AcroTEX.Net

The bargraph-js Package

D. P. Story

Prepared: May 20, 2019

www.acrotex.net
Version 0.7,2019/04/07

Table of Contents

1. Introduction

This package can create bar graphs that represent categorical or quantitative data; special techniques can be use to create bargraph representations of a discrete probability distribution.

Bar graphs created by this package are *dynamic*; that is, the user can enter data and the bar graph changes to reflect this new data. This magic is performed using Acrobat forms and Acrobat JavaScript. The individual "bars" of a bar graph are form fields, JavaScript is used to change their bounding rectangles.

The document requires, for the dynamic bar graph features to work, the user to be viewing the document in AR (or the Acrobat application (AA), of course). JavaScript even within AR is not fully supported on all devices, as a result, AR or AA on a desktop or laptop is required.

Before we start describing this package, we present a simple example. Enter nonnegative numbers in the text fields to the right.

This example appears in `examples/bgjs-basic1.tex`

Auto:

Truck:

SUV:

Van:

Optimize

Reset vertical bar graph: Reset

Motivation. The motivation for the package comes from the request of an AcroTEX user. He wanted a bar graph that summarizes the results of a questionnaire. Each of some eighty question is classified into one of eight categories. A bar graph is then generated based on the responses.

2. Requirements and options of the package

The bargraph-js package uses the eforms package dated 2019/03/16 or later. Use the package in the usual way,

```
\documentclass{article}
\usepackage{bargraph-js}
```

There are no package options. If you want to use options for the eforms package, load it earlier than bargraph-js:

```
\documentclass{article}
\usepackage[usealtadobe,setcorder]{eforms}
\usepackage{bargraph-js}
```

Part I
Bar graph with a fixed number of bars

In this part of the manual, the construction of bar graphs is discussed with a known number of bars.

3. The bar graph environments

The package defines two environments, these are `bargraphenv` and `bargraph`; the latter environment is placed within the former one.

```
\begin{bargraphenv}[⟨bgenv-options⟩]{⟨bgenv-name⟩}
   \presetsbarfor commands, optional
   \begin{bargraph}[⟨bg-options⟩]{⟨bg-name⟩}
      \barfor{⟨bar-name_1⟩}...\barfor{⟨bar-name_n⟩}
   \end{bargraph}
   additional bargraph environments, optional
\end{bargraphenv}
```

The ⟨*bgenv-name*⟩ argument must be unique throughout the document, the other names (⟨*bg-name*⟩ and ⟨*bar-name*⟩) may be reused in other `bargraphenv` environments. Within a `bargraphenv` environment, however, the ⟨*bg-name*⟩ must be unique, that is, not repeated. The `\presetsbarfor` commands are a way of changing the appearances of the individual bars. Within the `bargraph` environment, only `\barfor` or `\cmd` commands are recognized (the latter is not shown). The `\barfor` command gives a name to the bar so it can be referenced and manipulated. More than one `bargraph` environment within a `bargraphenv` environment is supported.

Only `\barfor` and `\cmd` are permitted

Before describing the individual components, we present the verbatim listing of the example given above.

```
\fbox{%
   \begin{bargraphenv}[width=(23bp*4),height=2in,o=vert]{vehiclesV}
      \presetsbarfor{vBar}{auto}{\BG{red}}
      \presetsbarfor{vBar}{truck}{\BG{green}}
      \presetsbarfor{vBar}{suv}{\BG{yellow}}
      \presetsbarfor{vBar}{van}{\BG{magenta}}
      \begin{bargraph}[nbars=4,gap=3]{vBar}
         \barfor{auto}\barfor{truck}\barfor{suv}\barfor{van}
      \end{bargraph}
   \end{bargraphenv}
}
```

The key-values of `bargraphenv` set the dimensions of the ‘display window’ (here, we use `\fbox` to enclose it). The option `o=vert` states these bar graphs will be vertically oriented. The next four lines are `\presetsbarfor` commands that pass appearances on to the bars themselves; this accounts for the coloring of the individual bars demonstrated in the interactive example above. The optional key-values for the `bargraph` environment are `nbars=4` (the number of bars in this bar graph, required); and `gap=3` (distance between bars in big points)

Description of `bargraphenv` arguments. `bargraphenv` is a `minipage` environment, in which the bars are drawn and displayed. The following are the key-values recognized within the optional argument of `bargraphenv`.

