonally to reach another number, based on the layout of a phone keypad.
- Talking Dialer announces the number.

When you have entered the whole phone number, tap anywhere at the bottom of the screen to start the call.

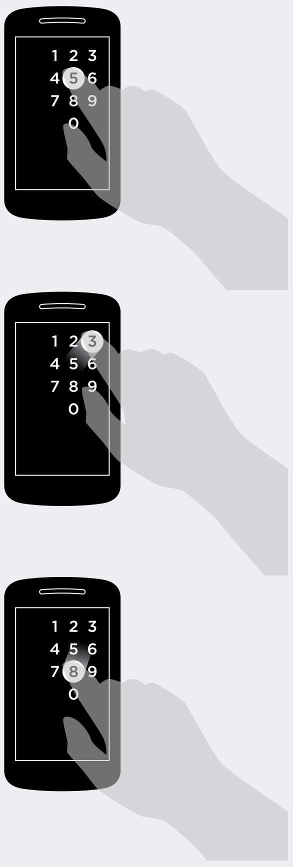


FIGURE 5.7
Starting from where your finger touches the screen, sliding it in any direction selects a number. For example, go to the left for a "4" or up diagonally to the right for a "3."

Don't trap keyboard focus

"Keyboard trap" can happen with embedded objects, such as videos, applets, and Flash. When the focus is trapped, users can't get in or out of an element without a pointing device, like a mouse. Techniques for avoiding keyboard trap are dependent in large part on the technology of the embedded object. Ideally, entering and exiting an embedded object uses the same navigation methods as a web page—namely, tabbing and arrow keys. For technologies that do not support standard navigation, provide a keyboard option and document it so that keyboard users can avoid getting trapped.

Make controls large enough to operate easily

Controls on-screen may not be three dimensional, but users still need dexterity to operate them. Physical issues from arthritis to tremors can make it hard to accurately use a control, but so can context like working on a crowded airplane without enough elbow room, or even wearing

Don’t Make Your Users Build a Frankenkindle

Glenn’s sister had a problem. She loved to read, but her cerebral palsy made handling both books and most ebook readers hard for her. Luckily, Glenn is an engineer with the skills to take on projects to make her life easier.

They found a children’s book reader that she could use, but it didn’t come with books she was interested in. The Kindle was a good start because it had buttons instead of a touchscreen, but the buttons were too small and stiff for her.

A little bit of homebrew electronics later, Glenn created the Frankenkindle, substituting a set of large buttons for the ones that come with the Kindle (see Figure 5.8).

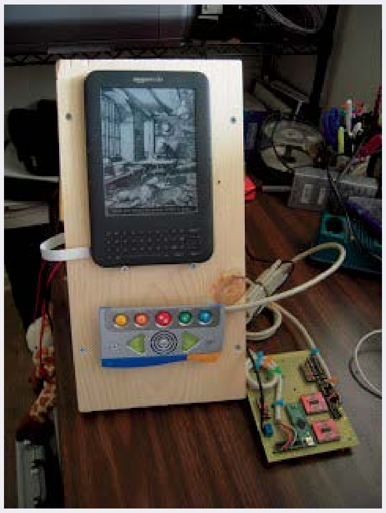


FIGURE 5.8
DIY assistive technology for the Kindle. http://breadboardconfessions.blogspot.com/2011/08/frankenkindle-prototype-demo.html

gloves. People navigating the web using a touchscreen mobile device can run into difficulty trying to, for example, select one of a set of radio buttons, or click a submit button. Even responsive sites may not take into account the differences in size between a pointing device and a fingertip.

The following guidelines help make controls easy to use:

- **Minimize the fine-motor skills needed for interactive elements.** Make buttons and touch points large enough.
- **Space controls.** Put enough space between controls so that users don’t accidentally activate the wrong one.
- **Minimize the complexity of the action required.** Choose controls that do not require timing or managing multiple actions when possible. Watch out for controls like multi-level menus that require a steady hand to operate.

Let users control the operation of the interface

Try to avoid making changes that are not triggered by an explicit user request. For example, the “carousel” of highlighted stories on a home page typically advances automatically, based on an estimation of time needed to get through the content. Uncontrolled motion in an interface is distracting and impacts comprehension. Some users need more time to take in the information, so at a minimum, provide a way to stop the action. (See Chapter 9, “Accessible Media” for more information about multimedia.)

A better approach is to load the first image and provide clear controls for advancing through the stories. This applies to all moving elements, including media such as video and audio. Don’t play media automatically. Instead, wait for users to elect to play the media. Autoplay is not only distracting, but can also cause problems for people in a quiet setting or using a low-bandwidth connection to the web (such as a weak mobile phone signal).

Another common practice is opening links in a new window, usually with the rationale that it will help users return to the originating website because they can just close the window. Unfortunately, opening a new window starts a new browsing history. When users navigate in this new window and try to use the back button to return to the first website, they can’t do so because the first website is not in the history for the programmatically opened window. Indeed, this practice could end up having the exact opposite of the desired effect—in that, users will not be able to find their way back.

Deciding whether to open a new window is a simple illustration of an important principle: **Don’t take actions on behalf of users that they can already accomplish on their own.**

People who like to open links in new windows can achieve this experience on their own, using built-in browser controls. People who do not like opening links in a new window cannot *not* use that behavior if you program it into the interface. As designers, we need to respect the boundaries of the user environment.

Design for contingencies

Like the fire protection and emergency exit systems of a building, digital products must also be built so that when something does go wrong, harmful effects are minimized or prevented through error response and recovery.

