Lecture 16

408/508 *Computational Techniques for Linguists*

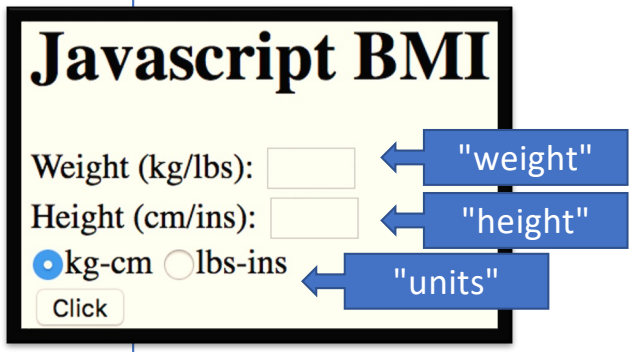
Today's Topic

- BMI Gauge animated display example
  - *using <form> elements*
  - *using <table>*
  - *using Javascript to compute BMI value*
  - *using the gaugeSVG.js widget to display the BMI value*
- Homework 7:
  - use the Gauge!

BMI: <form>… </form>

```
27</head>¶
28<body>¶
29<h1>Javascript BMI</h1>¶
30<div>¶
31<form action="" method="GET">¶
32Weight (kg/lbs): <input type="text" name="weight" size=5>¶
33<br>¶
34Height (cm/ins): <input type="text" name="height" size=5>¶
35<br>¶
36<input type="radio" name="units" value="kg" checked>kg-cm¶
37<input type="radio" name="units" value="lbs">lbs-ins¶
38<br>¶
39<input type="button" name="button" Value="Click" onClick="computeBMI(this)">¶
40</form>¶
41</div>¶
42<div id="output"></div>¶
43</body>¶
44</html>¶
```