`width=`⟨*length*⟩ The width of the environment.

`height=`⟨*length*⟩ The height of the environment.

`o=`⟨`horiz|vert`⟩ Declares the orientation of the bar graph contained within the environment: `o=horiz` specifies the bar graphs are to be horizontal; `o=vert` sets the bar graphs to be vertical. The default is vertical.

`origin=`⟨`0|.5`⟩ The position of the horizontal axis (when `o=vert`) or the vertical axis (when `o=horiz`). A value of `origin=0` is the default.

`showaxis`⟨`true|false`⟩ If `showaxis=true` (or just `showaxis`, the horizontal rule (if `o=vert`) or a vertical rule (if `o=horiz`) is drawn. The default is no axis is drawn.

Description of `bargraph` arguments. The `bargraph` environment is placed within the `bargraphenv` environment; it has a name (⟨*bg-name*⟩) and must contain, within its body, only the `\barfor` and `\cmd` commands. The optional arguments (⟨*bg-options*⟩) are used to describe the bars that is contained in the environment.

`nbars=`⟨*num*⟩ (Required) The number of bars to appear in this `bargraph` environment. The value ⟨*num*⟩ must be a positive integer. There is no enforcement
\nbars of this restriction; just disaster. The value specified by `nbars` is saved in the command `\nbars`; `\nbars` is globally defined and is overwritten then when the next `bargraph` environment is expanded. The default for `nbars` is zero (0); consequently, this is a required key.

`gap=`⟨*num*⟩ (Optional) The amount space between bars; ⟨*num*⟩ is a number of reasonable size; the number is dimensionless, but is interpreted as a big point. No enforcement, its your document. The gap can be access through the command
\bargap `\bargap`; if `gap=3`, then `\bargap` expands to `3bp`. The definition of `\bargap` is global, it is overwritten when the next `bargraph` environment is expanded. The default is `gap=0`.

`bardimen=`⟨*num*⟩ (Optional) A dimensionless positive number interpreted as the width (in the case of `o=vert`) and height (when `o=horiz`) in big points. The
\bardimen default is 20. The value of the `bardimen` key is saved in the `\bardimen` command as ⟨*num*⟩`bp`; this is global definition and is overwritten when the next `bargraph` environment is expanded.

In the example given earlier, the `bargraph` environment was,

```
\begin{bargraph}[nbars=4,gap=3]{vBar}
\barfor{auto}\barfor{truck}\barfor{suv}\barfor{van}
\end{bargraph}
```

We declare this environment to have a name of `vBar` (⟨*bg-name*⟩). It contains four bars, each with a gap of 3 (3bp). The environment contains four `\barfor` commands named `auto`, `truck`, `suv`, and `van` (these are the ⟨*bar-name*⟩s).

Description of `\presetsbarfor`. The `\barfor` command creates a (rectangular) push button used to represent a single 'bar' in a bar graph. The `\presetsbarfor` commands are the chosen way to pass (form field) options to the push buttons created by `\barfor`.

```
\presetsbarfor{⟨bg-name⟩}{⟨bar-name⟩}{⟨KV-pairs⟩}
```

To uniquely identify which form field these key-value pairs (⟨*KV-pairs*⟩) apply, we supply the name of the `bargraph` environment (⟨*bg-name*⟩) the bar is located in, and the name of the particular bar (⟨*bar-name*⟩). The ⟨*KV-pairs*⟩ argument consists of any key-value pairs supported by the `eforms` package for form fields.

```
\presetsbarfor{vBar}{auto}{\BG{red}}
\presetsbarfor{vBar}{truck}{\BG{green}}
\presetsbarfor{vBar}{suv}{\BG{yellow}}
\presetsbarfor{vBar}{van}{\BG{magenta}}
\begin{bargraph}[nbars=4,gap=3]{vBar}
   \barfor{auto}\barfor{truck}\barfor{suv}\barfor{van}
\end{bargraph}
\end{bargraphenv}}
```

These commands reference the bars within the `vBar` `bargraph` environment. For example, for the bar named `auto`, we assign a background (or fill) color of red, and so on.

