The PDF Expert

White paper

A look at the current state of accessibility of electronic forms, and an overview of a new model for accessible PDF forms developed during a two year research project for the Government of Canada.

Bryan Guignard

Introduction

Despite tremendous advances in electronic forms technologies, forms remain mostly unaccessible to the vast majority of people who use assistive technologies. This paper will introduce you to some of the problems and offer insights into what may well be the most powerful and cost effective accessible forms solution available today.

What is accessibility?

From: http://outreach.missouri.edu/hes/graphics/graphicterms.htm

As specified in Section 508 of the 1998 Rehabilitation Act, the process of designing and developing Web sites and other technology that can be navigated and understood by all people, including those with visual, hearing, motor, or cognitive impairments. This type of design also can benefit people with older/slower software and hardware.

Portable Document Format (PDF) forms are built around the paper metaphor, with its two dimensional, four sided, flat sheet style. It's even been referred to as *ePaper* by its creator, Adobe Systems. Despite its tremendous benefit to mankind, sheets of flat paper remain unaccessible to blind persons. Electronic technologies based on paper concepts, such as PDF, can overcome many of the accessibility limitations of paper by adding electronic features, but can still be challenging to many due to its popular page sizes, which can accomodate great quantities of static and interactive information on a single page. This problem is compounded when PDF forms span multiple pages. Forms that span multiple pages can lead to procedures that jump unexpectedly to other pages, and in the process a new level of complexity is added.

PDF forms are in the unfortunate situation of being a victim of their own remarkable richness and versatility. Coincidentally, it is these qualities that allow authors and designers to create accessible forms, yet they can lead to an excess of good things, which can be detrimental to usability. No matter which definition of accessibility you read (and there are many), none of them account for every design requirement needed to make a PDF form usable.

Consider these two common scenarios.
Scenario 1: a constituent downloads an electronic PDF tax form, opens it, fills it out, and forwards it to the government in the prescribed fashion. The constituent spends about 2 hours, once a year, performing this task. If the form could be made more efficient, so as to shave off 10 minutes off the time it takes to fill it out, the constituent may notice the increase in efficiency, but it would hardly be a significant matter due to the low number of times the form is used by the constituent.

Scenario 2: a government employee at the tax processing center where the tax form (from Scenario 1) arrives, spends 7.5 hours per day filling out the same tax assessment form. This employee, in a production environment, must maintain a quota of 50 assessments per shift. If the form could be made more efficient, so as to shave off only 1 minute in the time it takes to fill it out, the employee will notice the increase in efficiency in a big way. It would mean a savings of 50 minutes every working day for that employee alone! That can quickly add up to massive savings when applied to an entire group.

What is described in scenario 2 (which is based on a realistic situation) is an improvement to a form's **usability**, and has nothing to do with accessibility. Even if the tax assessment form was so inefficient that it took all day to fill out just once, it could still be deemed accessible. Usability often picks up where accessibility stops. In certain environments, a 100% accessible form can still be totally unusable, or only partially usable. Scenario 2 illustrates a lack of efficiency as one possible obstacle.

What is usability?

From: http://www.usability.gov/basics/index.html

Usability is the measure of the quality of a user's experience when interacting with a product or system, whether a Web site, a software application, mobile technology, or user-operated device.

Usability is a combination of factors that affect the user's experience with the system, including:

Ease of learning	How fast can a user who has never seen the User Interface before learn it sufficiently well to accomplish basic tasks?
Efficiency of use	Once an experienced user has learned to use the system, how fast can he or she accomplish tasks?
Memorability	If a user has used the system before, can he or she remember enough to use it effectively the next time or does the user have to start over again learning everything?
Error frequency and severity	How often do users make errors while using the system, how serious are these errors, and how do users recover from these errors?
Subjective satisfaction	How much does the user like using the system?

Accessibility is about putting information within every person's reach, whereas **usability** is about the quality of the experience once a person has access to the information.

Widespread obstacles to electronic forms accessibility and usability

These can easily be summed up as follows.