"output" is (currently) an empty division

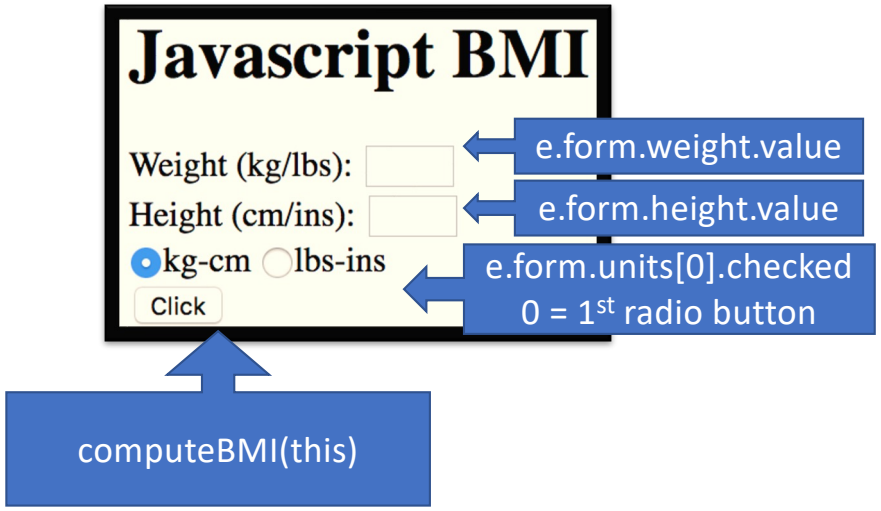


File: `bmi-js.html`

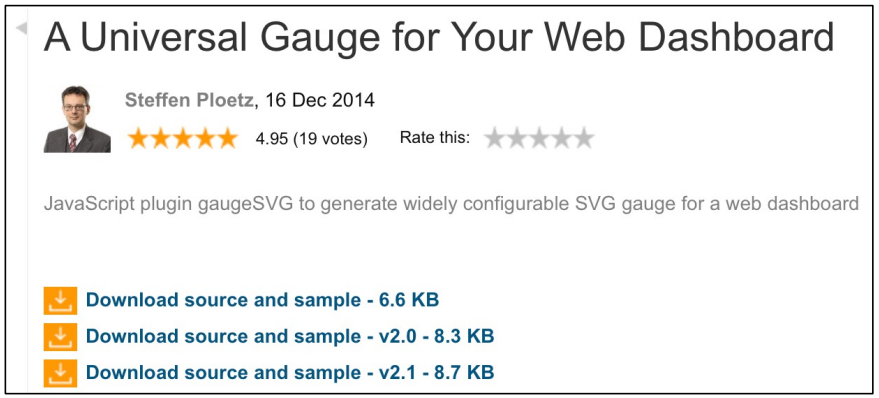
BMI: Javascript function `computeBMI()`

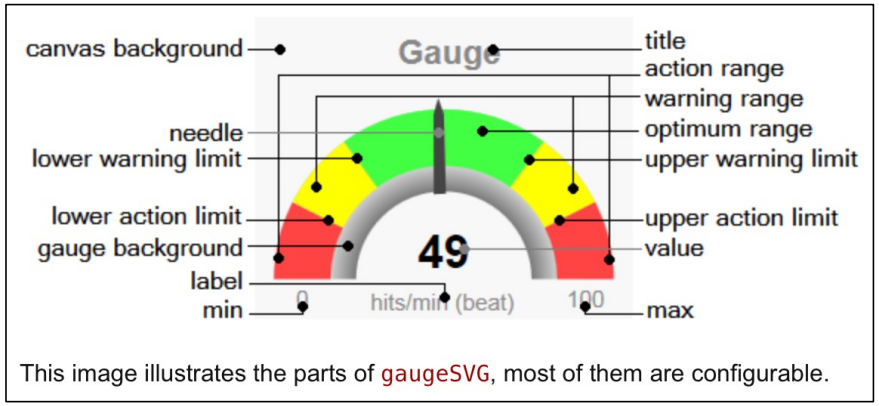
File: `bmi-js.html`

```
<!DOCTYPE html>
<html>
<head>
<title>Javascript BMI</title>
<script>
function computeBMI(e) {
    var weight = e.form.weight.value;
    var height = e.form.height.value;
    var output = document.getElementById("output");
    var scale = e.form.units[0].checked ? 10000 : 703; // kg-cm
    var bmi = weight * scale / (height *height);
    var range;
    if (bmi < 18.5) {
    range = "underweight"
    } else if (bmi < 25) {
    range = "normal"
    } else if (bmi < 30) {
    range = "overweight"
    } else {
    range = "obese"
    }
output.innerHTML =
    "BMI: " + bmi.toFixed(2) +
    "<br>Range: " + range;
}
</script>
```



BMI: `gaugeSVG.js`



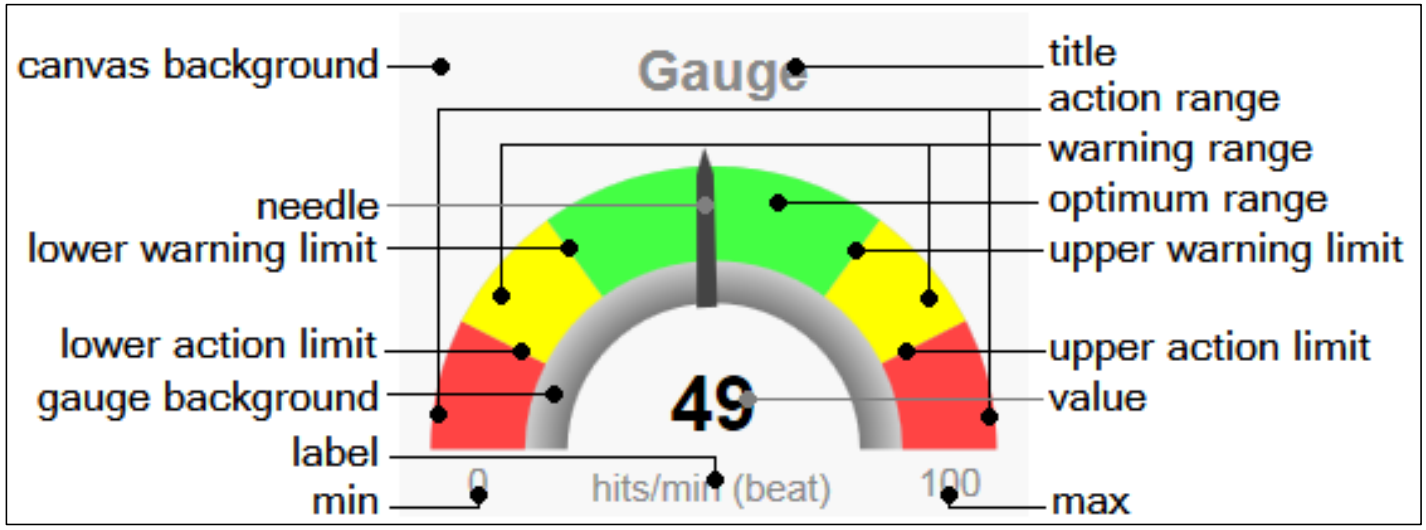


http://www.codeproject.com/Articles/604502/A-universal-gauge-for-your-web-dashboard