Defensive Design

37signals defines defensive design as "design for when things go wrong." In their book, *Defensive Design for the Web*, they define four ways defensive design supports users and helps them recover:

- **Validates** data to check for mistakes before they frustrate the user.
- **Expands** available options based on the user's implied intent.
- **Protects** site visitors from server errors and broken links with informative messages.
- **Assists** the user before mistakes happen.

Errors can occur on many levels. Some, like broken links or programming glitches, are a matter of writing valid code (see Chapter 4, *Code to standards*). Others occur because of confusion about how things work, or through simple mistakes, like clicking on the wrong menu item when your elbow is jostled, or poking a small screen. There is no such thing as a fail-safe system. No interface is intuitive to every user, and no user is on target every time.

Designing for contingencies is about using design to minimize the impact of errors and system failures when they can't be avoided.

For example, you should support users who are submitting information, ordering a product, or posting a comment.

- **Provide a review page.** Allow users to review their input before submitting.
- **Give options for editing the submission.** Support an iterative review/edit process to give users plenty of time and opportunity to be certain about their submission.
- **Provide a confirmation page.** When the information is submitted, confirm the transaction and provide instructions about making any additional changes. Confirmation pages not only provide a nice ending to the interaction, but they also work as conversation:

  User: I'd like to place an order. Here's all my information.

  Your site: Thanks. Got it. We'll send this to you within three days.

Good communication can also make the system easier to operate and to avoid errors. For example, if the system requires a specific date format, provide an example date right before the input field.

If an error does occur, provide helpful and accessible feedback in response to input errors. The feedback should appear with the element containing the error, and should provide clear instructions for how to correct the input, as shown in the sample sign-up form in Figure 5.3.

Allow users to request more time

Time is a challenge for many people. It may take more time for someone using assistive technologies, such as a screen reader, text enlarger, or alternative input device like a joystick. They may read more slowly or need more time to think about what they are reading. Other people need time to simply move their muscles, and it may take a long time to get their arm and hand to coordinate to interact with a link, button, or field. Time-outs can ruin their experience.

Some websites have features that are triggered by time—a common example is the timeout feature used for security reasons by many web applications. When a user logs into the system, the system notes the time and watches the activity. If a predetermined time passes with no activity from the user, the system times out and logs the user off.

A well-designed timeout process alerts users prior to logging them off, and provides them with the option to continue the session. Also, if the system ends up indeed logging off the user, it caches whatever activity they had initiated in the browser. This allows the user to log back in and pick up where they left off.

Who Is Responsible for Easy Interaction?

Design and development collaborate on easy interaction. It is possible to create interactions that are both innovative and accessible, but it takes coordination. Working together to come up with ideas and ways of coding the design can take experimentation, prototyping, and cleverness in solving problems.

When the design team understands what's possible within the constraints of the medium, and the developers understand what's required for universally usable interaction, the results are more likely to be accessible for everyone.

WCAG 2.0 and Easy Interaction

The guidelines for Easy Interaction map to the following WCAG 2.0 requirements.

A site with easy interaction is **Operable**, coded to support a variety of interactions, such as a mouse, keyboard or assistive technology. It also ensures that users can both **Perceive** and **Understand** how to interact with the site.

A site with easy interaction meets the guidelines:

- **2.1 Keyboard Accessible:** Make all functionality available from a keyboard (Guideline).
- **3.2 Predictable:** Make web pages appear and operate in predictable ways (Guideline).
- **3.3 Input Assistance:** Help users avoid and correct mistakes (Guideline).

The requirements for easy interaction are:

- **1.3.2 Meaningful Sequence:** Programs can determine the correct order of the content (Level A).
- **2.1.1 and 2.1.3 Keyboard:** All functionality can be operated through the keyboard without requiring specific timing for each keystroke with no exceptions (Level AAA) or except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints (Level A).
- **2.1.2 No Keyboard Trap:** The keyboard focus does not get "trapped" in a component (Level A).
- **2.4.3 Focus Order:** Elements on the page receive focus in a meaningful order (Level A).
- **2.4.7 Focus Visible:** Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible (Level AA).
- **3.2.1 On Focus:** The context does not change based only on a component receiving focus (Level A).
- **3.2.2 On Input:** The context does not change when a setting is changed, unless the user has been advised of the behavior before using the component (Level A).
- **3.2.5 Change on Request:** Changes of context are initiated only by user request or a mechanism is available to turn off such changes (Level AAA).

- **3.3.1 Error Identification and 3.3.3 Error Suggestions:** An item with an error is identified and the error is described to the user in text (Level A) and with a suggestion (Level AA) when possible.
- **3.3.2 Labels or Instructions:** Labels or instructions are provided when content requires user input (Level A).
- **3.3.4 and 3.3.6 Error Prevention:** Actions that submit information are reversible and checked for all (Level AAA) or for legal and financial transactions (Level AA).
- **3.3.5 Help:** Context-sensitive help is available (Level AAA).
- **4.1.2 Name, Role, Value:** Interface elements are identified so that their name and role can be read by assistive technology and other user agents. There is a way for the program to set any values that users can set (Level A).

The full text of the WCAG 2.0 requirements can be found in Appendix B.

Summary

Making the interaction easy for people with disabilities is an extension of making interaction easy for everyone. Interactive elements are identified clearly and are designed to be easy to use.

The site supports interaction with a keyboard, allowing assistive technology to emulate the keyboard. This also requires that the keyboard tab order make sense, matching the visual presentation.