1. Lack of a true universal and accessible forms client.
2. Lack of a robust electronic forms specification.
3. Lack of a universally accepted feature set.

Let's look at each one in more detail.

A universal client. This is the software that people use to fill out and work with electronic forms. Many believe web browsers to be this software, but that is a fallacy. The ongoing browser wars and lack of robust specifications have reduced this potential solution to a morass of half baked specs., browsers that routinely break what little bit of universality there is, and web developers who have become so frustrated by it all that they have resigned themselves to making only the simplest of forms, most of which are never designed for accessibility.

A robust specification. HTML, XForms, SVG are three of the leading open web specifications that support forms. On their own, and even when used in combination, they still fail to provide a robust electronic forms environment. There are also many closed, or proprietary forms specifications that are owned by profit seeking companies. A discussion of these is beyond the scope fo this paper, but suffice it to say that a few companies do offer robust specs, but fail to provide a true universal form client. The obstacle here is the cost of the client software.

A universal client is one that must be available to everyone, even people who cannot afford to buy one.

A universal feature set. This refers primarily to the way users interact with the form. This may be of less importance to able bodied people, but for those who do rely on assistive technologies, familiarity is of great importance. Consistent keyboard shortcuts, consistent navigation, and consistent electronic behaviour are just a few of the things involved. It's also very important for an electronic forms solution to maintain the look and feel and standards of the user's operating system.

Obstacles to accessibility and usability of Tagged PDF forms

The Tagged PDF forms format creates some particular obstacles that are usually not present in other formats. These seven issues are the most troublesome.

Lack of universal keyboard shortcuts (accelerator keys)

With Tagged PDF forms there is no access to standard keyboard shortcuts known as Alt keys. There are many standard key combinations that keyboard users have grown accustomed to using, but cannot be provided with Tagged PDF forms.

Adobe Systems only provides access to the Control key with PDF JavaScript. Most of the Control key combinations are used to provide shortcuts to many Adobe Reader and Acrobat commands. There are very few Control key combinations left for PDF form developers to use, and of those remaining, chances are poor that you will find a combination that bears any logical resemblance to the feature you want to make accessible with a keyboard shortcut.

Toggling between user modes with assistive technologies

Many assistive technologies offer a variety of operating modes. This is to give hardware and software tools as much versatility as possible for providing access to information, and in some cases special modes are implemented to overcome deficiencies in the formats these devices attempt to interpret. One of the modes used by many assistive technologies is called **forms** mode.

With Tagged PDF forms, users of assistive technologies are forced to constantly switch between the normal document reading mode (used for reading the content of a form), and the forms mode which is required for inputting data into editable fields. This creates a lot of confusion, missed information, and severely slows down the process of working with a Tagged PDF form. It is better for a form to be fully functional in the normal document reading mode.

Multipage forms increase complexity and create obstacles

The PDF specification allows for an arbitrary number of pages in a form. While multipage forms are necessary for printing, they offer no electronic advantage over single page forms. Actually, multipage forms are annoying when used electronically, and create many obstacles for users of assistive technologies.

The main problems that can be created by multipage forms are: complicated navigation, loss of focus, nonlinear workflow, separation of the data entry areas from critical information (think of an instruction page at the end of a 5 page form which contains instructions needed for filling out page 1). All these problems require the user to learn additional application features to overcome them, or they require additional effort by the developer to build additional features into the form to deal with these problems (which the user will also have to learn how to use).

Complex print layouts are hard to work in

Not only do multipage forms create obstacles, but many print layouts were never intended to be used on a computer screen. There are a multitude of issues, but only a few major ones will be discussed.

Some forms are known to have the following instruction written on them, "Attach receipts here." The intent is to get the user of a paper form to staple or clip receipts to the form. The unsuitability of such a statement on an electronic form is plain to see. It is left as an exercise for the reader to discover other illogical possibilities.

Paper layouts often follow complex paths across the page. Some layouts have only one long convoluted path, while others can have numerous disjointed paths with mandatory and optional sections. The flow of the paths can be vertical, horizontal, and with some poorly designed layouts the flow can even be diagonal, or backwards, or worse.