The `\barfor` and `\cmd` commands. The body of `bargraph` consists solely of `\barfor` and `\cmd` commands (`\cmd` is a local command, defined for the `bargraph` environment). The syntax for these are `\barfor{`⟨*bar-name*⟩`}` and `\cmd{`⟨*markup*⟩`}`, where ⟨*markup*⟩ refers to any LaTeX markup. The author created `\cmd` to add typeset labeling to the bar graph, but there are other applications. Illustrations of `\cmd` are found in Section 5 on drawing multiple bar graphs and Section 6 on labeling.

4. Inputting data for a bar graph

Now, given you have set up a `bargraphenv` environment with its various innards, how do you input data so the bar graph responds? There are several schemes.

4.1. Individual input via `\inputFor`

The method illustrated in the example on page 3 is for the user to input individual data using the package provided command `\inputFor`.

```
\renewcommand\presetinputfor[2]{⟨eforms-actions⟩}
\inputFor[⟨KV-pairs⟩]{⟨bgenv-name⟩}{⟨bg-name⟩}{⟨bar-name⟩}{⟨width⟩}{⟨height⟩}
```

The command `\inputFor` doesn't have to reside within a `bargraphenv` environment, so a full path is needed to characterize which of the bars this text field references. Use ⟨*KV-pairs*⟩ to change the appearance of the field. One such example from page 3 is,

```
\inputFor{vehiclesV}{vBar}{auto}{.5in}{13bp}
```

You need one `\inputFor` command for each of the bars in the targeted `bargraphenv` environment.

Use `\presetinputfor` to set any actions you require of your `\inputFor` commands; any re-definition of `\presetinputfor` should occur prior to the targeted `\inputFor` commands. The package initial definition is,

```
\newcommand{\presetinputfor}[2]{%
  \AAkeystroke{AFNumber_Keystroke(0, 0, 0, 0, "", true);}
  \AAformat{AFNumber_Format(0, 0, 0, 0, "", true);}
  \AAvalidate{AFRange_Validate(true, 0, false, 0);}% remove to allow all numbers
}
```

This requires the entries in the `\inputFor` commands to be nonnegative numbers. The two arguments of `\presetinputfor` are normally not used, but for the record `#1` is ⟨*bgenv-name*⟩ and `#2` is ⟨*bg-name*⟩`.`⟨*bar-name*⟩.

4.2. Providing comma-delimited data

Another way to enter data is through a comma-delimited list of numbers.

This example appears in `examples/bgjs-comma1.tex`

Symmetrical Skew left Skew right

Optimize 2 Rescale % Can perform simple arithmetic operations in text field

Reset

Enter comma-delimited nonnegative numbers into the text field above, or press any of the three buttons Symmetrical, Skew left, or Skew right to populate the bar graph with comma-delimited data. Press the Optimize button to automatically re-scale the graph; manually re-scale the graph by entering a scale factor in the text field, then press the Rescale button. Clear all fields by pressing the Reset button. These buttons are discussed in detail in a later section.

5. Multiple bar graphs within an `bargraphenv` environment

More than one `bargraph` environment can appear within a `bargraphenv`, as illustrated in the next two examples.

This example appears in `examples/bgjs-comma2.tex`

class1: Class 1

class2: Class 2

Optimize Reset

A partial verbatim listing is presented below:

```
\fbox{\begin{bargraphenv}[width=23bp*10,height=1.4in,o=vert]{math1}
...
\begin{bargraph}[nbars=5,gap=3]{class1}
\barfor{A}\barfor{B}\barfor{C}\barfor{D}\barfor{F}
\end{bargraph}
\begin{bargraph}[nbars=5,gap=3]{class2}\cmd{\hs{\bargap}}
\barfor{A}\barfor{B}\barfor{C}\barfor{D}\barfor{F}
\end{bargraph}
\end{bargraphenv}}
...
class1: \textField[\TU{Enter five natural numbers separated by commas}
  \AAvalidate{%
  var str=event.value;\r
  \populateCommaData("math1","class1",str,validateArrayNonNegNums)}
  ]{txtclass3}{2in}{13bp}\cgBdry[.5em]
  \pushButton[\CA{Class 1}\AAmouseup{%
    var str="12,15,23,10,15";\r
    \populateCommaData("math1","class1",str);
  }]{math1class1}{}{13bp}
```

To populate a bar graph with comma-delimited data, use the `\populateCommData` command, a LaTeX helper command for an underlying JavaScript function. Its arguments are

`\populateCommaData("⟨bgenv-name⟩","⟨bg-name⟩",⟨str⟩[,⟨validate⟩])`

Where, ⟨*str*⟩ is a comma-delimited string of numbers; the optional ⟨*validate*⟩ argument is a name of a validation function. In the example above, the package defined `validateArrayNonNegNums` is used.

