Animated rslidy
responsive HTML5 Slide Decks

Group 5

Rok Kogovšek, Alexei Kruglov, Fernando Pulido Ruiz, and Helmut Zöhrer

706.041 Information Architecture and Web Usability WS 2016
Graz University of Technology
A-8010 Graz, Austria

06 Feb 2017

Abstract

This project report specifies the upgrades done on the provided rSlidy application by Keith Andrews of TUG. The focus of the project was to make the already working version of the web slideshow application closer to well spread desktop solutions. This entitled improving the user interface to be more interactive, responsive and user friendly. A big focus of the project was on animation solutions.

Contents

List of Figures

List of Listings

Chapter 1

Introduction

HEre we tell we prepared a rSlidy only upgrade and a rSlidy supported by 3rd party software

Chapter 2

rSlidy

Fernando use that pdfyou found to write as much as possible about the original rslidy. And what where the reasons that this upgrade project was started - basicly the faults of the original rslidy.

Chapter 3

Changes of the Design

The design of rSlidy has undergone numerous major changes throughout our project. A comparison between the old and the new version is given in this chapter.

3.1 The Status Bar

The initial version of rSlidy was equipped with a permanent status bar, as shown in Figure 3.1.

It has been modified in terms of design and functionality. The final appearance of the status bar is shown in Figure 3.2. The following individual changes have been made.

3.1.1 Progress Bar

A simple blue progress bar has been added on top of the status bar. Its purpose is to give some visual feedback about the current progress within a presentation. Its implementation is fairly simple.

A progress bar container was added at the desired position whose width property is changed on each slide change (see Listing 3.1).

```
1 Rslidy.prototype.showSlide = function (slide_index) {
2   // ...
3   var progress_bar = document.getElementById("progress-bar");
4   progress_bar.style.width =
5         'calc(100%*' + (slide_index + 1) / this.num_slides + ')';
6   // ...
7 }
```

Listing 3.1: Adapting width of the progress bar container for authentic visual feedback [The code example is based on the users' implementation.]

3.1.2 Rearrangement / Extension of the Navigation Elements

We found it simply more intuitive to have the input field for jumping to a specific slide in the middle of the forward / backward buttons.

Apart from this, functionalities to jump to the first respectively the last slide have been added. These two starightforward implementations can be seen in Listing 3.2.

Figure 3.1: Design of rSlidy's original status bar. [Screenshot taken by the authors of this survey.]

images/status_bar_NEW.png

Figure 3.2: Design of rSlidy's modified status bar. [Screenshot taken by the authors of this survey.]

```
1  document.getElementById("status-bar-nav-button-first")
2    .addEventListener('click', function ()
3    {
4      this.showSlide(0);
5    }.bind(this));
6  document.getElementById("status-bar-nav-button-last")
7    .addEventListener('click', function ()
8    {
9      this.showSlide(this.num_slides - 1);
10   }.bind(this));
```

Listing 3.2: Implementation of the buttons for jumping to the first / last slide[The code example is based on the users' implementation.]

3.1.3 Pin Functionality

In opposition to the original rSlidy status bar, the new one features pinning / unpinning. The pinned status bar works the same as the old one. The unpinned status bar disappears when not hovering over it. When the mouse is not close to the bottom of the document, only the progress bar is visible in the unpinned mode. Two subtle triangles have been added to the unpinned status bar which are meant to function as little indicators for the actual bar. This implementation may not be the most elegant one, because it is relying on the title of the button to work properly. Some simple boolean variable which describes whether the bar is pinned or not may be a more robust solution. Still, this (see Listing 3.3) is what we came up with and it works fine as long as the title tag of the pin button in the rslidy.js file is either "Pin the status bar" or "Unpin the status bar" (depending on whether the user wants the bar to be pinned or not by default).

```
Rslidy.prototype.pinToggleClicked = function (close_only) {
  var pin_button = document.getElementById("status-bar-pin-button");
  var status_bar = document.getElementById("status-bar-content");
  var indicator_left = document.getElementById("progress-bar-indicator-left");
  var indicator_right = document.getElementById("progress-bar-indicator-right")
      ;

  if (pin_button.title == "Pin the status bar")
  {
    pin_button.title = "Unpin the status bar";
    status_bar.style = "transform: translateY(0);";
    pin_button.style.WebkitTransition = 'opacity 0.3s';
    pin_button.style.MozTransition = 'opacity 0.3s';
    pin_button.style.opacity = 0.5;
    indicator_left.style.visibility = "hidden";
    indicator_right.style.visibility = "hidden";
  }
  else
  {
    pin_button.title = "Pin the status bar";
    status_bar.removeAttribute('style');
    pin_button.style.opacity = 1;
    indicator_left.style.visibility = "visible";
    indicator_right.style.visibility = "visible";
  }
};
```

Listing 3.3: Implementation of the buttons for pinning / unpinning the status bar [The code example is based on the users' implementation.]

