Pozycja Menu - Przykładowe Treści

Treść

Case deseruisse ius no, usu aperiri laboramus cu. Ius ea zril nullam audire, sed ad velit indoctum scribentur. Postea sanctus eam an. Esse reprehendunt has ea. Sea eu veri soluta eripuit, paulo munere aliquid eu mei. Ex stet percipit has, nam no solum partiendo, duo inimicus dissentiet cu.

Case deseruisse ius no, usu aperiri laboramus cu. Ius ea zril nullam audire, sed ad velit indoctum scribentur. Postea sanctus eam an. Esse reprehendunt has ea. Sea eu veri soluta eripuit, paulo munere aliquid eu mei. Ex stet percipit has, nam no solum partiendo, duo inimicus dissentiet cu.

Case deseruisse ius no, usu aperiri laboramus cu. Ius ea zril nullam audire, sed ad velit indoctum scribentur. Postea sanctus eam an. Esse reprehendunt has ea. Sea eu veri soluta eripuit, paulo munere aliquid eu mei. Ex stet percipit has, nam no solum partiendo, duo inimicus dissentiet cu.

Adhuc torquatos comprehensam eu nam. Est ei sonet accusata repudiandae, ut nam esse feugait. Id mea summo fuisset, in est amet essent latine. Ut elitr euismod voluptatibus vis. Eu facilisi necessitatibus nam, id qui eros vulputate. Tale scripta cu ius.

Vero porro mucius sit ne, cetero theophrastus et vis. His oportere recteque ei, id ridens similique cum. Et dicant inciderint vituperatoribus sea, te per primis disputando. An luptatum imperdiet pertinacia sea, ad possim latine eum, cu per meliore reprimique. An eos nobis mediocrem.

Ea quo propriae intellegat. Justo integre imperdiet ei ius, est possim facilisi in, blandit appellantur an mel. Amet nobis at nec, ad est nisl homero. Vix meliore nominati deseruisse at, te mel dicta luptatum corrumpit. An dolor tantas vocibus est, ut quo prompta gloriatur. Probo prodesset cu pro, per minim perfecto te. Ocurreret interesset consectetuer ne ius, ad errem iudico ullamcorper vix. Ex labores quaestio pri, sed ei numquam ullamcorper, definitionem delicatissimi no cum. Harum similique ex sit.

Ei eam brute integre, his case ludus facilis te. Mea te affert aliquid dolorem, vis an atqui offendit. Sale impetus vituperata qui ea, illud graeco nec et. Ad vim graeco gloriatur, usu ad cibo iusto. Eu postea admodum scribentur sit. Est ei option aliquip civibus, sint mundi elitr eos id. Te probo gloriatur mea. Ius ut augue efficiantur. Aliquid patrioque ut cum, vix et nullam facilisi gloriatur. Cu vim ceteros persequeris, oblique persius usu ex. Et enim ridens pri, vix eu tota percipitur vituperatoribus. Quo in atqui ludus, in adhuc eligendi deserunt his.

No vel sint malis. Ridens quidam interesset ne quo, iusto consetetur definiebas in pri, sonet ceteros referrentur at nec. Vis utinam moderatius te. Nec liber epicurei electram no, vix denique minimum assueverit id. Pro et postulant erroribus definiebas, ut alii dicam albucius nam. Eu mea mentitum voluptua, eum at semper discere recteque, vim te dico bonorum.

Zril melius sed id, ea quo habeo maluisset. Nam et senserit expetendis contentiones, percipit vituperatoribus ut has, in has viris consulatu interpretaris. Cum nisl scripserit ei, affert homero complectitur nec in.

Eros legere mel ea, pri no offendit imperdiet argumentum, eu habeo quidam voluptatum vel. Vel eu virtute conceptam, vis ne virtute perfecto.

In order to assure that websites and web applications are accessible to and usable by everyone, designers and developers must follow web accessibility guidelines. Each of the following topics address issues that are especially common on UW websites. See also our annotated list of web accessibility Tools and Resources.

Features of Accessible Websites

- Good use of HTML headings
- Accessible with keyboard
- Accessible images
- Accessible menus
- Accessible forms
- Accessible tables
- Effective use of color
- Meaningful link text
- ARIA landmark roles
- ARIA for web applications

Additional information is available on the AccessComputing website 30 Web Accessibility Tips.

- Code header areas in the accordion as buttons. Using a <button> assures that accordions are usable with both screen readers and the keyboard.
- Use aria-expanded on buttons to express an accordion’s default state. Buttons should state if they are expanded by default with aria-expanded="true". The aria-expanded="false" attributes will be added to other buttons when the accordion is initialized by the

JavaScript.
- Use unique ids. Each button has a unique name aria-controls="id" that associates the control to the appropriate region by referencing the controlled element's id.
- The accordion uses javascript to set the hidden value of its content area. Each content area will have its hidden attribute set by the component, depending on its corresponding button's aria-expanded attribute. To ensure that your content is accessible in the event that the JavaScript does not load or is disabled, you should not manually set hidden on any of your content areas.