Important One more comment on the listing above: Observe the markup in bold font; the gap is not applied to the last bar for a `bargraph` environment, so to get same gap as we move from the `class1` bar graph to the `class2` bar graph, we insert `\cmd{\hs{\bargap}}` the desired gap. Read the comments following the next example for more detailed explanation of the (local) commands `\cmd` and `\hs`.

You can adjust the positions of the bar graph to have a more side-by-side comparison.

This example appears in `examples/bgjs-comma2.tex`

class1: ☐ Class 1 Test Optimize Reset

class2: ☐ Class 2

A partial listing is given next:

```
\begin{bargraph}[nbars=5,gap=13]{class1}
\barfor{A}\barfor{B}\barfor{C}\barfor{D}\barfor{F}
\end{bargraph}
\begin{bargraph}[nbars=5,gap=13]{class2}
\cmd{\hs{-33bp*5+10bp}}
\barfor{A}\barfor{B}\barfor{C}\barfor{D}\barfor{F}
\end{bargraph}
\end{bargraphenv}
```

`\cmd` *`\hs`* *`\vs`* This listing introduces two new commands, shown in bold font: `\cmd` is a generic command for placing LaTeX content into the bar graph; `\hs` (and `\vs`) are short-hand commands for entering a horizontal skip (`\hs`) or a vertical skip (`\vs`) into the page. These two should be used only within the `\cmd` command argument. In the above listing, we shift the contents of the second `bargraph` environment to the left to appear as shifted over 10bp from the first `bargraph`.

The markup shown in bold font horizontally shifts the second bar graph `33bp*5` to the left (`33bp` = `\bardimen` + `\bargap`) and 5 = `\nbars`. Since `\bardimen` = `20bp` (the default value), we shift `10bp` (`\bardimen`/2) to the right to get the two first bars offset by half their width.

6. Labeling the bars

Now we come to the difficult topic of labeling (or captioning) the bars. Because of the dynamic nature of these bar graphs, there can be no scaling on the axis; however, if you rollover any of the bars, you will see labeling that provides some information. Throughout this document, so far, the "default" labeling is applied, as exhibited by the red bar graph below; there is a richer scheme that is possible when the bars have a `\TU` (tool tip) markup. When you provide an appropriately coded tool tip you can build a much more meaningful labeling of the bars, as illustrated by the blue bar graph below.

class1: ☐ Class 1 Optimize Reset
class2: ☐ Class 2

A partial verbatim listing is presented below:

```
\fbox{\begin{bargraphenv}[width=23bp*10,height=1.4in,%
  o=vert]{math3}
\presetsbarfor{class1}{A}{\BG{red}}          % The \TU key is not provided
...
\presetsbarfor{class2}{A}{\BG{blue}
  \TU{Class2: @v@ students received an 'A'}} % The \TU key is provided
...
\begin{bargraph}[nbars=5,gap=3]{class1}
\barfor{A}\barfor{B}\barfor{C}\barfor{D}\barfor{F}
\end{bargraph}
\begin{bargraph}[nbars=5,gap=3]{class2}\cmd{\hs{\bargap}}
\barfor{A}\barfor{B}\barfor{C}\barfor{D}\barfor{F}
\end{bargraph}
\end{bargraphenv}}
```

Shown in bold font above is the `\TU` key for the first bar of `class2`. The value of the `\TU` key contains the string '`@v@`'; this three-character string is replaced by the actual value of the bar. Concepts and methods are outlined in the next section.

6.1. Methods for labeling bars

Fundamentally, there are two methods for labeling bars: (1) using the tooltip feature of PDF forms; and (2) non-tooltip methods. Within each of these two methods, several techniques have been developed, which are explained in the next two sections.

• Tooltip techniques

There are two commands, `\barLabelsTUs` and `\barLabelsNoTU`, to declare how the PDF tooltips are formed. They handle the cases of `\TU` key is provided and the `\TU` key is not provided. We look at each of these in turn.

