

\*

# *Lecture 23*

JavaScript Timed Events & Drawing

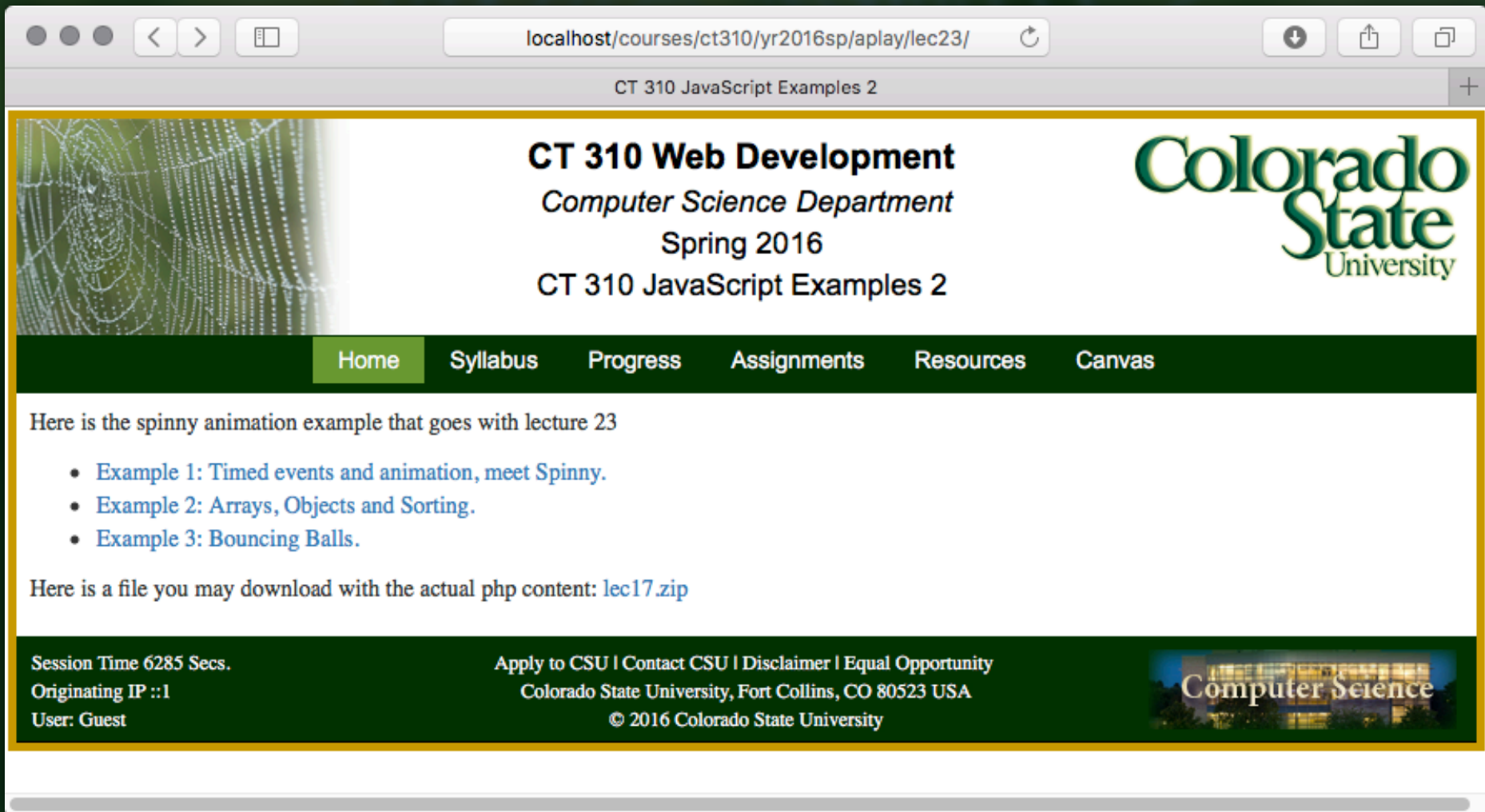
\*

Course logo spider web photograph from [Morguefile](#) openstock photograph by Gabor Karpati, Hungary.

# *Goals for Today*

- More about events including timers.
  - Essence of animation.
- Using JavaScript to modify the DOM
  - Effectively rewriting HTML.
- More about JavaScript Objects.
- Along the way, more visual elements.
  - Details about CSS and placement.
  - Semi-transparent images.