- Web Content Accessibility Guidelines 2.1 (WCAG) This is the latest version of the definitive web accessibility guidelines from the World Wide Web Consortium (W3C)
- Web Content Accessibility Guidelines 2.0 This is the previous version of WCAG, updated in June 2018. WCAG 2.0 Level AA is still the standards.
- Accessible Rich Internet Applications (ARIA) ARIA is a W3C specification that provides a way to make dynamic web applications and advanced user interface controls more accessible to people with disabilities.
- WAI-ARIA Authoring Practices This is an essential resource for anyone developing websites or web applications. It provides standard design patterns for dozens of common web widgets such as accordions, dialogs, and menus.

Checking a Website for Accessibility

You can go a long way toward assuring your website is accessible by following these simple steps:

1. Validate your HTML. If HTML is used incorrectly, assistive technology can have problems interpreting the page content, which

can result in access problems for users. Use an HTML validator to check your code.

2. Test with a keyboard. Set your mouse aside and use the tab key to navigate through your web pages. You should be able to access all interactive features (e.g., menus, links, form fields, buttons, controls) and operate them by pressing Enter, space, arrow keys or other intuitive keystrokes. If you are unable to access some of your site's features, your site is likely to have accessibility problems.
3. Use an accessibility checker. There are several free online tools that will check your web pages for accessibility. See our Tools and Resources page for an annotated list. Also, the UW has a subscription for Siteimprove, a powerful web-based tool that scans your site at regular intervals for broken links, spelling errors, and accessibility problems. See the Siteimprove page on this website for more information.
4. Test with users. You can test your site by simply recruiting and observing users as they interact with your site. To test for accessibility, recruit users who have a variety of skill levels and characteristics, such as those listed below under the heading *What Is Accessibility?*
5. Ask for help. The UW community is actively working toward the goal of full accessibility for all visitors to its websites. Since we're all working together toward this goal, there are many in the community who are happy to help. See Getting Help with Accessibility for more information.

Developing Accessible Sites Using WordPress

UW Marketing has created a UW WordPress Theme with input from UW-IT Accessible Technology Services. The theme includes a variety of accessible features, and accessibility is an ongoing consideration as features are added or updated. Therefore, web owners are encouraged to use this theme. Free WordPress hosting is also available for UW organizations.

WordPress is an easy-to-use, highly flexible content management system for creating and managing websites. Its functionality can be extended with any of several hundred widgets and plugins that are freely available. However, widgets and plug-ins can also introduce accessibility problems to a site, so you should select these very carefully.

What is Web Accessibility?

People who use the web have a growing variety of characteristics. As web developers, we can not assume that all our users are accessing our content using the same web browser or operating system as we are, nor can we assume they're using a traditional monitor for output, or keyboard and mouse for input. Consider these user characteristics:

- Unable to see. Individuals who are blind use either audible output (products called *screen readers* that read web content using synthesized speech) or tactile output (a refreshable Braille device).
- Has dyslexia. Individuals with learning disabilities such as dyslexia may also use audible output, along with software that highlights words or phrases as they're read aloud using synthesized speech.
- Has low vision. Individuals with low vision may use screen magnification software that allows them to zoom into all or a portion of the visual screen. Many others with less-than-perfect eyesight may enlarge the font on websites using standard browser functions, such as Ctrl + in Windows browsers or Command + in Mac browsers.
- Has a physical disability. Individuals with physical disabilities that effect their use of hands may be unable to use a mouse, and instead may rely exclusively on keyboard or use assistive technologies such

as speech recognition, head pointers, mouth sticks, or eye-gaze tracking systems.

- Unable to hear. Individuals who are deaf or hard of hearing are unable to access audio content, so video needs to be captioned and audio needs be transcribed.
- Using a mobile device. Individuals who are accessing the web using a compact mobile device such as a phone face accessibility barriers, just like individuals with disabilities do. They're using a small screen and may need to zoom in or increase the font size, and they are likely to be using a touch interface rather than a mouse. Also, Apple's iPhone and iPad do not support Adobe Flash.
- Limited bandwidth. Individuals may be on slow Internet connections if they're located in a rural area or lack the financial resources to access high-speed Internet. These users benefit from pages that load quickly (use graphics sparingly) and transcripts for video.
- Limited time. Very busy individuals may have too little time to watch an entire video or audio recording, but can quickly access its content if a transcript is available.

An accessible website works for all of these users, and countless others not mentioned.

The W3C summarizes web accessibility nicely in their Web Content Accessibility Guidelines 2.0. WCAG 2.0 is organized into the following four key concepts:

- Web content must be perceivable
- Web content must be operable
- Web content must be understandable
- Web content must be robust

There are many possible approaches to attaining accessibility as defined by these four concepts. The pages of this website were designed to help.

The following video, produced by UW-IT Accessible Technology Services, features university web designers and developers, including several from the UW, discussing the importance of creating websites that are accessible to all users.

Przewiń do początku