The `\TU` key is provided. The `\barLabelsTU` takes either a JavaScript function name or a JavaScript string argument.

```
\barLabelsTU{⟨JS-function⟩|⟨JS-string⟩}
```

`\barLabelsTU` *applies to all bars with a* `\TU` *key*. The `\barLabelsTU` *can be used within the body of the document* to change the method of assigning labels to bars. The tooltip can comprise of the variables `@env@`, `@barname@`, `@bar@`, and `@v@` to compose the string; these variables are replaces with their respective values ⟨*bgenv-name*⟩, ⟨*bg-name*⟩, ⟨*bar-name*⟩ and ⟨*value*⟩ when the bar is populated with ⟨*value*⟩.

Description of argument types

⟨***JS-function***⟩ Such a JavaScript function takes two arguments `fld` and `v`; the latter is the current value of the bar, the former is the field name of the bar. Such a function should return a string that will be displayed as the tooltip of the bar. The default is `customBarLabels`,

```
\barLabelsTU{customBarLabels} % applies to all bars with a \TU key
```

which is part of the bargraph-js package. Note that this declaration is equivalent to `\barLabelsTU{}` and `\barLabelsTU{""}`.

⟨***JS-string***⟩ When a string is provided, *the* `\TU` *key is ignored*. Instead a formatted string is used. Use the variables `@env@`, `@barname@`, `@bar@`, and `@v@` to compose the string, as seen below.

```
\barLabelsTU{"Within the \\"@env@\\" environment, within the
  \\"@barname@\\" environment, the bar \\"@bar@\\"
  has a value of @v@"}
```

When the empty string (`\barLabelsTU{""}`) or the empty argument (`\barLabelsTU{}`) is passed, the result is to default to `customBarLabels`.

No `\TU` key is provided. For the case that a bar does not have a `\TU` key (no tooltip specified), the argument of `\barLabelsNoTU`, which is JavaScript function or a JavaScript string, provides tooltips for the bar.

Declare *`\barLabelsNoTU`* *in preamble*

```
\barLabelsNoTU{⟨JS-function⟩|⟨JS-string⟩}
```

`\barLabelsNoTU` is specified in the preamble and cannot be changed in the body of the document. This labeling system applies to all bars with no `\TU` key.

Description of argument types

⟨***JS-function***⟩ When the argument is a JavaScript function, the referenced function must be written. In the sample file `bgjs-basic1.tex`, the JavaScript function `customLabelsForBars(fld,v)` is defined; use this function as a template for writing your own custom handlers.

```
\barLabelsNoTU{customLabelsForBars} % applies to all bars without a \TU key
```

⟨***JS-string***⟩ When the argument is a JavaScript string, use the variables `o.env`, `o.barname`, `o.bar`, and `o.value` to compose the string. The default is,

```
\barLabelsNoTU{o.barname+": "+o.bar+", Value: "+o.value}
```

This is equivalent to `\barLabelsNoTU{}`. Refer to the demo file `bgjs-basic1.tex` for more information.

• Non-tooltip techniques

In addition to using the `PDF` form feature of tool tips, the bars can be visually labeled by

- using typeset text between the bars, refer to `examples/bgjs-basic2.tex`;
- using form fields between the bars, refer to `examples/bgjs-basic3.tex`, also illustrated below;
- using layers (OCG) between the bars, refer to `examples/bgjs-pro1.tex`, this example requires a `dvips/Distiller` workflow.

Class instructions. Go to a road of your choice and count the number of vehicles of each type during rush hour (for one hour), fill out the form below and turn it in for credit.[1]

This example appears in `examples/ bgjs-basic3.tex`

Auto:

Truck:

SUV:

Van:

Optimize

Reset horizontal bar graph: Reset

One advantage of using forms over the other two methods is that the labeling can be dynamically revised using JavaScript; for example, by inserting the value of a bar (perhaps as a percentage).

---

[1] It is strongly recommended you read the documentation `bgjs-examples.pdf` on `bgjs-basic3.tex` as this example uses the command `\labelFld` not formally documented in this manual.

Part II

Bar graph with an unspecified number of bars

This package also supports the `bargraph` environment that contain no `\barfor` or `\cmd` tokens. These kinds of bar graphs are primarily designed to support the creation of probability mass functions (pmf) and cumulative distribution functions (cdf).

`dynamic` option

The creation of bar graph with an unspecified number of bars require the `dynamic` option:

```
\usepackage[dynamic]{bargraph-js}
```

This brings in a JavaScript function `displayDyBargraph()`.