A site with easy interaction enables users to control the interface, with large enough controls. It avoids taking unexpected actions for users that they can do on their own. Easy interaction also includes both preventing and handling errors in an accessible way.

Profile: Accessible Interaction with Derek Featherstone

Derek Featherstone is founder and lead of Simply Accessible, a consultancy that works to bring accessibility into organizations through consulting, training, workshops, and support. Derek is a well-known and well-respected advocate for accessible user experience, moving beyond technical accessibility to support good and successful experiences for everyone. Interactivity is a complex challenge for accessibility, making sure that sites and apps are operable in different ways on different devices. We wanted to learn from Derek how best to approach a moving target like accessible interaction.

People are the starting point.

Concern for people is the primary driver for Derek and his colleagues at Simply Accessible. "It has to be about the people who are actually using the site or application to accomplish something. If it's not about them, then what's the point?"

Best accessibility techniques are constantly changing.

Derek's team works from a knowledge base of accessible techniques, built over years of user research and usability testing. From there, they use observation and experimentation to come up with better techniques for coding, designing, and writing content that improves the user experience.

Technical remediation can help make interaction accessible.

Most often, Derek's team is called on to make an existing website or app accessible. Often, the designs are visually rich and appealing, and convey information and relationships among elements in a visual way. However, those visual details are not always in the code.

For example, many social media sharing badges don't have alt text that accurately reflects the contents of the badge. Visually, you can tell whether an item has been liked or shared, but the code doesn't contain the same information. Derek and his colleagues have been working on a script that would make state and status information available in the code where it's available to assistive technology.

> *We know it would be better if these elements were natively accessible. We also know that change doesn't happen right away. We keep our fixes on file so we can use them while the people in charge of social sharing badges work on making them more accessible.*

"Our entire team views accessibility as part of overall user experience and not as a separate thing that needs to be done afterwards." However, in practice, clients often come to accessibility late in the process, when there is little hope of changing the course of the design to produce a more accessible outcome. "What we usually do is say, given what you've designed, here's what you need to do to make it work in a more accessible way."

profile continues on next page

Profile: Accessible Interaction with Derek Featherstone (continued)

Integrated accessibility produces the best outcomes.

But in some projects, they are brought in early and remain part of the team throughout the project. These projects are the most successful. “What it comes down to is having people switched on right from the beginning of the project. They understand accessibility as more than a technical checklist.” With an early commitment, accessibility becomes part of the well-established user experience practice. “One of the most important things to achieve success is to have all accessibility touch points built into the process right from the beginning.”

Tools help teams integrate accessible components.

Derek and his colleagues have had good success providing teams with tools that make it easier to build accessible products. One tool is an accessibility guide, integrated into the organizational style guide or branding guide. Another is a code repository and pattern library containing elements like modal dialogs or tool tips that can be easily dropped into code. “One of our goals is to eliminate excuses. We think about all the different pressures people feel when building a site and try to address them.”

The most influential tool for accessibility is clear purpose.

In cases where Derek has been able to influence design, the best tool in guiding an accessible process has been clear purpose. “Asking the original question, what is the purpose of this page and what are we trying to help people do. And once we get answers, we can start asking questions like, how does somebody who uses voice recognition software because they don’t have the ability to use their hands, how are they going to activate that particular type of control?”

Solutions come from different places.

Derek tells a story of a woman who needed a very specific design approach for accessible interaction. She was a quadriplegic, but had enough strength in one arm to lift her hand and operate a large touchscreen. Her greatest challenges were radio buttons and checkboxes, which were too small for her to activate accurately. Derek created a stylesheet she could install in her browser that would resize radio buttons and checkboxes ten times the size they normally appeared.

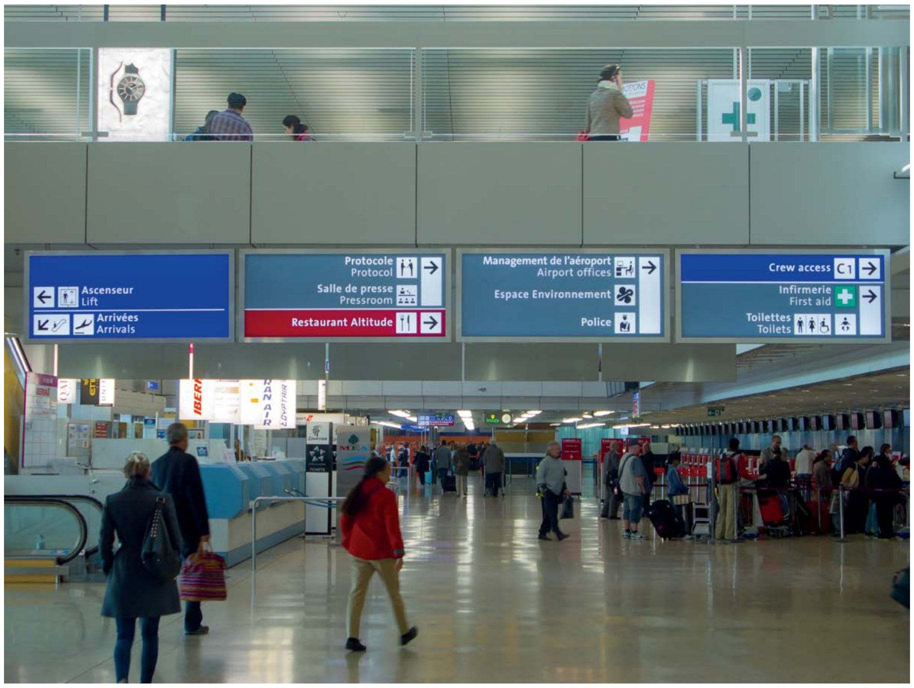
The story illustrates that accessible interaction isn’t about finding the one solution that works for everyone. “We can create one solution that works for most people’s abilities, but there are some people that need accommodation for their disabilities that a designer or developer can’t take into account in their work.” In these cases, the work of the designer and developer is to make the design flexible and adaptable. That way, browsers and assistive technology can take elements of the design and adapt them to meet the specific needs of the user.