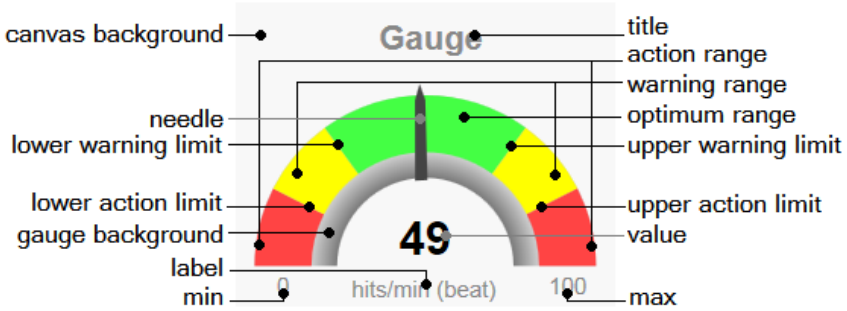
`bmi-gauge.html`
Download `gaugeSVG.js` from the course webpage
(*I've modified the original code a bit*)

BMI: `gaugeSVG.js`

- **Note**: I*'ve modified his code slightly to allow for different colors for lower and upper warning ranges*

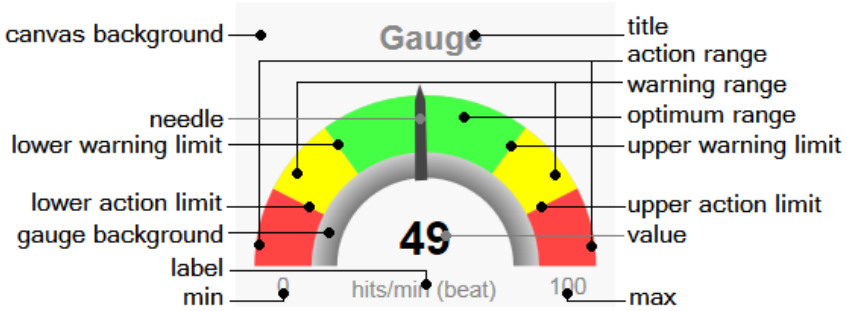


BMI: gaugeSVG.js



- `title:` [string] The title text that is displayed above the gauge. It can be an emty string to be suppressed. Default is an emty string.
- `titleColor:` [#rrggbb] The title color. Default is "#888888".
- `value:` [float] The value to display. Default is (max - min) / 2.0. Values below `min` are shown as `min`. Values above `max` are shown as `max`.
- `valueColor:` [#rrggbb] The value text color. Default is "#000000". The accurate value is shown in the center of the gauge as text.
- `label:` [string] The label displayed below the value text. It can be an emty string to be suppressed. Default is emty string. Typically used to display the value's measuring unit.
- `labelColor:` [#rrggbb] The label text color. Default is "#888888".
- `min:` [float] The minimum of the gauge display range. Will be displayed as text at the gauge start point, if is `showMinMax` true.
- `max:` [float] The maximum of the gauge display range. Will be displayed as text at the gauge start point, if is `showMinMax` true.
- `showMinMax:` [bool] Hide or display the `min` and `max` gauge display range values as text. Default is true.
- `minmaxColor:` [#rrggbb] The `min` and `max` value's text color. Default is "#888888".
- `canvasBackColor:` [#rrggbb] The background color of the gauge canvas. Default is "#f8f8f8".
- `gaugeWidthScale:` [float] The width of the gauge arc. Default is 1.0. Meaningful values range fron 0.15 to 1.5. Lower values show a smaller arc, higher values show a thicker arc.
- `gaugeBorderColor`: [float] The gauge arc border color. Default is "#cccccc".
- `gaugeBorderWidth`: [#rrggbb] The gauge arc border width. Default is 0.