Issues like these not only create problems for disabled users, they create problems for everyone.

PDF Tags are difficult to create and almost impossible to reuse

Ordinary fillable PDF forms are not accessible, it's that simple. To make a PDF form accessible, Adobe Systems recommends adding structured Tags to it. To make a data entry field accessible requires, at an absolute minimum 2 Tags. In addition, Short Description (alt) text must be added to the field's properties. Then there are numerous other Tags that are needed to deal with tables, pages, sections, language, page content, headers and footers, page numbers, graphics and logos, etc. But you're still not done. There are countless artifacts that must also be identified and dealt with. Things such as lines, arrows, pointers, inappropriate content, shaded objects and backgrounds, etc.

Suffice it to say that the average business form usually requires hundreds, and often thousands of Tags to make it accessible. Even with the best tools currently available, Tagging a PDF form is a significant task that requires specialized knowledge. Once you've completed this difficult task of Tagging a PDF form, you will find that your hard work has almost no chance of being reused in another form, which means starting from scratch for a new form will be a constant occurrence.

PDF Tags contain no logic or intelligence

To put it bluntly, PDF Tags are brainless. They are static, contain no logic, and just sit there in the order they were placed. This is sufficient for ordinary (static) documents that are merely meant to be read. Actually, PDF Tags work remarkably well for such documents. However, fillable PDF forms are not just ordinary documents. They are far more. They are usually more complex, support two way interaction with the user, often contain logic (in the form of JavaScript code), calculations, buttons that activate application features, database connections, and much, much more.

Except for the very simplest of forms, you would be wise to consider fillable PDF forms to be simple to moderately complex software applications, especially when they contain logic and calculations. To think of a fillable PDF form as just another document is to severely underestimate its complexity, and capacity, and to also overlook many of the intricacies involved in making it accessible and highly usable.

Paper concepts are foreign to some users

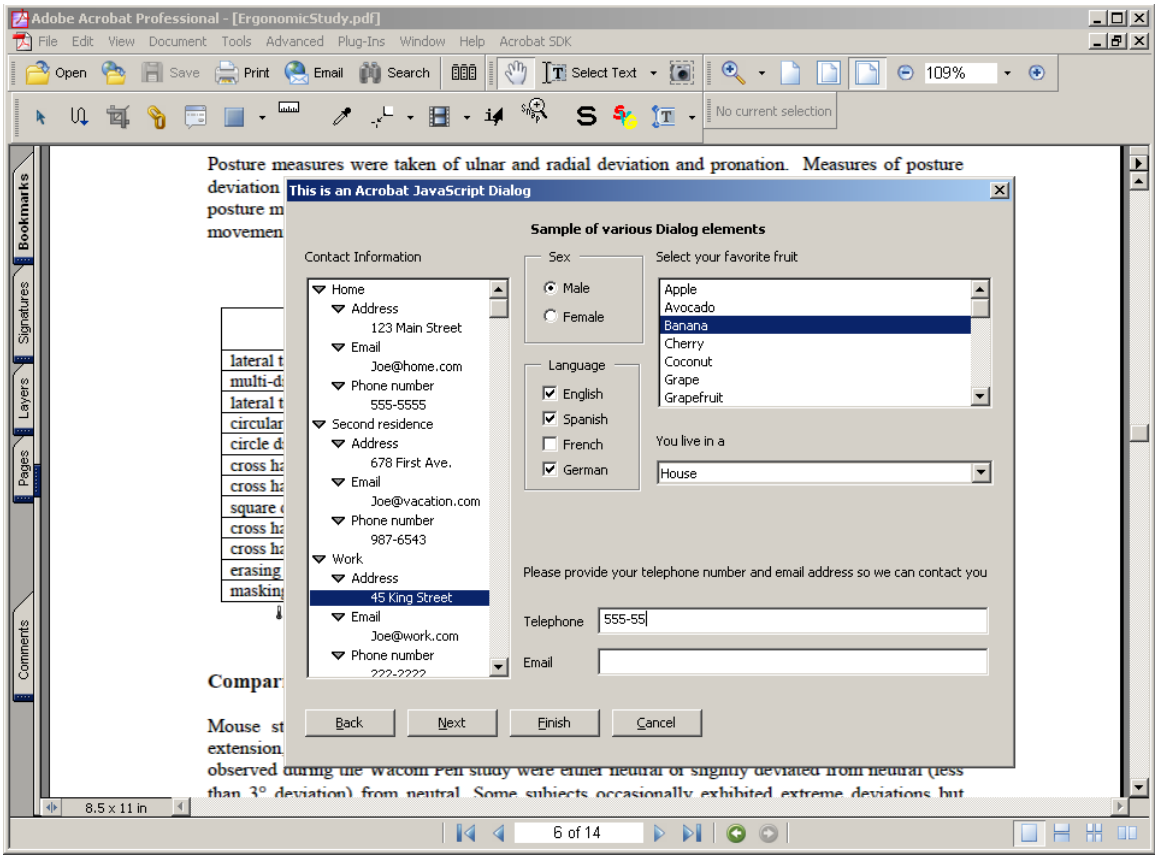
Blind users who have never seen paper, who have never developed an understanding and appreciation of complex paper layouts have extreme difficulty adapting to Tagged PDF forms.