When used to describe the bar graphs of a discrete probability distribution, it is assumed the weights are evenly distributed and contiguous on the number line.[2]

7. Dynamic bar graphs

As in **Part I**, the package defines two environments, `bargraphenv` and `bargraph`; the latter environment is placed within the former one.

```
\begin{bargraphenv}[⟨bgenv-options⟩]{⟨bgenv-name⟩}
   \begin{bargraph}{⟨bg-name⟩}\isdynamic\end{bargraph}
\end{bargraphenv}
```

The first thing to note is that the `bargraph` environment has no options; the options `nbars`, `gap`, and `bardimen` are automatically assigned. Multiple `bargraph` environments are not supported within a `bargraphenv` environment. As a signal to the package that the current `bargraph` environment is one that is dynamic, place the `\isdynamic` command within the `bargraph` content, and nothing else.

Multiple `bargraph` environments are not supported within the same `bargrapenv`, but you can have two `bargraphenv` environments, one to hold the `pmf` and another to hold the `cdf` bar graphs.

The function that does all the work is `displayDyBargraph`, it has four required arguments and one optional one.

```
displayDyBargraph(⟨bgenv-name⟩,⟨aPr⟩,⟨bPmf⟩,⟨bOptimize⟩[,⟨object⟩])
```

The arguments are described next.

⟨*bgenv-name*⟩ is the name of the target environmet, a JavaScript string that is enclosed in double (or single) quotes (`"⟨bgenv-name⟩"`).

⟨*aPr*⟩ is an array of probability mass information; each entry in this array has the form `[k,pmfPr,cdfPr]`, where `k` is a value of the distribution, `pmfPr` is the probability mass at that value; and `cdfPr` is the cumulative mass up to and including the value `k`.

---

[2] This restriction of contiguous can be bypassed with a little trickery.

⟨*bPmf*⟩ is true if we are to draw bar graph for a probability mass function (`pmf`), and false if we are to draw bar graph for a cumulative distribution function (`cdf`).

⟨*bOptimize*⟩ is true if we optimize the graph (applies to `pmf` only), otherwise, we use the current scale factor.

⟨*object*⟩ is a JavaScript object having the following properties:

- `bc:`⟨*color*⟩ is the border color for each of the bars, ⟨*color*⟩ is a JavaScript color: `["RBG",1,0,1]` or `color.red`. The default is `color.red`.
- `fc:`⟨*color*⟩ is the fill color for each of the bars, ⟨*color*⟩ is a JavaScript color: `["RBG",1,0,1]` or `color.blue`. The default is `color.blue`.
- `lbl:`⟨*JS-func*⟩ is a JavaScript function that provides the tool tips for each of the bars. When this property is not present, the built-in JavaScript function `_labelDyBars(pr,v,bPmf)` is used.

  A custom labeling function can be written and specified as the value of the `lbl` property. The arguments for a labeling function are `pr` (the current value of the random variable); `v` (the probability associated with `pr`, `v` is a `pmf` or `cdf` value depending on the switch `bPmf`); and `bPmf` is `true` when this is for a `pmf` and `false` when this is for a `cdf`.

The bar graph for the probability mass function (pmf)

Optimize Under normal scaling, the height of this region is 1 unit, when the bar graph is optimized, the height is the height of the tallest bar.

Random Reset Toggle Border

This example appears in `examples/bgjs-dyn1.tex`

Notice that the widths of the bars are automatically adjusted to allow all the bars to be displayed in the bar graph window. The minimum width is `bardimen`, which is set to 20(bp). In this example, maximum weight is randomly chosen less than or equal to 40. In this demo we do not display the `cdf`. Populating a table, as seen in the margin, is optional. Each row is one entry in the ⟨*aPr*⟩ array. In the sample file `bgjs-dyn1.tex`, several techniques are used to create the ⟨*aPr*⟩ array.

Part III

Postscript

We finish this documentation with additional miscellaneous commands, comments on demonstration files, and notes on my retirement.