# Example Overview



The screenshot shows a web browser window with the address bar containing `localhost/courses/ct310/yr2016sp/aplay/lec23/`. The page title is "CT 310 JavaScript Examples 2". The main content area features a header with the course title "CT 310 Web Development", the department "Computer Science Department", the semester "Spring 2016", and the specific page title "CT 310 JavaScript Examples 2". The Colorado State University logo is visible in the top right corner. Below the header is a navigation menu with links for "Home", "Syllabus", "Progress", "Assignments", "Resources", and "Canvas". The main text area contains the following content:

Here is the spinnny animation example that goes with lecture 23

- [Example 1: Timed events and animation, meet Spinny.](#)
- [Example 2: Arrays, Objects and Sorting.](#)
- [Example 3: Bouncing Balls.](#)

Here is a file you may download with the actual php content: [lec17.zip](#)

The footer contains session information on the left: "Session Time 6285 Secs.", "Originating IP :::1", and "User: Guest". In the center, it lists "Apply to CSU | Contact CSU | Disclaimer | Equal Opportunity", "Colorado State University, Fort Collins, CO 80523 USA", and "© 2016 Colorado State University". On the right, there is a "Computer Science" logo featuring a building image.

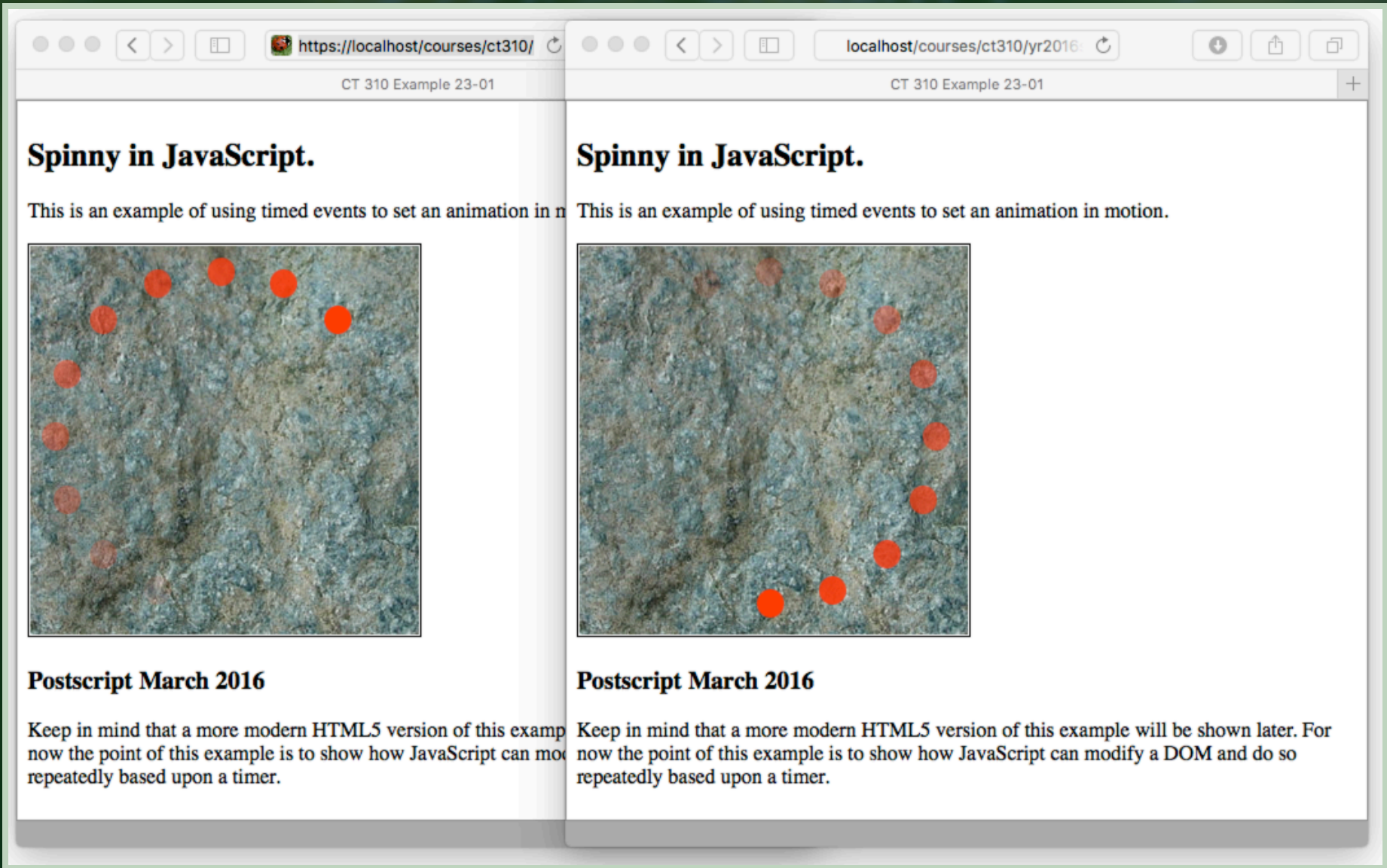
# *Timed Events*

- Consider Triggers for JavaScript
  - Mouse movement
    - Enter, leave, over,
  - Mouse Buttons
  - Move between form fields
  - Press of the return/enter key
- But, all require the user to instigate!

What if you want a page to hum along doing something without external user input?