CHAPTER 6

Helpful Wayfinding: Guides Users

A REAL WORLD EXAMPLE

Getting Around an Airport



Paul Mijkensanaar designed the wayfinding system for 10 international airports. This photo is from the bilingual wayfinding system at the Geneva International Airport. (www.mijksenaar.com)

In the physical world, we rely on maps, street signs, and how spaces are designed to help us get around. A corridor looks different than a meeting room, and we know one is for passage and the other is for gathering and discussion. Signs identify the routes and destinations, helping us get from here to there.

When signage uses consistent colors, typography, and icons and is placed in a visible location, it can be easy for people to find their way, even around a crowded airport.

In both the online and physical world, people rely on familiar patterns to help them find their way more easily. A mix of conventions and experience helps everyone locate cues in either the physical environment or an online interface, using the design of directional signs or interaction features like menus, buttons, and links. When these features are easily found where they are most expected, it takes less effort to use them. Table 6.1 compares the tools for wayfinding in the online and real worlds.

Wayfinding and orientation are especially important on the web where users have a relatively small window into the site (and an even smaller view on the tiny screens of mobile devices). Every site needs orientation features to help users know where they are, and it needs wayfinding features to help users find their way to the information or function they came to the site for. When these features anticipate users' needs, it makes using a site easier for everyone.

TABLE 6.1 WAYFINDING IN THE REAL WORLD AND ONLINE

Wayfinding in the Real World	Wayfinding Online
Architectural features define the space with paths, doorways, and so on.	Semantic features define the space with areas for information, interaction, orientation, and navigation.
Conventions for the placement of signs help people find them easily. There are often local standards, but they may vary.	Conventions for the placement of types of links or features help people find them easily, but there are no global standards.
Visually distinct landmarks help keep people oriented. (For example, "Let's meet at the clock at the station.")	Visual design, the types of content, and presentation details vary by the type of page, and help people stay oriented as they move around the web.
Signs mark places (shop signs, street signs).	Headings, landmarks, and visual cues mark places.
Signs point out paths to other destinations.	Menus and links point out paths to other destinations.
Signs have colors, shapes, and symbols to go along with the words.	Colors, icons, and typography go along with words to provide multiple cues.
Directories are organized in a logical order (alphabetical, by type, or by location).	Menus and page elements are organized in a logical order.
Space maps provide an overview.	Site maps provide an overview.

Trevor

When I can learn the pattern, I can find my way.

I like games. The ones where you have to find your way around a maze are good because I can go over them, and I can learn how they work. It's OK to get lost and have to figure out a game.

But when I'm trying to find something, like an assignment for school, I don't like getting lost. I want to know where I'm going—because it's easier, and it's easier to find things again when I need them. When it's clear and I can tell where I am, I like the site. It's like learning how to walk to school on my own. I practiced finding my landmarks, so I would know where to turn. I know that it's 500 steps from 1st Avenue to the first street I have to cross. Just like I know what to click on to get to my history class page on my school's website.

How Helpful Wayfinding Supports Accessibility

Wayfinding includes both navigation (features that let users move around the site) and orientation (identifying the current location). A site with wayfinding that works well for both visual navigation and all kinds of assistive technology makes the online environment usable for everyone.

With helpful wayfinding, **people can navigate a site, feature, or page following self-explanatory signposts.**

- **Navigation options make sense.** It can take more effort to navigate a website or web application using assistive technology, which makes every action more important. Links that aren't clear and menus that are confusing can result in ping-ponging among sections and backtracking. Clear wayfinding reduces the chance of making the wrong choice among navigation options.
- **The interface supports exploration.** No one wants to be pushed through a chute with no options. Sometimes the fun is in finding your own way through a site. When a site has good wayfinding, exploration is safe, because it offers clear cues about where each link or menu option leads, and it has consistent ways back.

- **People know where they are.** We all like to know where we are. When people are following a link or diving into the middle of a site, they need to know where they have landed, and what's available on this page. A site designed for easy wayfinding provides strong orientation cues so that everyone can use them, no matter how they are accessing the site.

How to Design Helpful Wayfinding

People rarely go online for the sheer joy of navigating around a site or app. (Perhaps people working on user experience do, but it's not the most usual reason.) The goal in designing navigation is to make it so intuitive that it becomes invisible to the user. With well-designed wayfinding, users can find their way around with minimal effort.

Create consistent cues for orientation and navigation

People tend to navigate partly by using the cues they experience in real time and partly by using a cognitive map, or mental construct, of the space (real or digital) they are navigating. Even small variations can be disorienting.

Although a clear and consistent model is important for any user, it is especially important for people who use screen readers and other technologies that read the page linearly. For linear access, consistent placement of elements and use of consistent semantic markup, including headings, helps users form mental models.

A visual user, for example, might quickly learn that there is a logo in a banner at the top of the page, search in the upper-right corner, and see a colorful title marking the beginning of the main content. A nonvisual user might equally quickly learn that the same title is the second heading on the page and use that model to jump rapidly to content on other pages.

