**Data Tokens**

**Definition**: any element within curly brackets is considered a token.

<%= tr("Hello World”)%>

**Number Based Tokens**

**Definition**: you can see/modify this definition in the **config/tr8n/config.json** file

**"numeric\_token"**: **{**

**"suffixes"**: **[**"count", "num", "age", "hours", "minutes", "years", "seconds"**]**,

**"method"**: "to\_i"

**}**

Number based sentence with a hidden token:

<%= tr("You have {count} {\_messages}", nil, **:count** => i, **:\_messages** => "message".pluralize\_for(i))%>

**Gender Based Tokens**

**Definition**: you can see/modify this definition in the **config/tr8n/config.json** file

**"gender\_token"**: **{**

**"suffixes"**: **[**"user", "profile", "actor", "target", "partner", "parent", "child", "sibling"**]**,

**"method"**: "gender",

**"values"**: **{"unknown"**: "u", **"neutral"**: "n", **"female"**: "f", **"male"**: "m"**}**

**}**

Simple token:

<%= tr("Dear {user}", nil, **:user** => current\_user) %>

Token with embedded method calls:

<%= tr("Dear {user.first\_name} {user.last\_name}", nil, **:user** => current\_user) %>

Token with decorative substitution:

<%= tr("Dear {user}", nil, **:user** => [current\_user, display\_profile(current\_user)]) %>

Token with decorative substitution using symbol method call:

<%= tr("Dear {user}", nil, **:user** => [current\_user, **:first\_name**]) %>

Token with decorative substitution using symbol method call with parameters:

<%= tr("Dear {user}", nil, **:user** => [current\_user, **:some\_method**, "value"]) %>

Token with decorative substitution using lambda method call:

<%= tr("Dear {user}", nil, **:user** => [current\_user, lambda{|val| html\_for(val)}]) %>

Token with decorative substitution using lambda method call with parameters:

<%= tr("Dear {user}", nil, **:user** => [current\_user, lambda{|val, test| html\_for(val, test)}], "test"]) %>

Gender based sentence with a hidden token (deprecated):

<%= tr("{user} changed {\_his\_her} name", nil, **:user** => [current\_user, :name], **:\_his\_her** => current\_user.his\_her) %>

**Tr8n Cheat Sheet**

**Decoration Tokens**

**Definition**: you can add/modify decorations in the **config/tr8n/default\_lambdas.json** file

**{**

**"bold"**: "<strong>{$0}</strong>",

**"italic"**: "<i>{$0}</i>",

**"link"**: "<a href='{$1}' style='{$2}'>{$0}</a>"

**}**

*Note: $0 parameter is always the translated value of the lambda*

Bold decoration with default docarator:

<%= tr("[bold: Hello World]”)%>

Link with style using default decorator:

<%= tr("[link: Hello World]”, nil, **:link** => [“http://www.google.com”, “text-decoration:none”])%>

Bold decoration with custom string substitution:

<%= tr("[bold: Hello World]”, nil, **:bold** => "<strong>{$0}</strong>")%>

Link with gender dependent tokens and lambda decoration:

<%= tr("{user} updated [link: {\_his\_her} profile]", nil,

**:user** => [current\_user, display\_profile(current\_user)],

**:\_his\_her** => current\_user.his\_her,

**:link** => lambda{|value| display\_profile(value)}

) %>

Link with number dependent tokens and lambda decoration:

<%= tr("You have [link: {count} {\_messages}]", nil,

**:count** => i,

**:\_messages** => "message".pluralize\_for(i),

**:link** => lambda{|value| link\_to(value, “/inbox”)}

) %>

Sentence with gender dependent and number dependent tokens and a link decoration:

<%= tr("{user} added {user1}, {user2}, {user3} and [link: {count} other family {\_members}] to {\_his\_her} tree", nil,

**:user** => [actor, display\_profile(actor)],

**:user1** => display\_profile(user1),

**:user2** => display\_profile(user2),

**:user3** => display\_profile(user3),

**:count** => i,

**:\_members** => "member".pluralize\_for(i),

**:\_his\_her** => actor.his\_her,

**:link** => lambda{|value| link\_to(value, “/tree”)}

) %>

Note: this sentence is an example of a complicated structure that would be translated into 9 variations in Russian language:

3 gender variation on {user} for “added” and “{\_his\_her}”

3 numeric variations on {count} for “{\_members}”

total: 3 \* 3 = 9 combintations