ITMD 465/565
Rich Internet Applications

Lecture 10

Fall 2019 – October 23, 2019

Tonight’s Agenda

- HTML 5 Storage APIs
- Canvas API Demo
- Tools we will use soon

HTML5 Storage

Storage

- There are a few major storage APIs
  - http://www.html5rocks.com/en/features/storage
- Web Storage
  - https://html.spec.whatwg.org/multipage/webstorage.html
- Indexed Database
  - https://www.w3.org/TR/IndexedDB/
- Web SQL Database – This has been deprecated and will no longer be developed or supported in the future.
  - https://www.w3.org/TR/webdatabase/
- File Access
  - https://www.w3.org/TR/FileAPI/
  - https://developer.mozilla.org/en-US/docs/Web/API/File

Web Storage

- Simple and fairly widely supported way to store data in the client browser.
- Simple string only Key/Value storage – **IMPORTANT if you want to store a JavaScript object you must use JSON.stringify() and JSON.parse().**
- Different than cookies as this is a JavaScript API
- There two areas you can store data in
  - localStorage – persistant storage after browser is closed
  - sessionStorage – only stores for current browser session
- Data is saved per origin and there is a data limit
  - Firefox is around 10MB
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API

Web Storage

- The storage object is exposed on the window object
- window.localStorage or just localStorage
- You can access key/value pairs in multiple ways

```
localStorage.colorSetting = '#a4509b';
localStorage['colorSetting'] = '#a4509b';
localStorage.setItem('colorSetting', '#a4509b');
localStorage.getItem('colorSetting’);
localStorage.removeItem('colorSetting’);
```

- There is also a storage event that fires that you can listen for

Indexed DB

- IndexedDB is a low-level API for client-side storage of significant amounts of structured data, including files/blobs, which also enables high performance searches of this data using indexes.
- Transactional based database system, non-sql based, JavaScript-based object-oriented database.
- Store and retrieve objects based on a key.
- Asynchronous API that uses a lot of function callbacks
- Storage limits do exist but much larger than Web Storage and differ by browser
  - https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Browser_storage_limits_and_eviction_criteria
- https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API

Indexed DB

- The basic pattern that Indexed DB encourages is the following:
  - Open a database.
  - Create an object store in the database.
  - Start a transaction and make a request to do some database operation, like adding or retrieving data.
  - Wait for the operation to complete by listening to the right kind of DOM event.
  - Do something with the results (which can be found on the request object).
- https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Using_IndexedDB
- http://code.tutsplus.com/tutorials/working-with-indexeddb--net-34673

HTML 5 Canvas

Canvas, 2d, web gl 3d

- These are all APIs used for drawing on a <canvas> element on the page using JavaScript
- Canvas is a raster-based where the canvas objects are drawn in immediate mode. Once the item is drawn to the canvas it is forgotten. If you are doing animation you redraw the entire scene each frame. You can not modify something once it has been drawn to the canvas.
- This is different than svg which is vector-based and the shapes are remembered as objects in the DOM and then rendered to a bitmap on the screen when the HTML renders. If an svg object attribute is changed then the browser can automatically re-render.
- There are many libraries for canvas that add scene-graph capabilities to the canvas element.
- Very low level api. Much easier to use a library, especially for hit detection.

Animation

- Animation can be canvas, web gl, or just dom manipulation
- Animation requires rendering a scene with changes at a frame rate
  - Old way is with setInterval() – new way is with requestAnimationFrame()

- https://css-tricks.com/using-requestanimationframe/

Canvas

- https://en.wikipedia.org/wiki/Canvas_element
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Drawing_shape
- https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D
- Many canvas libraries, some are listed on the Canvas_API page above, another is easeljs http://www.createjs.com/easeljs
- Web GL is a 3d implementation of canvas
  - Conforms closely to OpenGL ES 2.0
  - https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
  - Libraries to make it easier like three.js http://threejs.org/

Canvas Basics

- Tag is canvas.
- Should set width and height in html but you can also change it with css.
- Need an id to select it.
- Fall back content can be provided inside the canvas tag.
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial

```
<canvas id="stockGraph" width="150" height="150”>
   current stock price: $3.15 +0.15
</canvas>

<canvas id="clock" width="150" height="150">
   <img src="images/clock.png" width="150" height="150" alt=""/>
</canvas>
```