Present things that are the same in the same way

One way to help users find their way around a site is to be consistent in how elements of the site are presented and labeled, which doesn't mean that the site must be boring with no variation or texture. However, elements of the site that form important landmarks should be consistent:

- Describe or label the same thing in the same way each time.
- Be consistent about the location in the page structure of key navigational elements, including menus and links to features like contact pages.

Good Signposting Supports Cautious Clickers

Research with older adults found some distinctly different patterns in how they interacted with websites.

- They were "cautious clickers" who spent more time reading before deciding to click on a link.
- They were more likely to try to click on text that was not linked, hoping that it would let them find the information they wanted.
- They had more difficulty understanding where they were within a site. For example, they would click on the current page link in the left menu.

When the team redesigned pages to accommodate their needs, older adults had less trouble navigating. Their design recommendations were:

- Use action words as links.
- Present links in an obvious and consistent way.
- Include images used as icons or bullets in the link.
- Use simple, clear navigational cues and labels.
- Use secondary or pop-up windows rarely.

"Web Usability and Age: How Design Changes Can Improve Performance" by Ann Chadwick-Dias, Michelle McNulty, and Tom Tullis. http://dl.acm.org/citation.cfm?id=957212

The relationship between links and the pages they point to should also be consistent. Consider these basic rules:

- The text of the link should accurately communicate the page or feature that the link connects to.
- The text of the link and the title of the target page should be similar, if not identical.

Differentiate things that are different

When navigating in the real world, differences help you stay oriented. You recognize the corner where you turn not just by the name of the road, but because you recognize the building on the corner, or because the traffic on the main road sounds different than the quiet side streets.

It's the same on websites. Differences in layout and page elements help identify the page type or site section. For example, when encountering a carousel of big images and bold headlines, you might assume that you are on a home page. A page that is mostly information is probably a content page. And a page with a lot of links is for navigation, or what Ginny Redish calls a "pathway" page.

There can be too much consistency if it blurs important distinctions. People like predictability, so when the same words, images, or buttons do different things, it's disorienting and breaks their mental models.

Provide orientation cues

Orientation—knowing where you are—is an important part of successful wayfinding. When arriving at a page, how easily can you tell where you are and answer questions like "What is on this page?" or "What site is this page part of?" and "Where am I in this site?"

Orientation is important whether the journey starts from the home page or the user has landed in the middle of a site by following a link. In fact, users are more likely to start from the middle of a site than to navigate from the home page. Either way, the site needs clear "You Are Here" orientation cues, integrated into the design, content, and code.

- **Identify the site.** Be sure that it's easy to find the name of the site and to identify the organization behind the site.
- **Title the page.** The page title is the text that appears in the title bar of the browser. It's also the text that displays in a bookmarks list, in search results, and it is the first thing announced by screen reader software. The title tells what the page is about, and it also provides orientation cues. A good convention is to include page title, section name, and site name (in that order) in the `<title>` tag.
- **Provide good headings.** Headings describe the main topic of the page, as well as sections of content. Marking up headings using the correct markup (`<h1-h6>`) makes it easier for people who use assistive technology to find them.
- **Start with an overview.** It's common for users to leave sites if they don't see what they are looking for quickly. "Bouncing" is part of general behavior on the web, but you can help people make an accurate assessment by providing good content overviews.
- **Highlight the current location.** There are many ways to identify where a page fits into a site, from the page title to highlighting the menu item for the section to breadcrumb navigation.
- **Use multiple cues.** Providing more than one cue for orientation helps both visual and verbal thinkers. For example, using an icon with color-coding and a strong text label provides three ways to identify a page or feature.

When describing the interface, avoid using details like color or even location on the screen as the only cues in the instructions. Directions like "in the upper-left corner" don't mean much to someone navigating by audio. It's okay to use color and location along with other nonvisual cues: "The blue link labeled 'Contact Us' in the upper-left corner."

Provide clear landmarks within the page

Once visitors have gotten to a page, they still need to find their way around. Good visual design cues, good headings and other labels, and good underlying structure have to work together to make this process easy for everyone. When they do, the parts of the page are easy to differentiate both visually and in the code so they are available to assistive technology.

One way to provide a navigation map of the page is to use links that jump to specific areas of the page. These are often called *skip links* because they let a site visitor skip over sections of the page to go directly to key locations on the page. This is especially important as a way to skip over repeated blocks of content, like the links and logos in the heading of the page (see Figure 6.1).

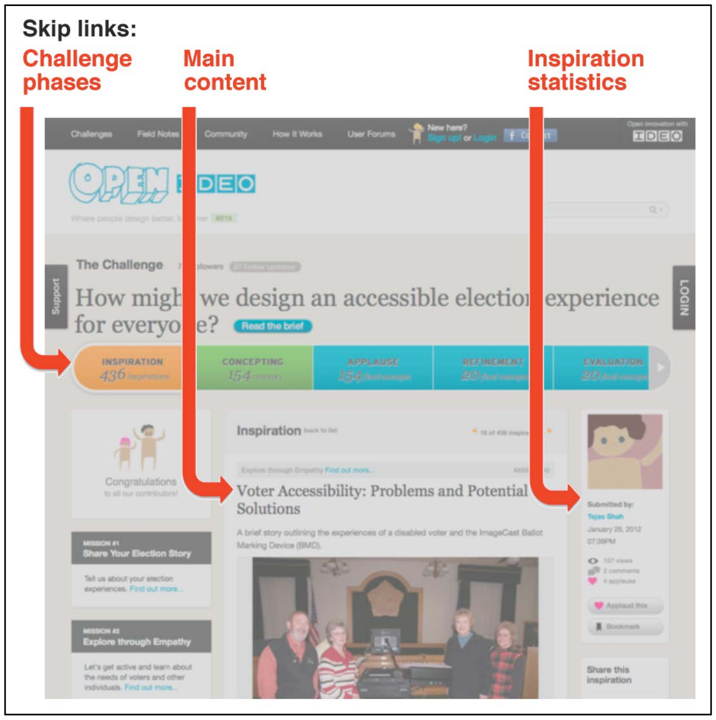
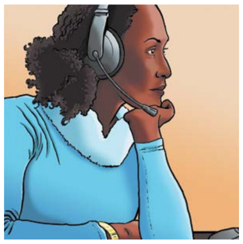


