© International Scientific Conference on Engineering, Technologies and Systems
TECHSYS 2018, Technical University – Sofia, Plovdiv branch
17 – 19 May 2018, Plovdiv, Bulgaria

OBJECT-BASED ENTITY RELATIONSHIP DIAGRAM DRAWING LIBRARY: ENTREL.JS

ERDİNÇ UZUN, TARIK YERLİKAYA, OĞUZ KIRAT

Abstract: *An entity relationship diagram (ERD) is a visual helper for database design. ERD gives information about the relations of entity sets and the logical structure of databases. In this paper, we introduce an open source JavaScript Library named EntRel.JS in order to design sophisticated ERDs by writing simple codes. This library, which we have developed to facilitate this task, is based on our obfc.js library. It generates a SVG output on the client side for modern browsers. The SVG output provides storage efficiency when compared to existing ERD drawings created with existing drawing applications. Moreover, we present our animation library to gain action for elements in your web page.*

Key words: *Entity relationship diagrams, SVG, JavaScript*

1. Introduction

The internet has been evolving for over 25 years with some standards including HTML (HyperText Markup Language), CSS (Cascading Style Sheets), JavaScript, XML (eXtensible Markup Language) and so on. These standards are shared by the W3C throughout the world, and browser developers release updates to support these standards. The W3C [1] continues to set standards on a number of different issues. In this study, one of these standards is SVG (Scalable Vector Graphics) [2] that is used for drawing entity relationship diagrams (ERDs) and JavaScript language is used for creating these ERDs with simple codes.

ER modeling was developed for increasing the clarity of database design by Peter Chen[3]. This model is the result for systematic data analysis of system. It is usually drawn in a graphical form named ER diagrams including entities, their attributes and relationships between entities. This model is typically implemented as a database. Software developers and database designers can easily discuss the design of database and software over the ER diagram. Therefore, this subject is the basis of the database courses. ER design can be created very quickly with our library introduced in this study.

The ERD drawing process can be done on the server or client side by using the drawing applications including Microsoft Visio, OpenOffice / LibreOffice Draw, Dia, Diagramly and so on. The drawing file stored on the server side is displayed in the browser in the <img> element. On the other hand, it can still be prepared on the server side and displayed in the CANVAS [4] or SVG elements used for drawing graphics in HTML 5. In this study, the SVG is suitable for our study instead of CANVAS. CANVAS is typically used in web-based games, while it allows the drawing process with JavaScript codes. Because SVG is a vector-based system, the browser is not affected by the magnification. It was chosen for this reason. It can also be drawn on the client side by using JavaScript's advantage and holding fewer code on the server side. This makes the server load lighter and the drawing works are made on the client side. However, with the JavaScript file loaded once, it is only necessary for utilizing the AJAX (asynchronous JavaScript and XML) [5] in order to upload the required code. AJAX improves the bandwidth performance of web-based applications. In this study, the object creation codes are stored in the server, and drawing with the EntRel.JS library is performed on the client side.

The EntRel.JS library (Entity Relationship) is based on the obfc.JS library (Object-Based Flow Charts) [6] that we develop is an object-based library for drawing SVG flow charts across modern web browsers. It makes easily to construct objects, links and connections. Moreover, it dispatches a click event when an object or a line is clicked and descriptions can be added for all clicks. The EntRel.JS allows you to design complex ERDs by writing short codes. SVG output is parsed from these texts and the SVG output is produced on the client side.

Technologies such as SVG, JavaScript and AJAX are used in many different subjects such as numerical graphics, networking, geography,

 ISSN Online: 2535-0048

medicine and electricity. For example, Saito and Ouyang [7] indicate how to draw data on the client side with ChartML. They use SVG and JavaScript in the client side and they produce graphs. Some studies [8-11] focus on how network topologies can be displayed on the browser by using SVG, JavaScript and AJAX. Yin and Zang [12] describe SVG and AJAX technologies in the Web geographic Information system. Fang and et al. [13] explain the use of these technologies in local thermal power plant management. Birr and et al. [14] introduces how three-dimensional medical data can be demonstrated in the Web environment. Alhirabi and Butler [15] perform the gene notation using SVG.