8. Miscellaneous commands

8.1. The bar graph data structure

The `bargraph-js` package associates with each `bargraphenv` environment a hidden text field named `internalData.`⟨*bgenv-name*⟩. This field uses the JavaScript object `dataForEnv` to store information about the environment. The data stored is,

```
if typeof dataForEnv=="undefined"
  var dataForEnv=new Object;
dataForEnv["⟨bgenv-name⟩"]=new Object;
dataForEnv["⟨bgenv-name⟩"].width=⟨width⟩;   // width in big points
dataForEnv["⟨bgenv-name⟩"].height=⟨heigh⟩;  // height in big points
dataForEnv["⟨bgenv-name⟩"].horiz=⟨true|false⟩;
dataForEnv["⟨bgenv-name⟩"].sf=⟨scale-factor⟩;
dataForEnv["⟨bgenv-name⟩"].bgs=[⟨list-of-bg-names⟩];
dataForEnv["⟨bgenv-name⟩"].origin=⟨0|.5⟩;
```

8.2. Setting the scale

The initial scale of a `bargraphenv` environment is set by the command

```
\scaleFactorDef{⟨pos-num⟩}
```

When placed in the preamble, `\scaleFactorDef` sets the default scale factor; the bargraph-js declares `\scaleFactorDef{2}`. When a bar graph is reset (cleared), the value ⟨*pos-num*⟩ is used to reset the scale value

When executed in the body, the initial scale factor of all `bargraphenv` environments that follow are affected, unless the declaration occurs in a group. For example, `\scaleFactorDef{3}` sets initial scale factor to 3; a more provocative example is found in sample file `bgjs-adv1.tex` where we declare,

```
\fbox{\scaleFactorDef{dataForEnv["math"].height/100}%
\begin{bargraphenv}[width=2in,height=2in,o=vert]{math}
...
\end{bargraphenv}}
```

Here we set the scale factor to be the height of the environment (in bp points) divided by 100; that way, data are re-scaled as a proportion of the height of the bar graph. We make this declaration inside the `\fbox` to make the declaration local.

8.3. Optimizing a bar graph

To optimize all the bar graphs within a `bargraphenv` environment, use the built-in JavaScript function `optimizeScaling("⟨bgenv-name⟩")`. Below is a "generic example" for optimizing an environment.

```
\pushButton[\TU{⟨tool tip text⟩}\CA{Optimize}
  \AAmouseup{optimizeScaling("⟨bgenv-name⟩");}]{optimize}{}{13bp}
```

The `optimizeScaling()` function is used in numerous examples of this package.

8.4. Manually scaling a bar graph

The user can be allowed to manually re-scale the bar graph. For this purpose, `bargraph-js` defines two commands `\displaysfFor` and `\manualsfFor`,

```
\displaysfFor[⟨eforms-opts⟩]{⟨bgenv-name⟩}{.5in}{13bp}
\manualsfFor[⟨eforms-opts⟩]{⟨bgenv-name⟩}{}{13bp}
```

where, `⟨bgenv-name⟩` is the name of the `bargraphenv` environment that these controls are to re-scale. This pair of commands appear in the sample files `bgjs-basic4.tex` and `bgjs-comma1.tex`.

Within the text field generated by `\displaysfFor`, simple arithmetic operations can be made; eg, if current scale factor is `3.3` (the optimized scale value, perhaps) then entering `3.3 / 2` and pressing the re-scale button, re-scales the bar graphs to half their height or width, depenting on whether `o=vert` or `o=horiz`.

8.5. Resetting a bar graph

Reset one or more bar graphs, as well as other fields using the JavaScript function `resetBargraphs(⟨various⟩)`

```
\pushButton[\CA{Reset}⟨other-options⟩
  \AAmouseup{resetBargraphs(⟨various⟩);}
]{reset}{}{13bp}
```

The `⟨various⟩` argument consists of a list of field names and `bargraphenv` names to be reset. The name of the `bargraphenv` environment on page 3 is `vehiclesV`. The code for the `Reset` button in that example is,

```
\pushButton[\CA{Reset}\A{\JS{resetBargraphs("vehiclesV");}}
```

Notice, the argument of `resetBargraphs` is `"vehiclesV"`, this will clear not only all bar graphs in the `bargraphenv` environment, but also the input fields (created by `\inputFor`); in the latter case, this is because the root name of the `\inputFor` fields is `"vehiclesV"`.

The function `resetBargraphs` is used extensively in the sample file of this distribution.

9. Demonstration files

This distribution comes with numerous sample files. The files are packed into the DTX file `bgjs-examples.dtx` (and can be unpacked by latexing `bgjs-examples.ins`, if not done already). Read the documentation file `bgjs-examples.pdf` for detailed descriptions of each sample file.

10. My retirement

Now, I simply must get back to it. DPS