BMI: gaugeSVG.js



- `gaugeBackColor:` [#rrggbb] The gauge arc background color. Default is "#cccccc".
- `showGaugeShadow`: [bool] Hide or display a gauge arc shadow. Dafault is true. The gauge shadow is made of a SVG radial gradient. The gradient start color is the `gaugeShadowColor`. The gradient stop color is the `gaugeBackColor`.
- `gaugeShadowColor`: [#rrggbb] The gauge arc shadow color. Dafault is "#000000".
- `gaugeShadowScale`: [float] The width of the gauge arc's shadow. Default is 1.0. Meaningful values range fron 0.8 to 1.5. Lower values show a smaller shadow, higher values show a thicker shadow.
- `lowerActionLimit:` [float] The lower action limit or a negative value, if not desired. Default is (max - min) * 0.15 + min.
- `lowerWarningLimit:` [float] The lower warning limit or a negative value, if not desired. Default is (max - min) * 0.30 + min.
- `upperWarningLimit:` [float] The upper warning limit or a negative value, if not desired. Default is (max - min) * 0.70 + min.
- `upperActionLimit:` [float] The upper action limit or a negative value, if not desired. Default is (max - min) * 0.85 + min.
- `needleColor:` [#rrggbb] The gauge needle color. Default is "#444444".
- `optimumRangeColor:` [#rrggbb] The optimum range color. Default is "#44ff44".
- `warningRangeColor:` [#rrggbb] The warning range color. Default is "#ffff00".
- `actionRangeColor:` [#rrggbb] The action range color. Default is "#ff4444".

BMI: <table>… </table>

```
<body>
<h1>Javascript SVG/BMI</h1>
<div>
<form action="" method="GET">
Weight (kg/lbs): <input type="text" name="weight" size=5>
<br>
Height (cm/ins): <input type="text" name="height" size=5>
<br>
<input type="radio" name="units" value="kg" checked>kg-cm
<input type="radio" name="units" value="lbs">lbs-ins
<br>
<input type="button" name="button" Value="Click" onClick="computeBMI(this)">
</form>
</div>
<div style="width:100px;color:red" id="output"></div>
<div id="gauge-div" style="width: 250px; height: 200px"></div>
<div>
  <table>
    <tr><th>Color</th><th>Range</th></tr>
    <tr>
      <td class="color" style="background-color: #eeee00"></td><td>Underweight</td>
    </tr>
    <tr>
      <td class="color" style="background-color: #44ff44"></td><td>Normal</td>
    </tr>
    <tr>
      <td class="color" style="background-color: #ff8800"></td><td>Overweight</td>
    </tr>
    <tr>
      <td class="color" style="background-color: #ff0000"></td><td>Obese</td>
    </tr>
</div>
</body>
```

<table>

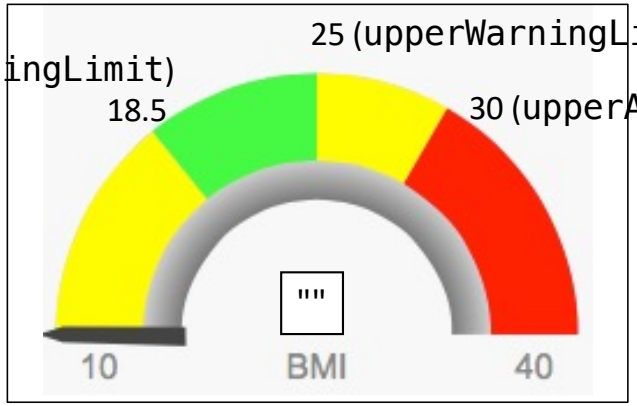
Color	Range
	Underweight
	Normal
	Overweight
	Obese

BMI: new GaugeSVG()

```
<!DOCTYPE html>
<html>
<head>
<title>Javascript/SVG BMI</title>
<style>
div { display: inline-block }
table { border: 1px solid black; border-collapse: collapse }
td, th { border: 1px solid black; padding: 5px }
.color { width: 30px; height: 30px }
</style>
<script src="gaugeSVG.js"></script>
<script>
var gauge;
window.onload = function(){
    gauge = new GaugeSVG({id: "gauge-div", value: 10,
                min: 10, max: 40,
                label: "BMI",
                lowerWarningLimit: 18.5, upperWarningLimit: 25,
                warningLowerRangeColor: "#eeee00",
                warningUpperRangeColor: "#ff8800",
                actionRangeColor: "#ff0000",
                upperActionLimit: 30, lowerActionLimit: -1});
    gauge.gaugeVAL.childNodes[0].textContent = ""; // don't display initially
};
```

BMI: new GaugeSVG()

```
<script src="gaugeSVG.js"></script>
<script>
var gauge;
window.onload = function(){
    gauge = new GaugeSVG({id: "gauge-div", value: 10,
                          min: 10, max: 40,
                          label: "BMI",
                          lowerWarningLimit: 18.5, upperWarningLimit: 25,
                          warningLowerRangeColor: "#eeee00",
                          warningUpperRangeColor: "#ff8800",
                          actionRangeColor: "#ff0000",
                          upperActionLimit: 30, lowerActionLimit: -1});
    gauge.gaugeVAL.childNodes[0].textContent = ""; // don't display initially
};
```