FIGURE 6.1
The OpenIDEO pages are complicated, with many different zones on the page. When the team retrofitted the site for accessible navigation, skip links provided a way to create an easy way to navigate to the key interactions on a complex page.



Lea

Links at the top of the page make navigation easier for me.

I like pages with links at the top of the page. It's really helpful on long pages with a lot of sections. I can figure out what's on the page without a lot of work. When I first saw a link to jump to the content, I didn't know what it was for, but it sure made navigating with a keyboard easier.

A better way to code navigation is with HTML5 elements and ARIA roles. These are complementary ways to identify parts of the page (see Table 6.2 and Figure 6.2). Adding elements and roles makes pages semantically rich by embedding information about the purpose of different elements. This information is then available for software—browsers, search engines, assistive technology—to use to enhance the user experience. For example, screen readers provide controls to move focus among the sections, which lessens the need for skip links as a way to bypass blocks of content. By coding boundaries around elements, you create clear landmarks that do not rely on visual properties, such as outlines or background color.

Because HTML is in a transitional phase, one of the challenges of working with these new standards is that using both elements and roles can create some duplication. For example, using both the `<main>` HTML element and the "main" ARIA role means that some screen readers will announce both, and users will hear, "main main." Over time, many of the ARIA roles will be incorporated into HTML5, and assistive technology

TABLE 6.2 HTML5 AND ARIA NAVIGATION

HTML5 Element	ARIA Role
`<article>`	"article"
`<aside>`	"complementary"
`<footer>`	"contentinfo"
`<header>`	"banner"
`<nav>`	"navigation"
`<section>`	"region"
`<main>`	"main"
	"search"

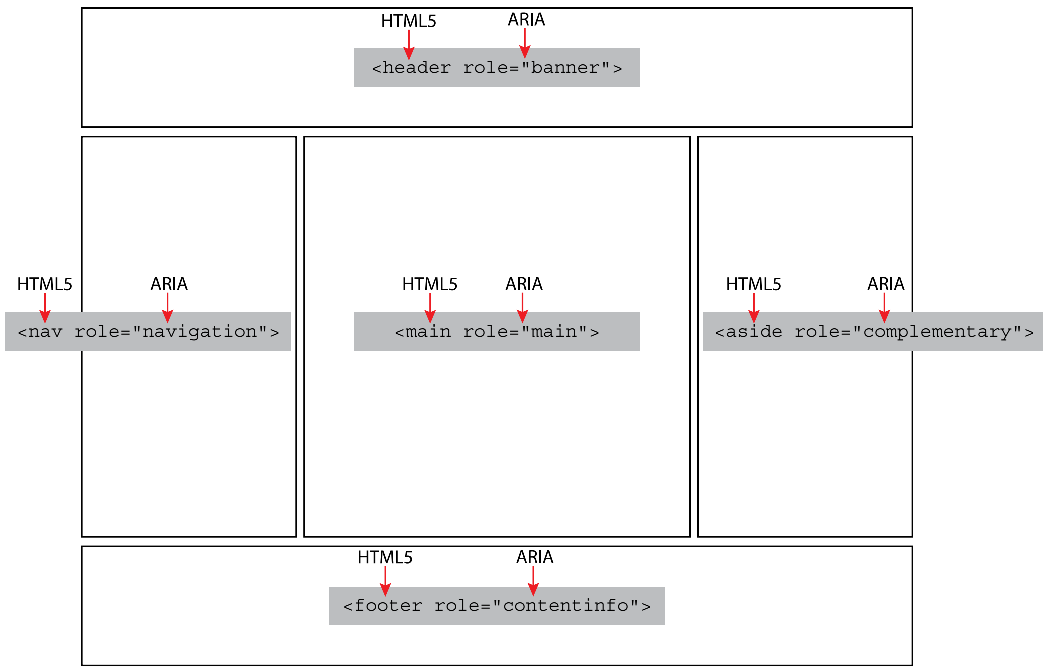


FIGURE 6.2
HTML5 elements and ARIA roles are complementary. Including both of them in your site provides a solid code structure and good navigation around the page.

will work with the standards in a more consistent way. Until then, using ARIA roles and HTML5 elements is the best approach to ensure that this important information is available to all users.