Rendering Context

- You need to get the rendering context to run the drawing commands on
- You can test for canvas support by seeing if the getContext function exists

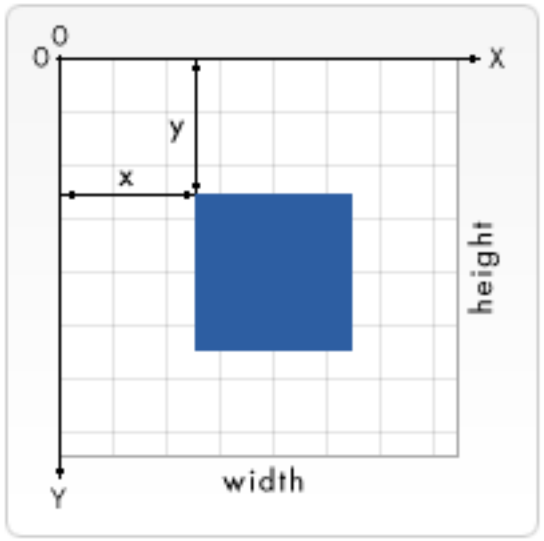
```
var canvas = document.getElementById('tutorial');
if (canvas.getContext){
        var ctx = canvas.getContext('2d');
        // drawing code here

} else {
        // canvas-unsupported code here
}
```

https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D

The Grid

- The canvas is our coordinate space.
- The **top left is (0, 0)** and it is as wide and tall as the canvas element
- Normally one grid unit corresponds to one pixel
- Grid units can be fractional
- Be careful drawing lines that do not align with the grid lines. They can not cleanly convert to pixels.



Styles and colors

- In general these properties use css syntax
- Two basic ones
- fillStyle = color Sets the style used when filling shapes.
- strokeStyle = color Sets the style for shapes' outlines.

```
// these all set the fillStyle to 'orange', ctx is the 2d context
ctx.fillStyle = "orange";
ctx.fillStyle = "#FFA500";
ctx.fillStyle = "rgb(255,165,0)";
ctx.fillStyle = "rgba(255,165,0,1)";
```

- Other properties described here.

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Applying_styles_and_colors

Rectangles

- Canvas only has one primitive shape, the rectangle
- These are functions on the context

fillRect(x, y, width, height) Draws a filled rectangle.

strokeRect(x, y, width, height) Draws a rectangular outline.

clearRect(x, y, width, height) Clears the specified rectangular area, making it fully transparent.

1, 2

Paths

- The only other primitive is paths

1. Create the path
2. Use drawing commands to draw into the path
3. Close the path
4. Fill or stroke to render the path

Paths

- beginPath() Creates a new path. Once created, future drawing commands are directed into the path and used to build the path up.
- Path methods Methods to set different paths for objects.
- closePath() Closes the path so that future drawing commands are once again directed to the context.
- stroke() Draws the shape by stroking its outline.
- fill() Draws a solid shape by filling the path's content area.

Path Methods

- Move the pen without drawing
  moveTo(x, y)
- Drawing straight lines
  lineTo(x, y)
- Draw arcs or circles (measured clockwise from the positive x axis and expressed in radians)
  Convert to degrees with this formula: (Math.PI/180)∗degrees
  arc(x, y, radius, startAngle, endAngle, anticlockwise)
  arcTo(x1, y1, x2, y2, radius)
- Bezier and quadratic curves
  quadraticCurveTo(cp1x, cp1y, x, y)
  bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)
- Rectangles added to the current path vs rectangle primitive
  rect(x, y, width, height)

3,4,5,6

Path 2d

- The path 2d object allows you to cache or record drawing commands.
- All path methods like moveTo, rect, arc or quadraticCurveTo, etc., which we got to know above, are available on Path2D objects.
- Path2D() The **Path2D()** constructor returns a newly instantiated Path2D object, optionally with another path as an argument (creates a copy), or optionally with a string consisting of SVG path data.
- new Path2D(); // empty path object
- new Path2D(path); // copy from another Path2D object
- new Path2D(d); // path from SVG path data
- Can also take in svg data

