

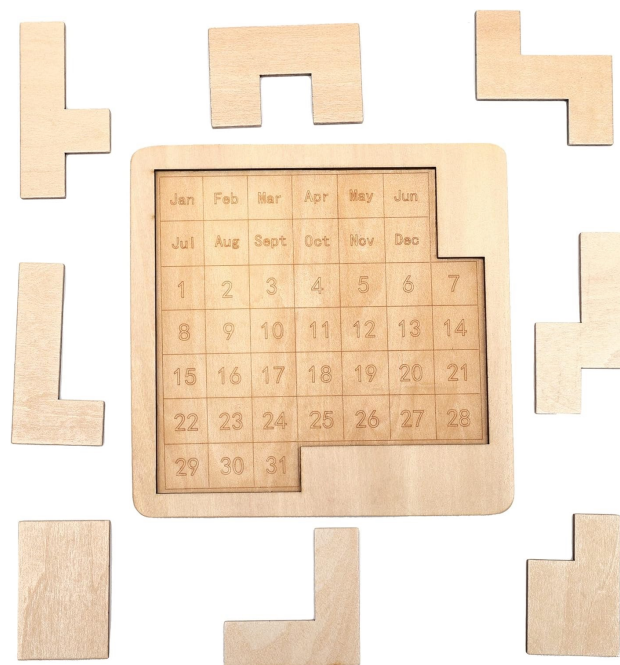
Lecture 16

# 408/508 *Computational Techniques for Linguists*

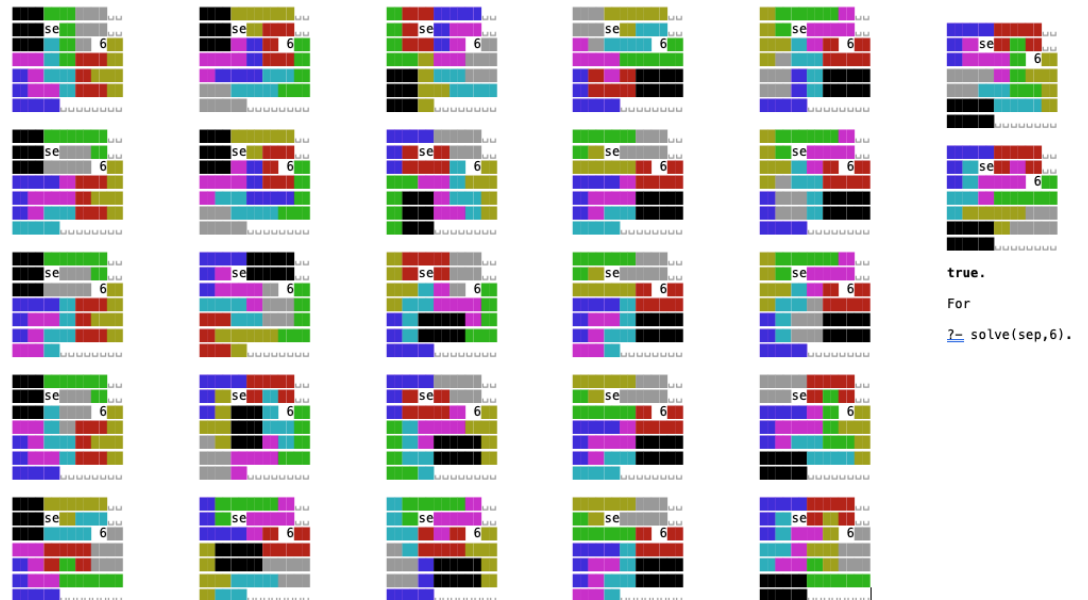
# Today's Topic

- **Reminder:** *Homework 6 (Tic-Tac-Toe) due this Sunday*
- Calendar Puzzle (*some more details*)
- Example of a "real" webpage's structure (*Guardian news article*)
- BMI Gauge animated display (**recap**)
  1. *using <form> elements*
  2. *using <table>*
  3. *using Javascript to compute BMI value*
  4. *using the gaugeSVG.js widget to display the BMI value*
- Gauge for a random number generator

