Web Advanced: Javascript

"We will learn JavaScript properly. Then, we will learn useful design patterns. Then we will pick up useful tools for making cool things better."

SPRING 2020

SESSION #10

CANVAS & HTML5 APIs

WORKFLOWS/DevOps

jaink@newschool.edu

https://canvas.newschool.edu/courses/1482285

https://repl.it/@jaink/webjavascripts20

RECAP

HTML5 DATA ATTRIBUTES

Allows for a more structured method for passing configs and attributes from html to javascript.

Used by all javascript/jQuery plugins to define settings for the plugin.

```
<div id="slideshow" data-type="carousel"
data-transition="fade-in" data-auto-start="true">
    <div>Slide 1</div>
    <div>Slide 2</div>
</div>

const slide = document.getElementById("slideshow");
const type = slide.dataset.type;
const transition = slide.dataset.transition;
const auto_start = slide.dataset.autoStart;

// alternative generic way to get attributes:
const type = slide.getAttribute("data-type");
slide.setAttribute('data-type', 'slider' );



// OR in jQuery
const type = slide.data("data-type");
```

HTML5 GEOLOCATION

Navigator contains current position information for the client and can be accessed through:

navigator.geolocation

-> returns a Geolocation object:

```
function youAreHere(position) {
  console.log("position: ", position);
}

if(navigator.geolocation) {

navigator.geolocation.getCurrentPosition(youAreHere);
}
```

Watch for changes:

```
function youHaveMoved(position) {
  console.log("changed position: ", position);
}
if(navigator.geolocation) {
  navigator.geolocation.watchPosition(youHaveMoved);
}
```

Reference:
https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API

HTML5 VIDEO/AUDIO

A clean and extensible approach to embedding media objects:

```
<audio src="assets/08_please.mp3" controls>
  Your browser does not support the audio element.
</audio>
```

Pre-html5:

```
<div>
    <object
classid="clsid:02bf25d5-8c17-4b23-bc80-d3488abddc6b"
codebase="http://www.apple.com/qtactivex/qtplugin.cab">
      <param name="src"  value="movie.mp4" />
      <param name="autoplay"  value="false" />
      <param name="controller"  value="true" />

    <!--[if !ie] -->
      <object type="video/mp4"  data="movie.mp4">
        <param name="controller"  value="true" />
        <param name="autoplay"  value="false" />
      </object>
    <!--[endif]-->
    </object>
</div>
```

NOW:

```
<video src="assets/Nearness_on_Vimeo.mp4">
  Your browser does not support the video element.
</video>
```

HTML5 VIDEO/AUDIO

Media attributes:

```
<video src="assets/Nearness_on_Vimeo.mp4"
autoplay>
  Your browser does not support the video
element.
</video>
```

```
<video src="assets/Nearness_on_Vimeo.mp4"
controls>
  Your browser does not support the video
element.
</video>
```

```
<video src="assets/Nearness_on_Vimeo.mp4" loop>
  Your browser does not support the video
element.
</video>
```

```
<video src="assets/Nearness_on_Vimeo.mp4"
poster="assets/casa2_hero_1452814189.jpg">
  Your browser does not support the video
element.
</video>
```

HTML5 VIDEO/AUDIO

Controlling media with Javascript:

```
const video = document.getElementsByTagName("video")[0];

video.play();
video.pause();
video.volume = 90;
video.muted  = true;
video.loop   = true;
```

Events:

```
video.addEventListener("pause", function(event) {
    console.log("video has been paused");
})

video.addEventListener("play", function(event) {
    console.log("video has been paused");
})

video.addEventListener("volumechange", function(event) {
    console.log("video volume has been changed");
})

video.addEventListener("timeupdate", function(event) {
    console.log("video timecode has changed");
})
```

HTML5 CANVAS

Canvas allows graphics to be drawn onto a web page in real time through JavaScript.

```
<canvas id="canvasDrawing1" width="200"
height="100">
   This browser doesn't support the canvas
element.
</canvas>

const canvas =
document.getElementById("canvasDrawing1");
```

Context:

An object containing all the methods used to draw onto and manipulate the canvas.

```
// 2D
let context = canvas.getContext("2d");

// 3D
let context3D = canvas.getContext("webgl");
```