Drawing Text

- Two methods for drawing text
- fillText(text, x, y [, maxWidth]) Fills a given text at the given (x,y) position. Optionally with a maximum width to draw.
- strokeText(text, x, y [, maxWidth]) Strokes a given text at the given (x,y) position. Optionally with a maximum width to draw.
- measureText() Returns a TextMetrics object containing the width, in pixels, that the specified text will be when drawn in the current text style.
- Text properties here
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Drawing_text

Images

- Importing images into a canvas is basically a two step process:
- Get a reference to an HTMLImageElement object or to another canvas element as a source. It is also possible to use images by providing a URL.
- Draw the image on the canvas using the drawImage() function.
- Be careful to wait until the image is loaded before using. Helpful to use the images onload attribute to trigger a callback function
- drawImage function has a few different signatures

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Using_images

Transform

- To do transforms you must transform the context or grid and then draw into it. When you are done you need to undo that transform.
- The save and restore function are helpful there and can also be used to save other context states like colors.
- save() Saves the entire state of the canvas.
- restore() Restores the most recently saved canvas state.
- We will look at the transforms here
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Transformations

Basic Animation

1. Clear the canvas
2. Save the canvas state
3. Draw the animated shapes
4. Restore the state

- You also need a way to trigger your draw frame function on an interval
- setInterval(function, delay) Starts repeatedly executing the function specified by function every delay milliseconds.
- setTimeout(function, delay) Executes the function specified by function in delay milliseconds.
- requestAnimationFrame(callback) Tells the browser that you wish to perform an animation and requests that the browser call a specified function to update an animation before the next repaint.
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Basic_animations

Canvas Libraries

- There are canvas libraries that make this easier
- EaselJS is one that turns the canvas into a retained mode drawing context
- http://www.createjs.com/easeljs
- http://code.tutsplus.com/tutorials/using-createjs-easeljs--net-34840

WebGL

- WebGL is a JavaScript API that allows 2d and 3d rendering using an API very close to OpenGL ES 2.0 and usually uses hardware rendering in compatible browsers.
- Difficult to work with if you are not familiar with OpenGL
- https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
- There are libraries that make it easier but still much more complex than canvas 2d
- ThreeJS is one of the most popular
- http://threejs.org/

Tools

Build tools for Web Applications

Babel

- Babel is a tool used to transpile newer JavaScript syntax into ES5 compatible code.
- Not a bundler but can be used with bundlers
- https://babeljs.io/
- Allows you to use next gen JavaScript today
- Will need to use it to convert React JSX syntax when we start using React
- Used as a CLI node module or plugin to a bundler
- Can also be used in a browser as a script tag for prototyping

Bundlers

- These are used to process your application’s various files and bundle or package them together.
- They allow you to write modular code in multiple files, combine them, and serve them using a single script tag at its simplest.
- Two popular ones are browserify and webpack
- http://browserify.org/
- https://webpack.js.org/
- Each have plugins that can add addition features

Compile to JavaScript languages

- There are many new languages that compile down to plan JavaScript
- https://github.com/jashkenas/coffeescript/wiki/list-of-languages-that-compile-to-js
- Early one was CoffeeScript: http://coffeescript.org/
- Google was pushing Dart: https://www.dartlang.org/
- Microsoft developed typescript: https://www.typescriptlang.org/
- All provide different ways to write code with additional features not present in JavaScript and when you compile them down you get plain JavaScript.
- Then you always have plain ES6+ & Babel.js http://babeljs.io
- http://es6-features.org/#Constants

TypeScript

- TypeScript has become a popular choice.
- Angular JS 2+ shows most of their examples in TypeScript
- Strict superset of JavaScript. Any JavaScript is valid TypeScript
- TypeScript complier is built in Node JS
- Install globally npm install –g typescript or locally to project
- Compile with the command: tsc myfile.ts
- Provides support for new language features and static typing
- Produces standard ES3 JavaScript
- https://www.typescriptlang.org/

Assignments

Reading/Assignments

- Lab 4 due Sunday Oct 27. New lab will be coming soon.
- Quiz may be assigned soon
- Review Mozilla MDN canvas docs linked on each slide in this presentation. Basically the MDN canvas tutorials starting here https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Basic_usage
- Take a look at easelJS examples in canvas demo folder to see how it is different that plain canvas.
- Start reading about webpack to get a start on next week. https://webpackjs.org/concepts/ & https://webpackjs.org/guides/getting-started/
- You will need nodejs to be working next week