To set the value:
`gauge.refresh(bmi, true);`

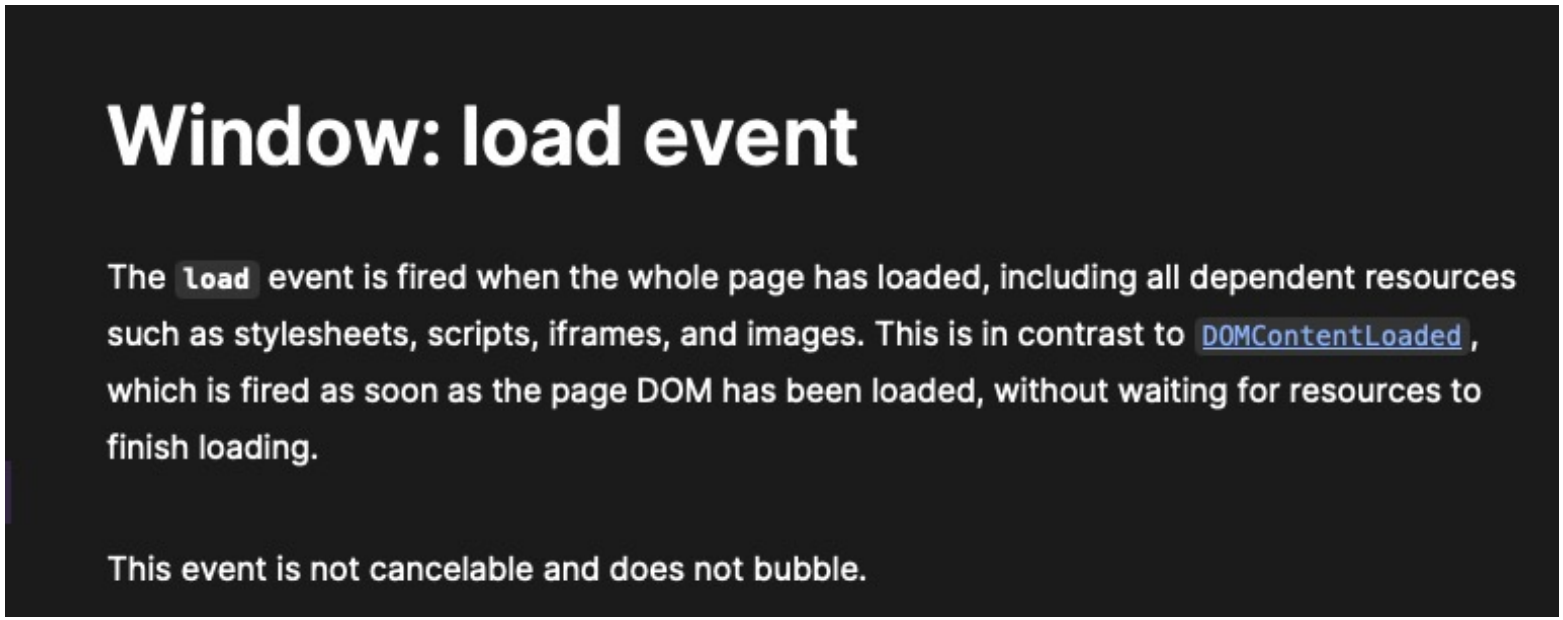
see animation?
`true|false`

bmi-gauge.html

```
25 function validNumber(x) {
26     return /^[0-9\.]+$/.test(x);
27 //    return ((x | 0) > 0 && x % 1 == 0); // convert to 32-bit integer and no fractional part
28 }
29 function computeBMI(e) {
30     var weight = e.form.weight.value;
31     var height = e.form.height.value;
32     var o = document.getElementById("output");
33     o.innerHTML = "";
34 
35     if (validNumber(weight) && validNumber(height)) {
36     var scale = e.form.units[0].checked ? 10000 : 703; // kg-cm
37     var bmi = weight * scale / (height *height);
38     var range;
39     if (bmi < 18.5) {
40         range = "underweight"
41     } else if (bmi < 25) {
42         range = "normal"
43     } else if (bmi < 30) {
44         range = "overweight"
45     } else {
46         range = "obese"
47     }
48     gauge.refresh(bmi.toFixed(2), true)
49     } else {
50     o.innerHTML = "Error: height and weight must be positive numbers"
51     }
52 }
```

window.onload()

