ECS 189H
WEB PROGRAMMING

4/24

Design questions

- What does the color scheme say?
- What is the font with the skinny letters? Is it the same as the regular font, only "thin"?
- How do you choose where to put regular and where to put thin font?
- The graphics are icon-like, rather than say cartoon-like or sketch-like. How does that work with the rest of the design?

Some debugging

- Web pages don't show error messages when their Javascript programs crash.
- This includes bugs in the CSS and HTML as well as Javascript.
- The error messages do show up on the console. Open it up if you suspect your program crashed!
- In Chrome, View->Developer->Javascript Console

Breakpoints and stepping

```
7     n = boxes.length;
8     for (i = 0; i < n; i++) {
9         boxes.style.left = "0px";
10    }
```

- Stops code while running (might have to reload page or push button again to get to that line…)
- Look at contents of variables by clicking on them (one reason it is nice to use lots of variables)
- Step from line to line to see execution

Using console.log()

- You can print-debug using the console.log() function
- This is your "printf" or "cout" function, results show up on Javascript console
- You can also use console.log() in the console

```
function showChildren(el) {
var children = el.childNodes;
for (var i=0; i<children.length; i++) {console.log(String(i)
   +String(children[i])+"\n");    }
}
```

Global Variables

```
var left = 0;
```

- You could define a global variable anywhere, but it is good practice to put them at the top of the file. Why?
- Globals are especially troublesome in Javascript

Accidental global variables

```
var x = "outside";
function f1 () {
  var x = "inside f1";
};
f1();  // global x contains "outside"

function f2 () {
  x = "inside f2";
};
f2();  // global x now contains "inside f2"
```

Inadvertent global variables

- Evil Javascript feature: variables assigned a value within a function but not defined with the "var" keyword are assumed to be global.
  - If there is no such global variable, it is created.
  - If there is, it is changed.
- Then if you use them by accident in another function, they'll remember the value from the first function instead of being undefined.
- Always be clear on where each variable is defined, and whether it is local or global.

Using an object instead of a global

- Instead of having left as a global, let's make it a property of the object that updates it and uses it.

```
var leftButton = {"left": 0};   // an empty object
// left is its property

// use alternative function declaration syntax
// to define a method for the object
leftButton.action = function () { … };
```

Two syntaxes to define a function

- A function:

```
var buttonAction = function () { …. }
```

- A method:

```
leftButton.action = function () { …. }
```

- Emphasizes that functions are values like any other

Using a property inside a method

- Refer to the object as "this" within it's own methods.

```
if (this.left < width-((200*n)+25)) {
      this.left = this.left+100;  // slide all boxes
      ….
```

Helpful, but not perfect

- We're less likely to mistakenly set leftButton.left than left.
- But it is still a global variable, accessible throughout the program.
- How to make it really hidden inside the leftButton object?

Object constructor with "new"

```
function CityWeather (cityParam,weatherParam) {
        this.city = cityParam;
        this.weather = weatherParam;
}

var davisWeather = new
   CityWeather("Davis","sunny");
var chicagoWeather = new
   CityWeather("Chicago","raining");
```

Constructor functions

- Usually the name of a constructor function begins with a capitol letter
- If it has parameters, they often control the initial settings of properties
- The constructor function refers the object properties using "this" since the constructor is a function, belonging to an object, referring to its own properties

Method in constructor function

```
…
 this.report = function() {        console.log("The
   weather in ",this.city," is ",this.weather);
…
davisWeather.report();
```

- As usual, a method is a property that happens to contain a function. In the function, the object itself is referred to using "this"

Private data in an object

- Constructor functions give us the opportunity to define private data that can only be accessed by methods of the object itself
- Variables defined inside a constructor function, using the "var" keyword, are local to the function (and hence private).
- This is very useful for encapsulation: making data change only through well-defined interfaces

Private data

```
 …
 var today = "Monday";
 this.report = function() {        console.log("The
   weather in ",this.city," is ",this.weather," on ",today);
```

- The method can print out the property "today"
- But "today" cannot be read or written from outside the object

Private version of left

```
function leftButtonConstruct() {
   var left = 0;
   this.action = function () {
         …
   }
}

var leftButton = new leftButtonConstruct();
```

Variable scope

- Private variables are available everywhere inside their objects.
- Any variable declared in a function is available throughout the function (not just inside its block, like in C)
- Global variables (declared when the Javascript file is loaded, outside any function) are available throughout the file.
- You can have both a global and a local variable with the same name. But it is a terrible idea. Why?