This study presents a JavaScript Library to create ERDs with objects. Moreover, you can also easily link objects to each other. Draw function of these objects produces a SVG output in browsers. Moreover, we introduce an animation library (animation.JS) that we developed.

2. obfc.JS library

Before explaining the EntRel.JS library, we give information about obfc.JS library that we developed earlier. obfc.JS has 24 different SVG shapes for drawing flow charts. In this section, we will describe the most basic features of this library.

2.1. Creating an object

Before creating an object, you add obfc.js file and SVG element to body of a web page. Then, you connect library to the id of the SVG element. Code 1 indicates these lines.

Code 1. *Preparing library and creating an object*

```
<script src="obfc.min.js"></script>
<svg id="demo" width="600" height="700">
</svg>
<script>
prepare_SVG("demo");
var object2 = add_theObject(new Process(300, 150, 1, ["Line 1", "Line 2"], 10));
</script>
```

To draw an object, add_theObject function can be used for the given SVG element. add_theObject is a function that adds the object into a given SVG element and returns this object. This returned object is used for drawing lines between two objects. There are 24 different SVG objects in obfc.JS. “Process” is the one of objects from 24 different objects. There are 9 parameters for creating an object. First two parameters are required, others are optimal.

Object_Name(_middle_x, _middle_y, _size, _text, _text_size, _description, _fill_color, _stroke_color, _text_color);

- _middle_x and middle_y: centre of the object. (Required)
- _size: For example, default size of a process object width is 125 and height is 50. _size value is multiplied by these values.
- _text and _text_size: Text is written in the center of an object. This parameter can be defined string value or array[“”, “”…]. If your text is too long, you can use array for creating lines.
- _description: These value can be coded in HTML format. These value is displayed in a HTML element that contains “desc” id after clicking an object or a line.
- _fill_color: Default value is white. But, the object color can be determined with this parameter.
- _stroke_color: Default value is black.
- _text_color: Default value is black.

2.1. Creating lines between objects

After creating all objects, objects can be linked by using draw_theLine function.

Code 2. *Creating lines between two objects*

```
<script>
….
var o_line1 = draw_theLine(new Line(object1, object2));
</script>
```

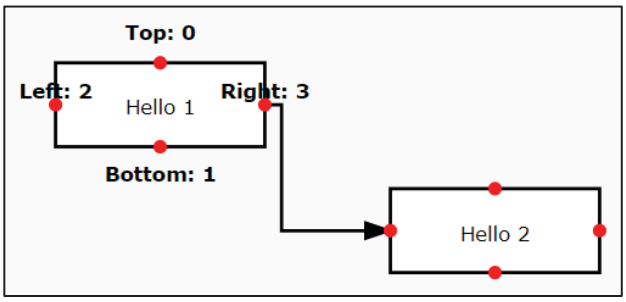

Fig. 1. *Output of Code 2*

Line is a function that determines the path for given two objects and their positions. This function has 9 parameters. First two parameter is required and others are optimal.

Line(object1, object2, position1, position2, _text, _text_size, _description, _stroke_color, _text_color);

- object1 ve object2: are variables that is defined in the previous section.

- position1 and position2: are position information of objects. There are four positions for all shapes. Top=0, Bottom=1, Left=2 and Right=3. But, when these values are not entered or entered "-1", this function automatically determines these position by calculating differences between all unused positions. (Unused position means that this position is used for creating lines)
- _text and _text_size: text in line. This function selects the longest sub-line for writing text.
- description, _stroke_color, _text_color: (same with previous section)

Moreover, you can determine connection points manually as shown in Code 3.