# Calendar Puzzle (*from last time*)

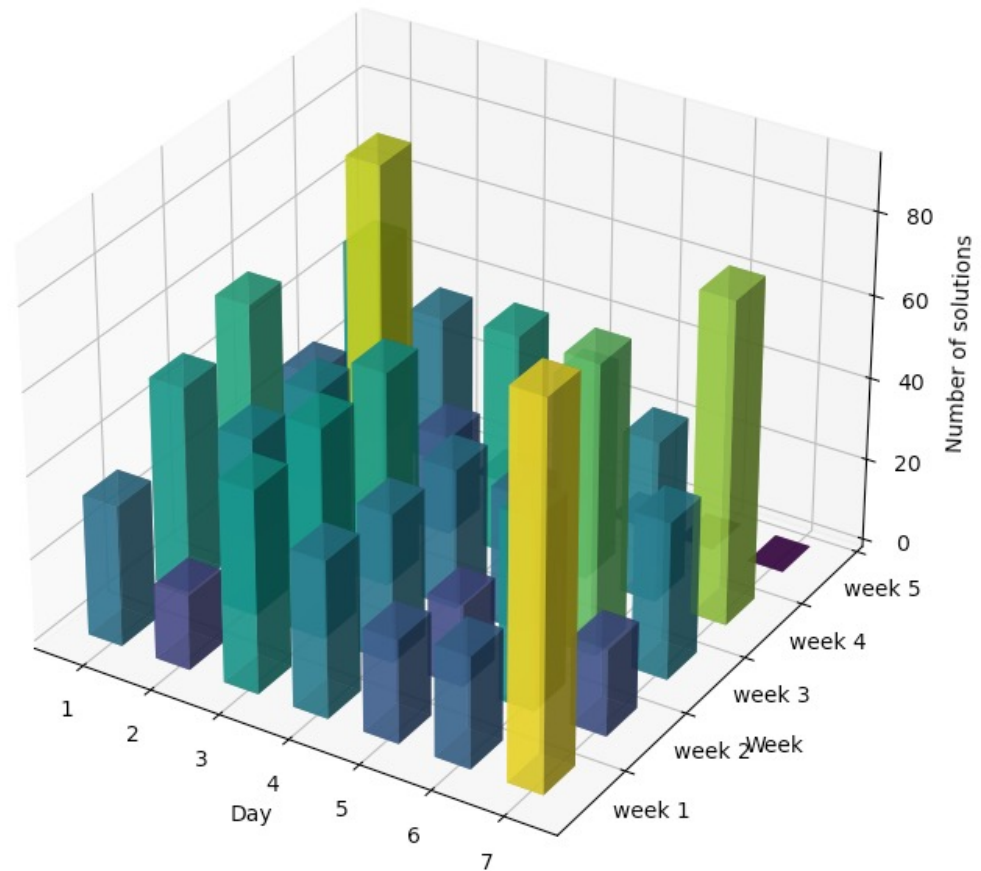


Solutions for Sept 6<sup>th</sup> (*my ASCII graphics code*)

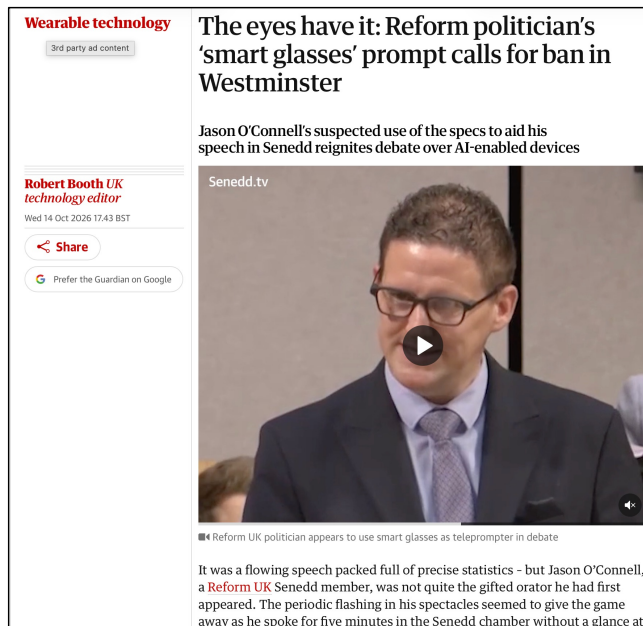


# of solutions  
for September  
1<sup>st</sup> – 31<sup>st</sup>

Calendar Puzzle Solutions: September



# Guardian News Article (on AI glasses)



Guardian News  
Article (on AI  
glasses)



