World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:10, No:3, 2016

HTML5 Online Learning Application with Offline Web, Location Based, Animated Web, Multithread, and Real-Time Features

Sheetal R. Jadhwani, Daisy Sang, Chang-Shyh Peng

***Abstract*—**Web applications are an integral part of modem life. They are mostly based upon the HyperText Markup Language (HTML). While HTML meets the basic needs, there are some shortcomings. For example, applications can cease to work once user goes offline, real-time updates may be lagging, and user interface can freeze on computationally intensive tasks. The latest language specification HTML5 attempts to rectify the situation with new tools and protocols. This paper studies the new Web Storage, Geolocation, Web Worker, Canvas, and Web Socket APIs, and presents applications to test their features and efficiencies.

***Keywords*—**HTML5, Web Worker, Canvas, Web Socket.

I. Introduction

HTML, HyperText Markup Language, is the main markup language for creating web pages and other information that can be displayed in a web browser. HTML 1.0 debuted as a hybrid of Standard Generalized Mark-up Language (SGML), and enabled documents to link other documents. HTML 2.0 was the first official set of standards by which all browsers were measured [1]. It featured supports for math and scientific documents, and for style sheets with the STYLE tag and CLASS attribute [2]. HTML 3.0 draft included new and improved functionality for HTML, but was quickly abandoned due to the size of the overhaul and multitude of proprietary browser-specific tags [3]. The World Wide Web Consortium (W3C) soon standardized HTML 3.2, which included features such as tables, applets, text flow around images, subscripts, superscripts, and so on [4]. The Language quickly evolved to HTML 4.0 to support international languages, Cascading Style Sheets (CSS), form extensions, scripting, and much more [5]. Shortly after, the final version of classic HTML, HTML 4.01 was published which contained minor revisions and corrections [6].

In early 2000 W3C issued the specifications of Extensible Hypertext Markup Language (XHTML) 1.0. XHTML has most of the same syntax conventions as HTML, but enforces stricter rules. However, many XHTML's planned benefits (e.g. interoperability with Extensible Markup Language tools, efficient page processing for automated programs, portability with mobile platforms, language extensibility, etc.) never came to light [7]. Accordingly, XHTML 2.0 was defined in hope to meet the expectation. From a theoretical point of view, XHTML was cleaner and more intelligible. But from a practical point of view, it imposed new coding styles without adding much new functionality [8]. Development of XHTML 2.0 quickly lost steam. The Web Hypertext Application Technology Working Group (WHATWG) and the W3C then jointly designed the new specification HTML5 [9].

HTML5 inherited many earlier features, but also added plenty new elements, attributes and abilities [10], [11]. The core HTML5 is the official W3C's contribution. It includes many new tags and features such as semantic elements and form widgets, which provide clear, concise, and controlled markup to HTML applications. Features furnished by WHATWG are mostly specifications where JavaScript is required to manipulate the contents of a document, respond to user interaction, and support rich web applications. There are also next-generation features such as CSS3, web workers, geolocation, and canvas [12].

HTML5 specifications are massive and challenging. While there are a few online demonstrations of the new APIs, they don't stipulate the detailed theoretical foundation. This paper implements a HTML5 Online Learning Application with Offline Web, Location Based, Animated Web, Multithread, and Real-Time Features (HOLA) to demonstrate the key foundations and features. Paper is divided into sections for the five main features: Offline Web Applications, Location Based Applications, Animated Web Applications, Multithreaded Web Applications, and Real-time Web Applications. These sections cover the foundation principles and implementation issues. Paper is finished with conclusion.

II. Offline Web Applications

HTTP is stateless. Neither the client nor the server keeps track of earlier communications or transactions. Cookie can be implemented to ease the job of user identification entries and web page customization. Cookies are small, often encrypted, text files located in browser directories. Cookies store information such as name, address, online buying patterns, login details, etc. The server to provide corresponding dynamic interaction and advertisement can use such information. Cookies can automatically fill online forms. Cookies can also be used on the statistics of client activities. Cookies do come with concerns. Each visit to a web site leaves some client information behind, e.g. client's computer

Sheetal R. Jadhwani and Daisy Sang are with the Computer Science Department, California State Pollytechnic University, Pomona, CA 91768 USA.

Chang-Shyh Peng is with Computer Science Department, California Lutheran University, Thousand Oaks, CA 91360 USA (phone: 805-493-3819, e-mail: peng@callutheran.edu).

name and IP address, the brand of browser, the operating system, the URL, and so on. The security implication can be problematic. For instance, an eavesdropper together with a packet sniffer could intercept the cookies and obtain unrestricted access to the supposedly secure sites [13], [14]. These issues are carefully addressed in the HTML5 Offline Web Applications, which provides virtual connection to website of interest when users are offline. The Offline Web Applications enable users to work with the resources that don't require network connectivity, e.g. HTML, image, CSS, JavaScript, media files, etc. The two major components in HTML5 Offline Web Applications are Application Caching and Web Storage.

Web Storage stores named key/value pairs locally within the client web browser. The data has a size limit of 5MB per domain or subdomain, and is persistent even after navigating away from the web site, closing the browser tab, or exiting the browser. The data is however never transmitted to the remote web server. Such implementation provides the flexibility of offline computation, the option to sync data back to the server while online, better performances, and a more streamlined development environment. The data storage is divided into session storage and local storage [15]. Session storage is for the duration of a single page session. Reloading or restoring the page will not clear the storage, while opening a new page creates a new session. Page navigation within the same domain also maintains the session storage. Session storage is for any pages from the same website opened in the same window. For security purposes, session storage does not cross the domains. Local storage is on the other hand tied to one origin and persists across all sessions on the entire origin, but cross-origin does draw the line at the subdomain level.

