Semantic Form Challenge

Introduction

Effective front-end development is not about churning out pixel-perfect conversions of a designer's Photoshop files, or helping every stakeholder get their pet feature highlighted on the home page. It's about ensuring that the entire team's mission is realized through semantic, performant code that ensures the best user experience across the multitude of devices and access issues inherent to the modern web.

Progressive enhancement as a philosophy places an emphasis on semantic markup, accessibility, and the use of external technologies in a layered fashion. A site's content and functionality should be available to all users, regardless of device or capability. Users with more capable devices can get an enhanced experience that is layered on top of the core experience, but no user is locked out due to personal or device restrictions.

Accessibility is a complementary principle that is baked into the idea of progressive enhancement. Simply stated, accessibility means removing barriers to access that may be experienced by people with physical or cognitive disabilities.

Background Research

- Understanding Progressive Enhancement by Aaron Gustafson
- Reframing Accessibility for the Web by Anne Gibson
- Best Practices: HTML & CSS by Scott O'Hara
- HTML Element Reference at MDN
- A Look Into Proper HTML5 Semantics by Hongkiat
- The Web Accessibility Basics by Marco Zehe
- Creating Accessible Forms at WebAim
- Using Voiceover to Evaluate Web Accessibility at WebAim
- HTML5 Form Validation With the "pattern" Attribute by Thoriq Firdaus
- Touch Keyboard Types at Baymard Institute

- WTF, forms? by Mark Otto

Challenge Outline

The challenge is to take an image of a form that has already been designed and develop a front-end system that accommodates the various elements required in form design as demonstrated by elements in the example, and then to develop that specific form using your system.

Your work must follow the principles of progressive enhancement and accessibility. You must use proper semantic markup and test with VoiceOver to verify that it is accessible to a user with visual impairments. The form must display in a comprehensible way without CSS enabled, and it must be usable without JavaScript. Browsers that are capable of using HTML5 form enhancements must use those instead of JavaScript. Browsers that do not have advanced HTML5 functionality must load in JavaScript to add the missing features.

Recommended Process

1. Write your markup first. DO NOT write any CSS or JS until you've confirmed that the HTML works properly on its own and is semantic and accessible.
2. Write CSS with a systems-oriented approach, i.e. style your markup so that a team could use your CSS to handle basic form elements across an entire site.
3. Apply your system to the specific form example you've been provided with.
4. Load in JS enhancements only in browsers that need them. It is acceptable to use jQuery if desired. Do not load any JS into a browser that already has native functionality replicated by that specific JS.

Deliverable

- A page with the finished example form. Expect the HTML, CSS and JavaScript you've written to be evaluated in detail. Your form will also be tested with CSS off, JavaScript off, and using Voiceover. It should also be capable of being navigated and filled out with just the keyboard (no mouse).

Resources

- Example form (source, specced and export)
- Image assets (in standard and @2x resolutions in case you want to try optimizing for hi-res displays. Check and warning icons are white with transparent backgrounds so might be hard to see in image previews before placing them on the page on the proper background.)
- Modernizr, for testing availability of features in the browser
- jQuery Validation plugin, for enhancing older browsers with form validation
- Parsley JS, an alternate validation plugin
- jQuery Credit Card Validation, a plugin for validating credit card numbers