UI Implementation

John Jannotti

/course/cs0320/www/docs/lectures/

Mar 20, 2018
Announcements

- Come down to front for github goodies.
- This week is about getting started on maps.
  - The CLI checkpoint is “integration and maintenance.”
  - No snow day extension!
  - The final UI part has a lot of room to be creative.
  - Do NOT split the work along front-end / back-end.
  - No snow day extension!
- You’re not expect to work over break, but by having maps and your group project at top of mind, you can mull ideas.
- Today is free cone day at Ben & Jerry’s.
- Thursday is TA application deadline. Consider applying for CS33.
- Thursday will be a guest lecture on Kotlin.
Two topics today

- How can you cleanly add functionality to HTML/JS frontends?
  - We’ll spend some time on Javascript itself.
  - Then we’ll figure out how to use and create functionality as jQuery plugins.
- What goes into drawing a abstract model to a canvas?
  - Mostly mapping abstract coordinates to screen coordinates.
  - A bit about color, event handlers, etc.
Javascript

- Javascript *looks* a lot like Java.
- Javascript even *sounds* a lot like Java.
- But there are many differences, and the similarities were marketing choices.
- Fortunately, JS is still pretty nice. Just different.
- One downside, JS doesn’t impose much structure. You should.
- Another: A lot of “dark corners”
- Best book: JavaScript: The Good Parts by Douglas Crockford
- On his webpage, http://javascript.crockford.com/, you can find many worthwhile videos and other resources (like jslint).
Javascript (can be) OO, but different.

- In Java, methods are found by considering the class (and superclasses) of an object.
- In Javascript, there are only objects, not classes.
- But object fields can be functions, so you end up with syntax that looks just like Java’s method invocation.

```javascript
1 > cat = {
2    talk: function() { console.log("meow"); }
3 } 
4 { talk: [Function] } 
5 > cat.talk()
6 meow

(Using “node” as a convenient command-line js interpreter.)
```
Prototypes

- So how does “inheritance” work?
- Objects have Prototypes, not classes.
- Field lookup proceeds through prototype “chains”.
- The prototype can be set explicitly (Don’t do that.)
- Even for built in types. (Don’t ever do that.)

```
1 > a = {a1: 1, a2: 2, a3: 3}
2 { a1: 1, a2: 2, a3: 3 }
3 > b = Object.create(a);
4 {} 
5 > b.b1 = 1
6 1
7 > b
8 { b1: 1 }
9 > b.a2
10 2
```
Constructors

- Prototypes are normally set automatically by constructors.
- Any function can be a constructor, by invoking with `new`.

```javascript
1 > function Point(x, y) {
2     this.x = x;
3     this.y = y;
4 }
5 undefined
6 > pt = Point(1, 3);
7 undefined
8 > [x, y] [1, 3]
9 [1, 3]
10 > pt = new Point(4, 5);
11 { x: 4, y: 5 }
12 > pt.x
13 4
```
In Javascript, \textit{this} must be used explicitly.

Further, \textit{this} should be thought of as “context” and is set a few ways.

- During object construction with \texttt{new}, \textit{this} is a fresh object.
- Globally it begins set to some root object. \texttt{window} in browsers.
- Most Java-like: When a function is invoked “through” an object.
- \texttt{f.apply}, and \texttt{f.call} are used to invoke \texttt{f} with a specific \texttt{this}.
- \texttt{f.bind(o)} produces a new function, like \texttt{f()}, but where \texttt{this} will be \texttt{o}.
- When browsers invoke event handlers, \textit{this} is set to the DOM object.

Most confusing: \textit{this} is \textit{not} set to an object, \texttt{X}, just because you call a function that you got by saying: \texttt{const f = X.m;}
More about new

- new F() allocates a new object, and sets this to it.
- F() is conventionally crafted to set up properties on this.
- By convention, name the function with capitals. Think of it as a class.

```javascript
function Point(x, y) {
  this.x = x;
  this.y = y;
  this.manhattan = function(pt) {
    return Math.abs(this.x - pt.x) + Math.abs(this.y - pt.y);
  }
}
```
The previous slide is creating a new `manhattan` function for every `Point` created.

When `new` is used, the `prototype` field of the function is set up as the prototype of the new object.

So `Point()`’s prototype can be the one place where a single `manhattan` function object is created.

```javascript
function Point(x, y) {
    this.x = x;
    this.y = y;
}

Point.prototype.manhattan = function(pt) {
    return Math.abs(this.x-pt.x) + Math.abs(this.y-pt.y);
}
```
In EcmaScript 6

```javascript
class Point {
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }
  manhattan(pt) {
    return Math.abs(this.x - pt.x) + Math.abs(this.y - pt.y);
  }
}
```

This is “syntactic sugar” for the previous slide. Use it. You’ll cut down on your cognitive load.
So it all works out

- Taken together, these constructs make a perfectly servicable language.
- You can build abstract data types with as much or as little data hiding as you prefer.
- With the Java handcuffs off, resist the urge to “hack it until it works.”
- Use class and method in js, not just a pile functions and if statements.
- Every single cs32 student tells me their js code descends into madness. Resist! (and admit Java’s “verbose” type system was helping)
Which of these will do what it intends, without error? $ is jQuery.

```javascript
class Point {
    constructor(x, y) {
        this.x = x; this.y = y;
    }
    distance(point) {
        /* use this.x, point.x, point.y */
    }
    distance(line) {
        /* use this, line */
    }
}

class Counter {
    constructor() {
        this.value = 0;
    }
    add() {
        this.value++;
    }
}

let ctr = new Counter();
```

A) Entire Point declaration
B) `$('#link').click(alert('clicked'))`
C) Lines 6-10, then: `$('#link').click(ctr.add)`
D) `const a = [4,2,3]; a.sort(); alert("two: "+a[0]);`
E) `$('#message').style.color = '#FF2200';`
The next layer of convention

jQuery, and its plugins constitute another set of conventions.