Application Cache institutes full offline browsing capability by implementing a server-side manifest file, which is a list of URLs referencing all needed resources such as HTML, CSS, JavaScript, images, etc. The web browser that implements HTML5 Offline Applications will read the manifest file, download the resources, cache locally, keep the local copies up to date, and sync with the server when online again [16]. Cache manifest file has three sections: explicit section, online whitelist section, and fallback section. The explicit section lists resources that will be downloaded and cached locally. The online whitelist section denotes resources that are not supposed to be available offline and thus cannot be downloaded. The fallback section defines substitutions for online resources that could not or were not cached as planned.

Fig. 1 Sticky Notes Weekly Shopping List Screenshot

Based upon the HTML5Sticky [19], HOLA presents the Sticky Notes Application. Sticky Notes Demo allows users to create and manage a list of up to 50 sticky notes for weekly shopping needs. Each sticky notes comes with a randomly assigned color. Saved sticky notes are available till deletion. Sticky notes can be accessed and interacted with either online or offline. Fig. 1 is a sample screenshot.

HOLA's Sticky Notes Application implements both Web Storage and Application Cache, and uses the JavaScript library Modernizr [18]. As shown below, it stores all sticky notes data in browser local storage with the web storage setItem() method.

```
if (Modernizr.localstorage){
localStorage.setItem(index, index);
localStorage.setItem(index + '|pos',
```

```
parseInt($(stickynote).offset().left, 10) + '|' + parseInt($(stickynote).offset().top, 10));
localStorage.setItem(index + '|text', $(stickynote).find('textarea.note-title').val() + '|' + $(stickynote).find('textarea.note-content').val());
localStorage.setItem(nindex + '|settings', bgcolor );
}
```

Web storage getItem method is invoked when sticky notes data is retrieved.

```
if (localStorage.getItem(index + 'stickynote')){
note_index = index;
// get color
temp_array = localStorage.getItem(i + 'stickynote' + '|settings').split('|');
bgcolor = temp_array[0];
// get position info
temp_array = localStorage.getItem(i + 'stickynote' + '|pos').split('|');
pleft = temp_array[0];
ptop = temp_array[1];
// get text info
temp_array = localStorage.getItem(i + 'stickynote' + '|text').split('|');
}
```

To remove selected sticky notes, the removeItem() method is executed with corresponding indices.

```
if (Modernizr.localstorage){
    var identifier = html5sticky.getIdentifier($(el));
    localStorage.removeItem(identifier);
    localStorage.removeItem(identifier + '|pos');
    localStorage.removeItem(identifier + '|text');
    localStorage.removeItem(identifier + '|settings');
}
```

and, following is the cache manifest file.

```
OfflineAppDemo.html
index.html
about.html
stickyDemo.html
Style/html5stickynotes.css
Style/Demo.css
Style/design.psd
Style/main.css
Style/menu.css
Style/styles.css
http://fonts.googleapis.com/css?family=Architects+Daughter
http://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js
Script/html5sticky.js
Script/jquery1.6.2.js
Script/modernizr.custom.23610.js
Images/add.png
Images/bg.png
Images/close.png
Images/hideButton.png
Images/navigation.jpg
Images/navigationmenu.jpg
Images/pushpin.png
Images/remove.png
Images/restore.png
Images/save.png
Images/showButton.png
```

Testing verifies that HOLA Sticky Notes Application successfully stores and maintains data locally, updates and checks current data in different browser tabs, and provides consistent and persistent offline functionality and real-time online synchronizations.

III. Location Based Applications

Geolocation is the determination of the geographic position. Various techniques have been deployed since the early ages, including magnetic compass, chronometers, radio triangulations, Doppler, GSM/CDMA, Wi-Fi, Bluetooth, Internet Protocol (IP) Address, and Global Positioning System (GPS) [19]. To meet the growing needs of location-aware tasks, HTML5 institutes the optional Geolocation API. Geolocation is an opt-in service. The primary function is getCurrentPosition, which returns a timestamp and coords, where timestamp is the date and time when the location was calculated and the coords has the latitude and longitude of user's location [20].

Near Me? in HOLA demonstrates HTML's Geolocation API by looking up user's current location and nearby places of interest. It uses Google Maps JavaScript API [21] to load the Google map. There are four types of maps. Roadmap displays the normal 2D map. Satellite map displays photographic map. Hybrid map shows a mixture of photographic map and overlay of other relevant information. Terrain map displays elevation and water features. After loading the map, Modernizr is used to detect whether the browser supports Geolocation. If not, users are notified accordingly. Otherwise, current location is queried with the navigator.geolocation.getCurrentPosition function call which returns a LatLng object denoting the latitude and longitude of the current location. Reverse Geocoding is then used to convert the LatLng object to a common readable address. Following is the sample code for Reverse Geocoding.

```
html5geolocation.getCurrentAddress = function(position){
        geocoder = new google.maps.Geocoder();
        geocoder.geocode({'latLng': position}, function(results, status){
        if (status == google.maps.GeocoderStatus.OK) {
            if (results[1])
                address = results[0].formatted_address; }
    else {
        alert("Geocoder failed due to: " + status);
        address = "";
    }
}
```

Near Me? continues with Google Map Places API to search for nearby places of interests within a specified radius. Several parameters are required for the search, including an API key for quota management, latitude/longitude, radius, etc. The Places API returns an array of up to 20 establishment results per query. Each entry of the results array contains details of a point of interest, such as an ID which uniquely identifies the place even across separate searches, geometric information, business name, price level, user review rating, reference for optional retrieval of additional information, types, vicinity, etc. [22]. Map marker overlays are generated and depicted on

the map accordingly. It's shown that HOLA's Near Me? loads the map correctly, plots current location, and displays the nearby establishments accurately. Fig. 2 is a screenshot of Near Me? that is centered in Los Angeles. Fig. 3 depicts a sample search result.

