Author:
Tanay Gavankar (tgavanka)

NOTE: All the code positions I've given are just examples of that feature. There are similar (and other) usages of that technology elsewhere, and if for some reason you need more examples and are unable to find them, please let me know. I have also included more than 10, so please pick the top 10 that you feel were the strongest for grading.

Required Elements:

Javascript

- Objects / prototypes
  - *public/js/presenter.js and public/js/viewer.js*
  - *views/pdf/script.ejs*

HTML

- Tables
  - Used to display the list of presentations linked to your account as well as available options for them.
  - *views/list.ejs:32*
- Forms w/ validation
  - Validates inputs on forms that have requirements.
  - HTML5 "required" and "pattern" attr
    - *views/register.ejs:58,62,66*
    - *views/login.ejs:51,55*
- HTML5 Accessibility Tags
  - Uses accessibility tags to support screen-readers, etc.
  - <header> - *views/parts/top.ejs:34*
  - <footer> - *views/parts/bottom.ejs:18*
  - <nav> - *views/parts/bottom.ejs:3*

CSS

- Reset
  - Reset all styles for browser consistency.
  - *public/css/reset.css* (used in all pages, i.e. *views/parts/top.ejs:7*)
- Pseudo-selectors
  - Used for navigation bar and list page.
  - :last-child - *public/css/style.css:133*
  - :nth-of-type - *public/css/style.css:389-396,422-429,440-448*
  - :before - *public/css/style.css:431*
    - *views/present.ejs:124,129*
- Media queries
  - Used to reflow layout depending on screen size.
  - min-width: *public/css/style.css:310*
  - max-width: *public/css/style.css:399*
- Fluid Layouts
  - Resize homepage on desktop
    - Navbar moves from top to bottom

        - Homepage splash image disappears
        - Pure CSS (media-queries)
    - Resize /list on desktop
        - Table reflows from full table to vertical accordian
        - Pure CSS, including vertical accordian (uses :target / media-queries)
- Animations
    - Used for any AJAX form submit (i.e. "/login")
    - Ajax Loading Icon
        - *public/css/style.css:266-299*

DOM Manipulation

- Show/hide AJAX loading icon
    - *views/login.ejs:7,32*
    - All other forms
- Disable form submit on AJAX
    - *views/login.ejs:8,33*
    - All other forms
- Update AJAX response (i.e. success/error)
    - *views/login.ejs:20*
    - All forms
- Presentation changes page on page turn

jQuery

- See DOM Manipulation section
- Event handling
    - *public/js/viewer.js:29,39*

AJAX client

- Submit POST
    - *views/login.ejs:36*
    - All forms

AJAX server

- Handle POST
    - *loginRoutes.js:31,42*
    - *presRoutes.js:98,174,215*

Node.js (with Express)

- [*]Routes.js
- Express used for routes
- Custom middleware for auth checking
    - *middleware/ensureAuthenticated.js*
- CSRF middleware
    - *app.js:39*
- Custom 404 page
    - *app.js:53*
- All fields validated server-side, with 400 response on invalid (Bad Request)

- *loginRoutes.js:46,50,54,etc*
- *presRoutes.js:158,177,180,etc*

Socket.IO

- Used for communication between clients and server for updating slide position.
- *socketSrv.js* (server)
- *public/js/presenter.js:19,28*
- *public/js/viewer.js:15,20*

MongoDB via Mongoose

- Used to store saved presentations and user accounts.
- *presRoutes.js:13,89,136,228*
- *loginRoutes.js:57*

Redis

- Used to store generated security keys (nonces) for presenters. This helps verify that the update slide messages are correctly authenticated.
- *redisRoutes.js:24*
- *socketSrv.js:20*

window.postMessage

- Used to communicate and send messages between windows open on the client machine (i.e. from parent frame to iframe). These messages are wrapped to also be forwarded via socketIO so as to enable communication across multiple viewers' machines.
- *views/pdf/script.ejs:105*
- *public/js/presenter.js:21*