Code 3.

```
<script>
var object1 = add_theObject(new Process(100, 75, 1, "Hello 1", 12));
var object2 = add_theObject(new Process(300, 150, 1, "Hello 2", 12));
var o_line1 = draw_theLine(new Line(object1, object2, 0, 1));
var o_line2 = draw_theLine(new Line(object1, object2, 2, 2));
var o_line3 = draw_theLine(new Line(object1, object2, 1, 3));
</script>
```

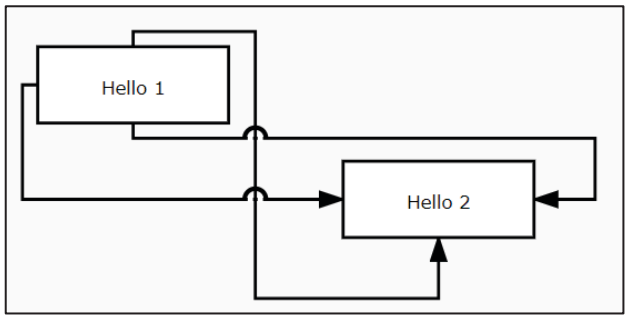

Fig. 2. *Output of Code 3*

In this example, position informations are added to link two objects. For example, 0 means top of object1 and 1 means bottom of object2. There are three lines in this example. Moreover, obfc.js has jumping mechanism in collision of lines. For more information, you can visit the following web page:

https://www.e-adys.com/obfc-js/object-based-flow-charts-obfc-js/

3. EntRel.JS

EntRel.JS is JavaScript Library for creating ERDs with simple JavaScript methods. Figure 3 gives information about the location of the libraries in the Web. Libraries of obfc.JS, EntRel.JS and Animation.JS takes the drawing code from a server and parse these codes for drawing into an SVG element. The file sizes of bfc.JS, EntRel.JS and Animation.JS are 78 KB, 3.85 KB and 3.62 KB. The first installation of these libraries is loaded into the browser cache, and these libraries are not updated on other requests.

There are two main classes for creating diagrams as FlatLine and Entity. Entity class derived from Process class of obfc.js. For better understanding this object, you can examine Section 2.1.

Entity(_middle_x, _middle_y, _size, _text, _attributes, _text_size, _weak, _description, _fill_color, _stroke_color, _text_color)

The fifth parameter contain extra information about attributes of an entity. You can define 8 attribute in array format for an entity. For example:

[null, "AuthorID(PK)", "AuthorName", "AuthorSurName"]

The first attribute contain a null value so that it can be drawn. The first four attributes are in top and others are in bottom. Weak parameter is used for a weak entity that cannot be uniquely identified by its attributes alone.

FlatLine connects the objects. FlatLine class derived from Line class of obfc.js. FlatLine don't contain an arrow in drawing. Code 4 is an example for Entity and FlatLine. Moreover, Fig 4 is output of Code 4.