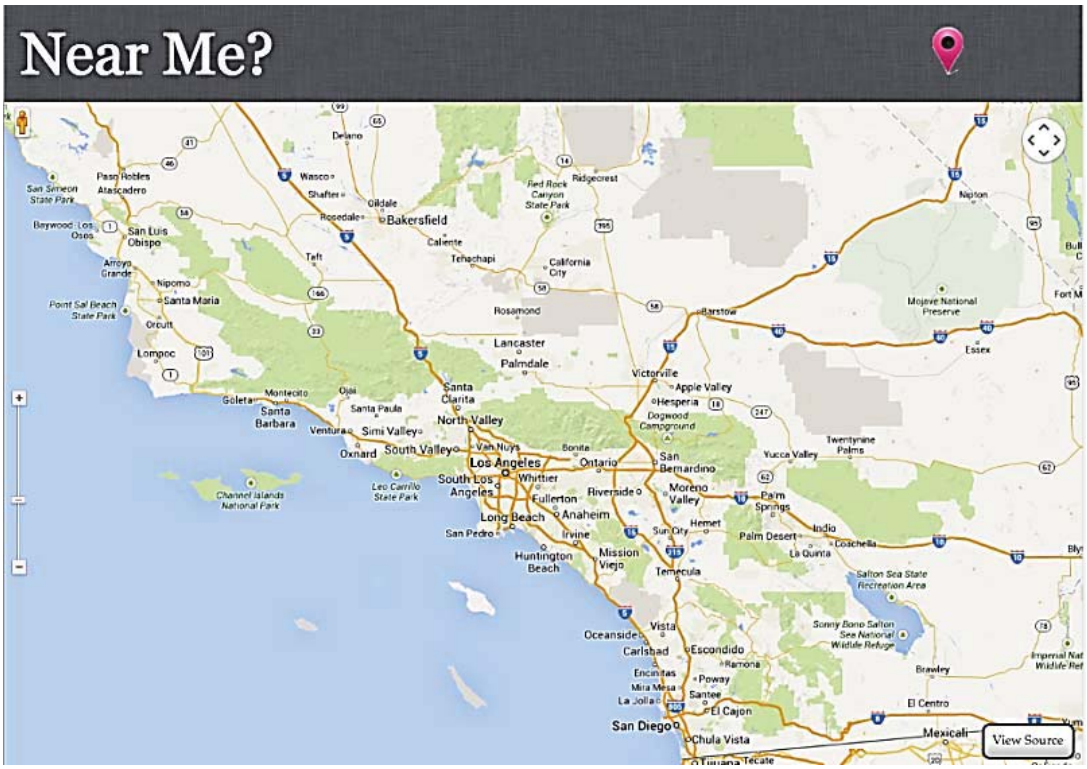

Fig. 2 Near Me? Screenshot on Los Angeles

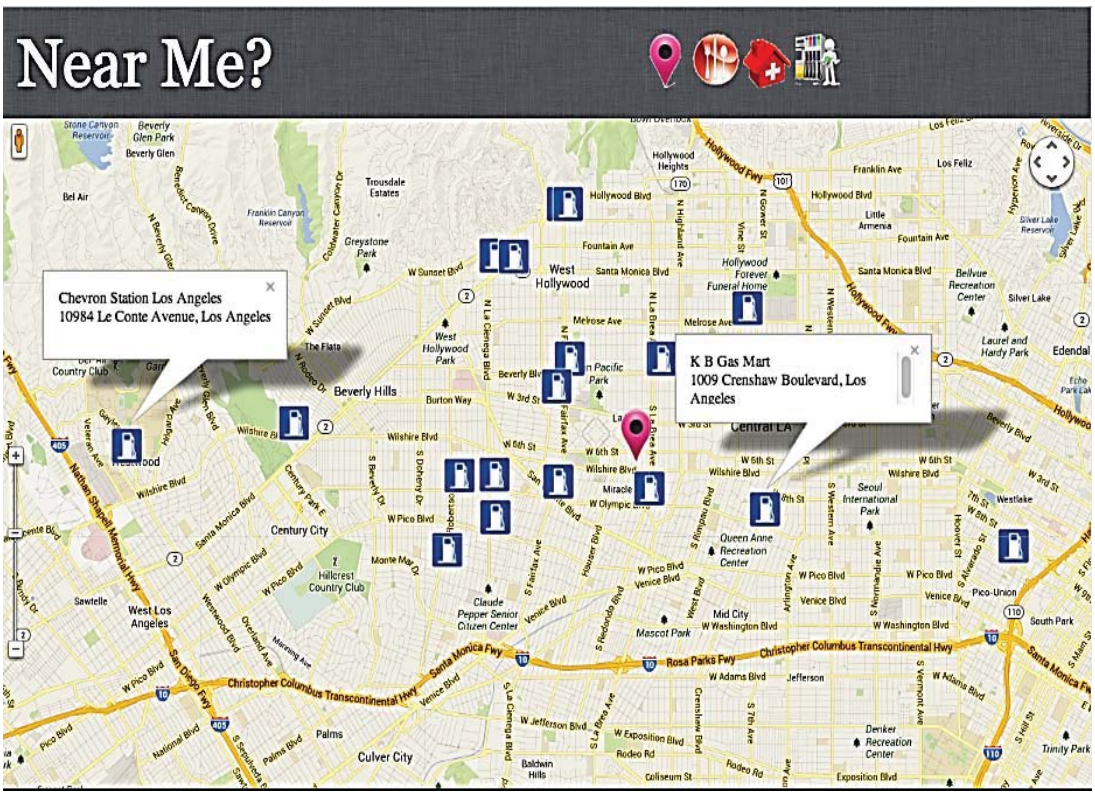

Fig. 3 Near Me? Sample Search Result

IV. Animated Web Applications

Graphics rendering is nowadays an integral part of many web applications. Bitmap rendering and scalable vector graphics are two main camps [23]. Bitmap rendering is the technique behind many popular formats such as bmp, png, and jpeg. With the help of advanced tools such as Adobe Photoshop and Image Editor, designers are able to generate professional and sophisticated web imagery. Scalable Vector Graphics (SVG) takes a completely different approach. SVG is an XML based vector image format for two-dimensional graphics that has native support for interactivity and animation. Vector drawing first creates paths by connecting dots with straight lines, curved lines, rectangles, polygons, ellipses, and so on. Enclosed areas are then filled with various colors and shades. In general, paths are calculated and plotted on a unit scale, such as pixel. The entire graphic can be rendered to the resolution of the user's content. The image quality can remain intact regardless of the resolution. Vector drawings are thus considered scalable as they can potentially scale to any resolution without the loss of imagery quality.

Despite the nice features of bitmap rendering and SVG, both have drawbacks. Bitmap image is static. Though can be implemented with animated gif, animation is a string of predefined static images and generally does not accommodate user interaction. Besides, bitmap rendering is suitable for pre-defined resolutions only. Change of resolution can result in image pixilation, and user experience quickly deteriorates. As for SVG, its performance can degrade drastically in computationally intensive drawings of complex shapes. The Canvas API of HTML5 can bridge the gap [24].

