Tempo 1.1

Tempo is a tiny JSON rendering engine that enables you to craft data templates in pure HTML.

- Download
- Twitter Example
- Solr Example
- View on GitHub

Why use Tempo?

- Clear separation of concerns: no HTML in your JavaScript files
- It makes AJAX content easier to work with
- It's as friendly as a bunny rabbit (which are quite friendly)

Key Features

- Works in Safari, Chrome, FireFox, Opera, and Internet Explorer 6+
- Itty-bitty footprint, lightning-fast rendering!
- No dependencies - Works with or without jQuery
- Degrades gracefully if JavaScript is not enabled
- Supports nested and conditional templates

1. Include the Tempo script

```
<script src="js/tempo.js" type="text/javascript"></script>

<script>Tempo.prepare("tweets").render(data);</script>
```

2. Compose the data template inline in HTML

```
<ol id="tweets">
        <li data-template>
                <img src="{{profile_image_url}}" />
                <h3>{{from_user}}</h3>
                <p>{{text}}</p>
        </li>
</ol>
```

3. Booyah!

VisitorsCenter

Groucho Marx once said "Humor is reason gone mad." If you are looking for a laugh, "Groucho" A Life in Revue,... http://fb.me/UiD1h2lA

GiftedPreacher

RT @ronkarr: Only one man in a thousand is a leader of men -- the other 999 follow women.-Groucho Marx #leadership

Usage

- JSON
- JavaScript
- HTML
- CSS
- jQuery
- Advanced Topics

JSON

Tempo takes information encoded as JSON and renders it according to an HTML template. Below is a sample array of JSON data.

```
var data = [
        {'name':{'first':'Leonard','last':'Marx'},'nickname':'Chico','born':'March 21, 1887','actor': true,'solo_endeavours':[{'title':'Pa
        {'name':{'first':'Adolph','last':'Marx'},'nickname':'Harpo','born':'November 23, 1888','actor':true,'solo_endeavours':[{'title':'T
        {'name':{'first':'Julius Henry','last':'Marx'},'nickname':'Groucho','born': 'October 2, 1890','actor':true,'solo_endeavours':[{'ti
        {'name':{'first':'Milton','last':'Marx'},'nickname':'Gummo','born':'October 23, 1892'},
        {'name':{'first':'Herbert','last':'Marx'},'nickname':'Zeppo','born':'February 25, 1901','actor':true,'solo_endeavours':[{'title':'
];
```

JavaScript

Include script

You only need to include a single, 4kb script.

```
<script src="js/tempo.js" type="text/javascript"></script>
```

Tempo.prepare(element)

To initialize Tempo, run the `prepare()` function to scan an HTML container for data templates, cache them in memory, and remove the data template HTML elements from the page. `Tempo.prepare(element)` returns an instance of a renderer that knows how to layout the data you provide to it.

element

The **ID** of the HTML element (or the element itself) containing your data template. If you're using jQuery, you may pass in a jQuery object instead.

If the container does not contain a default (that is without conditions and not nested) `data-template` the entire contents of the container will be considered to represent the template.

renderer.render()

The Tempo.prepare() function returns an instance of a renderer. Once the JSON data is available, run the `render(data)` function to add the data to the page.

data

The JSON **data** to be rendered. You'll first need to perform an AJAX call to the JSON data source (see below).

```
Tempo.prepare( element ).render( data );
```

```
Tempo.prepare('marx-brothers').render(data);
```

HTML

data-template

Any tag with the `data-template` attribute will be flagged as a data template.

{{fields}}

Any field represented in the JSON data may be retrieved by referencing the field name inside double brackets.

```
<ol id="marx-brothers">
        <li data-template>{{nickname}} {{name.last}}</li>
</ol>
```

The above example would be rendered as:

1. Chico Marx
2. Harpo Marx
3. Groucho Marx
4. Gummo Marx
5. Zeppo Marx

If the JSON data represents an array of arrays (which can not be referenced by field/member name) for example:

var data = [['Leonard','Marx'], ['Adolph','Marx'], ['Julius Henry','Marx'], ['Milton','Marx'], ['Herbert','Marx']];

You can reference array elements with the following notation:

```
<ol id="marx-brothers">
        <li data-template>{{[0]}} {{[1]}}</li>
</ol>
```

The above example would be rendered as:

1. Leonard Marx
2. Adolph Marx
3. Julius Henry Marx
4. Milton Marx
5. Herbert Marx

Nested data-templates

Data templates can even be nested within other data templates. Multiple nested templates are supported.

```
<li data-template>
    {{nickname}} {{name.last}}
    <ul>
        <li data-template="solo_endeavours">{{title}}</li>
    </ul>
</li>
```

