Rebecca D. Johnson

Brother Gary T. James

CIT 261-02

20 July 2019

Project Submission

# Application URL:

<https://johnson-rebecca.github.io/Project/blairgifts/>

# GitHu Repo URL:

<https://github.com/johnson-rebecca/johnson-rebecca.github.io>

# Project Description

“Blair Gifts” is designed to be a mobile application used by family members participating in our annual family Christmas craft gift exchange. This application allows a user to set up a local profile (or load a previously saved profile), save a setting of which calendar year they are working on (normally current year, but allows others in case participant is behind from a previous year. The user can also select from a list of other participants to record their assigned recipient. (In the future, I envision the assigned recipient being a feature managed through an administrative function and loaded from a central database.)

The mobile application is designed to help users search for craft ideas through other users saved Pinterest boards (pulled from the 3rd party Pinterest API) and to save ideas as favorites or set an item as the planned gift. Ideas marked as favorites or planned gift are saved to local storage, so that the user can use this application to help them fin ideas and sort through their ideas for their craft. This will allow some of our older users to access to Pinterest ideas without needing to register for and learn another application. It also helps to remind our participants on when they

The application was designed to have a simple and clear interface without too many distractions so that the interface will not be too hard for a wide variety of ages to interact with. The application is designed with a responsive menu design that presents a standard horizontal menu on larger screens which auto-transitions into a standard hamburger menu on smaller devices. This general design of the navigation should be familiar to most users.

# CSS3 Features

CSS3 is used throughout in various ways to try to help draw the users attention to issues, no information, etc.

## CSS Transitions Highlights

Transitions are used pretty extensively throughout the application to make changes more subtle and less jarring to the user. A few key highlights are:

1. Revealing the hidden divs to display recipient preferred subjects and colors when the recipient is selected from the dropdown on the Recipient page.

.hiddenDiv {

max-height:0px;

opacity:0;

width: 30rem;

transition: all 2s ease-in-out;

}

.revealDiv {

max-height:200px;

opacity:1;

}

1. Buttons use transitions on all properties in order to allow subtle changes as the user interacts with them, including a slight glow on hover using box-shadow.

.btn5:hover {

box-shadow: 0 5px rgba(0,0,0,0.35), 0 0 20px #839D56;

}

1. Inputs have transitions applied to all properties. This is used to apply a highlight “glow” effect on the border when the inputs have focus. This is also used to transition a red “glow” when the input has an error.

input, select, textarea {

border: 1px solid transparent;

transition: all .5s ease-out !important;

-webkit-transition: all .5s ease-out;

-moz-transition: all .5s ease-out;

-o-transition: all .5s ease-out;

}

input:focus, select:focus, textarea:focus {

outline: none;

border-color: #9edeed;

box-shadow: 0 0 2px #9edeed, 0 0 5px #9edeed, 0 0 10px #9edeed;

}

## CSS Transforms Highlights

Transforms were used extensively throughout the CSS3 animations that were used,.

Animations (detailed in the animations section) utilizing transforms:

1. Jello-horizontal uses a variety of scale3d to build its effects
2. Move-up uses translate
3. Shake uses translateX
4. scaleFromUp uses scale
5. error-vibrate uses translate extensively

A few other examples are:

1. The hamburger menu icon uses a combination transition an transform to animate the icon change on. The transform includes several uses of rotate.
2. The holly icons used in the page headers use the rotate transform to change how they are angled.

.hollyLeft {

transform: rotate(30deg);

}

.hollyRight {

transform: rotate(-30deg);

}

1. All authenticated pages use a scale transform to show or hide either the page form or the div directing the user to login first to access the page.

#authenticated {

-webkit-transform: scale(1);

transform: scale(1);

}

#unauthenticated {

-webkit-transform: scale(0);

transform: scale(0);

height: 0;

margin-top: 0;

margin-bottom: 0;

}

## CSS Animations Highlights

1. Jello – this animation creates a sort of jello-ish horizontal jiggle effect and is used on buttons in the application to draw attention to the next action to perform.
   * Index page on the Open Profile and New Profile buttons – Open Profile jiggles when the username matches a record in local storage; New Profile jiggles when the username entry does not exist in local storage and would need to be added.