HTML5 Canvas API is an immediate mode fire-and-forget bitmap graphics technique. Canvas is all about drawing and manipulating pixels. It completely redraws the bitmap screen on every frame. Once the graphics is completely rendered, there is no memory of what was drawn. The basic HTML5 Canvas API includes a 2D context that allows a programmer to draw various shapes, render text, and display images directly onto a defined area of the browser window. It includes methods to apply colors, rotations, gradient fills, alpha transparencies, pixel manipulations, and various types of lines, curves, boxes, and fills. Canvas API has a few context methods and attributes: fillStyle gets the colors for fills and strokes, strokeStyle sets the colors for fills and strokes, fillRect fills a rectangle with the current fillStyle, strokeRect outlines a rectangle with the current strokeStyle, clearRect clears the canvas context at the specified rectangle, beginPath resets the current path on the canvas context, closePath marks the current subpath closed, lineWidth gets or sets the width of paths, stroke traces the current path with the color defined by strokeStyle and a thickness defined by lineWidth, fill fills the current path with the fillStyle, moveTo creates a new subpath, lineTo adds a line to the current subpath, rect creates a new subpath in the shape of a rectangle and closes that subpath, quadraticCurveTo draws a quadratic curve, bezierCurveTo draws a cubic Bezier curve, arcTo draws an arc with the given control points and radius, arc draws an arc, lineCap gets or sets the style of the end of lines, lineJoin gets or sets the style of corners, save saves the current state, and restore recalls the current state of the canvas.

HTML Canvas API does not make SVG obsolete. In fact, as shown in Fig. 4, they can complement each other [25]. Many 2D games adopt the cross over model. Canvas is used where intensive dynamic graphics and animations are needed, and SVG is applied when rich user interaction is expected.

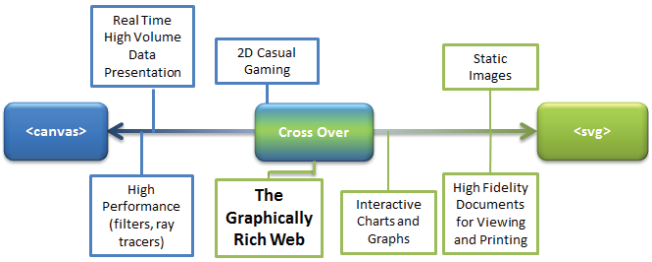

Fig. 4 Spectrum of Canvas to SVG

HOLA's Revolving Earch uses HTML5 Canvas API to animate the partial solar system. Sun is drawn as the center of the Solar system. Earth follows an orbit to rotate and revolve about itself and the Sun respectively. An Earth sprite, shown in Fig. 5, is used for Earth's incremental motion along the orbit.

