ECS 89

5/28

Announcements

- Next assignment due Tu June 3
- Final in this room, Wds June 11, 8AM

- Agenda for today:
  - Assignment
  - Objects
  - DOM events
  - Animation loop
- Theme: functions in Javascript are objects

Assignment: A game. Any game.

- What could we do with this?

Objects

- Creator function sets attributes.
- Python "self" → Javascript "this"

```
// creator function for object
function Paddle(halfWidth) {
   this.x = 200;
   this.halfWidth = halfWidth;
   this.hot = false;
   ….
```

Methods

- A method is just an object attribute that happens to be a function.

```
// a method
this.draw = function() {
      if (this.hot) { ctx.fillStyle =  "rgb(255, 255, 100)"; }
      else { ctx.fillStyle = "rgb(170, 215, 130)";  }
      ctx.fillRect(this.x-this.halfWidth, 400-20,
                                    this.halfWidth*2, 10);
}
```

Note: function has no name

```
this.draw = function() {
   …..
```

- "this.draw" is an attribute of canvas; it contains a function-object
- The function itself has no name
- It is certainly possible to put a function with a name into an attribute; we've seen that in HTML:

```
<button type="button" onclick="myFunction()">Try it</button>
```

Two methods to change the color

```
// another method - called when mouse is pressed
this.beHot = function(e) {
    this.hot = true;
}

// another method - called when mouse is released
this.beCool = function(e) {
    this.hot = false;
}
```

DOM events

- We have already seen one kind of DOM event:

```
<button type="button" onclick="myFunction()">Try it</button>
```

- Interaction with mouse clicks, motion, keyboard clicks can be associated with any DOM element.
- For the game, we associate them with the canvas.
- The DOM element is an object (eg. canvas); these attributes of the object are functions that are called when the event happens.

DOM events

- We can put a function into the attribute in the Javascript code instead of in the HTML:

```
var canvas = document.querySelector("canvas")

function grabEvents() {
  // let paddle respond to all events
  canvas.onmousedown = function(e) {pad.beHot(e)};
  canvas.onmouseup = function(e) {pad.beCool(e)};
}
```

Possible mouse events

- These are called by touchpad, trackball, etc.

onclick, ondblclick

onmousedown, onmouseup

onmousemove

onmouseover, onmouseout

- There is a touch interface that does similar things for fingers on a touchscreen.

Event object

```
canvas.onmousedown = function(e) {pad.beHot(e)};
```

- The event function is called … by what? …with a parameter that is an event object; it has a number of useful attributes. For a mouse event:
  - clientX
  - clientY
  - button – which mouse button was pressed or released
- You'll need to Web surf about these

onmousemove

- We'll need this to get one of our custom objects to follow the mouse…something like this:

```
canvas.onmousemove = function (e) {pad.follow(e);}
```

And in the Paddle object creation:

```
this.follow = function (e) {
    x = e.clientX;
    y = e.clientY;
    …
```

Animation

- Also handled by running a function
- It draws a (possibly) slightly different picture – a frame - each time, creating the illusion of motion

Very nice animation mechanism

```
function frame() {
   requestAnimationFrame(frame);  // request the next frame
   updateAnimation();  //draw
 }
```

- requestAnimationFrame says this function – frame – should be called …by who? … the next time the screen refreshes (typically in 1/60th of a second)
- Then we draw the picture
- So this is kind of an infinite loop