.jello-horizontal {

-webkit-animation: jello-horizontal 0.9s both;

animation: jello-horizontal 0.9s both;

}

@keyframes jello-horizontal {

0% {

-webkit-transform: scale3d(1, 1, 1);

transform: scale3d(1, 1, 1);

}

30% {

-webkit-transform: scale3d(1.25, 0.75, 1);

transform: scale3d(1.25, 0.75, 1);

}

40% {

-webkit-transform: scale3d(0.75, 1.25, 1);

transform: scale3d(0.75, 1.25, 1);

}

50% {

-webkit-transform: scale3d(1.15, 0.85, 1);

transform: scale3d(1.15, 0.85, 1);

}

65% {

-webkit-transform: scale3d(0.95, 1.05, 1);

transform: scale3d(0.95, 1.05, 1);

}

75% {

-webkit-transform: scale3d(1.05, 0.95, 1);

transform: scale3d(1.05, 0.95, 1);

}

100% {

-webkit-transform: scale3d(1, 1, 1);

transform: scale3d(1, 1, 1);

}

}

1. Move up – this animation is

@keyframes moveUp {

0% {

transform: translateY(50%);

opacity: 0;

}

100% {

opacity: 1;

transform: translateY(0);

}

}

1. Error-vibrate – this animation is applied to text fields to shake them when they switch to an error state. Used:
   * Index page on the username field when left blank.

.error-vibrate {

-webkit-animation: error-vibrate 0.3s linear 6 both;

animation: error-vibrate 0.3s linear 6 both;

outline: none;

border-color: red;

box-shadow: 0 0 10px red;

}

@keyframes error-vibrate {

0% {

-webkit-transform: translate(0);

transform: translate(0);

}

20% {

-webkit-transform: translate(-2px, 2px);

transform: translate(-2px, 2px);

}

40% {

-webkit-transform: translate(-2px, -2px);

transform: translate(-2px, -2px);

}

60% {

-webkit-transform: translate(2px, 2px);

transform: translate(2px, 2px);

}

80% {

-webkit-transform: translate(2px, -2px);

transform: translate(2px, -2px);

}

100% {

-webkit-transform: translate(0);

transform: translate(0);

}

}

1. Shake – this animation is applied to buttons to shake them when they switch to an error state. Used:
   * Index page on the Open Profile and New Profile buttons when there are errors.
   * Recipient page on the Save Profile button

.btnerror {

-webkit-animation: shake 0.5s;

-moz-animation: shake 0.5s;

animation: shake 0.5s;

}

@keyframes shake {

0%, 100% {transform: translateX(0);}

10%, 30%, 50%, 70%, 90% {transform: translateX(-10px);}

20%, 40%, 60%, 80% {transform: translateX(10px);}

}

1. ScaleFromUp – This animation is used to animate an error text into the button display when the error class is applied to the button. Used:
   * Index page on the Open Profile and New Profile buttons when there are errors.
   * Recipient page on the Save Profile button.

.btnerror:after {

content: "Error!";

-webkit-animation: scaleFromUp 0.5s;

-moz-animation: scaleFromUp 0.5s;

animation: scaleFromUp 0.5s;

}

@keyframes scaleFromUp {

0% {

transform: scale(0);

opacity: 0;

}

100% {

opacity: 1;

transform: scale(1);

}

}

1. Tracking-in-expand – used to introduce a validation message on a form field. Used:
   * Index page to show a descriptive error message if the user attempts to add a new profile for a username that already exists in local storage.

.tracking-in-expand {

-webkit-animation: tracking-in-expand 0.7s cubic-bezier(0.215, 0.610, 0.355, 1.000) both;

animation: tracking-in-expand 0.7s cubic-bezier(0.215, 0.610, 0.355, 1.000) both;

}

@keyframes tracking-in-expand {

0% {

letter-spacing: -0.5em;

opacity: 0;

}

40% {

opacity: 0.6;

}

100% {

opacity: 1;

}

}

1. Tracking-out-contract– used to remove a validation message on a form field. Used:
   * Index page to remove the descriptive error message if the user attempts to add a new profile for a username that already exists in local storage after its been shown when the username field gains focus again.

.tracking-out-contract {

-webkit-animation: tracking-out-contract 0.7s cubic-bezier(0.550, 0.085, 0.680, 0.530) both;

animation: tracking-out-contract 0.7s cubic-bezier(0.550, 0.085, 0.680, 0.530) both;

}

@keyframes tracking-out-contract {

0% {

opacity: 1;

}

50% {

opacity: 1;

}

100% {

letter-spacing: -0.5em;

opacity: 0;

}

}

1. Heartbeat – Animation applied to Days ‘til shipping due on Status page when it reaches between 1 and 15 days due (calculated based on difference between today’s date and 11/20 of the gift year.)