## 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

**Javascript BMI**

Weight (kg/lbs):

Height (cm/ins):

☒ kg-cm ☐ lbs-ins

← "weight"

← "height"

← "units"

**File:** bmi-js.html

# BMI: Javascript function computeBMI ( )

File: bmi-js.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Javascript BMI</title>
5 <script>
6 function computeBMI(e) {
7     var weight = e.form.weight.value;
8     var height = e.form.height.value;
9     var output = document.getElementById("output");
10    var scale = e.form.units[0].checked ? 10000 : 703; // kg-cm
11    var bmi = weight * scale / (height * height);
12    var range;
13    if (bmi < 18.5) {
14        range = "underweight"
15    } else if (bmi < 25) {
16        range = "normal"
17    } else if (bmi < 30) {
18        range = "overweight"
19    } else {
20        range = "obese"
21    }
22    output.innerHTML =
23        "BMI: " + bmi.toFixed(2) +
24        "<br>Range: " + range;
25 }
26 </script>
```

The diagram illustrates the user interface for the BMI calculator and how it interacts with the JavaScript function. The form, titled "Javascript BMI", contains the following elements:

- Weight (kg/lbs):
- Height (cm/ins):
- Units: ☒ kg-cm ☐ lbs-ins
- Click:

Arrows indicate the data flow and function calls:

- From the Weight input to `e.form.weight.value`.
- From the Height input to `e.form.height.value`.
- From the selected radio button to `e.form.units[0].checked`, with a note "0 = 1<sup>st</sup> radio button".
- From the Click button to the `computeBMI(this)` function call.

# BMI: gaugeSVG.js

◀ A Universal Gauge for Your Web Dashboard

 Steffen Ploetz, 16 Dec 2014

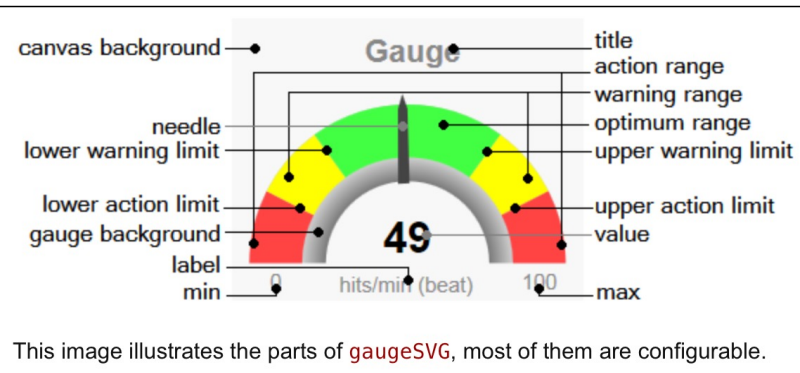
★★★★★ 4.95 (19 votes) Rate this: ★★★★★

JavaScript plugin gaugeSVG to generate widely configurable SVG gauge for a web dashboard

 Download source and sample - 6.6 KB

 Download source and sample - v2.0 - 8.3 KB

 Download source and sample - v2.1 - 8.7 KB



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

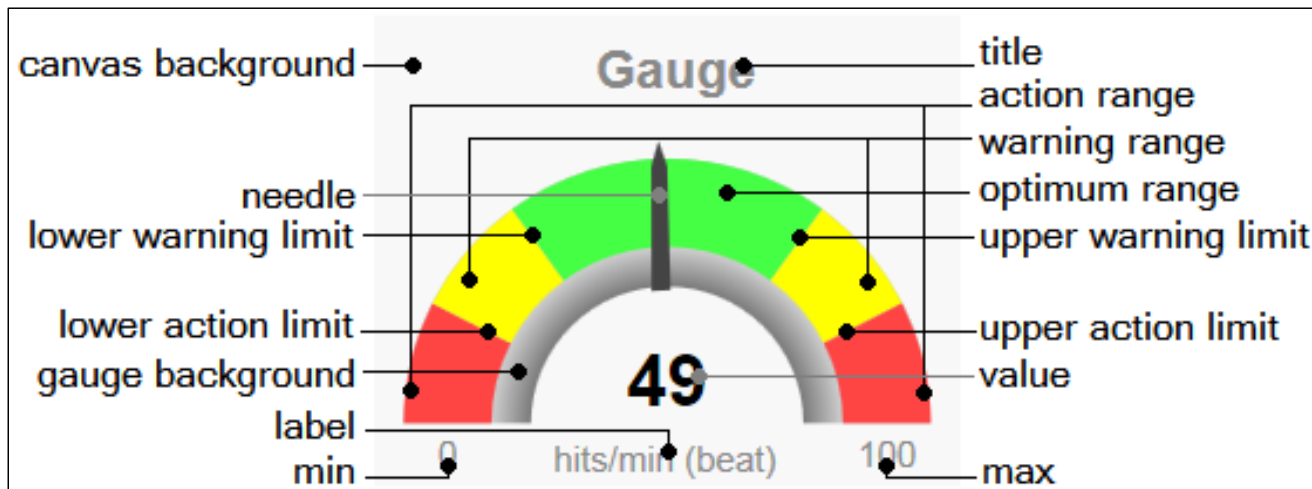
bmi-gauge.html

Download gaugeSVG.js from the course webpage

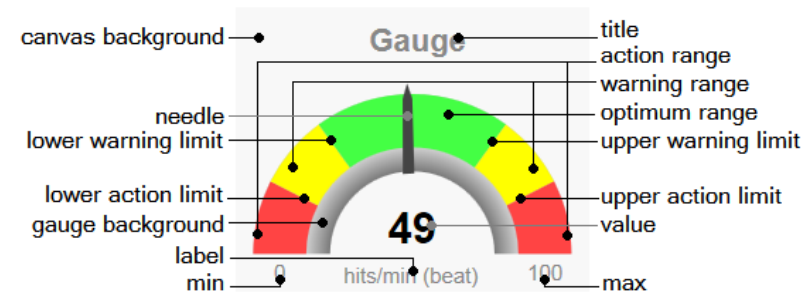
*(I've modified the original code a bit)*

