

## **Resolving Zero-Initiated Zoom Issues on the Timeline Website: Case Study**

The project is on a timeline website and it is built in javascript. This open source project is done to make storytelling simple with dynamic stories on the web and anyone can build visually rich, interactive timelines. It is convenient for beginners and experts.

### **Introduction:**

TimelineJS, an open-source project in JavaScript, is a pivotal tool that simplifies web-based storytelling by enabling the creation of visually rich and interactive timelines. Its significance lies in fostering accessibility and inclusivity, welcoming users of all skill levels to contribute and customize their storytelling experiences. The collaborative development model facilitates a diverse community of global contributors, resulting in continuous improvement, community engagement, and an enduring support network. This project serves as a valuable learning resource, empowering developers to enhance their skills and contribute to a versatile tool that has a global impact on web storytelling. TimelineJS embodies the principles of openness, transparency, and innovation, ensuring the longevity and sustainability of a dynamic and flexible platform that transcends borders and empowers storytellers worldwide.

### **Typical users:**

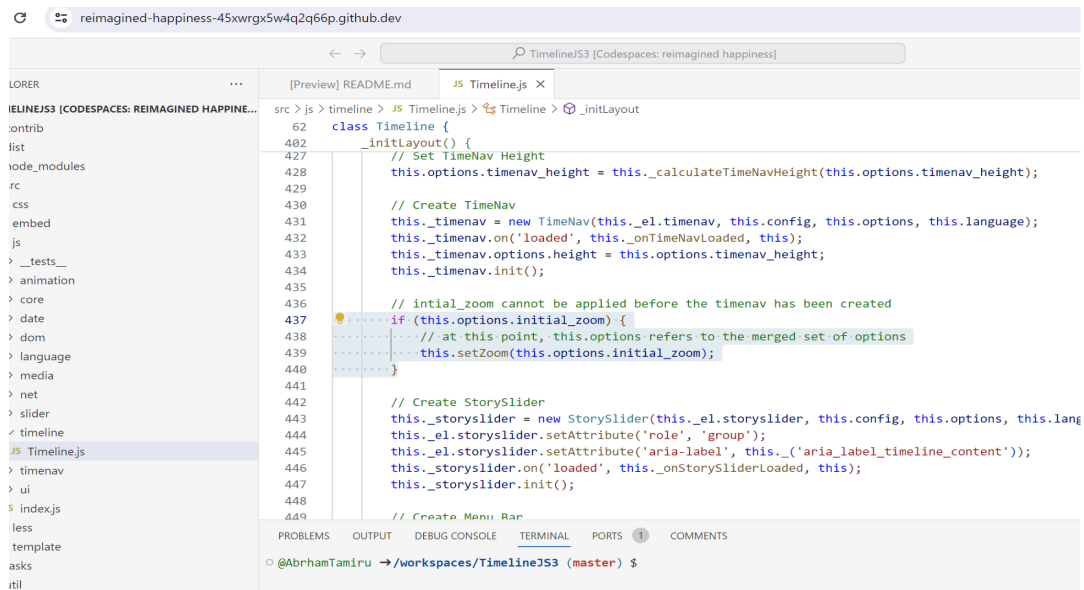
The typical user of TimelineJS is diverse and includes journalists, educators, historians, content creators, and anyone interested in presenting chronological events in an engaging and interactive format. They use TimelineJS to craft dynamic narratives, showcase historical timelines, tell compelling stories, and enhance educational content.

For example, an educator teaching a history class could utilize TimelineJS to create an interactive timeline illustrating key events during a specific historical period

### **The Issue**

The problem with the zoom feature on the Timeline website is that it restricts users from setting the initial zoom level to zero. The zoom range is from 0 to 10, but when users try to set their own initial zoom level to zero, it defaults to the system's default zoom level instead. This issue stems from the implementation of zero in JavaScript, where it is interpreted as false. Consequently, when users input zero as their

initial zoom level, the system interprets it as false and defaults to the system's default zoom level instead of considering zero as the minimum zoom size. This behavior occurs due to a specific line of code at line 437, which overrides user input of zero and sets the default zoom level instead.