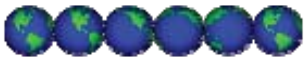

Fig. 5 Earth Sprite

Following is the code to draw revolving earth using sprite image on the canvas.

```
html5SolarCanvas.drawRevolvingEarth = function(){
context = html5SolarCanvas.getCanvasContext();
context.clearRect(0,0,canvas.width,canvas.height);
shadowTimer = 0;
html5SolarCanvas.drawSun();
html5SolarCanvas.drawArc();
if(flag){
      context.save();
      context.translate(canvas.width/2 -10,canvas.height/2 -10);
      if(reverse){
         xunits = Math.cos(-angle) * 60 ;
         yunits = Math.sin(-angle) * 60 ;
      }else{
         xunits = Math.cos(angle) * 60 ;
         yunits = Math.sin(angle) * 60 ;}
   context.drawImage(img,sourceX, sourceY, 23,imgHeight, xunits,
yunits, imgWidth, imgHeight);
   context.translate(-canvas.width/2 - 10,-canvas.height/2 - 10);
   context.restore();
   if(i< 5){
      sourceX += 23;
      angle += angleIncrement;
      i++}
   else{
      sourceX = 0;
      angle += angleIncrement;
      i =1} }
}
```

Earth Revolution also contains a control panel, including play, stop, pause, forward, and reverse. Play, stop, and pause are self explanatory. Forward increases the speed of rotation and revolution. Reverse also increases the rotation and revolution speed but in the reverse direction. Following is the code for reverse.

```
html5SolarCanvas.reverseAnimation = function(){
angleIncrement = 0.1;
reverse = true;
flag = true;
$('#btnRvs').find('.flagImg').attr('src',"Images/reverse_off.png");
$('#btnPlay').find('.flagImg').attr('src',"Images/play_off.png");
$('#btnFwd').find('.flagImg').attr('src',"Images/forward.png");
$('#btnStop').find('.flagImg').attr('src',"Images/stop.png");
$('#btnPause').find('.flagImg').attr('src',"Images/pause.png");
clearInterval(originalTimer);
clearInterval(newTimer);
newTimer = setInterval(html5SolarCanvas.drawRevolvingEarth,
100);
};
```

Fig. 6 depicts a screenshot. Earth Revolution clearly shows the effectiveness in rendering the initial graphics as well as the animation at various speeds.

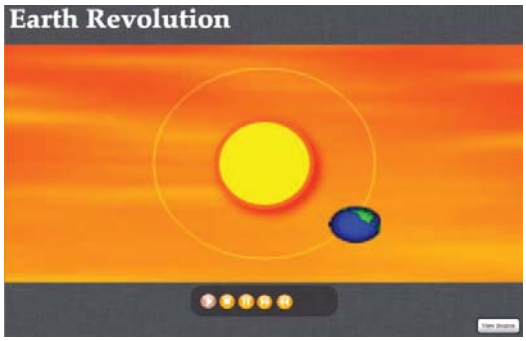

Fig. 6 Earth Revolving Screenshot

V. Multithreaded Web Applications

JavaScript has been a key tool to introducing and enhancing interactivity in web browsing. As shown in Fig. 7 [26], all scripts required by the application are first loaded and readied for user interaction. Upon validation, the Document Object Model is processed and updated. JavaScript is designed to handle one task at a time, and is thus classified single-threaded. Single-threaded computation is straightforward, and works great for light tasks. However, if the task is computationally intensive, performance can degrade significantly. User will be interrupted, browser can be frozen till completion of task, events are stacked up in the pipeline, and the "Application not responding" error message may be thrown at the user.

HTML5 Web Workers, depicted in Fig. 8 [26], is designed to amend JavaScript Single Threading's limitation. Web Workers are in effect parallel threads of execution. Web Workers is in a self-contained execution environment, and communicates with the main thread only through asynchronous message passing. Web Worker's specification includes Worker Object and WorkerGlobalScope [27]. Worker Object is created with the Worker constructor, and can receive message via postMessage. The postMessage method clones the message in current browsers. Messages from a Worker can be received by listening for message events on the Worker object. The global object for that corresponding execution environment is a WorkerGlobalScope object.

HOLA's Web Worker to Rescue! is an application that demonstrates the capabilities of Web Worker API by performing multi-threaded tasks. Using the HTML5 Canvas [28], Web Worker to Rescue! first draws 250 objects of user specified shapes on the canvas. It then move the shapes indefinitely along a vector. A vector has two parameters: direction and speed. Direction is defined in radians, and speed is the movement in pixels per drawing. Web Worker to Rescue!'s main thread is responsible for drawing the objects on the canvas. Dedicated threads calculate and store selected objects' properties, and forward them to the main thread. Following is the main code.

```
html5CanvasWorker.createBouncingShapes = function(canvasWidth,
canvasHeight, numBalls, radius,      speed, shape){
for(vari = 0; i<numBalls; i++){
      tempX = (Math.floor(Math.random()* canvasWidth);
      tempY = (Math.floor(Math.random()* canvasHeight);
      tempAngle = Math.floor(Math.random() * 360 );
      tempRadians = tempAngle * Math.PI/180;
      tempXunits = Math.cos(tempRadians) * speed;
      tempYunits = Math.sin(tempRadians) * speed;
      tempShape = {
         x: tempX,
         y: tempY,
         radius: radius,
         speed : speed,
         angle :tempAngle,
         xunits: tempXunits,
         yunits :tempYunits};
      shapes.push(tempShape);}
   return shapes;
     }
```

Upon reaching the boarder of the canvas, objects bounce back in reversing the direction according to the law of reflection. Following is the calculation for bouncing triangles.

```
for (vari = 0; i<numBalls; i++){
triangle = triangles[i];
triangle.x += triangle.xunits;
triangle.y += triangle.yunits;
context.strokeStyle = "white";
if(color == 0)context.fillStyle = '#52D017'
elsecontext.fillStyle = color;
context.beginPath();
triangle.x1 = triangle.x - radius;
triangle.y1= triangle.y;
triangle.x2= triangle.x + radius;
triangle.y2= triangle.y;
triangle.x3= triangle.x;
triangle.y3= triangle.y - radius;
context.moveTo(triangle.x1,triangle.y1);
context.lineTo(triangle.x2,triangle.y2);
context.lineTo(triangle.x3,triangle.y3);
```

```
context.lineTo(triangle.x1,triangle.y1);
context.stroke();
context.closePath();
context.fill();
if(triangle.x+ radius >canvas.width || triangle.x+radius< 0)
        triangle.angle = 180 - triangle.angle;
elseif(triangle.y+radius>canvas.height || triangle.y+radius<0)
        triangle.angle = 360 - triangle.angle;
html5CanvasWebWorker.updateTriangle(triangle);
    }
```