## BMI: gaugeSVG.js

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

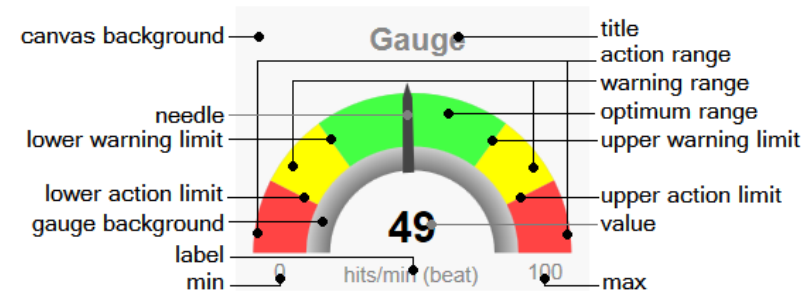


# BMI: gaugeSVG.js



- **title**: [string] The title text that is displayed above the gauge. It can be an empty string to be suppressed. Default is an empty string.
- **titleColor**: [string] The title color. Default is "#888888".
- **value**: [float] The value to display. Default is  $(\text{max} - \text{min}) / 2.0$ . Values below **min** are shown as **min**. Values above **max** are shown as **max**.
- **valueColor**: [string] 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 empty string to be suppressed. Default is empty string. Typically used to display the value's measuring unit.
- **labelColor**: [string] 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**: [boolean] Hide or display the **min** and **max** gauge display range values as text. Default is true.
- **minmaxColor**: [string] The **min** and **max** value's text color. Default is "#888888".
- **canvasBackColor**: [string] 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 from 0.15 to 1.5. Lower values show a smaller arc, higher values show a thicker arc.
- **gaugeBorderColor**: [string] The gauge arc border color. Default is "#cccccc".
- **gaugeBorderWidth**: [float] 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. Default 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. Default is "#000000".
- **gaugeShadowScale**: [float] The width of the gauge arc's shadow. Default is 1.0. Meaningful values range from 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  $(\text{max} - \text{min}) * 0.15 + \text{min}$ .
- **lowerWarningLimit**: [float] The lower warning limit or a negative value, if not desired. Default is  $(\text{max} - \text{min}) * 0.30 + \text{min}$ .
- **upperWarningLimit**: [float] The upper warning limit or a negative value, if not desired. Default is  $(\text{max} - \text{min}) * 0.70 + \text{min}$ .
- **upperActionLimit**: [float] The upper action limit or a negative value, if not desired. Default is  $(\text{max} - \text{min}) * 0.85 + \text{min}$ .
- **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>