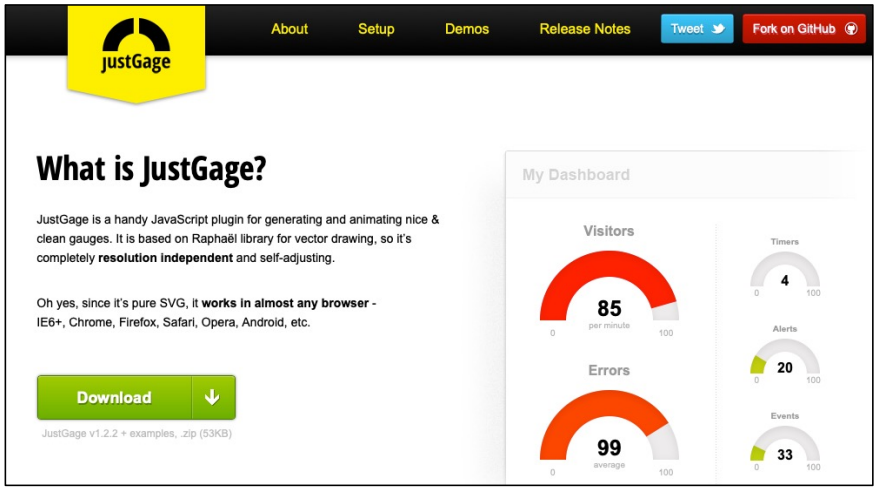
- https://developer.mozilla.org/en-US/docs/Web/API/Window/load_event



JustGage

- gaugeSVG author cites JustGage as inspiration.

https://toorshia.github.io/justgage/



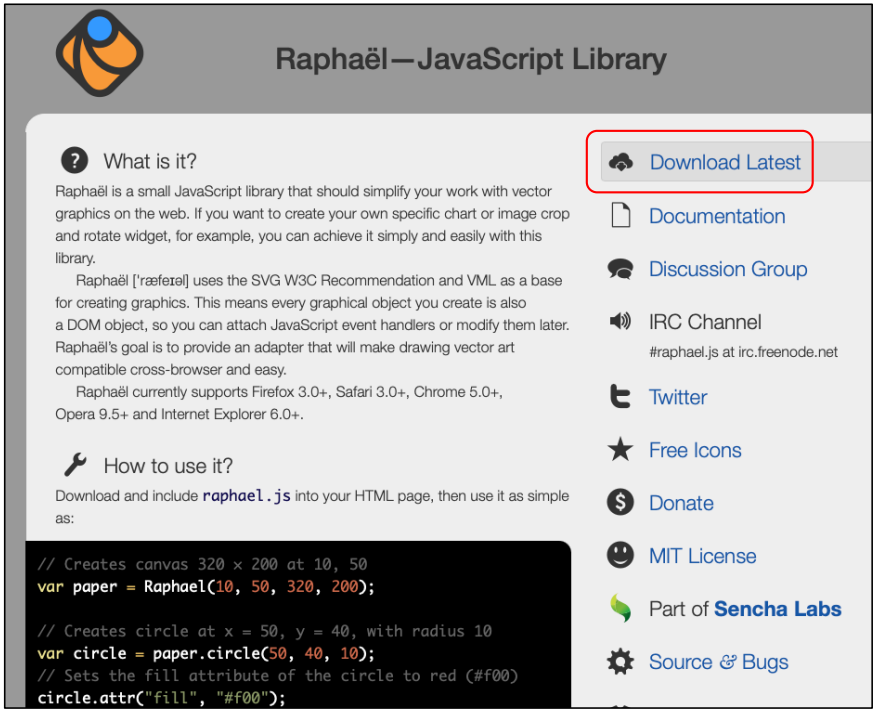

```
<script src="raphael.2.1.0.min.js"></script>
<script src="justgage.1.0.1.min.js"></script>
```

```
<div id="gauge" class="200x160px"></div>
```

```
<script>
  var g = new JustGage({
    id: "gauge",
    value: 67,
    min: 0,
    max: 100,
    title: "Visitors"
  });
</script>
```