HTML5 CANVAS - ADDING SHAPES

Create Shapes

```
// set defaults
context.fillStyle = "#0000cc"; // a blue fill color
context.strokeStyle = "#ccc"; // a gray stroke color
context.lineWidth = 4;

// draw
context.fillRect(10,10,100,50);

// draw
context.strokeRect(10,100,100,50);


// Lines
context.beginPath();
context.moveTo(20, 50);
context.lineTo(180, 50);
context.moveTo(20, 50);
context.lineTo(20, 90);
context.strokeStyle = "#c00";
context.lineWidth = 10;
context.stroke();

// ARCs
context.arc(200, 200, 30, 0, Math.PI * 2, false);
context.strokeStyle = "#ff0";
context.lineWidth = 4;
context.stroke();
```

HTML5 CANVAS - ADDING SHAPES

```
// text
context.fillStyle = "#cc0033"; // fill color
context.font = "bold 26px sans-serif";
context.fillText("Hello", 20, 200);

// Image
let img = document.createElement('img');
img.src = 'assets/no_image.gif';
img.addEventListener('load', function() {
    context.drawImage(img, 10, 10 );
});


//Transform
context.scale(1,2) // works on anything drawn
after
context.rotate(0.1*Math.PI)
context.translate(50,100)
```

HTML5 APIs

- Prefetch API
- Camera API
- Speech Synthesis API
- Geolocation API
- Fullscreen API
- etc...

A nice list: https://github.com/diegocard/awesome-html5

WHAT IS A WORKFLOW?

- ➔ Organize the js and scss, css, assets
- ➔ better integration with source control
- ➔ Automate repetitive tasks like joining, minifying, parsing SASS, moving and renaming files etc.
- ➔ allow easy replication on other environments/team systems without changing the source code
- ➔ No more FTP!!!

COMPONENTS OF A WORKFLOW

- Source Control: Git

- Allows managing code changes over time, along with actions like alternative copies (branches), reverting the code to previous states (commits) whenever needed etc.
- Also allows better code management when working with teams in parallel.
- Github - a service used for hosting git repositories (free for open source projects)
-
- Easy guide here: http://rogerdudler.github.io/git-guide/

COMPONENTS OF A WORKFLOW

- ➔ JS Transpiler: Babel/Typescript

- ➔ Required to convert modern/edge code like ES6, Typescript etc. for all browsers.
- ➔ Required to convert the language into ES5.
- ➔ Eventually ES6 will be 100% supported and this component will not be necessary if all code is written in ES6 directly.
- ➔ Babel is still handy to completely future proof the code as it will work with the latest js release and transpile to an older format.

COMPONENTS OF A WORKFLOW

- ➔ CSS Preprocessors: LESS/SASS

- ➔ SCSS (SASS) is a scripting language that extends CSS that eventually flattens/compiles into regular CSS.
- ➔ Allows for more programmatic approaches to writing CSS styles.
- ➔ Allows features like reusable variables, nested definitions, importable modules, mixins/functions etc.

COMPONENTS OF A WORKFLOW

- ➔ Task Runner: GRUNT/GULP/WEBPACK

- ➔ Runs automated tasks on code to generate a cleaner/optimized output
- ➔ Handle all repetitive tasks, manages all the heavy lifting

COMPONENTS OF A WORKFLOW

- ➔ Code Linting: JSLINT/ESLint

- ➔ Linting checks for bugs or inconsistency in code before compiling or processing.
- ➔ Issues can be simple typos, missing punctuation etc. and most lint systems allow a customizable definition of standards to test the code against, in real time.

COMPONENTS OF A WORKFLOW

- ➔ Integrated Testing: Mocha/Jasmine/Selenium

- ➔ Requires writing specific code for each functionality in the application that tests all possible conditions.
- ➔ These tests are then run through the framework used and produces results, without manually debugging/logging etc.
- ➔ Requires time/patience and experience to write clean and comprehensive tests.
- ➔ Unit tests create small pieces of code like functions, and run isolated to verify the cleanliness of data going in and out.
- ➔ Integration tests overall system integration and needs proper scripting.
- ➔ Functional tests performs actual browser and UI testing.

