ECS 189H
WEB PROGRAMMING

4/26

Objects so far

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

```
var leftButton = {"left": 0};
// left is its only property for now
// define a method:
leftButton.action = function () { … };
```

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","Windy");
```

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 def 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 and methods that can only be accessed by the 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 day = "Wednesday";
this.report = function() {        console.log("The
  weather in ",this.city,"is",this.weather,"on",day);
```

- The method can use the value of property day
- But day cannot be read or written from outside the object

Private data example

davisWeather.day = "Thursday"

- Does not cause an error (hey, this is Javascript…)

davisWeather.report()

- What will it say?

Private data example

davisWeather.day = "Thursday"

- Does not cause an error (hey, this is Javascript…)

davisWeather.report()

- What will it say?

The weather in Davis is Sunny on Wednesday

Private method

- Can be called from within the object, but is invisible to code outside of the object.

```
function initString() {       // private method
  …
  }
```

Public vs privileged methods

- Public methods are created by code outside the object constructor. They can't access private data. They are limited to the outside world's view of the object.

```
davisWeather.showSecret = function()
  { console.log(day); }

// causes an error since as far as it knows, day is
  undefined
```

Privileged methods

- Privileged methods are defined in the object constructor, using "this". They can use private variables and call private methods, but they can be called (but not modified or examined) by code outside the object.
- Thus, data hiding!

Variable scope

- A local variable declared in a function is available throughout the function (not just inside its block, like in C)
- Private variables and methods are available everywhere inside their objects.
- Global variables (declared outside any function, or without the var keyword anywhere, including within an object) are available throughout the file.

Private version of left

```
function LeftButtonConstruct() {
  var left = 0;
  this.action = function () {
      …
}}
var leftButton = new LeftButtonConstruct();
```

In HTML: <button onclick = "leftButton.action()">

Using the private variable

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

- Do not use the "this" keyword! "this" means that it is an external property of the object; it would silently create the property. Without it, we are using the private variable.

My favorite string methods

```
str = "Tue, 25 Apr 2017 10:00 AM PDT";

str.split(" ");
```

- *["Tue,", "25", "Apr", "2017", "10:00", "AM", "PDT"]*

```
dayOfWeek = str.split(" ") [0];

dayOfWeek.slice(0,-1);
```

- *"Tue"*

Split

- Split divides its string into an array of substrings, by cutting out the split-character or substring you give it.

"http://www.cs.ucdavis.edu".split("/");

- *["http:", "", "www.cs.ucdavis.edu"]*

- Why is the empty string in the array?

Slice vs substring

- There are two Javascript methods that do almost the same thing.
- Also there's substr, which is slightly different.

dayOfWeek = "Tue,"

d = dayOfWeek.slice(1,2)

- *"u"*

d = dayOfWeek.substring(1,2)

- *"u"*

Slice vs substring

- There are two Javascript methods that do almost the same thing.

d = dayOfWeek.slice(0,-1)

- *"Tue"*

d = dayOfWeek.substring(0,-1)

- ""

- Slice uses negative numbers as offsets from end of string. Substring takes them as negative from beginning. I prefer slice.