Provide alternative ways to navigate

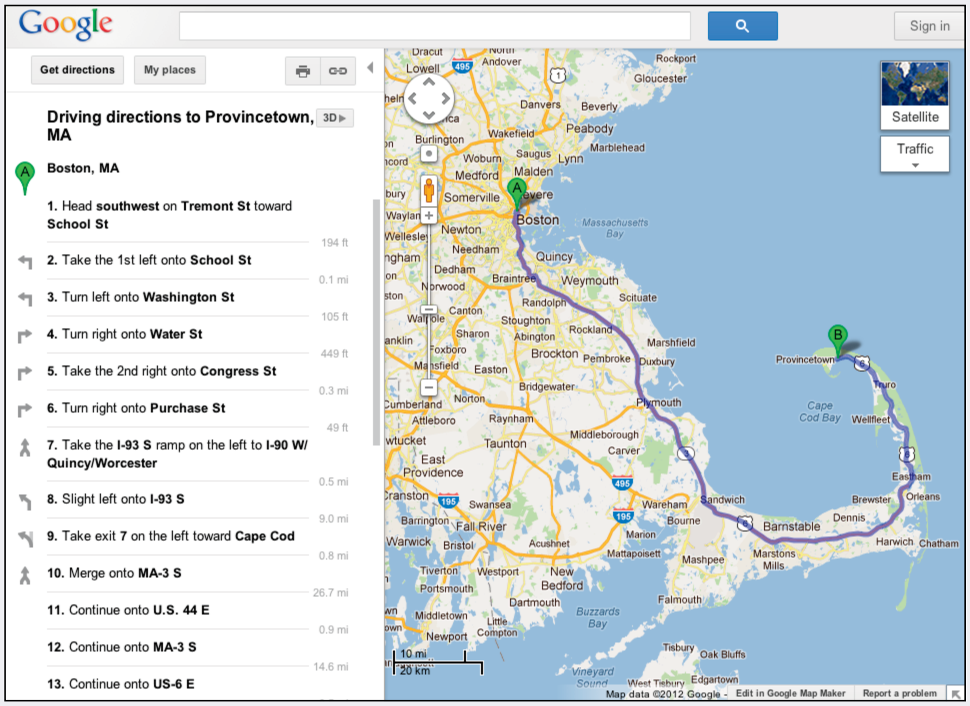
It's important to reiterate: there is no *one way* to provide accessibility. The solution instead is to provide alternatives. For helpful wayfinding, this means offering different navigational options.

Most sites include more than one way to move around the site: menus, links, a sitemap, or search. Providing alternatives improves the chance that people will find what they are looking for.

For many people, given the choice of browsing through menus to explore options versus using search, search wins. For people with disabilities, for whom navigation can be slower and more difficult, search provides the means to jump directly to the right page. Many sites have realized the universal appeal of search and have made it prominent on the home page so it's easy to use to navigate.

Google Maps

You might not realize that the turn-by-turn text directions in Google Maps that so many people rely on started as an accessibility feature (see Figure 6.3).



Driving directions to Provincetown, MA

Boston, MA

1. Head **southwest** on **Tremont St** toward **School St**

194 ft

2. Take the 1st left onto **School St**

0.1 mi

3. Turn left onto **Washington St**

105 ft

4. Turn right onto **Water St**

449 ft

5. Take the 2nd right onto **Congress St**

0.3 mi

6. Turn right onto **Purchase St**

49 ft

7. Take the **I-93 S** ramp on the left to **I-90 W/ Quincy/Worcester**

0.5 mi

8. Slight left onto **I-93 S**

9.0 mi

9. Take exit **7** on the left toward **Cape Cod**

0.8 mi

10. Merge onto **MA-3 S**

26.7 mi

11. Continue onto **U.S. 44 E**

0.9 mi

12. Continue onto **MA-3 S**

14.6 mi

13. Continue onto **US-6 E**

FIGURE 6.3

The original interface to Google Maps was entirely visual. Users would explore the map, including all the rich information about local features, and follow the guide on the visual map to get from place to place.

T.V. Raman, a research scientist at Google, created the first Textual Maps UI. As he wrote in his blog in 2006, "When using spoken output, this visual richness can get in the way of quickly listening to the results of a maps query."

His solution was an alternative interface that "serves up directions very efficiently when working with a screen reader or a braille display. … It's extremely useful for blind and visually impaired users, as well as an effective solution for those times when you're at a nongraphical display and need to quickly look up a location. Just type a simple English query of the form start address to end address and quickly get the information you're looking for. Though we added this option to enhance the accessibility of Google Maps for blind and low-vision users, perhaps others will find this alternative view a useful addition to their maps arsenal."

You can read his blog entry from December 26, 2006: http://googleblog.blogspot.com/2006/12/speech-friendly-textual-directions.html

Who Is Responsible for Wayfinding?

Creating helpful wayfinding relies on both good design practices and user research. Understanding how your audience thinks about the information architecture is critical.

The design and content teams are responsible for strong information architecture, backed up by good user research. The teams should look for opportunities to help users make their way through the space and provide clear, distinctive landmarks to help them stay oriented.

In addition, it's important that wayfinding elements in the design and content are supported by the code, with landmarks that provide alternative ways to navigate a site or application.

WCAG 2.0 and Helpful Wayfinding

The guidelines for Helpful Wayfinding map to the following WCAG 2.0 requirements.

A site with helpful wayfinding is **Operable** and **Understandable**, with landmarks for orientation and wayfinding presented both in the content and code.

Wayfinding relies on being:

- **2.4 Navigable:** Provide ways to help users navigate, find content, and determine where they are (Guideline).
- **3.2 Predictable:** Make web pages appear and operate in predictable ways (Guideline).