- They make it easier to combine functionality from many sources to operate on DOM objects.
  - Calendar widgets, color pickers, fancy scrolling, etc.
- Each plugin is “tidy” with respect to namespace “pollution”.
- Each adds to the set of functions that can be called on jQuery results.
- Each functions in a similar way.
Example: Date picker from jQuery UI

- jQuery UI is a single “plugin” that offers many new functions.
- http://jqueryui.com/datepicker/
- To use:
  - Include the css and js for the plugin (and jQuery itself)
  - Create the DOM object to operate on.
  - Invoke the plugin (possibly “onload”)

```html
<script>
  $(function () {
    $('#datepicker').datepicker();
  });
</script>
<p>Date: <input type="text" id="datepicker"></p>
```
Example: Autocorrect

- Consider the “multiple inputs” design question from Autocorrect.
- Maybe you’d like to make your autocorrect into a jQuery plugin.
- http://learn.jquery.com/plugins/basic-plugin-creation/
- Think about how you’ll keep track of which of several text boxes might be in action.
- Follow conventions to keep code tidy.
(function($)
{
    // vars here are scope by the IIFE
    var privateVariable = 2;

    $.fn.autocorrect = function()
    {
        return this.each(function(textbox)
        {
            textbox.on('keyup_change', function()
            {
                $.get('/api/suggest').done(function(data)
                {
                    // Insert the suggestions into the DOM.
                    data = JSON.parse(data);
                    for (var i = 0; i < data.length; i++)
                    {
                        // use 'this' to locate the textbox
                    }
                });
            });
        });
    };
})();
Moving on to free form interfaces

- jQuery is mostly about HTML, CSS, and the DOM.
- For the most flexibility, you might need to use canvas.
- Introduced in a Lab, canvas is about totally free drawing.
  - You’ll want to use it for Maps.
- If you have a geometric model (say, roads)
  - Maybe you fetch details (roads in view) with AJAX.
  - Lines are based on latitude, longitude end points.
  - How do you translate them for drawing in canvas?
Device Coordinates – Pixels
- 0,0 in upper left (Why?)
- Pixels can be 1, 8, 24, 32 bits
- These days, in HTML/CSS a pixel (px) is not a device pixel.

Provide finest control
- The UI looks exactly as you draw it.
- Most toolkits provide access to this level, but used sparingly.
- Often called a canvas (as it is in HTML)
Coordinate Transformations

- Translation: \((x+u, y+v)\)
  - Moved everything by \((u,v)\)
  - If you want to set the origin in the middle.
- Scaling: \((Ax, By)\)
  - Everything is scaled.
  - Used to map \((0-1, 0-1)\) to an arbitrary window.
- Reflection: \((-x, y)\) or \((x, -y)\)
  - Map y coordinate so up is positive.
- Rotation \((x \cos \theta + y \sin \theta, -x \sin \theta + y \cos \theta)\)
  - Rotation around the origin of \(\theta\), degrees.

How will you draw Latitude / Longitude features?

Remember that \(1^\circ\) latitude \(\neq 1^\circ\) longitude.
Expressing transforms with matrices

- Notice that most of those transforms are just multiplication.
  - So they compose nicely.
- Matrix-vector multiplication is great notation.
- But what about translation?
Expressing transforms with matrices

- Notice that most of those transforms are just multiplication.
  - So they compose nicely.
- Matrix-vector multiplication is great notation.
- But what about translation?
- An extra (unity) dimension allows translation.
- Express the input as \((x, y, 1)\) and multiply by a 3x3 matrix.
Colors

- **Red-green-blue (RGB) color model**
  - 0-255 (in hex) for each \#rrggbh format
  - Don’t like that? rgb(128, 256, 45) works fine in HTML/CSS.

- **RGBA color model adds “alpha” channel — transparency**
  - rgba(128, 128, 128, 0.5) — half see through gray
  - Blends with what is underneath

- **HSL/HSLA color model**
  - Useful for information visualization
  - Closer to human intuition
  - hsl(120, 100%, 50%), hsla(120, 80%, 75%, 0.3);

- Choose a nice, consistent palette. Lots of sites to help.

- Need chart colors? Don’t reach for Red, Green, Blue.
Control the assembly in a composite
  - Handle window size changes

Simple layouts
  - Grids, sequential, box

Complex layouts
  - Gridbag layout
  - Spring layout

Browser: tables, style-sheets

Can be the most painful part of setting up a UI.
  - Some things in CSS are easy, some are hard. Sorry.
  - Handling resize is tricky, some designs “punt” with static layouts.
  - Testing is tricky — what constitutes a unit test?
  - We won’t hate you for using tables sparingly.
  - Look into “flexbox” if you’re ambitious.
Modern GUI Programming

- Inverted control structure
  - Normally your program has control
  - In a GUI application, the GUI has control
- Calls your program as needed
- Takes a little getting used to
- Two phase process
  - Set up the interface by defining widgets
- Register callbacks for actions
  - Start the user interface
  - Handle events as callbacks
Callback Handling

- Callback is passed an Event object
- How to handle multiple events
  - Multiple callback objects
  - Callback object looks at event
  - JS callbacks also have `this` set appropriately.
- Callback hints
  - Avoid long-running callbacks.
  - Javascript runs in a single thread.
  - “Web workers” can get around that limitation.
  - You don’t have to use anonymous functions!
    - But if you do, use “arrow” functions.