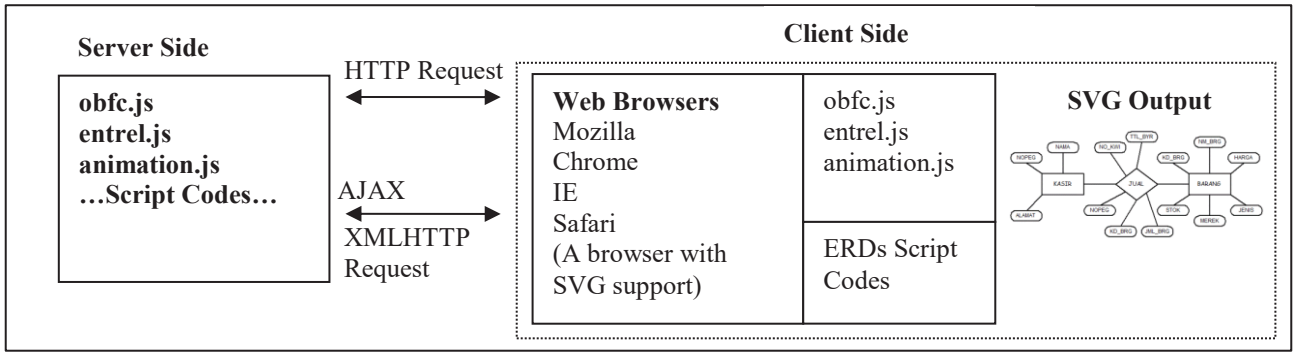

Fig. 3. *Location of the EntRel.JS and Animation.JS in the Web*

Code 4. An example ERD

```
<script>
prepare_SVG("demo");
var object1 = add_theObject(new Entity(300, 350, 0.75, "Books", [null, null, null, null, null, "BookID(PK)", "Title"], 16));
var object2 = add_theObject(new Entity(65, 150, 0.75, "Authors", [null, "AID(PK)", "AName", "ASurname"], 16));
var object3 = add_theObject(new Entity(300, 150, 0.75, "Types", [null, "TID(PK)", "TName"], 16));
var object4 = add_theObject(new Entity(500, 150, 0.75, "Publishers", [null, "PID(PK)", "PName", "Location"], 16));
var object5 = add_theObject(new Decision(65, 250, 0.75, ["R1"], 12));
var object6 = add_theObject(new Decision(300, 250, 0.75, ["R2"], 12));
var object7 = add_theObject(new Decision(500, 250, 0.75, ["R3"], 12));

var o_line1 = draw_theLine(new FlatLine(object2, object5, null, null, "N"));
var o_line2 = draw_theLine(new FlatLine(object5, object1, 1, 2, "N"));
var o_line3 = draw_theLine(new FlatLine(object3, object6, null, null, "N"));
var o_line4 = draw_theLine(new FlatLine(object6, object1, null, null, "N"));
var o_line5 = draw_theLine(new FlatLine(object4, object7, null, null, "N"));
var o_line6 = draw_theLine(new FlatLine(object7, object1, 1, 3, "1"));
</script>
```

