

# How to create an interactive timeline using CSS and JavaScript

Timeline is a powerful visualization tool that helps users navigate and understand events along a timeline. Let's explore with TipsMake.com.com how to create interactive timelines using CSS and JavaScript.

Using the basics of HTML, CSS, and JS, you'll have an interactive timeline up and running in no time.



**Timeline** is a powerful visualization tool that helps users navigate and understand events along a timeline. Let's explore with TipsMake.com.com **how to create an interactive timeline using CSS and JavaScript** .

## Build a timeline structure

To start, outline the HTML structure in **index.html** . Create events and dates as separate components, laying the foundation for an interactive timeline.

## Timeline

Here is the breakdown of the time we anticipate using for the upcoming event.

### Occasion 1

Lorem ipsum dolor sit amet consectetur adipisicing elit. Corporis, explicabo.

12 Dec. 2023

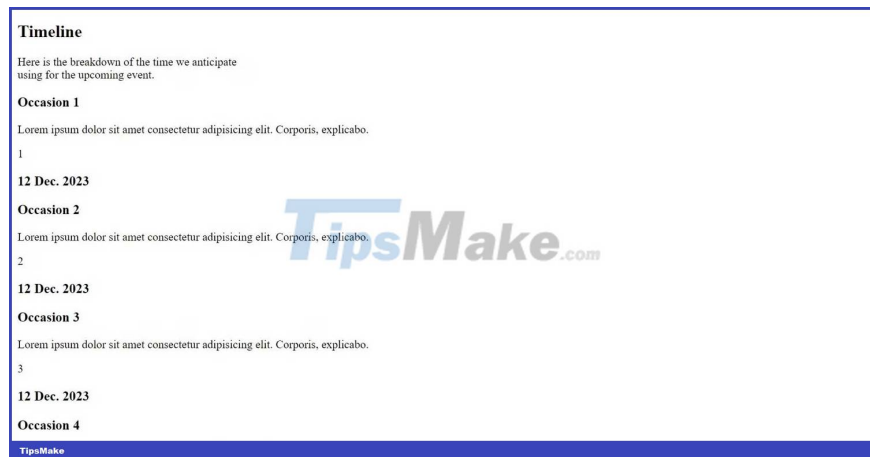
## Occasion 2

Lorem ipsum dolor sit amet consectetur adipiscing elit. Corporis, explicabo.

2

12 Dec. 2023

Currently, the component looks like this:



## Choose a vertical or horizontal timeline layout

When designing an interactive timeline, you can choose a vertical or horizontal style. A vertical timeline is easy to use, especially on phones, because this is the typical web scrolling direction. If the timeline has a lot of content, this may be the most convenient layout.

However, horizontal timelines are more eye-catching on wide screens and are a great choice for creative pages with less detail that can be scrolled in turn. Each style has its own characteristics, suitable for different web types and user experiences.

## Style the timeline with CSS

You have three types of visual elements to style your timeline: line, node, and date stamp.

1. **Lines:** A vertical line in the middle, created with the `Timeline__content::after` pseudo-element, that acts as the 'spine' of the timeline. It is styled with a specific width and color, positioned to align in the middle of the timeline items.

```
.Timeline__content::after { background-color: var(--clr-purple); content: ""; po
```

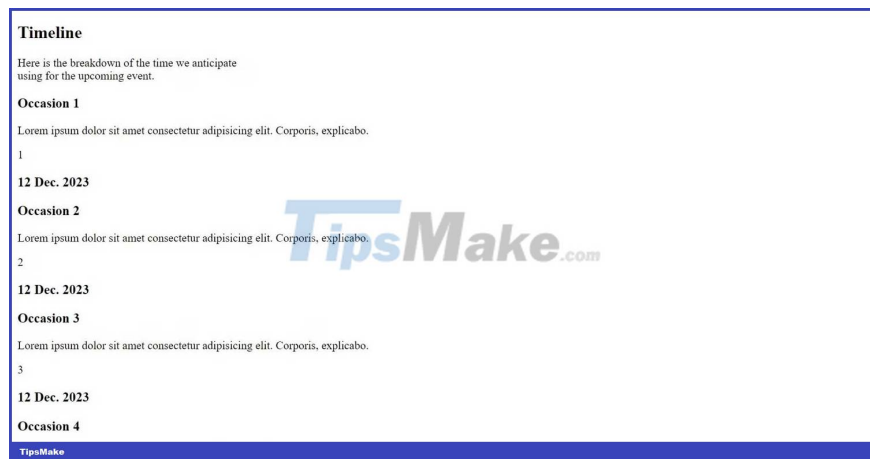
1. **Nodes:** Are circles, styled with the circle class, that act as nodes on the timeline. They are positioned absolutely in the center of each timeline item and act as intersections on the timeline. They are positioned absolutely at the center of each timeline item and are visually distinct from the background color, forming key points along the timeline.

```
.circle { position: absolute; background: var(--clr-purple); top: 50%; left: 50%;
```

**Date markers:** Dates, styled with the Timeline\_\_date class, appear on either side of the timeline. Their position alternates between left and right for each timeline item, creating a balanced, staggered image along the timeline.

```
.Timeline__text, .Timeline__date { width: 50%; } .Timeline__item:nth-child(even)
```

Once styled, your component will look like this:



## Create animations with JavaScript

To animate this element, use the Intersection Observer API to animate timeline items as they scroll. Add the following code to script.js.

### 1. Initial setup

First, select the entire element using the Timeline\_item class.

```
const timelineItems = document.querySelectorAll(".Timeline__item");
```

### 2. Initial styling of timeline items

Set each item's initial opacity to 0 (invisible) and apply a CSS transition to create a smooth blur effect.

```
timelineItems.forEach((item) => { item.style.opacity = 0; item.style.transition =
```

### 3. Callback Intersection Observer

Define a function `fadeInOnScroll` to change the opacity of items to 1 (visible) when they intersect the viewport.

```
const fadeInOnScroll = (entries, observer) => { entries.forEach((entry) => { if
```

#### 4. Intersection Observer options

Set choices for the observer, with a threshold of 0.1 indicating that the animation will trigger when 10% of the items are visible.

```
const options = { root: null, rootMargin: "0px", threshold: 0.1, }
```

#### 5. Create and use Intersection Observer

Finish by creating an `IntersectionObserver` with these options and applying it to each timeline item.

```
const observer = new IntersectionObserver(fadeInOnScroll, options); timelineItems
```

Building an interactive timeline not only presents information but also creates a more engaging experience for the web. By combining HTML structure, CSS styling, and JavaScript animations, you can do it with ease. Good luck!

You finished reading the article "**How to create an interactive timeline using CSS and JavaScript**" edited by the [TipsMake](#) team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.