# A Spinning Pattern



The image displays two browser windows side-by-side, illustrating a JavaScript animation. Both windows show a page titled "CT 310 Example 23-01" with the heading "Spinny in JavaScript." and the text "This is an example of using timed events to set an animation in motion." Below the text is a square image of a textured blue and green background with ten red dots arranged in a circular pattern. The left window shows the dots in their initial position, while the right window shows the same pattern rotated clockwise. Below the image in both windows is the text "Postscript March 2016" and a paragraph: "Keep in mind that a more modern HTML5 version of this example will be shown later. For now the point of this example is to show how JavaScript can modify a DOM and do so repeatedly based upon a timer."

**Spinny in JavaScript.**

This is an example of using timed events to set an animation in motion.

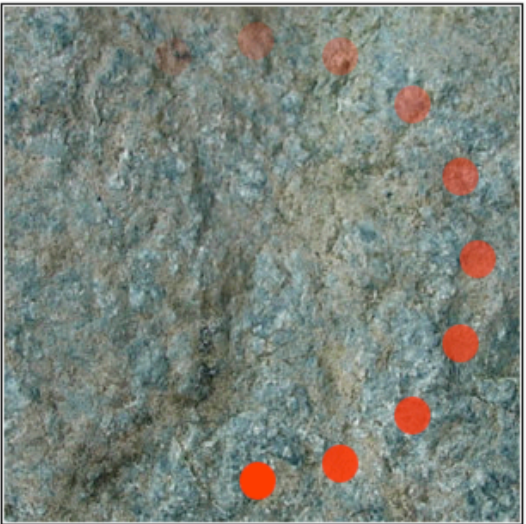


**Postscript March 2016**

Keep in mind that a more modern HTML5 version of this example will be shown later. For now the point of this example is to show how JavaScript can modify a DOM and do so repeatedly based upon a timer.

**Spinny in JavaScript.**

This is an example of using timed events to set an animation in motion.



**Postscript March 2016**

Keep in mind that a more modern HTML5 version of this example will be shown later. For now the point of this example is to show how JavaScript can modify a DOM and do so repeatedly based upon a timer.

# *What it does ...*

- A moving pattern over an image.
- Apparent motion of a single object.
- Object moves in a circle.
- Tail fades to transparent.
- Object stays in center of picture
  - Even if content moves picture on page.

# Ongoing Sequence of Events

- Initialization is critical, then
- Note that atomic action 'placeEm()'

```
17= function init() {  
18     dots    = document.getElementsByName("dot");  
19     stones = document.getElementById("stones");  
20     placeEm();  
21     setTimeout(doStep, 100);  
22     //setInterval(doStep,100);  
23 }  
24= function doStep() {  
25     theta = theta - delta;  
26     placeEm();  
27     setTimeout(doStep, 100);  
28 }
```