REQUIREMENTS

- ➔ A little familiarity with the Terminal
- ➔ Xcode (OSX)
- ➔ Homebrew (OSX)
- ➔ NPM

SETUP

Xcode:

```
https://developer.apple.com/download

gcc -v

xcode-select --install
```

Homebrew: package manager for OSX

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/inst
all/master/install)"

brew update

brew doctor
```

Add the brew location in the profile file:

```
export PATH="/usr/local/bin:$PATH"
echo 'export PATH="/usr/local/sbin:$PATH"' >>
~/.bash_profile
```

Install Nodejs (also installs NPM):

```
brew install node
to upgrade: brew upgrade node
(or download install from nodejs.org - LTS 12.x)
```

NODE PACKAGE MANAGER

- ➔ NPM: package manager for javascript package libraries
- ➔ Installed with Node
- ➔ Contains a massive number of libraries of reusable code for Node and other javascript based applications
- ➔ https://www.npmjs.com/

```
node -v
```

To update to latest Node:

```
npm install npm@latest -g
```

(if installed without brew)

Or download latest package from nodejs.org

TESTED STEPS (OSX)

The following steps have been tested on a brand new mac laptop:

```
// for those who do not have xcode installed
xcode-select --install

//install Homebrew
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"


//these update the brew install
brew update
brew doctor

//Add the brew location in the profile file - open a new terminal
window after doing this
export PATH="/usr/local/bin:$PATH"
echo 'export PATH="/usr/local/sbin:$PATH"' >> ~/.bash_profile

brew install node
brew postinstall node

// (alternatively install from Node self-installing package available
on https://nodejs.org)

// install git
brew install git

// install gulp globally
npm install gulp gulp-cli -g

// in case of permissions error"
sudo npm install gulp gulp-cli -g

//go to project folder and initialize the project by entering default
values eg. cd <folder path>

npm init

// use the package.json file from the boilerplate files hereon
npm install
```

GIT INSTALLATION

- ➔ Git will track all versions and changes to the code
- ➔ https://git-scm.com

```
brew install git
```

Create project:

```
cd /<path>/project1
```

Create a new repo:

```
git init
```

Or clone an existing one:

```
git clone username@host:/path/to/repository
./project_folder
```

Typical commands:

```
git add *

git commit -m "Commit message"
git checkout master
git checkout -b feature_x
```

GUI (OSX): https://www.sourcetreeapp.com/

GULP INSTALLATION

- ➔ **Gulp is a task runner to handle common and frequently run tasks to automate it through a script and plugins. eg.**
- ➔ Lint JS and CSS
- ➔ Minify CSS and JS
- ➔ Autoprefix CSS
- ➔ SASS, LESS Compilation
- ➔ Minify Images
- ➔ Auto Generated SVG Sprites
- ➔ Build production ready files with file size reporting
- ➔ Uglify JS and CSS for production and finally,
- ➔ BrowserSync
- ➔ https://www.npmjs.com/

To install globally:

```
npm install gulp gulp-cli -g
```

The above allows running gulp in CLI. if this throws an error, gulp needs to be added as an npm script.

PROJECT INITIALIZATION

- ➔ Each project needs a config file called package.json that will record all the package dependencies needed for the tasks.
- ➔ All packages installed "LOCALLY" will get added to this file.
- ➔ All dependencies get downloaded into a folder inside this project ready to be used.

In Terminal go to the project folder:

```
cd "~/Documents/D&T/Faculty 2018/class 10"
```

Run this to set up the base package file. Type gulp for :

```
npm init

// for latest:
npm install --save-dev gulp
```

Install some commonly used plugins:

```
npm install --save-dev gulp-sass gulp-cssnano
gulp-sourcemaps gulp-autoprefixer
```

If gulp is not installed globally, edit package.json and add this to scripts:

```
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "gulp"
  },
```

SETUP THE TASKRUNNER

➔ **Create gulpfile.js**

```
'use strict';

// all plugins
const gulp = require('gulp');
const sass = require('gulp-sass');
const cssnano = require('gulp-cssnano');
const sourcemaps = require('gulp-sourcemaps');
const autoprefixer = require('gulp-autoprefixer');

// Expose the sas to css task
function sassworkflow() {
    return gulp
        .src('./src/sass/**/*.scss')
        // tasks go here
        .pipe(sourcemaps.init())
        .pipe(sass().on('error', sass.logError))
        .pipe(cssnano())
        .pipe(autoprefixer({
            browsers: ['last 2 versions'],
            cascade: false
        }))
        .pipe(sourcemaps.write('./'))
        .pipe(gulp.dest('./dist/css/'));
};

// Expose the task by exporting it
// This allows you to run it from the commandline using
// $ gulp sassworkflow
exports.sassworkflow = sassworkflow;

// build the parallal task
const build = gulp.parallel(sassworkflow);

// what runs when typing gulp
gulp.task('default', build);
```

Next Steps

- ➔ Workflows continued