Interactive Fiction with Twine

Documentation: https://twinery.org/wiki/twine2:guide

Getting Started

Twine stories can be found on https://ifdb.tads.org/. A good short story I recommend for trying branching text-based fiction and understanding the capabilities of the Twine system is Tangaroa Deep, available at https://astriddalmady.com/deep.html. When ready to start your own story, follow these directions:

- Go to the website: **twinery.org**.
- Click the link in the top right corner to **use it online**.
- You can either read or **skip** the online documentation.
- Click the green button to **Start a new story**.
- **Name the story**.

There are 3 different story formats. Branching links are created the same way in each format. However, each format has a different syntax for writing scripts to program more complex behaviors like stats, inventory, and random events. Students learning JavaScript should change the story format to Snowman, which provides full support for the JavaScript language.

The rest of the directions in this document assume you are using the Snowman format.

- Click your **story name** in the lower left corner of the browser.
- Click **Change story format**.
- **Select Snowman 2**.

Passages

The content of each story node is stored in a passage.

Double-click the **Untitled Passage** box in the Twine editor to begin.

Node titles are used to identify nodes and generate links. Rename the node from Untitled Passage to something you will remember later.

Tags are optional features that can be used to target specific passages in CSS and JavaScript code.

Passage text is entered in the textarea box. This text is displayed to the reader.

Links

Links are created using double square bracket syntax.

- Start with two opening square brackets – [[
- Write the link text shown to the readers
- Type the separator character – |
- Write the title of the node for the link target passage
- End with two closing square brackets –]]

```
[[Give Me a Mullet!|mullet]]

[[I want the Mutton Chops!|mutton chops]]
```

In those examples, the reader will see two links.

If the user clicks the “Give Me a Mullet!” link text, they will be redirected to the passage with the “mullet” node title.

If the user clicks the “I want the Mutton Chops!” link text, they will be redirected to the passage with the “mullet” node title.

Tips:

- Choose node titles that clearly explain the target content, like “mullet.”
- Editing your story later will be harder if you choose node title codes that aren’t semantic, like “node1.”
- Node titles are case sensitive, so using a standard pattern like all lowercase letters is recommended.

HTML in Passages

The Snowman format supports HTML elements in passages.

While you can simply type plain text, if you know HTML you can take advantage of those tags for more advanced formatting.

```
<h1>Important Fashion Decision</h1>

<p>Make your choice:</p>

[[Give Me a Mullet!|mullet]]

[[I want the Mutton Chops!|mutton chops]]
```

Images and videos must be hosted on a web server. Use absolute links to add them to your passage.

```
<img src="http://david.emerson.build/image/mullet.png" alt="mullet guy">

<iframe width="560" height="315" src="https://www.youtube.com/embed/rSjW-
_xUuNE" frameborder="0" allow="accelerometer; autoplay; encrypted-media;
gyroscope; picture-in-picture" allowfullscreen></iframe>
```

Snowman HTML Template

The following shows you the finished HTML generated by the Twine Snowman system. The online editor only allows you to change the contents of the tw-passage custom element. **Do not type any of the generated HTML into the passage boxes.**

```
<html>
   <head>
      <title>Story Name</title>
      <meta charset="utf-8">
      <style>
         <!-- Snowman Styles -->
      </style>
   </head>

   <body>
      <tw-story>
         <tw-passage class="passage" aria-live="polite">
            Your Passage Text is Inserted Here
         </tw-passage>
      </tw-story>

      <tw-storydata>
         <!-- Twine 2 data -->
      </tw-storydata>

      <script>
         /* Snowman code* /
      </script>
   </body>
</html>
```

Stylesheet

You can edit the Story Stylesheet from the menu in the lower left corner to add custom CSS.

Here is a sample stylesheet. Get your own color and font codes from sites like Paletton and Google Fonts.

```
@import url('https://fonts.googleapis.com/css2?family=Oswald&display=swap');
@import url('https://fonts.googleapis.com/css2?family=Lato&display=swap');
/* copy and paste font embed code from the @import tab in google fonts */

html {
   font-size: 100%;
}

body {
   font-family: 'Lato', sans-serif;
   line-height: 1.4;
   background-color: #333333;
   color: #CCCCCC;
}

h1 {
   font-family: 'Oswald', sans-serif;
}

a, a:link {
   color: #F590CB;
   text-decoration-color: #F590CB;
}

a:visited {
   color: #98597E;
   text-decoration-color: #98597E;
}

.passage {
   /* rules that only apply to passage content goes here */
}
```

Stylesheet: Centering

The following CSS rules can be used to center headings, images, and videos on the page.

```
h1 {
   /* code to center images */
   text-align: center;
}

img {
  /* code to center images */
  margin: auto;
  display: block;
}

iframe {
  /* code to center videos */
  margin: auto;
  display: block;
}
```

Stylesheet: Classes

You can add classes to your HTML, just like in regular webpages, and select the classes in the stylesheet.

```
<p class="lead">Lead Paragraph</p>     ← Passage HTML
<p>Regular Paragraph</p>

...

.lead {                                ← Story Stylesheet
   font-size: 140%;
   font-weight: bold;
   color: cornflowerblue;
   margin-top: 3em;
}
```

Stylesheet: Style Elements and Attributes

Other ways of adding CSS rules directly in the passage using HTML will also work.

```
<style> ... </style>

<p style="...">
```

Scripts

The Twine Snowman format allows JavaScript to be included in passages.

Documentation can be found here: https://videlais.github.io/snowman/2/

Scripts can be placed inside <% and %> tags. You can write any sort of script using variables, arrays, and JavaScript functions. Snowman also has additional built-in properties and functions for your use. The documentation includes code samples for adding features like the following:

- Selecting a random value from an array
- Checking if a passage node was visited by the reader
- Dynamically replacing the content of a story node
- Loading external stylesheets into the passage
- Conditional displaying of passage content and links
- Displaying of current date and time
- Random dice rolling
- Keypress event handling
- Creating rooms with locked doors and keys
- Player statistics tracking
- Timed passages
- Counting total number of passages visited
- Displaying text with a typewriter effect.

Scripts – Custom Page Backgrounds

You can only add custom classes to HTML is passages, which means you can't change the style of the body element. But you need to change the body style if you want different background colors or images for different passages.

I wrote a JavaScript solution that copies the passage tags to the body element class list.

Step 1: Create tags on the passages where you want a customized body style.

Step 2: Copy the code on the next page into the Story JavaScript from the lower left corner menu.

Step 3: Add CSS rules to the Story Stylesheet using class names with the same names as the passage tags.

```
<%                                          ← Passage HTML
setup.tagsToClasses();
%>

...

// Use or create window.setup               ← Story JavaScript
window.setup = window.setup || {};

// Create global "Tags to Classes" function
window.setup.tagsToClasses = function() {
      let tag;

      // remove existing classes from the body tags
      let classList = document.body.classList;
      while (classList.length > 0) {
            classList.remove(classList.item(0));
      }

      // copy the passage's tags to the body element class list
      for (tag of window.passage.tags) {
            document.body.classList.add(tag);
      }
}

...

.intro {                                    ← Story Stylesheet
  background-color: #333333;
  color: #CCCCCC;
}
```