# Trigonometry Is Important

- Review exactly what happens each time step in the animation.

```
3 var radius = 128;
4 var theta = 0.0;
5 var delta = Math.PI / 8;
6 var voff = 138;
7 var hoff = 138;
8
9 function placeEm() {
10     for (var i = 0; i < dots.length; i++) {
11         row = voff + (radius * Math.cos(theta + (i * delta)));
12         col = hoff + (radius * Math.sin(theta + (i * delta)));
13         dots[i].style.top = parseInt(row) + 'px';
14         dots[i].style.left = parseInt(col) + 'px';
15     }
16 }
```



# JavaScript Debugging

**Spinnny in Java**

This is an example of u

```
1 var dots, row, col;
2 var stones; // This will be the image object displaying the stone
3 var radius = 128; spinnny.js
4 var theta = 0.0;
5 var delta = Math.PI / 8;
6 var hoff = 138; // This will be the image object displaying the
7 var voff = 138;
8 var theta = 0.0;
9 var delta = Math.PI / 8;
10 for (var i = 0; i < dots.length; i++) {
11   row = voff + (radius * Math.cos(theta + (i * delta)));
12   col = hoff + (radius * Math.sin(theta + (i * delta)));
13   function placeEm() {
14     dots[i].style.top = parseInt(row) + 'px';
15     dots[i].style.left = parseInt(col) + 'px';
16   }
17   dots[i].style.top = parseInt(row) + 'px';
18   dots[i].style.left = parseInt(col) + 'px';
19 }
20 function init() {
21   dots = document.getElementsByTagName("dot");
22   stones = document.getElementById("stones");
23   placeEm();
24   setTimeout(doStep, 100);
25 }
26 //setInterval(doStep, 100);
27 function doStep() {
28   theta = theta - delta;
29   placeEm();
30   setTimeout(doStep, 100);
31 }
32 }
```

**Postscript March**

Keep in mind that a mo  
now the point of this ex  
repeatedly based upon.

**Postscript Ma**

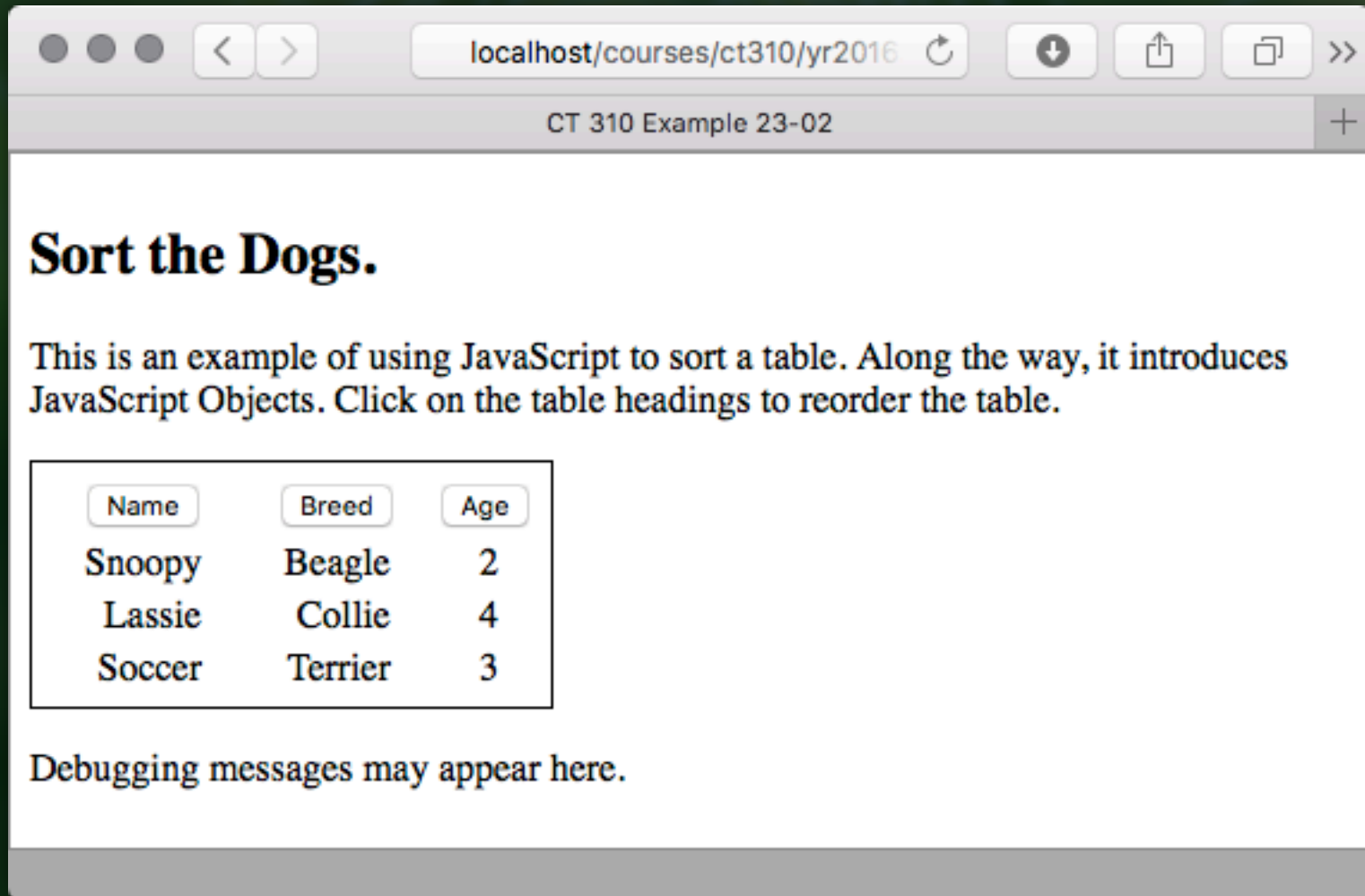
Keep in mind that a mo  
now the point of this ex  
repeatedly based upon.