The requirements for helpful wayfinding are:

- **2.4.1 Bypass Blocks:** Users can bypass blocks of repeated content (Level A).
- **2.4.2 Page Titled:** Web pages have titles that describe topic or purpose (Level A).
- **2.4.4 and 2.4.9 Link Purpose (In Context or Link Only):** The purpose of each link can be determined from the link text alone (Level AAA) or from the link text together with context (Level A).
- **2.4.5 Multiple Ways:** More than one way is available to locate a web page within a set of web pages, except for pages that are a step in a process (Level AA).

- **2.4.6 and 2.4.10 Headings and Labels:** Headings and labels describe topic or purpose (Level AA) and are used to organize content (Level AAA).
- **2.4.8 Location:** Information about the user's location within a set of web pages is available (Level AAA).
- **3.2.3 Consistent Navigation:** Navigational mechanisms that are repeated on a set of web pages are presented consistently (Level AA).
- **3.2.4 Consistent Identification:** Components that have the same functionality within a set of web pages are identified consistently (Level AA).

The full text of the WCAG 2.0 requirements can be found in Appendix B.

Summary

Helpful wayfinding depends on consistent, understandable cues that help users know where they are (orientation) and how to get where they want to go (navigation).

Some aspects of wayfinding are part of the content and presentation design: titles for pages, naming things that are the same in the same way each time (and differentiating things that are different), and including information about where users are in a page, site, or process.

Helpful wayfinding also relies on cues in the code, including ARIA landmarks, which act as descriptive landmarks for navigating around the page, and descriptive coding for navigation elements.

Profile: Coding Accessibility with Steve Faulkner

PATRICK H. LAUKE

Steve Faulkner has been an accessibility engineer since 2001, first with Vision Australia and currently with The Paciello Group. He has a hand in developing HTML5 and WAI-ARIA specifications as a member of W3C working groups, and is editor of W3C specifications on HTML5, Using ARIA in HTML, accessibility APIs, and text alternatives. In short, Steve has accessibility chops.

Because much of what's needed is beneath the surface of a page, we asked Steve to explain what user experience designers should know about how code supports accessibility.

Elements of an accessible user interface.

Web accessibility is largely about providing the information needed to make a user interface accessible to assistive technology (AT). This is accomplished via an accessibility application programming interface (API)—a standardized way of specifying elements of an interface. For accessibility, Steve explains, "One piece of the puzzle is for information to be defined by an accessibility API. The second piece of the puzzle is for AT to make use of it."

For the web, every HTML element has a role, states, and properties that are defined in the technology specification—"what it is, what it does, and where it's at, at this point in time." That information is expressed by the accessibility API in the browser so that assistive technology can make use of it. "For example, with a heading, one of the properties might be its level, and whether it's editable, visible, or linked." Information in the accessibility API allows assistive technology to, for example, create a list of headings or move keyboard focus to the main heading of a page.

An accessibility API needs more than HTML.

There are gaps in how native HTML elements support the accessibility API. "Some decisions about what goes into HTML and what doesn't don't take into account accessibility requirements." For example, the role "main" is not represented in the HTML5 specification as an element, despite the expressed need for AT users to move cursor focus to the main content area. Other roles are defined, such as header, footer, and navigation: "I guess the thinking was, anything that's not something else must be the main content."

Also, HTML elements are not always used correctly. As an example, Steve explained that HTML buttons have been part of the specification for 15 years. However, developers build custom buttons using elements like images and links that are coded to behave like buttons. Typically, this is because "They have to please whoever makes the design decisions. To achieve a certain visual effect across different browsers and platforms, they can't always use standard controls." In these cases, "WAI-ARIA is the only way to supplement the necessary information for assistive technologies."

WAI-ARIA fills the gaps.
WAI-ARIA fills out the accessibility API, enhancing the descriptive information contained in native HTML controls and back-filling information for elements that are used for other than their intended purpose.

For example, ARIA provides explicit roles and properties to assist in wayfinding for users of assistive technology. Like "stepping stones," ARIA landmark roles allow assistive technology users to step through the content of the page to find the content area of interest.

HTML5, ARIA, or both?
In general, support for ARIA is more robust than accessibility support for new HTML5 features, because it's been around longer and has benefited from successful collaborations between accessibility experts and software vendors. "I'm not saying that it's good across the board, mainly due to some AT vendors just putting their heads in the sand, which has the effect of doing a disservice to their users." And you can count on support for ARIA in the future. "As far as ARIA is concerned, it will remain relevant for many years to come. As new issues and technologies emerge, there will be new related updates to the ARIA specs to fill in any gaps."

It's best to think of ARIA as complementary to HTML. "A good rule of thumb is, when there's a native HTML structure, element, or attribute that's well supported, use it. If that's not enough, use the appropriate ARIA semantics."

Advice for project teams.
Steve encourages developers to add ARIA landmark roles. "The good thing about landmarks is that you can add them to your current code, and they don't have any design effects." For interactive widgets that provide complex interactions, he recommends looking to existing libraries, such as JQuery or Dojo—code with WAI-ARIA already built in.

But he also sees the need for a change in mindset in how we build websites and applications. Project efforts often focus on the business transactions that occur on the back end, with little thought given to the user interface. "You tend to find back-end coders developing front ends without understanding what makes a user interface usable, let alone accessible."

On the design end, Steve urges designers to understand that design elements are not just "pixels on a page," but rather semantic containers for information described in code. "I don't think designers need to know how to code. They do need to understand that there's a give and take between what can be done, given the requirement to actually code."

Overall, he believes project teams must include user interface expertise. "What's needed is the realization within a team of the value of having people who understand usable UI design and can bring it to fruition using the code."