```
55 <body>
56 <h1>Javascript SVG/BMI</h1>
57 <div>
58 <form action="" method="GET">
59 Weight (kg/lbs): <input type="text" name="weight" size=5>
60 <br>
61 Height (cm/ins): <input type="text" name="height" size=5>
62 <br>
63 <input type="radio" name="units" value="kg" checked>kg-cm
64 <input type="radio" name="units" value="lbs">lbs-ins
65 <br>
66 <input type="button" name="button" Value="Click" onClick="computeBMI(this)">
67 </form>
68 </div>
69 <div style="width:100px;color:red" id="output"></div>
70 <div id="gauge-div" style="width: 250px; height: 200px"></div>
71 <div>
72 <table>
73 <tr><th>Color</th><th>Range</th></tr>
74 <tr>
75 <td class="color" style="background-color: #eeee00"></td><td>Underweight</td>
76 </tr>
77 <tr>
78 <td class="color" style="background-color: #44ff44"></td><td>Normal</td>
79 </tr>
80 <tr>
81 <td class="color" style="background-color: #ff8800"></td><td>Overweight</td>
82 </tr>
83 <tr>
84 <td class="color" style="background-color: #ff0000"></td><td>Obese</td>
85 </tr>
86 </div>
87 </body>
```

<table>

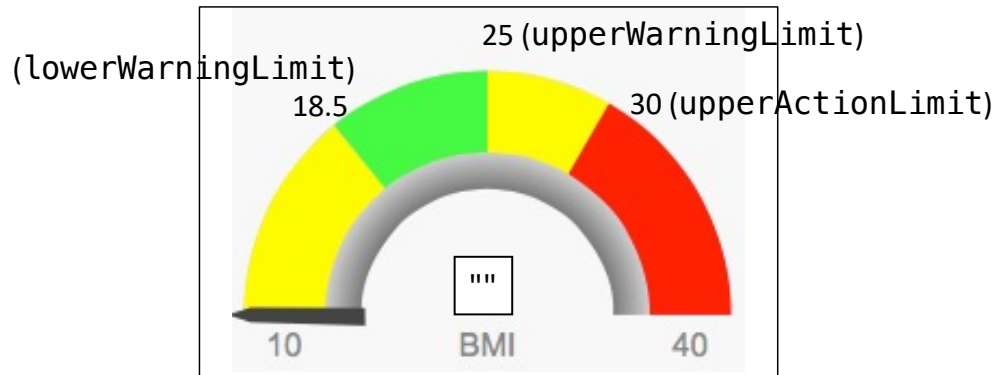
| Color | Range       |
|-------|-------------|
|       | Underweight |
|       | Normal      |
|       | Overweight  |
|       | Obese       |

## BMI: new GaugeSVG()

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Javascript/SVG BMI</title>
5 <style>
6 div { display: inline-block }
7 table { border: 1px solid black; border-collapse: collapse }
8 td, th { border: 1px solid black; padding: 5px }
9 .color { width: 30px; height: 30px }
10 </style>
11 <script src="gaugeSVG.js"></script>
12 <script>
13 var gauge;
14 window.onload = function(){
15     gauge = new GaugeSVG({id: "gauge-div", value: 10,
16         min: 10, max: 40,
17         label: "BMI",
18         lowerWarningLimit: 18.5, upperWarningLimit: 25,
19         warningLowerRangeColor: "#eeee00",
20         warningUpperRangeColor: "#ff8800",
21         actionRangeColor: "#ff0000",
22         upperActionLimit: 30, lowerActionLimit: -1});
23     gauge.gaugeVAL.childNodes[0].textContent = ""; // don't display initially
24 };
```