Learn to set break points and watch variables.

# *Take Home Lessons*

- Another example showing interplay:
  - CSS for naming
  - JavaScript for actions on named objects
- Timed events for animation
- Relative and absolute placement
- Collecting sets of objects by name
- The Math object

# Sort A Table Example



The screenshot shows a web browser window with the address bar containing 'localhost/courses/ct310/yr2016'. The page title is 'CT 310 Example 23-02'. The main content area has the heading 'Sort the Dogs.' followed by a paragraph explaining that JavaScript is used to sort a table and that clicking on the table headings will reorder it. Below the text is a table with three columns: Name, Breed, and Age. The table contains three rows of data: Snoopy (Beagle, 2), Lassie (Collie, 4), and Soccer (Terrier, 3). Below the table, there is a note that debugging messages may appear here.

## Sort the Dogs.

This is an example of using JavaScript to sort a table. Along the way, it introduces JavaScript Objects. Click on the table headings to reorder the table.

Name	Breed	Age
Snoopy	Beagle	2
Lassie	Collie	4
Soccer	Terrier	3

Debugging messages may appear here.

# *What it Does*

- Presents a table with three dogs
  - .. true name for Wish Bone was Soccer.
- Table content is loaded dynamically.
- Table content adjusted dynamically.
- Events and buttons used.
- All changes are client side.



Side by side of the page view and show source. Stop thinking in terms of HTML source.

## Sort the Dogs.

This is an example of using JavaScript Objects. Click on the table

Name	Breed	Age
Snoopy	Beagle	2
Lassie	Collie	4
Soccer	Terrier	3

Debugging messages may appear here.

Table has only one row!

```
ct310lec23ex02.php
1 <!DOCTYPE html>
2 <html lang="en-US">
3 <head>
4 <title> CT 310 Example 23-02 </title>
5 <link href="ct310lec23.css" rel="stylesheet" type="text/css" />
6 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
7 <link href="woof.css" rel="stylesheet" type="text/css" />
8 <script type="text/javascript" src="woof.js"></script>
9 <script type="text/javascript">
10     window.onload = init;
11 </script>
12 </head>
13 <body>
14     <div id="contents">
15         <h2 style="margin-left: auto; margin-right: auto">Sort the Dogs.</h2>
16         <p>This is an example of using JavaScript to sort a table. Along the
17             way, it introduces JavaScript Objects. Click on the table headi
18             reorder the table.</p>
19         <table id="wooftable">
20             <tr>
21                 <th><input type="button" onclick="sortName()" value="Name" ,
22                 <th><input type="button" onclick="sortBreed()" value="Breed"
23                 <th><input type="button" onclick="sortAge()" value="Age" />
24             </tr>
25         </table>
26         <p id="debug">Debugging messages may appear here.</p>
27     </div>
28     <!-- end of the page contents -->
29 </body>
30 </html>
31
```

# Think About the DOM

The screenshot shows a web browser window with a page titled "Sort the Dogs." The page contains a table with three columns: Name, Breed, and Age. The table has three rows of data. The first row is highlighted in green, and the second row is highlighted in yellow. A tooltip is visible over the second row, showing "td.woofit 81px x 20px" and "Role gridcell".