Raphaël Javascript graphics library

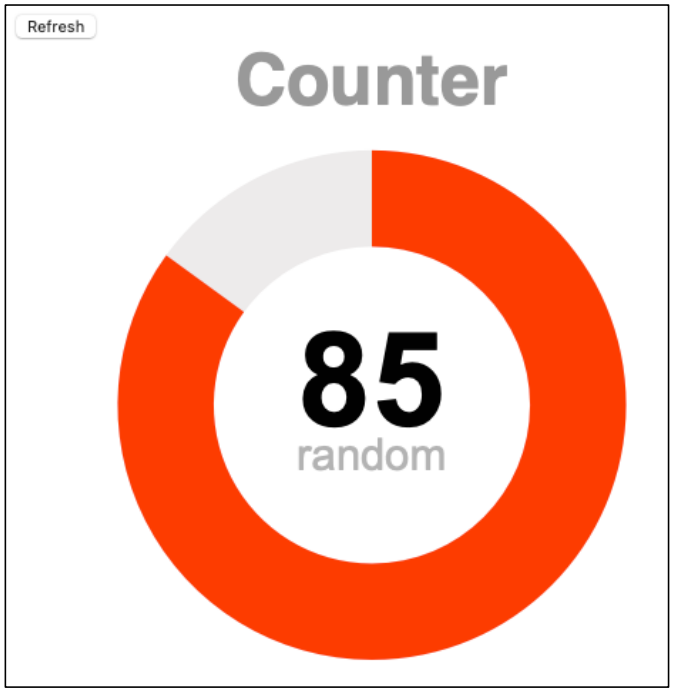
https://dmitrybaranovskiy.github.io/raphael/



JustGage

- copy `raphael-2.1.4.min.js` and `justgage.js` into the current directory where you have your `.html` file
- inside directory examples, there is a simple file `counter.html`
  - I've modified it slightly to refer to the .js files for the same directory
  - and simplified the code

counter.html




```
<script src="raphael-2.1.4.min.js"></script>
<script src="justgage.js"></script>
<script>
  var g1;
document.addEventListener(
  "DOMContentLoaded",
  function(event) {
  g1 = new JustGage({
  id: "g1",
  value: getRandomInt(0,100),
  min: 0,
  max: 100,
  donut: true,
  gaugeWidthScale: 0.9,
  label: "random",
  title: "Counter",
  counter: true,
  hideInnerShadow: true
  });
});
</script>
 </head>
 <body>
  <button onclick="g1.refresh(getRandomInt(0, 100))">Refresh</button>
  <div id="g1" style="width:500px; height:500px"></div>
 </body>
```

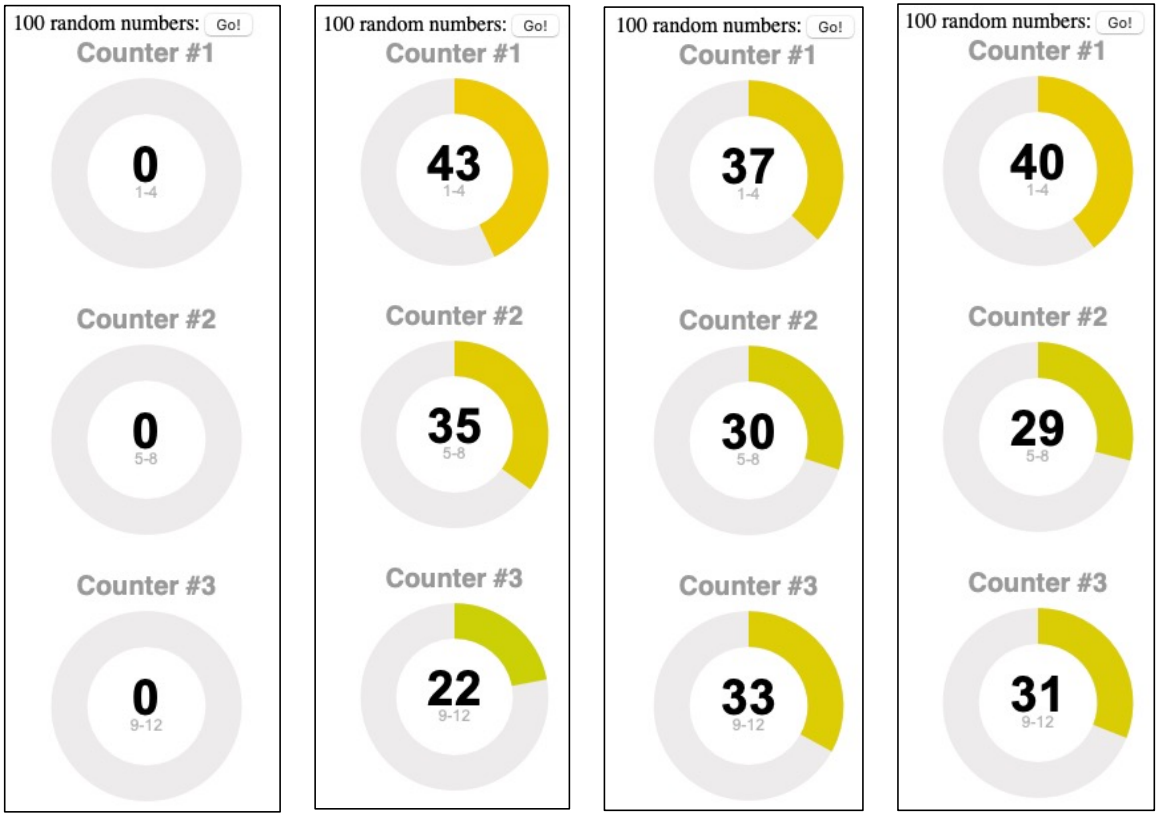
JustGage

- justgage.js

```
1116/** Random integer  */¶
1117function getRandomInt(min, max) {
1118  return Math.floor(Math.random() * (max - min + 1)) + min;¶
1119}¶
```