3.1.4 Buttons animations

Similar to an animated hamburger icon, which changes its shape on click, the buttons within the status bar of rSlidy are animated now as well. Listing 3.4 shows how the animated flip was created. These style changes and the button's text changed to an "X" lead to a simple and intuitive animation of a button which turns around to change its functionality.

```
#button-overview, #button-toc, #button-menu{
  animation-duration: 0.3s;
  animation-timing-function: ease-in-out;
  animation-fill-mode: forwards;
  animation-name: flip2Face;
}

#button-overview.clicked, #button-toc.clicked, #button-menu.clicked{
  animation-name: flip2Back;
  transform: rotateY(180);
}
```

Listing 3.4: Implementation of the animation of the buttons in the status bar [The code example is based on the users' implementation.]

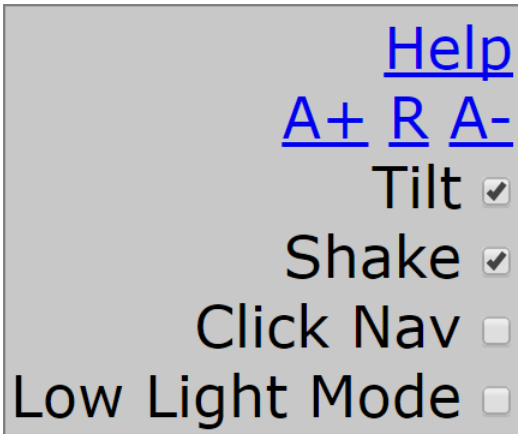


Figure 3.3: Design of rSlidy's original menu. [Screenshot taken by the authors of this survey.]

Figure 3.4: Design of rSlidy's modified menu. [Screenshot taken by the authors of this survey.]

3.2 The Menu

The menu has been modernized and harmonized as seen in a direct comparison of Figure 3.3 and Figure 3.4. Implementation-wise these changes were mostly straightforward:

- Partial transparency has been added to the menu. The actual document thus slightly shines through the menu.
- The corners have been corners in order to get a smoother look in general.
- Shadow effects have been added to the edges of the menu in order to get a very basic 3D-effect.
- Harmonization has taken place. All links were exchanged with buttons. This means more consistency in the design altogether.
- One more checkbox has been added. It allows the user to switch between a version which, as usual, shows the address of a link on hover and a version which suppresses that standard function by removing

the "href" property from all initial links.

3.3 The Help Information

This information, which can be opened from the menu, used to be a usual alert box. Due to the instructor's wish to have no third party libraries included, there are two versions of our implementation for the new help popup.

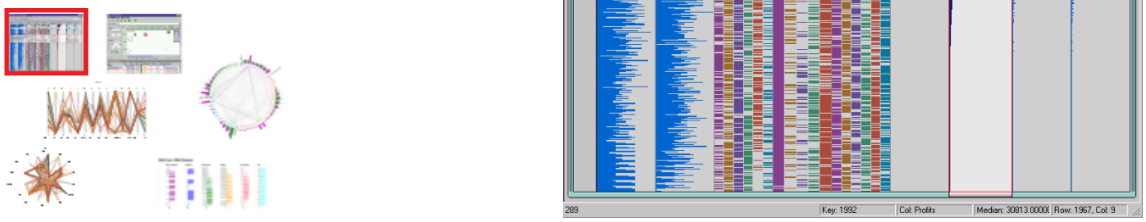
The initially intended version is based on a library called "sweetalert" (https://limonte.github.io/sweetalert2/). It allows animated popup messages with focusing on the actual text. The advantage of this way of implementation is the the polished design while having to rely on a third party library.

The revised version simply opens a new tab to display the help message. This does not look as sophisticated as the other version, but still works fine and is an independent way of solving the popup problem.

Chapter 4

Image magnification

In this part of project we have done following enhancements to our task. We have added zoom feature, which opens picture in new window. The functions, which we implemented work on many different picture formats like svg, jpeg, gif and png.Our enhancements give better possibility for users to see detailed part of image, when they zoom it in, and to see whole picture, without details, when they zoom it out.



(a) Realy size of image in slides

(b) One part of image in after popup window

Figure 4.1: Screenshot of realy size image and after pop up window. [Screenshot taken by the authors of rslide project.]