The Web Inspector shows the DOM tree for the page. The selected element is a `tr` element within a `tbody` element, which is part of a `table` element with the ID "wooftable". The table structure is as follows:

```
<!DOCTYPE html>
<html lang="en-US">
  <head>...</head>
  <body>
    <div id="contents">
      <h2 style="margin-left: auto; margin-right: auto">Sort the Dogs.</h2>
      <p>
        "This is an example of using JavaScript to sort a table. Along the way, it
        introduces JavaScript Objects. Click on the table headings to reorder the
        table."
      </p>
      <table id="wooftable">
        <tbody>
          <tr>...</tr>
          <tr>
            <td class="woofit">Snoopy</td>
            <td class="woofit">Beagle</td>
            <td class="woofit">2</td>
          </tr>
          <tr>...</tr>
          <tr>...</tr>
        </tbody>
      </table>
```

Pretty obvious that contents are inserted client side using JavaScript.

# JavaScript Dog Object

- Objects key in this example.
- Here is addRow, also need a loadRow.

```
1 var woofs = new Array();
2
3 function Dog(name, breed, age) {
4     this.name = name;
5     this.breed = breed;
6     this.age = age;
7
8     this.addRow = function() {
9         var i = document.getElementById('wooftable').rows.length;
10        var rr = document.getElementById('wooftable').insertRow(i);
11        var rt = "<tr><td class=\"woofit\">" + this.name
12        + "</td> <td class=\"woofit\">" + this.breed
13        + "</td> <td class=\"woofit\">" + this.age + "</td></tr>";
14        rr.innerHTML = rt;
15    }
```

# Let Others Sort for You

## Array.prototype.sort()

### SEE ALSO

Standard built-in objects

#### Array

##### ▼ Properties

Array.prototype

Array.prototype.length

##### ▼ Methods

Array.from()

Array.isArray()

 Array.observe()

Array.of()

Array.prototype.concat()

Array.prototype.copyWithin()

Array.prototype.entries()

Array.prototype.every()

The `sort()` method sorts the elements of an array *in place* and returns the array. The sort is not necessarily [stable](#). The default sort order is according to string Unicode code points.

## Syntax

```
arr.sort([compareFunction])
```

## Parameters

### compareFunction

Optional. Specifies a function that defines the sort order. If omitted, the array is sorted according to each character's [Unicode](#) code point value, according to the string conversion of each element.

## Description

If `compareFunction` is not supplied, elements are sorted by converting them to strings

[https://developer.mozilla.org/en-US/docs/web/JavaScript/Reference/Global\\_Objects/Array/sort](https://developer.mozilla.org/en-US/docs/web/JavaScript/Reference/Global_Objects/Array/sort)



# Basics for Dogs Example

- JavaScript let functions be defined and passed as arguments.

```
37= function sortName() {
38     woofs.sort(function(a, b) { return(res = a.name > b.name ? 1 : -1) });
39     redoDogs();
40 }
41
42= function sortBreed() {
43     woofs.sort(function(a, b) { var res = a.breed > b.breed ? 1 : -1; return(res) });
44     redoDogs();
45 }
46
47= function sortAge() {
48     woofs.sort(function(a, b) { var res = a.age > b.age ? 1 : -1; return(res) });
49     redoDogs();
50 }
```

