Using JavaScript with Twine

Cool effects to polish your interactive story!

What is JavaScript?

- A 19-year-old programming language that is mainly used on the web.
- Allows dynamic interaction and effects to happen based on conditions and events.

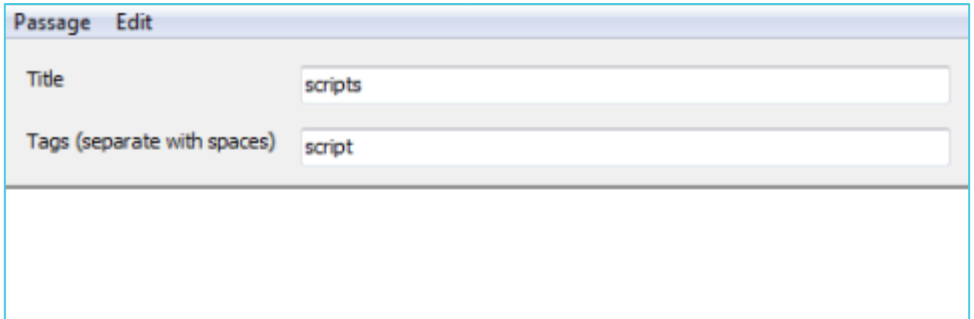
What is jQuery?

- A JavaScript framework, or set of pre-made tools and functions.
- jQuery makes creating animation, interaction effects, etc. using JavaScript easier.

Using jQuery in Twine

Step 1:

Create a new passage and use the **script** tag to include custom Javascript in your story.



Using jQuery in Twine

Step 2:

In the passage area, write:

```
//requires jquery
```

Basic concepts of JavaScript

JavaScript Variables

Useful for storing data that may change or be referenced throughout the course of your game.

For example, best friends may change but the label stays the same:

```
var myBestFriend = “Isaiah”;
var myBestFriend = “Rebecca”;
```

JavaScript Functions

- A group of code that performs a specific task.

```
var fetch = function (object) {
    run to the object;
    pick up the object;
    bring back the object;
};

fetch(ball);
```

Conditional Statements

- Perform a task if something is true or false.

```
var beAnAdult = function (day) {
    if (day != “Saturday” || “Sunday”) {
        go to work;
    } else {
        party;
    }
};
```

Twine Functions & Conditional Statements

There are many native functions in Twine. Check out twinery.org/wiki/function if you want to include them in your game.

You can read about Twine's conditional statements at http://twinery.org/wiki/if

Handling Player Interaction

Event handlers

Runs a function when an interaction (click, hover, etc.) has happened.

Suggestion!

JavaScript is best for dynamic effects. Use CSS to style things before the page loads.

```
.passage a {
   display:none;
}
```

Click events

Use the **click** event to trigger code when an object is clicked once.

```
$("div").click(function() {
  console.log("You clicked me!");
});
```

Double click events

Use the **dblclick** event to trigger code when an object is clicked twice.

```
$("div").dblclick(function() {
  console.log("You double clicked me!");
});
```

Mouseover events

Use the **mouseover** event to trigger code when an object is moused over.

```
$("div").mouseover(function() {
  console.log("You moused over me!");
});
```

Mouseout events

Use the **mouseout** event to trigger code when an object is no longer being moused over.

```
$("div").mouseout(function() {
  console.log("You moused off of me!");
});
```

Hover events

Use the **hover** event to trigger code when an object is moused over AND out.

```
$("div").hover(function() {
  console.log("You hovered over me!");
});
```

Creating effects with jQuery

Finding HTML objects

Use the **find** function to locate & modify HTML elements, classes, or IDs.

```
$("div").click(function() {
  $(".passage").find(".body").css("background", "red");
});
```

Appending text and HTML

Use the **append** function to add text and/or HTML elements to the bottom of the selected object.

```
$("div").click(function() {
  $(".passage").append("hey!");
});
```

Appending text and HTML

Use the **before** function to add text and/or HTML elements above the selected object.

```
$("div").click(function() {
  $(".passage").before("hey!");
});
```

Modifying text

Use the **text** function to change the text value of HTML elements.

```
$("div").click(function() {
    $(".passage").find(".body").text("hey!");
});
```

Modifying HTML

Use the **html** function to change the contents of HTML elements.

```
$("div").click(function() {
  $(".passage").find(".body").html("<p>hello!</p>");
});
```

Adding CSS classes

Use the **addClass** function to add a class to an HTML element.

```
$("div").click(function() {
  $(".passage").find(".body").addClass("myClass");
});
```

Removing CSS classes

Use the **removeClass** function to remove a class from an HTML element.

```
$("div").click(function() {
  $(".passage").find(".body").removeClass("myClass");
});
```

Toggling CSS classes

Use the **toggleClass** function to toggle a class.

```
$("div").click(function() {
  $(".passage").find(".body").toggleClass("myClass");
});
```

Hiding elements

Use the **hide** function to hide HTML elements.

```
$("div").click(function() {
  $(".passage").find("a").hide();
});
```

Showing elements

Use the **show** function to show hidden HTML elements.

```
$("div").click(function() {
  $(".passage").find("a").show();
});
```

Toggling elements

Use the **toggle** function to switch between showing and hiding HTML elements.

```
$("div").click(function() {
  $(".passage").find("a").toggle();
});
```

Fading elements

Use the **fadeOut** function to hide HTML elements with a fade.

```
$("div").click(function() {
  $(".passage").find("a").fadeOut();
});
```

Fading elements

Use the **fadeIn** function to show HTML elements with a fade.

```
$("div").click(function() {
  $(".passage").find("a").fadeIn();
});
```

Fading elements

Use the **fadeToggle** function to toggle HTML elements with a fade.

```
$("div").click(function() {
  $(".passage").find("a").fadeToggle();
});
```

Sliding elements

Use the **slideUp** function to hide HTML elements by sliding up.

```
$("div").click(function() {
  $(".passage").find("a").slideUp();
});
```

Sliding elements

Use the **slideDown** function to show HTML elements by sliding down.

```
$("div").click(function() {
  $(".passage").find("a").slideDown();
});
```

Sliding elements

Use the **slideToggle** function to toggle HTML elements with a sliding effect.

```
$("div").click(function() {
  $(".passage").find("a").slideToggle();
});
```

More Twine JavaScript Concepts

JavaScript has all kinds of cool uses.

Prerender

Manipulate objects on the page before the page has been rendered.

```
prerender.titleLog = function() {
   console.log(this.title);
};
```

Postrender

Manipulate objects on the page after the page has been rendered.

```
postrender.hello = function() {
   alert("hello!");
};
```

Macros

Create custom macros you can use throughout your Twine game.

```
macros[“hello”] = {
    handler: function() {
        alert(“hello, world!”);
    }
}
```

Other Programming Terminology

JavaScript has all kinds of cool uses.

Object Variables

Stores sets of data in one variable.

```
var Catt = {
   height: 164,
   age: 24,
   occupation: “Product Designer”
}
```

Object Variables

Uses **dot notation** to reference and/or define properties.

```
Catt.hairColor = “dark brown”;
Catt.hasPets = true;
```

Array Variables

Stores sets of data in numbered list form.

```
var inventory = [ “sword”, “potion” ];
```

Access and modify array items with **brackets**.

```
inventory[2] = “crescent moon wand”;
```

Challenge for the week!

Finish your Twine game.

Thanks! Questions?

@cattsmall
catt@codeliberation.org