# 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](https://developer.mozilla.org/en-US/docs/Web/API/Window/load_event)

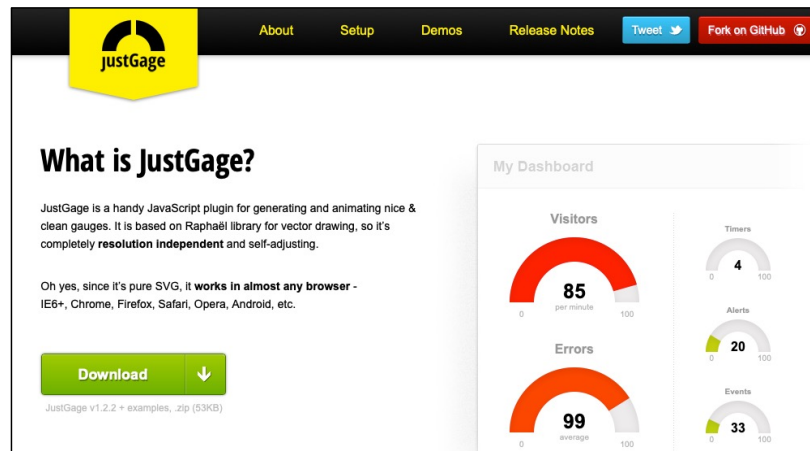
## Window: load event

The **load** event is fired when the whole page has loaded, including all dependent resources such as stylesheets, scripts, iframes, and images. This is in contrast to [DOMContentLoaded](#), which is fired as soon as the page DOM has been loaded, without waiting for resources to finish loading.

This event is not cancelable and does not bubble.

# 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/>

|                        |               |                          |                       |
|------------------------|---------------|--------------------------|-----------------------|
| raphael-2.3.0          | --            | Folder                   | Today, 1:30 PM        |
| yarn.lock              | 118 KB        | Document                 | Today, 1:30 PM        |
| webpack.config.js      | 715 bytes     | JavaScript script        | Today, 1:30 PM        |
| raphael.no-deps.min.js | 90 KB         | JavaScript script        | Today, 1:30 PM        |
| raphael.no-deps.js     | 295 KB        | JavaScript script        | Today, 1:30 PM        |
| raphael.min.js         | 93 KB         | JavaScript script        | Today, 1:30 PM        |
| <b>raphael.js</b>      | <b>311 KB</b> | <b>JavaScript script</b> | <b>Today, 1:30 PM</b> |
| package.json           | 1 KB          | JSON Document            | Today, 1:30 PM        |
| license.txt            | 1 KB          | Plain Text               | Today, 1:30 PM        |
| history.md             | 7 KB          | Markdo...cument          | Today, 1:30 PM        |
| dev                    | --            | Folder                   | Today, 1:30 PM        |
| bower.json             | 500 bytes     | JSON Document            | Today, 1:30 PM        |
| README.md              | 3 KB          | Markdo...cument          | Today, 1:30 PM        |
| Gruntfile.js           | 2 KB          | JavaScript script        | Today, 1:30 PM        |
| CONTRIBUTING.md        | 1 KB          | Markdo...cument          | Today, 1:30 PM        |



## Raphaël — JavaScript Library

**What is it?**

Raphaël is a small JavaScript library that should simplify your work with vector graphics on the web. If you want to create your own specific chart or image crop and rotate widget, for example, you can achieve it simply and easily with this library.

Raphaël ['ræfæɪəl] uses the SVG W3C Recommendation and VML as a base for creating graphics. This means every graphical object you create is also a DOM object, so you can attach JavaScript event handlers or modify them later. Raphaël's goal is to provide an adapter that will make drawing vector art compatible cross-browser and easy.