.heartbeat {

-webkit-animation: heartbeat 1.5s ease-in-out infinite both;

animation: heartbeat 1.5s ease-in-out infinite both;

}

@keyframes heartbeat {

from {

-webkit-transform: scale(1);

transform: scale(1);

-webkit-transform-origin: center center;

transform-origin: center center;

-webkit-animation-timing-function: ease-out;

animation-timing-function: ease-out;

}

10% {

-webkit-transform: scale(0.91);

transform: scale(0.91);

-webkit-animation-timing-function: ease-in;

animation-timing-function: ease-in;

}

17% {

-webkit-transform: scale(0.98);

transform: scale(0.98);

-webkit-animation-timing-function: ease-out;

animation-timing-function: ease-out;

}

33% {

-webkit-transform: scale(0.87);

transform: scale(0.87);

-webkit-animation-timing-function: ease-in;

animation-timing-function: ease-in;

}

45% {

-webkit-transform: scale(1.5);

transform: scale(1.5);

-webkit-animation-timing-function: ease-out;

animation-timing-function: ease-out;

}

}

1. Ping -- Animation applied to Days ‘til shipping due on Status page when the difference between today’s date and 11/20 of the gift year are passed.

.ping {

-webkit-animation: ping 0.8s ease-in-out infinite both;

animation: ping 0.8s ease-in-out infinite both;

}

@keyframes ping {

0% {

-webkit-transform: scale(0.2);

transform: scale(0.2);

opacity: 1;

}

80% {

-webkit-transform: scale(1.2);

transform: scale(1.2);

opacity: 0.3;

}

100% {

-webkit-transform: scale(2.2);

transform: scale(2.2);

opacity: 0;

}

}

# Other Code Topics

* JavaScript Arrays – I use arrays throughout the application. I use arrays to maintain the list of participants (potential recipients) and also to manage the list of “favorited” pins.
* JavaScript Objects – I use JS Ojects to matain the user profile data, the data ehind recipients, and.

function Profile(username, giftYear, assignedRecipient, jsonresults, savedFavorites, lastSearchedUser, plannedGift, shippedDate, trackingnbr) {

this.username = username;

this.giftYear = giftYear;

this.assignedRecipient = assignedRecipient;

this.jsonresults = jsonresults;

this.savedFavorites = savedFavorites;

this.lastSearchedUser = lastSearchedUser;

this.plannedGift = plannedGift;

this.shippedDate = shippedDate;

this.trackingnbr = trackingnbr;

}

// define object to save Pinterest users for viewing boards

function PinterestUserBoard(firstname, lastname, id, boardname, interests, colors) {

this.firstname = firstname;

this.lastname = lastname;

this.id = id;

this.boardname = boardname;

this.interests = interests;

this.colors = colors;

this.displayname = function() {return this.lastname + ", " + this.firstname;};

this.displaynameFL = function() {return this.firstname + " " + this.lastname;};

}

* JSON & AJAX – I use AJAX calls to load JSON data from a 3rd part Pinterest API. I also use JSON parse and stringify to convert my objects into formats that can be saved to local storage and parsed ack out into objects. The json behind a favorited pin is saved into an array of json objects which are stringified to save to local storage and parsed ack into objects to display/interact with previously saved favorites.
* Local Storage – the User profile is saved to local storage including all saved favorites and a planned gift, if selected. Additionally, local storage I used to save the last logged in username, so that what multiple profiles can be saved on the same machine and when the application is opened, it will persist the last logged in user.
* DOM: The application queries the DOM extensively to create a dynamic application with updating data, styles and loaded values without distracting page refreshes.
* CSS using Javascript: -- Many functions apply or remove classes from elements on the page. Also, on the Status page, the color of days ‘til shipping is due is changed to yellow when with 15 days due and red when the shipping date is past due.

// function to validate that recipient is selected applies an error class and later reoves tht error class with a setTimeout function

function validateRecipient(){

var lst = document.getElementById('recipient');

var selRecipient = lst.options[lst.selectedIndex].value;

if(selRecipient == '' || selRecipient == '0') {

//add class to recipient

console.log(selRecipient);

document.getElementById("recipient").classList.add('error-vibrate');

setTimeout(function(){

document.getElementById("recipient").classList.remove('error-vibrate');

}, 3000);

return false;

}

else {

return true;

}

}

//color change on the days due til ship date based on the result

const getRemainingShip = (dt, id, timer) => {

const end = new Date(dt)

const now = new Date()

const distance = end - now;

const daysTil = Math.ceil(distance / timer().day);

document.getElementById(id).innerHTML = daysTil;

if(daysTil <= 15 && daysTil > 0){

document.getElementById(id).style.color = '#ffff66';

//add animation to draw attention (subtle)

document.getElementById(id).classList.add('heartbeat');

}

if(daysTil <= 0){

document.getElementById(id).style.color = '#cc0000';

//add animation to draw attention (more immediate)

document.getElementById(id).classList.add('ping');

}

}

* JavaScript Events: -- The application uses a lot of event listeners. Most pages use a DOMContentLoaded event listener to ensure JavaScript functions aren’t fired before the page is ready. All buttons use event listeners to check for clicks. There are also events on change of selects on blur of the username input, and on focus on some inputs.