1. Chico Marx
    - Papa Romani
2. Harpo Marx
    - Too Many Kisses

- Stage Door Canteen

3. Groucho Marx
    - Copacabana
    - Mr. Music
    - Double Dynamite
4. Gummo Marx
5. Zeppo Marx
    - A Kiss in the Dark

Conditional Templates

Tempo provides boolean and value-based conditionals, as well as the ability to define multiple data templates per container (the first matching template wins).

```
<ul id="marx-brothers3">
        <li data-template data-if-nickname="Groucho"">{{nickname}} (aka {{name.first}}) was grumpy!</li>
        <li data-template data-if-actor>{{name.first}}, nicknamed '<i>{{nickname}} {{name.last}}</i>' was born on {{born}}</li>

        <!-- Default template -->
        <li data-template>{{name.first}} {{name.last}} was not in any movies!</li>
</ul>
```

- Leonard, nicknamed '*Chico Marx*' was born on March 21, 1887
- Adolph, nicknamed '*Harpo Marx*' was born on November 23, 1888
- Groucho (aka Julius Henry) was grumpy!
- Milton Marx was not in any movies!
- Herbert, nicknamed '*Zeppo Marx*' was born on February 25, 1901

Template Tags

Tempo also supports tag blocks.

```
{{if javascript-expression}} ... {{endif}}
```

The body of the tag will only be rendered if the JavaScript expression evaluates to true.

CSS

To ensure that your data template is not visible before being rendered (either because of JavaScript being disabled or due to latency retrieving the data), it's best practice to hide your data templates with CSS. If you add an inline rule of `style="display: none;"` Tempo will simply remove the inline rule once the data has been rendered.

Coming soon: Tempo 1.2 will support `data-template-fallback` which will be shown if JavaScript is not enabled.

```
<ol id="tweets">
        <li data-template style="display: none;">
                ...
        </li>
</ol>
```

Using Tempo with jQuery

jQuery's getJSON() method provides an easy means of loading JSON-encoded data from the server using a GET HTTP request.

```
var twitter = Tempo.prepare('tweets');
$.getJSON("http://search.twitter.com/search.json?q='marx brothers'&callback=?", function(data) {
        twitter.render(data.results);
});
```

Advanced Topics

renderer.notify(eventlistener)

After preparing a template you can register an event listener by providing a callback function to be notified of events in the lifecycle. The event listener will be called with a single argument, a `TempoEvent` which has the following properties:

type

The **type** of the event. Constant values are defined in `TempoEvent.Types`.

- `TempoEvent.Types.RENDER_STARTING`: Indicates that rendering has started, or been manually triggered by calling `starting()` on the renderer object.
- `TempoEvent.Types.ITEM_RENDER_STARTING`: Indicates that the rendering of a given individual item is starting.
- `TempoEvent.Types.ITEM_RENDER_COMPLETE`: Indicates that the rendering of a given individual item has completed.
- `TempoEvent.Types.RENDER_COMPLETE`: Indicates that the rendering of all items is completed.

item

The **item** being rendered. This is only available for the `ITEM_RENDER_STARTING` and `ITEM_RENDER_COMPLETE` events.

element

The HTML **element** or template being used for rendering. This is only available for the `ITEM_RENDER_STARTING` and `ITEM_RENDER_COMPLETE` events.

starting()

In some cases you prepare the templates ahead of calling render. In those cases if you would like to e.g. set loader graphics, call the renderer's `starting()` method just prior to issuing e.g. a jQuery request. This will fire the `ITEM_RENDER_STARTING` event.

The following example demonstrates use of both methods above. In this case we prepare the templates, and register our event handler function. The event handler is then notified that the jQuery request is about to be issued (when we manually fire `RENDER_STARTING` with a call to `starting()`) adding a CSS class to the container. We are then notified that rendering is complete so we can remove the CSS class.

```
var twitter = Tempo.prepare('tweets').notify( function (event) {
        if (event.type === TempoEvent.Types.RENDER_STARTING || event.type === TempoEvent.Types.RENDER_COMPLETE) {
                $('#tweets').toggleClass('loading');
        }
});
twitter.starting();
$.getJSON(url, function(data) {
        twitter.render(data.results);
});
```

- Download
- Twitter Example
- Solr Example
- View on GitHub

Brought to you by the friendly guys at TwigKit. Follow Stefan Olafsson and Tyler Tate on Twitter.