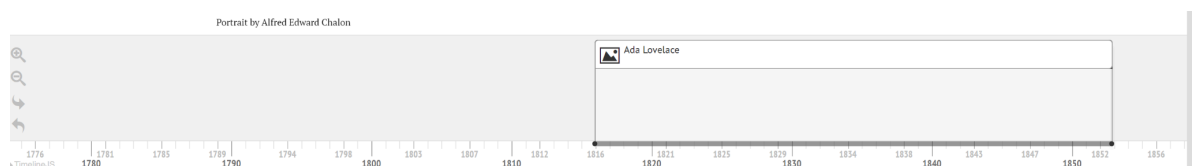


```
src > js > timeline > JS Timeline.js > Timeline > _initLayout
62 class Timeline {
402   _initLayout() {
427     // Set TimeNav Height
428     this.options.timenav_height = this._calculateTimeNavHeight(this.options.timenav_height);
429
430     // Create TimeNav
431     this._timenav = new TimeNav(this._el.timenav, this.config, this.options, this.language);
432     this._timenav.on('loaded', this._onTimeNavLoaded, this);
433     this._timenav.options.height = this.options.timenav_height;
434     this._timenav.init();
435
436     // initial_zoom cannot be applied before the timenav has been created
437     if (this.options.initial_zoom) {
438       // at this point, this.options refers to the merged set of options
439       this.setZoom(this.options.initial_zoom);
440     }
441
442     // Create StorySlider
443     this._storyslider = new StorySlider(this._el.storyslider, this.config, this.options, this.lang
444     this._el.storyslider.setAttribute('role', 'group');
445     this._el.storyslider.setAttribute('aria-label', this._('aria_label_timeline_content'));
446     this._storyslider.on('loaded', this._onStorySliderLoaded, this);
447     this._storyslider.init();
448
449     // Create Menu Bar
```

```
// initial_zoom cannot be applied before the timenav has been created
if (this.options.initial_zoom) {
    // at this point, this.options refers to the merged set of options
    this.setZoom(this.options.initial_zoom);
}
```

Old code:

- When the user initialize initial==0  
if(this.options.initial\_zoom) will be false  
So the default zoom size will be kepted,  
When initial zoom =0

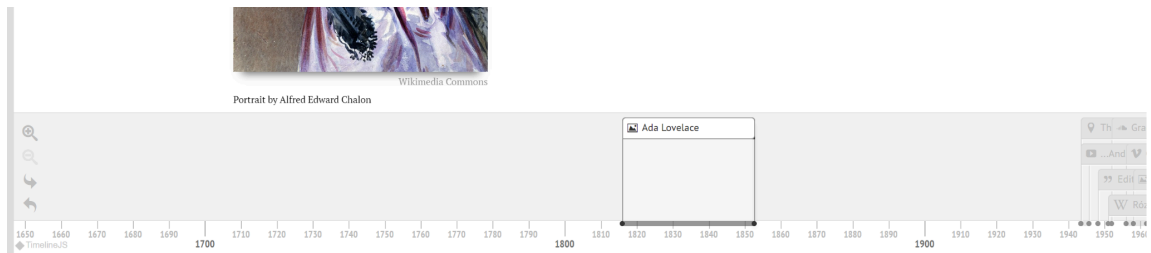


```
// initial_zoom cannot be applied before the timenav has been created
if (this.options.initial_zoom != null && this.options.initial_zoom != undefined)
{
    // at this point, this.options refers to the merged set of options
    this.setZoom(this.options.initial_zoom);
}
```

New code:

To resolve the issue, we adjust the conditional statement to only set the default zoom level if the user hasn't specified their own initial zoom. This is done by checking if the initial zoom is either null or undefined. If it is, the system sets the default zoom level; otherwise, it uses the zoom level specified by the user. This ensures that zero can be properly interpreted as the minimum zoom size.

When initial zoom =0 , it looks like this



## System overview:

TimelineJS uses a Google spreadsheet as a data source. Users can edit their timeline by going back to the spreadsheet. Changes made to the spreadsheet are automatically available to your Timeline. It incorporates a variety of media types including Text, Images, Videos, Audio, Links, and HTML. TimelineJS can also pull in media from a variety of sources. Twitter, Flickr, YouTube, Vimeo, Vine, Dailymotion, Google Maps, Wikipedia, SoundCloud, Document Cloud and more

## Tech stacks

JavaScript	<ul style="list-style-type: none"> <li>- is used to create interactive web pages and web applications.</li> <li>- enable dynamic and interactive features within the timeline.</li> </ul>
------------	---