These forms force them to struggle with unfamiliar paper concepts instead of using the software application concepts they are commonly used to.

Even though Tagged PDF forms can be made accessible, they are usually hard, if not impossible, for totally blind persons to use because of the complex paper concepts they are based on.

An accessible electronic forms solution that overcomes all of these limitations and obstacles.

With the release of Adobe Acrobat 6 and the new PDF 1.5 specification (and also with Acrobat 7 and PDF 1.6), a new paradigm for accessible electronic forms emerged. Adobe Systems added a Dialog object to their JavaScript extensions. This is a very important addition as you'll see shortly. In brief, the Dialog object is a device used to create application like Graphical User Interfaces (GUI) that can be executed within the Acrobat Standard, Professional, and Adobe Reader applications as plugins or as scripts embedded in a PDF file, as illustrated in this screen shot. Since the Dialog object works with the ubiquitous Adobe Reader, we now have a universal form client, and since it is based on the mature PDF specification, many subsets of which are now ISO standards, we also have a robust specification. As the screenshot also shows, the Dialog object follows standard GUI features which have become universally accepted. As a result of all these things the widespread obstacles to electronic forms accessibility and usability mentioned previously, are finally overcome. Continue reading to discover how the Dialog object, with its application like User Interface, also solves the obstacles to accessibility and usability created by Tagged PDF forms and the paper concepts they are based on.



These Dialogs can range from simple to complex. This example shows a sampling of the most popular element types. As you can see, it is possible to design complex and versatile interfaces that can be attached to forms. These Dialogs can offer the user an alternate way of filling out and interacting with a fillable PDF form. These Dialogs can even be daisy-chained into intelligent wizards (notice the Back, Next, and Finish wizard buttons in the screenshot). Complicated paper layouts can be completely avoided by the user! Yet the paper layout remains in the background, ready to be called up when the user is ready to print the completed form.

What is important to understand is that these Dialogs conform to most operating system standards for accessibility by default! That's right, by default!! In case you're still uncertain what that translates to, it means this: A Dialog (or wizard) is automatically accessible simply by the fact that it exists. There is no need to create PDF Tags, no need for form field Short Descriptions (or Alt text), no need for complicated tabbing sequences, no special reading order, no special navigation to help visually impaired users navigate complex paper layouts. Just robust, standard, universal, user friendly accessibility, straight out of the box, at no cost and no effort to the user. You'll even notice the little underscores on the first letters of the [Back] [Next] [Finish] [Cancel] buttons. These are the vital keyboard accelerators (Alt keys) that are missing from Tagged PDF forms.

These Dialogs can exploit the full power of JavaScript. They can read from, and write information to the underlying fillable PDF form, they can use existing field formatting, validation, and calculations, so you won't lose your existing work. In one word, they're **wonderful**. So what's the catch?