Server-side Filesystem API

- Used to handle uploaded PDFs.
- *presRoutes.js:123-128*

HTML5 History API

- Slide positions are stored in the URL hash fragment and updated via the HTML5 history API (replaceState). This allows for navigating the slides without a new server request (saves on bandwidth costs) and additionally means that users do not have to hit the back button on their browser for every slide change. (Specifically, when viewing a presentation, hitting the back button will take you straight back to the previous page, rather than the previous slide.)
- Event listeners are bound to the onhashchange event so the page is updated quickly.
- *views/pdf/script.js:91*

General Security

- CSRF Protection
    - All AJAX calls are protected from CSRF attacks by CSRF tokens.
- Replay and imposter attack prevention
    - The presenter view has to sign each "update page" action with a key generated by the server, which is received upon registering as an authorized user. This protects from replay attacks and imposter attacks.

- HTTPS (planned)
    - In production, SlideSync would be served under HTTPS - this app is not because introducing SSL certs causes a lot of issues (it's not a simple "change to https") that I did not have time to address, as well as the additional monetary costs of having a valid SSL cert (self-signed certs can also bring up issues).
- Input Sanitization
    - Allowing for custom user input opens XSS vulnerabilities, so all inputs are sanitized and XSS vulnerabilities are for the most part removed. (Of course, it is not possible to protect against ALL forms of XSS, as that is its very nature - new XSS vectors are constantly discovered.)

Design Process / User Testing

Initial Competitive Analysis and Research:

I have used dzSlides (http://paulrouget.com/dzslides/) before and it was very clean and stable. This performs the cross-platform aspect of creating the presentation, but it does nothing about syncing across various sessions. I plan on using dzSlides to power the HTML5 presentation templating.

Google I/O 2012 Slides (http://code.google.com/p/io-2012-slides/) seems to do some things that I'd like (specifically the "presenter mode"), but it seems like creating the actual presentation requires technical knowledge of HTML and CSS. Hopefully, my product will be accessible to those who both do and do not have such technical knowledge. Furthermore, the presenter mode seems to be geared towards serving locally (as it only runs a python SimpleHTTPServer) and so would not work well for long-distance presentations.

Socket.IO (http://socket.io) is a library that enables socket communication for web browsers. The HTML5 spec has websockets already defined, but there are many idiosyncracies across browsers and various lack of support that makes using pure websockets difficult. I plan on using Socket.IO to bridge this gap and use it for network communication.

Mozilla's pdf.js (https://github.com/mozilla/pdf.js) is a project to get PDFs to render entirely in the browser with pure javascript. It is still in its infancy, but I believe it may be an excellent start in being able to support cross-platform PDFs for this idea.

Initial Storyboard:

1. User creates an account / logs into their account
2. User gets to make a new presention (HTML5 or PDF), edit an existing one, or view one of their own
    a. Make new: Allows input with the dzslides template for the HTML/CSS
    b. Edit: Gives editor to an existing presentation
    c. View: gives public link to view presentation as well as private "presenter mode" link. Links can be shared via email, QR code, and are generally "shortlinks". The viewer mode has the option to "stop following" the presenter and can always browse the presentation on their own. The presenter mode has large forward/back buttons, a clock, current slide position (k of n), and current slide. Moving in the presenter mode automatically moves all connected viewers to the same slide.

User Testing:

QR Code Button:

My original design did not have anything about a QR code - I had envisioned that the users would simply type in the given URL (since they were short). During my first user testing round, I had two people tell me that it would be nice to have a way to easily transfer the link from one computer to another device. Most commonly, users would open the presentation on desktop and then want to transfer the presenter mode to their phones, so a QR code made the most sense. Thus, the QR code option was added. This also gave the benefit of being able to display the QR code to viewers and they can pull up the presentation on their own devices (since no installation beyond a modern browser is needed).