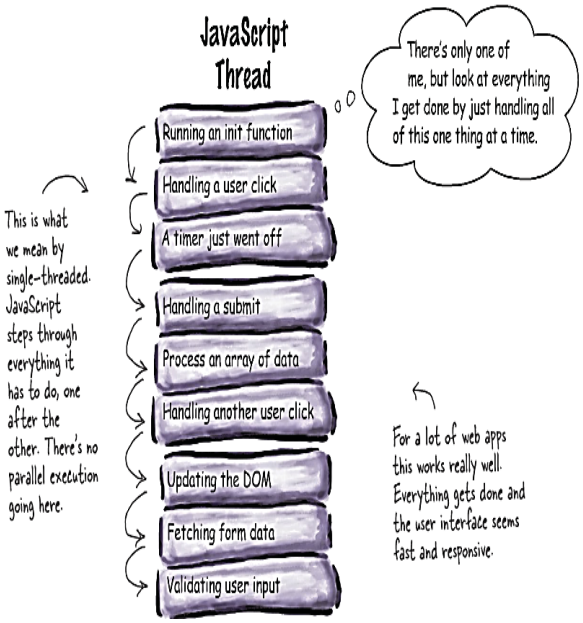

Fig. 7 JavaScript Single Thread Execution

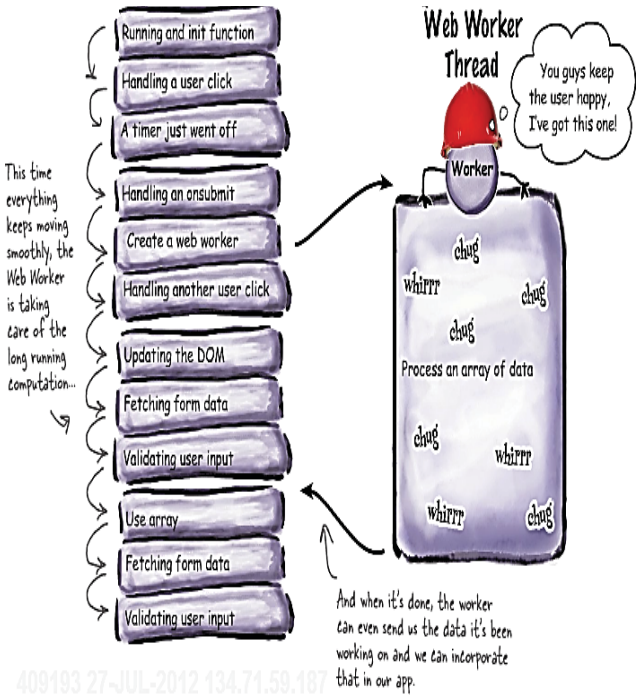

Fig. 8 JavaScript with Web Worker

Web Worker to Rescue! shows that user interface is responsible while switching the objects amongst various shapes. Application works flawlessly regardless of the number of objects and the speed of movement. Fig. 9 shows a screenshot where objects are chosen to be circles. Fig. 10 is a screenshot with triangles and Fig. 11 depicts when user is given a color picker.

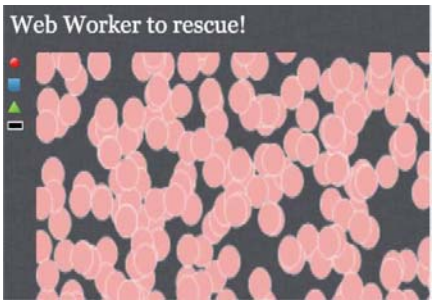

Fig. 9 Web Worker to Rescue! with Circles

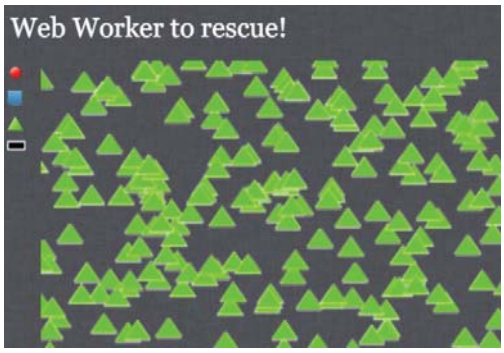

Fig. 10 Web Worker to Rescue! with Triangles

VI. Real-Time Web Applications

Real-time web applications enable users to receive information as soon as it is available. Examples include online gaming, social stream updates, business applications, and web-based monitoring. Most such applications use the request-response model from the Hypertext Transfer Protocol (HTTP) suite. Client first sends a HTTP request, and then waits until the HTTP response is returned. To create more interactive web applications, polling protocols such as AJAX and Comet have been developed over the HTTP connections. With AJAX, browser application performs HTTP requests based on the XMLHttpRequest API. The XMLHttpRequest API can handle the HTTP request in the background asynchronously without blocking the user interface. Yet, server response still requires a corresponding HTTP request from the client. Comet, also known as Reverse Ajax, overcomes this limitation by creating and maintaining long-lived HTTP connections which enable servers to push data to clients. While such polling architecture help realize real-time applications, there are some drawbacks. If the polling frequency is low, applications can suffer from higher event latency. If the polling frequency is high, there can be many redundant requests and empty response and thus poor network resource utilization.HTML5 defines the Web Socket that provide an efficient solution.