# *Onclick now Simple*

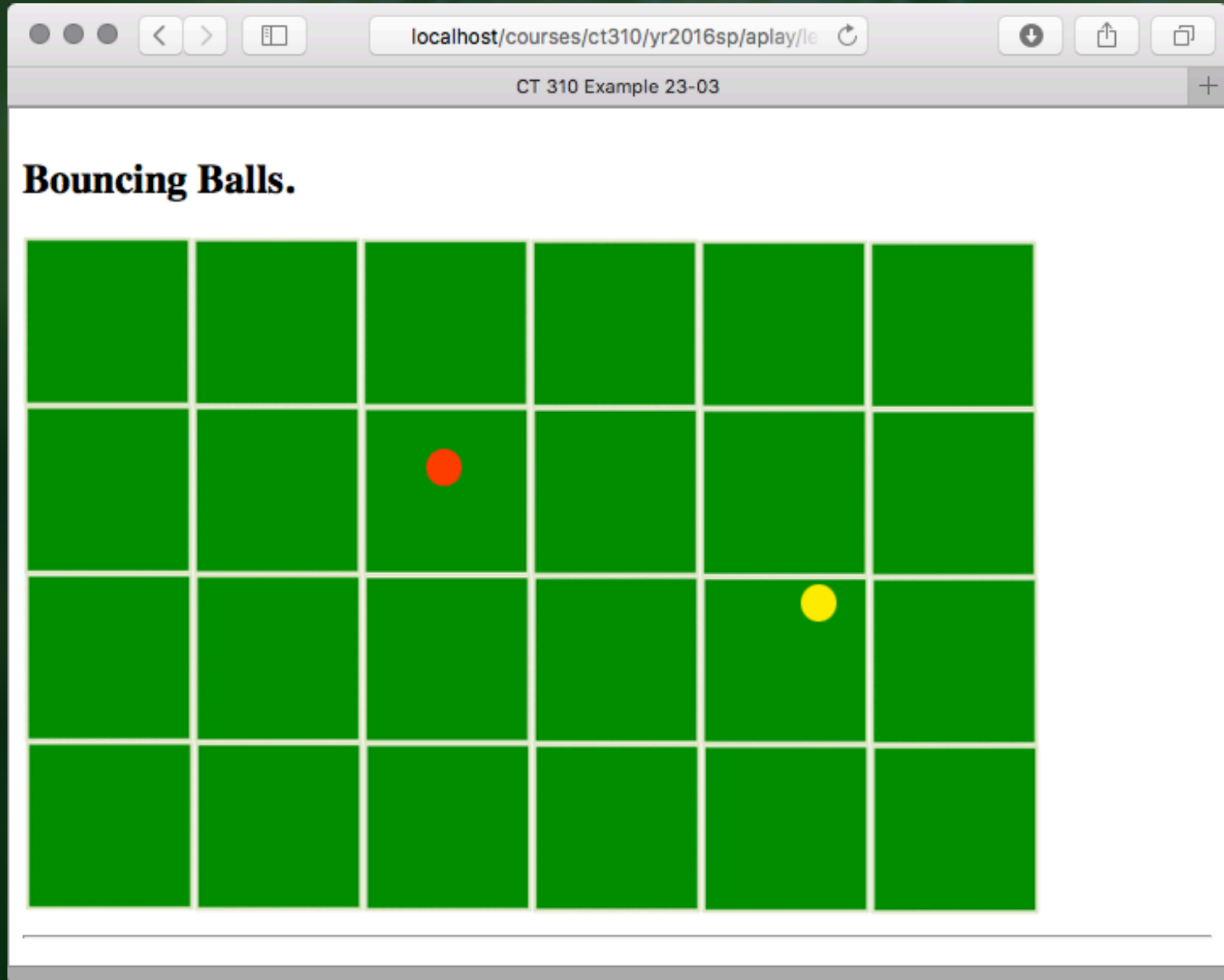
- Now revisit the header row in the table.
- Because of encapsulation in JavaScript the HTML is simple and straight forward.

```
<table id="wooftable">
  <tr>
    <th><input type="button" onclick="sortName()" value="Name" /></th>
    <th><input type="button" onclick="sortBreed()" value="Breed" /></th>
    <th><input type="button" onclick="sortAge()" value="Age" /></th>
  </tr>
</table>
```

# *Take Home Lessons*

- JavaScript objects
  - Properties and methods
- Accessing table contents
  - Many ways to fill a table
- Arrays of objects and sorting
  - Sort based upon user defined function

# Bouncing Balls





# *What it Does*

- Presents a playing board.
- Two balls are in motion.
- They 'hit' the sides and are reflected.
- Motion appears essentially smooth.
- There is not collision detection.

# *Take Home Lessons*

- As with Spinny, combine relative and absolute placement.
- Updates per second – how smooth?
- Fakery – apparent physics of hitting edge of the image.
- Next step – collision detection
  - ... and new directions.