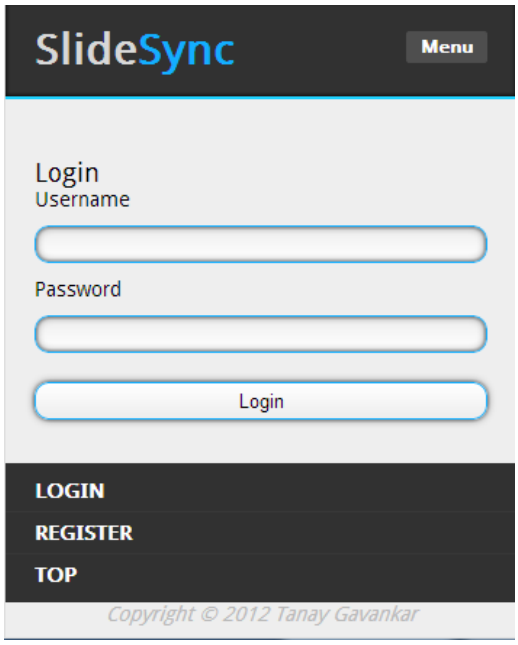
Better presenter interface:
The original presenter interface did not have many labels for the information it supplied, and so one user suggested to make it more clear what things meant. Specifically, the slide k of n label was not discernable in the initial version, and so it was clarified by adding a label. Screenshots of before and after are included below.

Before

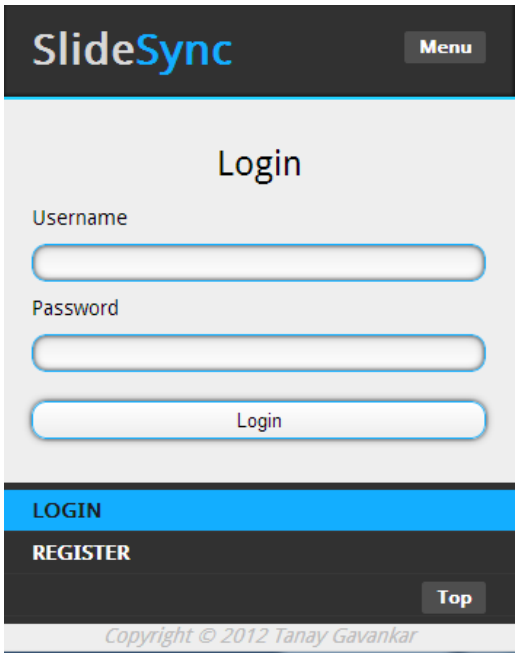
After

Better navigation bar:
The original navigation bar had proper reflowing for mobile, but the "top" button was the same style as the rest of the links. One user said to make it more explicit (so it doesn't blend in). Another user also said that it would be useful for the navbar to indicate which page you are currently viewing. Before and after is included below.

Before



After



Iterative Design:
You can see features be added at Github, https://github.com/tgavankar/SlideSync.

Known Issues/Bugs

Cordova uses does not support websockets, and so SocketIO uses XHR polling. This means that if a message is attempted to be sent when the network connection is down, the entire app will crash (on Android and possibly on iOS, but only in Cordova - this is a Cordova Browser bug).

Cordova/Phonegap does not perform HTML5 input validation, so all the required/pattern attributes are ignored. This works in Chrome/Firefox.

Cordova/Phonegap does not support HTML5 history API, and so there is no history.replaceState. This means that the fallback of window.location.hash is used, which results in lots of history entries. Thus, when viewing a presentation, after scrolling through some slides, pressing the back button will only take you through the previous slides rather than to the actual previous page.

In iOS6, click events on absolutely positioned elements don't work properly / consistently. This means the "stop following" / "back button" on the view presentation on IOS sometimes does not work.
See: http://stackoverflow.com/questions/9145025/jquery-click-events-not-firing-on-positionabsolute-td-tag-on-ios-after-scroll

iOS Cordova does not support PDF rendering. This may be a bug in Mozilla's PDF.js library, but they claim that it works on all modern browsers (which iOS Cordova may not fall under).

iOS does not size iframes correctly. This means the presentation view's rendering looks broken, but functionality is still intact. See: http://stackoverflow.com/questions/9814256/iframe-on-ios-ipad-content-cropping-issue (the given solutions did not solve the issue).