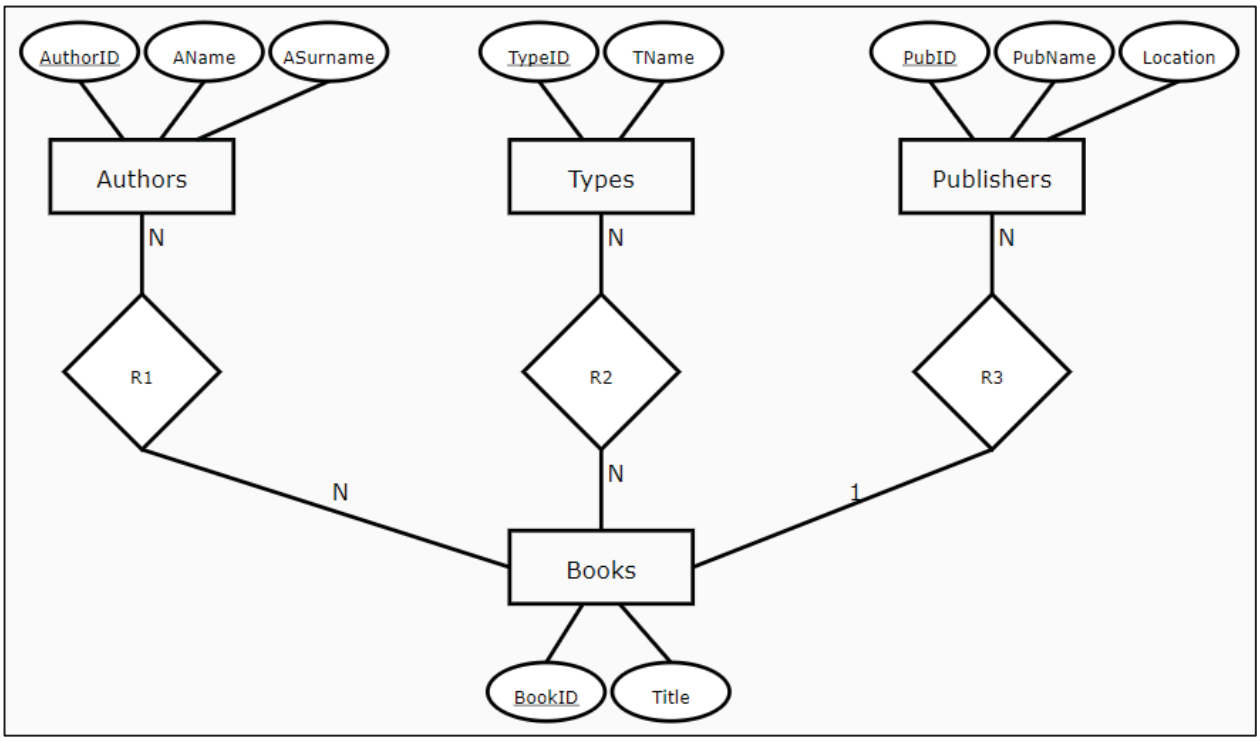

Fig. 4. *Output of Code 3*

In Code 4, (PK) keyword defines a primary key of an entity. Moreover, (DE) keyword defines derived attributes and (MV) keyword is for multivalued attributes. There are four entity objects

and three Decision objects created from obfc.JS library. And these objects link with six FlatLine. Now let's add animation to this drawing.

4. animation.JS

With animation.js, you can easily animate your a desired element of child elements in your web page. Navigation bar of this animation contains links: Previous, Next, Show All, Hide All, Start Animation, and Stop Animation. Code 5 is a simple animation code.

Code 5. Creating an animation a web page

```
<script src="animation.min.js"></script>
<script>
initializeAnimation(null, ".animation", "div,p",
"#masthead,#secondary");
</script>
```

- Add animation.min.js your web page, then call initializeAnimation method for configuration of the animation.
- The first parameter of this method is the number of elements. But this parameter is used by other libraries: obfc.js and entrel.js.
- The second parameter is the base element of animation. You can use a selector for setting this parameter.
- Third parameter is inner elements in the base element. You can use more than one selector for selecting the desired elements.
- Fourth parameters is used to set opacity of the selected elements.

Moreover, you can set time interval for animation and opacity for elements. For example: (Append the following code to Code 5)

```
opacity = 0.1; //opacity of elements
timeInterval = 3000; //3 seconds
```

For creating links for navigating animation in a web page, you can append Code 6 for your web page.

Code 6. Navigation bar

```
<div id="anavbar" class="anavbar"
style="display:none">
<ul class="horizontal">
<li><a id="pre">Previous</a></li>
<li><a id="next">Next</a> </li>
<li><a id="show">Show All</a> </li>
<li><a id="hide">Hide All</a> </li>
<li><a id="start">Start Animation</a> </li>
<li><a id="stop">Stop Animation</a> </li>
</ul>
</div>
```

5. The use of obfc.JS and EntReel.JS with amination.JS

For creating animations, you can group the objects created by obfc.JS and EntReel.JS using an array. If two objects are together, these objects can be grouped together. (For example ([object5, object2])). You can append the following codes to Code 4 for creating animation in Fig 4.

```
var groups = [object1, [object5, object2],
[o_line1, o_line2], [object6, object3], [o_line3,
o_line4], [object7, object4], [o_line5, o_line6]];
prepareClassforAnimation(groups);
initializeAnimation(groups.length - 1);
```

prepareClassforAnimation method prepares objects for animation. initializeAnimation method starts animation. For testing animation, you can visit the following web page:

https://www.e-adys.com/web_tasarimi_programlama/entrel-js-creating-entity-relationship-diagrams-with-javascript-and-svg/

6. Conclusion

In this study, the open source EntRel.JS and animation.JS Libraries that we developed and the functions of these libraries to draw ERDs in the Web environment are introduced. Thanks to this library, drawings are made quickly with very little code. Unlike other drawing applications, the links between shapes are automatically feasible. SVG output occupies very little space according to the other image formats.

In future studies, we plan to develop a design library that enables drawing and drag-and-drop functionality through the Web page without writing codes. We are also aiming to group some designs and share them in a web environment. Finally, we intend to develop new libraries for different subjects including logic circuits, data structures, database management systems, software engineering and system analysis in order to draw different shapes.