Homework 7

- Example:

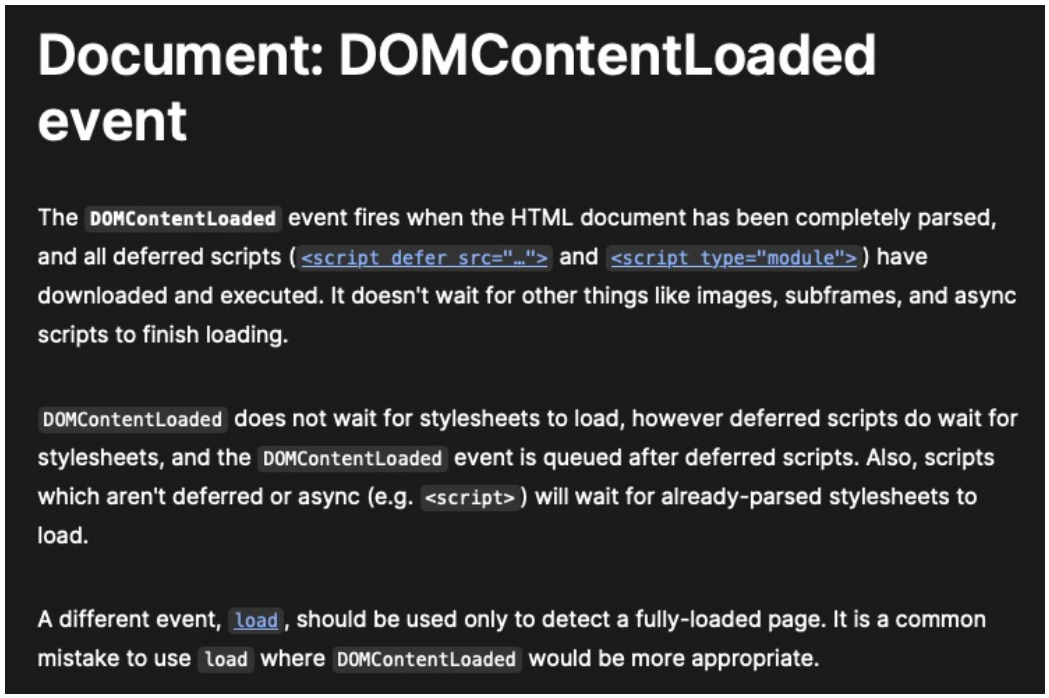


Homework 7

- Use either the gaugeSVG.js or the JustGage widget
- Write a html page that generates 100 random whole numbers between 1,2,…,12 when a button is clicked
  - **Hint**: recall tic-tac-toe code `Math.floor(Math.random()*n)+1`
- Displays 3 gauges that counts the total number for ranges 1-4, 5-8, 9-12
  - Hint: use three totals, one for each range, each a variable initialized to 0.
  - refresh the appropriate gauge when its total is incremented

DOMContentLoaded

- https://developer.mozilla.org/en-US/docs/Web/API/Document/DOMContentLoaded_event



Document: DOMContentLoaded event

The `DOMContentLoaded` event fires when the HTML document has been completely parsed, and all deferred scripts (`<script defer src="…">` and `<script type="module">`) have downloaded and executed. It doesn't wait for other things like images, subframes, and async scripts to finish loading.

`DOMContentLoaded` does not wait for stylesheets to load, however deferred scripts do wait for stylesheets, and the `DOMContentLoaded` event is queued after deferred scripts. Also, scripts which aren't deferred or async (e.g. `<script>`) will wait for already-parsed stylesheets to load.

A different event, `load`, should be used only to detect a fully-loaded page. It is a common mistake to use `load` where `DOMContentLoaded` would be more appropriate.

Homework 7

- Email to me
- Subject: 408/508 Homework 7 *YOUR NAME*
- Due date: Sunday midnight
- A PDF file (showing screenshots)
- Attachment: your .html file so I can test it