//index page

document.addEventListener("DOMContentLoaded", function() {

checkLoggedIn();

// event listeners

document.getElementById("txtUserName").addEventListener("blur", checkUser, false);

document.getElementById("txtUserName").addEventListener("focus", resetButtons, false);

document.getElementById("btnLoad").addEventListener("click", loadProfile, false);

document.getElementById("btnNew").addEventListener("click", addProfile, false);

document.getElementById("lnkLogout").addEventListener("click", logOut, false);

});

//status page

document.addEventListener("DOMContentLoaded", function() {

document.getElementById("lnkLogout").addEventListener("click", logOut, false);

checkLoggedIn();

buildStandardListUsers();

sortListUsers();

loadStatus();

getRemaining('12/25/' + statusYear, 'countdown', timer);

getRemainingShip('11/20/' + statusYear, 'daystilShipDue', timer);

});

// recipient page

document.addEventListener("DOMContentLoaded", function() {

window.addEventListener("load", loadPage);

document.getElementById("lnkLogout").addEventListener("click", logOut, false);

document.getElementById("recipient").addEventListener("change", loadRecipientDetails, false);

document.getElementById("giftYear").addEventListener("change", updateGiftYear, false);

document.getElementById("giftYear").addEventListener("blur", validateGiftYear, false);

document.getElementById("btnSave").addEventListener("click", updateProfile, false);

});

//discover ideas page

document.addEventListener("DOMContentLoaded", function() {

window.addEventListener("load", loadPage);

document.getElementById("lnkLogout").addEventListener("click", logOut, false);

document.getElementById("recipient").addEventListener("change", loadRecipientDetails, false);

});

//favorites page

document.addEventListener("DOMContentLoaded", function() {

window.addEventListener("load", loadPage);

document.getElementById("lnkLogout").addEventListener("click", logOut, false);

});

# Design Process

My design process started with planning a to do list application. Then I began building a basic structure for a semi-responsive menu and nav site using some example code that I found online that I adapted to build what I wanted. This could be improved upon, and would be much easier with Bootstrap, but I wanted to push myself and try to do it. I tried to keep each page relatively simple so that the learning curve, even for older family members should not be too hard.

As I began planning and designing, I found myself more inspired y building an application to help my family members with planning craft ideas for our family Christmas exchange.

I knew that since I would not yet have a database backend that I would need to use a lot of local storage, some of it to simulate actions that might be changed in the future to be saved in a central database. I based much of the Pinterest functionality I did on the code I turned in for my JSON review assignment, and used some of the buttons and animations I had in my CSS3 Transitions/Transforms code review assignment. I expanded a lot on the Pinterest example though, by building in interactive buttons into the pin data returned from JSON, to allow users to mark pins as favorites or as the planned gift.

As I began building the pages, I planned going in to work on finding ways to incorporate transforms, transitions, and animations as holistically as possible to introduce errors/alerts or new data as naturally as possible.

# Changes from Original Concept

This differs dramatically from my original concept, since I had originally conceptualized a to-do list. I found myself bored with this idea – it felt too pedestrian and didn’t inspire me. I knew I could keep myself more committed to an end product I could be proud of if I cared about the application I was making. The Pinterest idea is something I can easily share with my mother and Aunts (as well as cousins, siblings, and nieces) to help them, since not all of them have Pinterest accounts or know how to use it. As I switched conceptual directions, I found much more motivation to create this, since it now felt meaningful and not just an exercise that would never be used y any actual user.

# Problems and Resolutions

One of the large problems I ran into was discovering limitations in the functionality offered through the Pinterest API. I had originally hoped to allow users to search on specific keywords of subjects to find results to save. When I discovered this functionality did not appear to be supported, I spent a while searching to find if there were any possible solution to provide it. Eventually I found something that might eventually be able to be adapted, although it’s not guaranteed to be possible. Since this approach is using something that does not appear to be part of Pinterest’s official API, I chose to shelve this feature for now. Instead, I chose to adapt the “Discover Ideas” function to include an expanded list of participants and to allow a user to select from any of the participants to view their pinned ideas to search for more ideas instead of being limited to searching on just your assigned recipient (which it still defaults to for ease).

Several times I ran into issues where event listeners did not trigger as expected because I needed to make sure that the event listener was not trying to apply before the DOM content was loaded. I debugged and watched what was happening when, and this helped to trigger a memory, so that I knew to add another event listener to make sure the other event listeners only happened on DOMContentLoaded.