JavaScript libraries	<ul style="list-style-type: none"> <li>• D3.js: to create interactive charts, graphs, and other data visualizations. It enhance visualization of timeline data</li> <li>• jQuery: is used to simplify HTML DOM manipulation.</li> <li>• Moment.js: used for parsing, manipulating, and formatting dates and times.</li> <li>• Lodash: provides utility functions for working with arrays, objects, and strings.</li> </ul>
HTML	to create the timeline markup. HTML is a markup language that is used to create web pages. it generated dynamically or statically to represent the timeline's structure, including event entries, media content (such as text, images, videos), and any additional interactive elements
CSS	to style the timeline. CSS is a style sheet language that is used to control the appearance of web pages.it applied to customize the appearance of various timeline components, including fonts, colors, spacing, and layout

## Flow control

Let's look the flow of control on making a new timeline:

### 1. Create a spreadsheet:

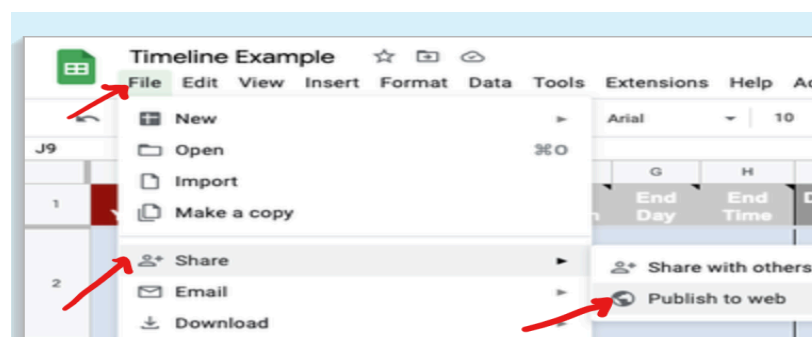
Build a new Google Spreadsheet using our template. You'll need to copy the template to your own Google Drive account by clicking the "Make a Copy" button.

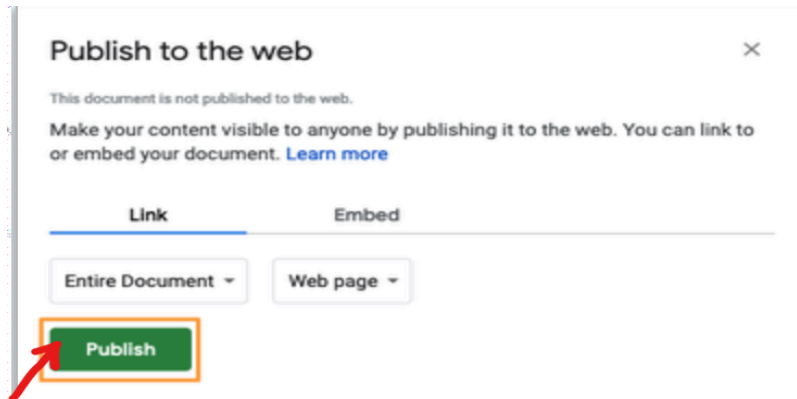
Drop dates, text and links to media into the appropriate columns. For more about work with our template, see [Making a timeline from a Google Spreadsheet](#)

Get the Spreadsheet Template 

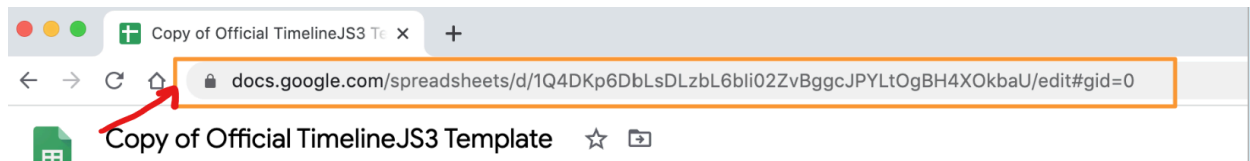
### 2. Publish to the web:

Under the File menu, Share submenu, select "Publish to the Web."





After you close the window, copy the URL in your browser's address bar. You'll use this in the next step



### 3. Generate your timeline:

Copy/paste spreadsheet URL into the box, Set the Width and Height, For more features click show.

Google Spreadsheet URL

Width       Height

**Optional settings (show)**

*Set language, fonts, starting slide and more.*

### 4. Share your timeline

Use this to link directly to your timeline

```
<iframe src="https://cdn.knightlab.com/libs/timeline3/1:
source=1xuY4upIooEeszZ_1CmeNx24eSFWe0rHe9ZdqH2xqVNk&fon
width="100%" frameborder="0"></iframe>
```

**Preview**      Open Preview in a new window

## Challenges

## Solution