Web Sockets is a new communications protocol and JavaScript API that enable two-way interactive communication between a client browser and a server [29]. It defines full-duplex channel over a single socket on the Web, and uses its own protocol instead of that of HTTP. To establish a Web Socket connection, client and server are upgraded from the HTTP protocol to the Web Socket protocol during their initial handshake. Upon successful handshake, data transfer starts. Client and server can initiate the transaction independent from each other. A Web Socket object can dispatch four different events: open, message, error, and close [30]. Open event is triggered when a connection is established. Message event takes place when messages are received. Error event is set off in response to unexpected failures. Close event is launched when the Web Socket connection is to be closed.

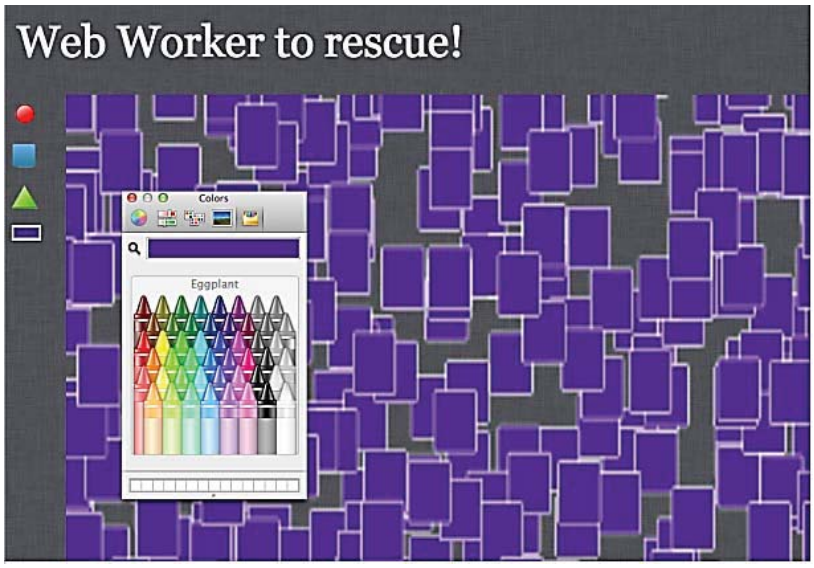

Fig. 11 Web Worker to Rescue! with Color Picker

HOLA's Real-Time Voting Application was designed to try out the Web Socket API and demonstrate its efficiency. This application polls for the popular smartphone devices. Users can select from Android, iPhone, Microsoft and Blackberry device. Real-Time Voting Application uses the node.js platform [31] and the socket.io interface [32]. The server object is created as follows where port 8015 is the chosen port.

```
vario = require('socket.io').listen(8015);
io.sockets.on('connection', function (socket) {
    socket.broadcast.emit('user connected');
    socket.on('message', function (message) {
        console.log("Message from server: "+ message);
        io.sockets.emit('message', message);
        });
});
```

Client can be connect to the server as follows.

```
html5WebSocketClient.connect = function(){
    socket = io.connect("http://localhost:8015");
    socket.on("connect", html5WebSocketClient.onOpen);
    socket.on('message', function (msg) {
    console.log("Message recieved on client: "+ msg);
    html5WebSocketClient.onMessage(msg);
    });
};
```

Upon successful connection, voting counter is updated for any newly connected clients with the following codes.

```
var message = {'android': androidCounter, 'iphone':
iphoneCounter, 'windows': windowsCounter,'blackberry':
blackberryCounter};
if (Modernizr.localstorage){
localStorage.setItem("android", androidCounter);
localStorage.setItem("iphone", iphoneCounter);
localStorage.setItem("windows", windowsCounter);
localStorage.setItem("blackberry", blackberryCounter);
}
socket.emit("message", JSON.stringify(message));
}
```

When a client polls for a device, its local counter is updated and the message is sent to notify and update the server. The server then broadcast that the update to all connected clients. Following is the implementation.

```
var message = { 'android': androidCounter, 'iphone': iphoneCounter,
'windows': windowsCounter,'blackberry': blackberryCounter};
if (Modernizr.localstorage){
    localStorage.setItem("android", androidCounter);
    localStorage.setItem("iphone", iphoneCounter);
    localStorage.setItem("windows", windowsCounter);
    localStorage.setItem("blackberry", blackberryCounter);
    }
    socket.emit("message", JSON.stringify(message));
}
```

A comparison study was conducted between the HOLA Real-Time Voting Application's Web Socket and the corresponding polling implementation. Case A involves 1000

clients polling every second. Case B has 10000 clients polling every second. Case C has 100000 clients polling every second. As shown in Fig. 12, the Web Socket implementation exhibits a very favorable reduction of overhead in data polling.