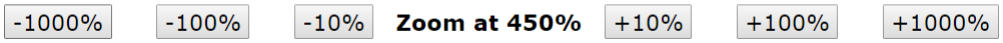


Figure 4.2: Buttons function in popup window [Screenshot taken by the project of rslides.]

In popup window we can zoom it for +-10%, +-100%, and +-1000%. We can also zoom it with +/- buttons on the keyboard. To switch it in default state, where zoom is 100%, we can do it with the press of button 0 on the keyboard.

On click on image we selected one image and open new window, to which we supply content through javascript document.write function.

```
    // Input listeners
    var images = document.getElementsByTagName("img");
    var images  = content_section.getElementsByTagName("img");

    for (var i=0, len=images.length, img; i<len; i++) {
      img = images[i];
      img.addEventListener("click", function() {
      openImageTab(this.src);
      });
    }
  };
}

function openImageTab(imgSrc) {
  var newWindow = window.open();

  var htmlCode ="<head><title>rSlidy Image View</title><link rel='stylesheet'
      href='css/reset.css'><link rel='stylesheet' href='css/normalise.css'>" +
      "<link rel='stylesheet' href='css/rslidy.css'><link rel='stylesheet' href
          ='css/slides-default.css'></head>" +
      "<body><div class='slide imageAlert'><h1><button>-1000\%</button><button
          >-100\%</button><button>-10\%</button>Zoom at <span id='zoomNumber
          '>100</span>\%<button>+10\%</button><button>+100\%</button><button
          >+1000\%</button></h1>" +
      "<div><img id='zoomedImg' src='"+ imgSrc + "'></div></div>"+
      "<script type='text/javascript'>" + String(openImageTabListeners) + ";
          openImageTabListeners();</script></body>";
  newWindow.document.write(htmlCode);
}
```

Listing 4.1: In this function we implements a detection and selection one image in slides and open her in new window.

In content we describe the page layout with buttons, picture, and javascript, which is needed for image resizing.

```
1
2   window.addEventListener('keypress', function (e) {
3     if (e.key == '+' || e.key == '-' || e.key == '0') {
4       var zoom = parseInt(titleElement.innerHTML);
5       if(e.key == '+'){
6         zoom = zoom + 10;
7       }
8       else if(e.key == '-')
9       {
10        zoom = zoom - 10;
11      }
12      else{
13        zoom = 100;
14      }
15      if(zoom > 0){
16        img.style.height = zoom * heightPer + "px";
17        img.style.width = zoom * widthPer + "px";
18        titleElement.innerHTML = zoom;
19      }
20    }
21  }, false);
22
23  var isCtrl = false;
24  window.addEventListener('keydown', function (e) {
25    if (e.which === 17) {
26            isCtrl = true;
27        }
28    }, false);
29  window.addEventListener('keyup', function (e) {
30    if (e.which === 17) {
31            isCtrl = false;
32        }
33    }, false);
34
35  window.addEventListener("mousewheel", function (e) {
36    if(isCtrl){
37      var delta = Math.max(-1, Math.min(1, e.wheelDelta));
38      var zoom = parseInt(titleElement.innerHTML);
39      if(delta > 0){
40        zoom = zoom + 10;
41      }
42      else if(delta < 0)
43      {
44        zoom = zoom - 10;
45      }
46      if(zoom > 0){
47        img.style.height = zoom * heightPer + "px";
48        img.style.width = zoom * widthPer + "px";
49        titleElement.innerHTML = zoom;
50      }
51    }
52  }, false);
53 }
```

Listing 4.2: In this function we implements a image resizing in zoom function.

Chapter 5

Animated Slideshow

The biggest downside of the original rSlidy when compared to alternative solutions, with focus on well spread desktop solutions, would be its static state. Namely the presentation flow achieved with the application was an immediate switch between states or slides. Even the user interface was behaving similarly in a state-switching way. In the design changes the effects of hiding elements with hover and transition CSS elements also include a better user experience. This follows the observations from the web animation survey, also done by our group, were the importance of animation for user experience was stressed out. With knowledge gathered from the mentioned survey we enhanced both the user interface as well the presentation flow by incorporating animation with CSS and JavaScript.

5.1 Initialization

5.2 Button animation

5.3 Hiding elements

5.4 Slide transitions

Chapter 6

Concluding Remarks

Through the course of further investigating web animations, we realized that animations are not merely there to make a website appear more beautiful, but to carry meaning as well. So, if a user sees a hamburger icon, they should, and nowadays probably they do, know what this icon stands for. We showed many other useful applications for animation in web UI design, how we can achieve them with CSS, SVG and JS. Numerous examples of code show, how powerful CSS by istelf can be and how each addition on top of it enhances it, which makes it great for RWD development