Raphaël currently supports Firefox 3.0+, Safari 3.0+, Chrome 5.0+, Opera 9.5+ and Internet Explorer 6.0+.

**How to use it?**

Download and include **raphael.js** into your HTML page, then use it as simple as:

```
// Creates canvas 320 x 200 at 10, 50
var paper = Raphael(10, 50, 320, 200);

// Creates circle at x = 50, y = 40, with radius 10
var circle = paper.circle(50, 40, 10);
// Sets the fill attribute of the circle to red (#f00)
circle.attr("fill", "#f00");
```

[Download Latest](#)

[Documentation](#)

[Discussion Group](#)

[IRC Channel](#)  
#raphael.js at irc.freenode.net

[Twitter](#)

[Free Icons](#)

[Donate](#)

[MIT License](#)

Part of [Sencha Labs](#)

[Source & Bugs](#)

# JustGage

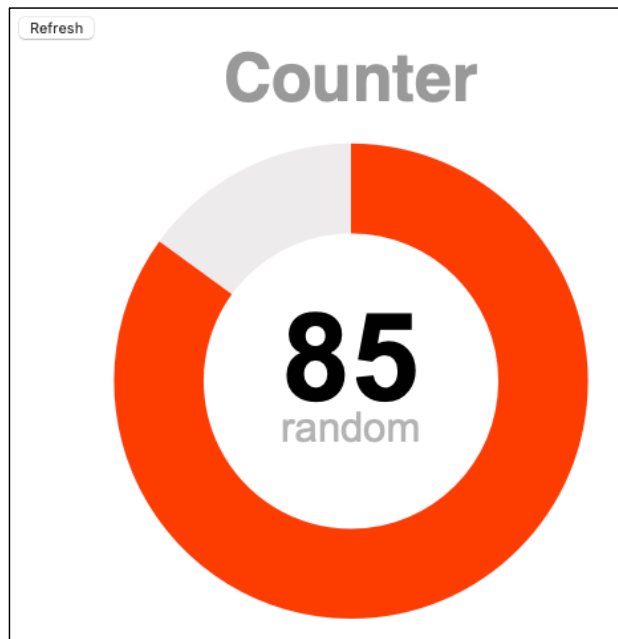
|   |                      |       |                   |                |
|---|----------------------|-------|-------------------|----------------|
| ▼ | justgage-1.2.2       | --    | Folder            | Today, 1:27 PM |
|   | raphael-2.1.4.min.js | 93 KB | JavaScript script | Today, 1:27 PM |
|   | justgage.js          | 38 KB | JavaScript script | Today, 1:27 PM |
| > | examples             | --    | Folder            | Today, 1:27 PM |

- 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

on the course website:

*let's run it a few times!*



```
7 <script src="raphael-2.1.4.min.js"></script>
8 <script src="justgage.js"></script>
9 <script>
10   var g1;
11   document.addEventListener(
12     "DOMContentLoaded",
13     function(event) {
14       g1 = new JustGage({
15         id: "g1",
16         value: getRandomInt(0,100),
17         min: 0,
18         max: 100,
19         donut: true,
20         gaugeWidthScale: 0.9,
21         label: "random",
22         title: "Counter",
23         counter: true,
24         hideInnerShadow: true
25       });
26     });
27 </script>
28 </head>
29 <body>
30   <button onclick="g1.refresh(getRandomInt(0, 100))">Refresh</button>
31   <div id="g1" style="width:500px; height:500px"></div>
32 </body>
```

an event to be signaled (by the browser)

# JustGage

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

- `getRandomInt()`:
  - `Math.random()` returns value [0,1) (*float*)
  - `Max - Min + 1` gives range, e.g. 50-20+1=31, so [0,31)
  - `Math.floor()` gives 0..30 (*whole number*)
  - `+ Min` offset, so 20..50

# DOMContentLoaded

- [https://developer.mozilla.org/en-US/docs/Web/API/Document/DOMContentLoaded\\_event](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.