Additional Information

All codes are open-source. Web addresses and help documents are as follows.

- https://github.com/erdincuzun/obfc.js
- https://github.com/erdincuzun/entrel.js
- https://github.com/erdincuzun/Animation.js
- https://www.e-adys.com/

REFERENCES

1. World Wide Web Consortium, https://www.w3.org/, (14.06.2018)
2. Scalable Vector Graphics, https://www.w3.org/Graphics/SVG/, (14.06.2018)
3. Chen, P. (1976). The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems*. 1 (1): 9–36.
4. HTML 5 Canvas, https://www.w3.org/TR/2dcontext/, (14.06.2018)
5. AJAX, https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX, (14.06.2018)
6. Uzun, E., Buluş, H. N. (2017). Object-based flowchart drawing library. International Conference on Computer Science and Engineering (UBMK 2017), Antalya, Turkey, 5-8 Oct. 2017, pp. 110-115.
7. Saito, T. and Ouyang, J. "Client-side data visualization," 2009 IEEE International Conference on Information Reuse & Integration, Las Vegas, NV, 2009, pp. 194-199. doi: 10.1109/IRI.2009.5211550.
8. Valle, R. D. T., Passos, D., Albuquerque, C. and Muchaluat Saade, D. C. (2008). Mesh Topology Viewer (MTV): an SVG-based interactive mesh network topology visualization tool. *2008 IEEE Symposium on Computers and Communications*, Marrakech, pp. 292-297.
9. Lin, T., Zou, F., Kienle, H. M. and Muller, H. A. (2008). A domain-customizable SVG-based graph editor for software visualizations. *2008 IEEE International Conference on Software Maintenance*, Beijing, 2008, pp. 466-467.
10. Fan, C., Wu, Y. and Wang, F. (2009). SVG based on Ajax and its application in graphical network topology management. *2009 IEEE International Conference on Communications Technology and Applications*, Beijing.
11. Kehe, W., Tingting, W., Yanwen, A. and Wenjing, Z. (2015). Study on the Drawing Method of Project Network Diagram. *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*, Hangzhou, 2015, pp. 95-98.
12. Yin, F. and Zhang, L. (2010). Research of WebGIS based on SVG and Ajax technology. *2nd IEEE International Conference on Information and Financial Engineering*, Chongqing, pp. 629-632.
13. Fang, W., Zhang, J., Hu, B., Zhang, Q. and Ha, X. (2011). Graphics and data web publishing for local thermal power plant management information system. *2011 International Conference on Multimedia Technology*, Hangzhou, 2011, pp. 337-340.
14. Birr, S., Mönch, J., Sommerfeld, D., Preim, U. and Preim, B. (2013). The LiverAnatomyExplorer: A WebGL-Based Surgical Teaching Tool. *in IEEE Computer Graphics and Applications*, vol. 33, no. 5, pp. 48-58.
15. Alhirabi, N. and Butler, G. (2015). A visual spreadsheet using HTML5 for whole genome display. *2015 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, Niagara Falls, ON, 2015, pp. 1-7.

Authors' contacts

Erdinç Uzun,
Organization: Namık Kemal University, Çorlu Faculty of Engineering, Computer Engineering Department
Address: NKÜ Çorlu Mühendislik Fakültesi Dekanlığı, Silahtarağa Mahallesi Üniversite 1.Sokak, No:13, 59860 Çorlu / Tekirdağ / TURKEY
Phone (optional): +90 (282) 250 2325
E-mail: erdincuzun@nku.edu.tr

Tarık Yerlikaya (Corresponding author)
Oğuz Kırat
Organization: Trakya University, Faculty of Engineering, Computer Engineering Department
Address: Trakya Üniversitesi Ahmet Karadeniz Yerleşkesi Mühendislik Fakültesi 22020 Merkez / Edirne /TURKEY
Phone (optional): +90 (284) 226 1217 / 2215
E-mail: tarikyer@trakya.edu.tr, ogzkirat@gmail.com