There are two actually. The first catch is that not all the element types are user friendly when used with assistive technologies. The second catch is that (as you may have noticed) these Dialogs look a lot more like applications than PDF forms, and indeed they are. They are created entirely in JavaScript, and if you've ever written a significant amount of PDF JavaScript you know that it can be a challenging task (unless you happen to be an expert programmer). These Dialogs require a large quantity of hard-to-write, and even harder to debug JavaScript code. Coding them up by hand is a challenge even for an experienced programmer.

Now the good news. Highly usable Dialogs (or wizards) can be created for even the most complex PDF forms, using only the elements that are user friendly to assistive technologies. The following element types and properties have been identified as stable, well behaved, and useful for building accessible PDF forms:

static_text	Used to display text captions associated with data input fields (edit_text).
edit_text	Used for collecting data from the user. Always used in combination with a static_text element.
popup	Equivalent to a PDF combobox. Displays a list of selectable options.
check_box	Equivalent to a PDF checkbox.
button	Equivalent to a PDF button field with a text caption.
gap	Element used to provide horizontal and vertical spacing between elements.
view	Mandatory element for controlling the tabbing sequence. Not visible to the user.

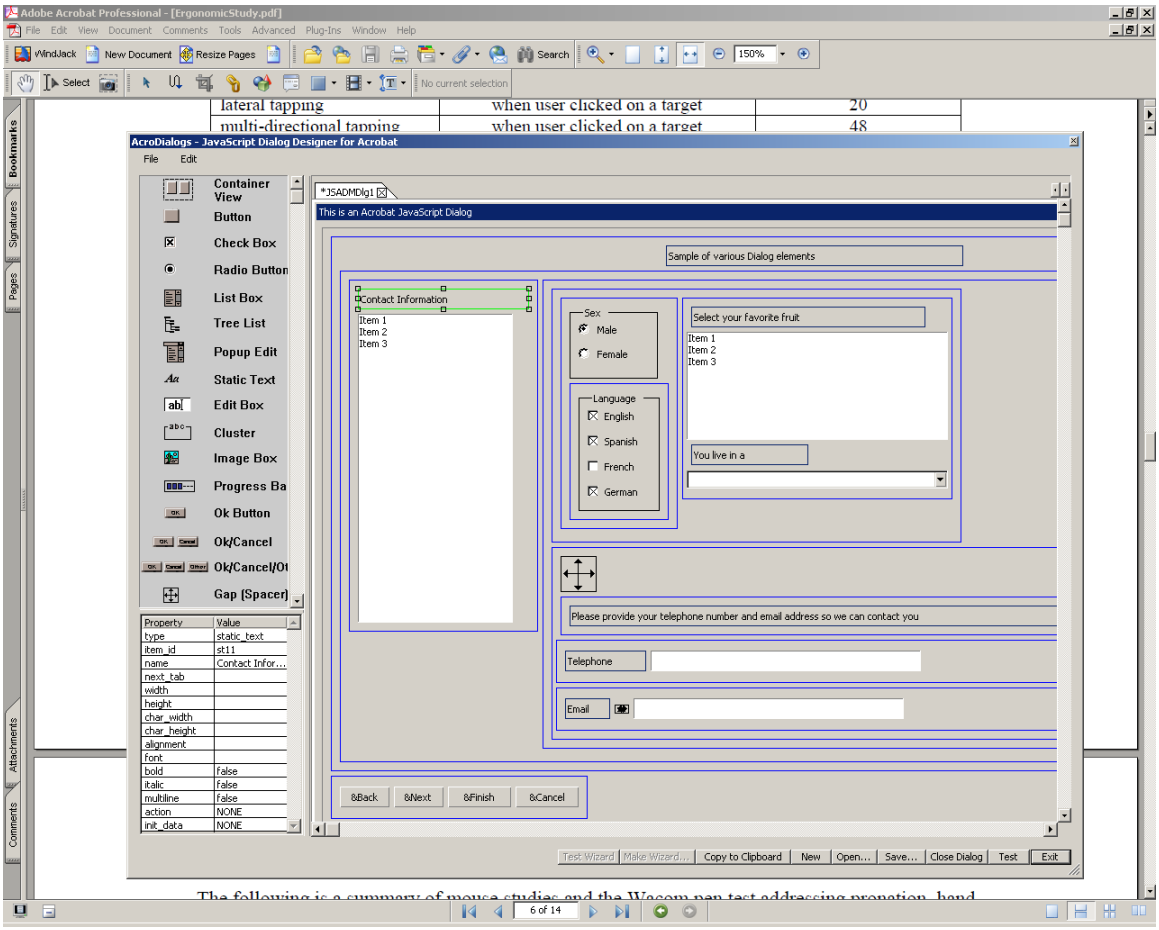
All other element types should be viewed with suspicion, and must be fully tested with assistive devices before being used. Some have always been bad choices for accessibility, like **Radio Buttons**, which are hard to use in any format. Others, like the **Hierarchical Tree List** behave inconsistently with different assistive technologies and are just to complicated for some users, while some are simply useless for accessibility, like the **Image** and **Progress Bar** elements.