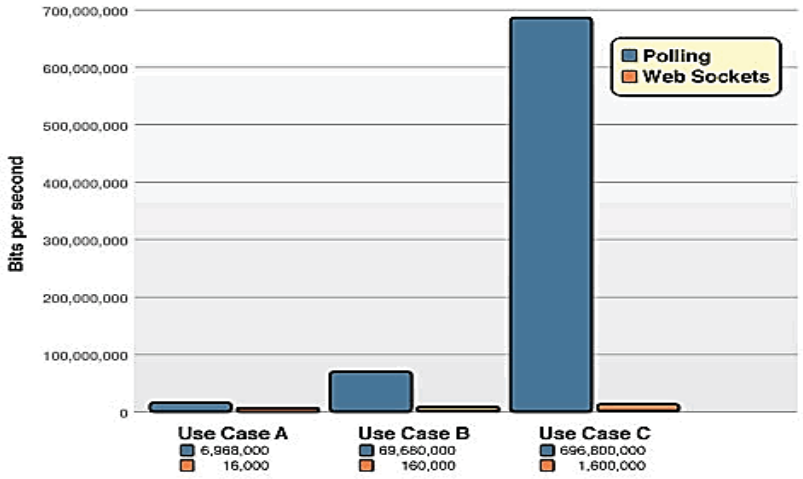

Fig. 12 Overhead Comparison Polling vs. Web Socket

VII. Conclusion

HTML5 is not just a markup language, it is a platform that empowers developers on more advanced and efficient web applications. HOLA's source code is available online [33]. Sticky Notes demonstrates the benefits of Offline Web Application and Web Storage in offline access and computation. Near Me? uses the Geolocation API to provide accurate location-aware information. Revolving Earth animates a partial solar system with Canvas API. Web Worker to Rescue! benefits from the Web Worker API in multi-threading computation. And Web Socket API enables the low-overhead Rea-Time Voting Application. HTML5 is expected to further enrich the web development and user interaction, as well as pave the road to much better use of network resources.

References

[1] D. Ragett, J. Lam, I. Alexander, and M. Kmiec, Raggett on HTML4, Addison Wesley, 1997.
[2] HTML 2.0 Specifications, https://www.w3.org/MarkUp/html-spec/, last access 2015.
[3] HTML 3.0 Specifications, https://www.w3.org/MarkUp/html3/Contents.html, last access 2015.
[4] HTML 3.2 Specifications, https://www.w3.org/TR/REC-html32, last access 2015.
[5] HTML 4.0 Specifications, https://www.w3.org/TR/1998/REC-html40-19980424/, last access 2015.
[6] HTML 4.01 Specifications, https://www.w3.org/TR/html4/, last access 2015.
[7] XHTML 1.0 Specifications, https://www.w3.org/TR/xhtml1/, last access 2015.
[8] XHTML 2.0 Specifications, https://www.w3.org/TR/xhtml2/, last access 2015.
[9] Web Hypertext Application Technology Working Group, https://whatwg.org/, last access 2015.
[10] M. MacDonald, HTML5: The Missing Manual, O'Reilly, 2014.
[11] Gartner Inc., Gartner Recommends a Hybrid Approach for Business-to-Employee Mobile Apps, Gartner Inc., 2013.
[12] HTML5 Specifications, https://www.w3.org/TR/html5/, last access 2015.
[13] L. Shevchik, HTML5 Web Storage – Cookies Are So 1994!, http://blog.newrelic.com/2012/09/18/html5-web-storage-cookies-are-so-1994/, 2012.
[14] W. Peng, and J. Cisna, HTTP cookies – a promising technology, MCB UP Ltd., 2000.
[15] M. Pilgrim, "The Past, Present & Future of Local Storage for Web Storage",HTML5: Up and Running, O'Reilly, 2010.
[16] M. Pilgrim, "Let's take this Offline",HTML5: Up and Running, O'Reilly, 2010.
[17] S. Ahmed, HTML5Sticky – Sticky Notes for the Web!, http://sarfraznawaz.wordpress.com/2011/10/08/html5sticky-sticky-notes-for-the-web/, 2011.
[18] HTML5 Modernizr, http://modernizr.com/, last access 2015.
[19] A. Holdener III, HTML5 Geolocation, O'Reilly, 2011.
[20] M. Pilgrim, "You are here (And so is everybody else)", HTML5: Up and Running, O'Reilly, 2010.
[21] Google Developers, Google Maps Javascript API, https://developers.google.com/maps/documentation/javascript/tutorial, last access 2015.
[22] Google Developers, Google Places API Place Types, https://developers.google.com/places/documentation/supported_types, last access 2015.
[23] S. Malik, Scalable Vector Graphics and bitmap rendering using Flex, http://www.ibm.com/developerworks/library/wa-svgbitmap/, last access 2015.
[24] S. Sarris, "Canvas",HTML5 Unleashed, Sams Publishing, 2013.
[25] P. Dengler, How to Choose Between Canvas and SVG for your Site, http://msdn.microsoft.com/en-us/hh552482, last access 2015.
[26] E. Freeman and E. Robson, "Putting Javascript to work: Web Worker", Head First HTML5 Programming, O'Reilly, 2011.
[27] D. Flanagan, "HTML5 APIs", JavaScript: The Definitive Guide, O'Reilly, 2011.
[28] S. Fulton and J. Fulton, "Math, Physics and Animation",HTML5 Canvas, O'Reilly, 2011.
[29] P. Lubbers and F. Greco, HTML5 Web Sockets: A Quantum Leap in Scalability for the Web, https://www.websocket.org/quantum.html, last access 2015.
[30] V. Wang, F. Salim, and P. Moskovits, The Definitive Guide to HTML5 WebSocket, Apress, 2013.
[31] J. Seidelin, "Going online with Web Sockets",HTML5 Games Creating Fun with HTML5, CSS3, and WebGL, Wiley, 2011.
[32] socket.io, a Javascript library for real-time and bi-directional web communications, http://socket.io/, last access 2015.
[33] HOLA: an HTML5 Online Learning Application Source Code, https://goo.gl/hrHhCd, last access 2015.