For complete details about the Dialog object and all its elements and properties see the Acrobat 7 JavaScript manual. A free copy can be downloaded from the Adobe Systems web site at: http://partners.adobe.com/public/developer/en/acrobat/sdk/AcroJS.pdf

Note: The Dialog object is present in Acrobat 6, but undocumented in the Acrobat 6 JavaScript manual, so you need the Acrobat 7 manual.

Now that the first catch is out of the way, what about the second one? There is now a great new visual design tool that you can use to create Dialogs and wizards without hardly touching the JavaScript code. This tool is called **AcroDialogs**. It's an Acrobat plugin sold for a reasonable price by WindJack Solutions. http://www.windjack.com/

Here's a screenshot of the **AcroDialogs** plugin being used to design the sample Dialog you saw earlier. It has built-in support for all the element types and properties of the Dialog object, plus it has several additional features, like a wizard building option, and an option to save all or part of a layout in a user definable library for easy reuse later on.



Design strategies

There are two approaches that can be used when designing for accessibility.

Technique 1: Design for a specific disability. This is usually for situations where you have control over the intended audience, process, and equipment. Such a situation might be making a company timesheet accessible when you know that blindness is the only disability that needs special accommodation in your company. In such a case you can create a highly specialized accessible form with features that may only be useful to blind persons.

Technique 2: Design for all disabilities. This would be appropriate for forms that will be in wide spread use, such as forms distributed to the public over the internet, or in situations where you don't have full control over your audience, process, and equipment. A couple of examples would be a tax form, and a manufacturer's product registration form. In these cases you want to appeal to the broadest audience possible, and make the form's accessibility features as universal as possible, by designing for various disabilities, sticking to universal processes, and testing the form with a wide variety of assistive technologies.

Design guidelines

Here are a few things to consider. Always be on the lookout for paper concepts that do not translate well for onscreen use. Avoid using references to locations on a page, such as "attach receipts here," or "read the instructions below." These are visual cues. Be careful of paper concepts that do not translate well to electronic media. For example, "complete the back side of this form" makes no sense in cyberspace. A paper page printed on both sides usually becomes two electronic (or PDF) pages. Whether they print as two separate pages or as a single double sided page is usually a function of the printer's duplex printing option, and has nothing to do with the online use of a form. So the example above should read "complete page two of this form" instead of "complete the back side of this form."

In a typical Dialog or wizard you should include some or all of the following features.

- A brief description of the form in the window's title bar. This is usually the name or number of the form.
- Simple, efficient, and intuitive layout.
- Navigational buttons. Back, Next, Finish, and Cancel buttons are standard.
- Hot keys. Only Alt key combinations are used.
- Check boxes.
- Drop down lists.
- Text and number edit boxes.
- Basic data validation.
- Business intelligence.
- Pre-populated or calculated values where applicable.
- Navigational aids to assist with complex tables.
- Dialog size should fit on 640 by 480 pixel monitors.
- All text and features optimized for accessibility.

Every Dialog or wizard should use a standard set of hot keys. These are accessed by pressing the Alt key in combination with another key. The following hot keys are commonly encountered.

- Back button Alt + b
- Next button Alt + n
- Finish button Alt + f
- Cancel button Alt + c
- Print Form button Alt + p
- Reset Form button Alt + r

The golden rule

The single most important design rule one can follow is to keep things as simple as possible.

The second most important design rule is to know the features, and limitations, of the electronic format you will be publishing to. PDF forms come in a variety of flavors. Every version is different. To adequately deal with PDF forms accessibility requires a good understanding of the format, and the environments in which it will be used.

Remember: **Complex layouts and irregular formatting = too much concentration and memorization for the user**

This is particularly true for the visually impaired and those with learning disabilities.

Forms often require very complex layouts that pose serious usability challenges for designers and users. Designing forms for accessibility further complicates these challenges. Using complex layouts also greatly complicates the task at hand for blind users because it requires too much concentration and memorization.

Complexity is the enemy of usability any time a document needs to be read. Forms only compound this problem, because not only are they designed to be read, they also require user interaction. Electronically fillable forms add yet another layer of complexity on top of all this, since they often need to control some or all of the user interaction and input. Making this control intuitive and usable by everyone is a demanding task that increases almost exponentially with the complexity of a form.

Forms designers are in a difficult position when dealing with accessibility. The demands are far more stringent than for regular documents. The full power of a programming language is often required to achieve acceptable levels of accessibility and usability.

Techniques for testing form accessibility

Adobe recommends the following techniques. To ensure that your forms are accessible to a wide variety of users, you should test them with a variety of assistive technologies. You can test your forms simply and inexpensively using these techniques.

- Ensure that the form can be filled using only the keyboard. Be sure to fill the entire form and test all fields and buttons. As you complete the form, determine whether improvements are required based on your answers to the following questions:
    - Are there any operations that cannot be performed?

- Are any operations awkward or difficult to perform?
- Are keyboard mechanisms well-documented?
- Do all controls and menu items have underlined access keys?

- Demo versions of screen reader software can be downloaded free from the Internet. To test screen reader results, turn your monitor off and use only the screen reader to navigate and fill the form. Because you are the form author, your familiarity with the form may make it difficult to determine if the information read by the screen reader is sufficient and makes sense. If possible, have someone else test your form in this way.
- Demo versions of screen magnification software are also available for testing from the Internet.
- Speech to text software is available at a nominal cost from local computer stores. Test the form by using voice input only.
- Many users with vision impairment rely on high contrast between text and the background to read the form. Microsoft Windows has a high contrast color scheme that provides a display similar to what many users with vision impairment will be using to complete your form. To set your display to high contrast mode, enable the feature through Accessibility Options in the Windows Control Panel. As you complete the form in this mode, determine whether improvements are required based on your answers to the following questions:
  - Do parts of the form become invisible, unrecognizable, or difficult to use?
  - Do any areas continue to appear black on a white background?
  - Are any elements improperly sized or truncated?

Conclusion

It has been a very long time in coming. Many have tried and failed. There have been many setbacks along the way. The browser wars continue to rage on and cause grief for everyone. Numerous half baked specifications continue to appear on the scene. As a result of all this, those with disabilities continue to struggle with, and are often denied, the tremendous benefits of electronic forms.

Assistive technologies continue to improve, but the constant feedback from the disabled community is loud and clear! An application like User Interface is the preferred approach to providing accessible electronic forms. The Acrobat JavaScript Dialog object is an excellent complement to PDF forms. It provides a versatile, user friendly, and universal UI to disabled users. It also maintains all the benefits of the PDF format, but without confronting the user with the complex and unfamiliar concepts of paper layouts.

Thanks to Adobe Systems, and WindJack Solutions, a truly accessible and usable electronic forms solution is now at hand. It is inexpensive, user friendly for developers, and above all, it makes possible what countless people with disabilities around the world have been asking and waiting for. A robust, universal, ubiquitous, accessible, and usable electronic forms solution.