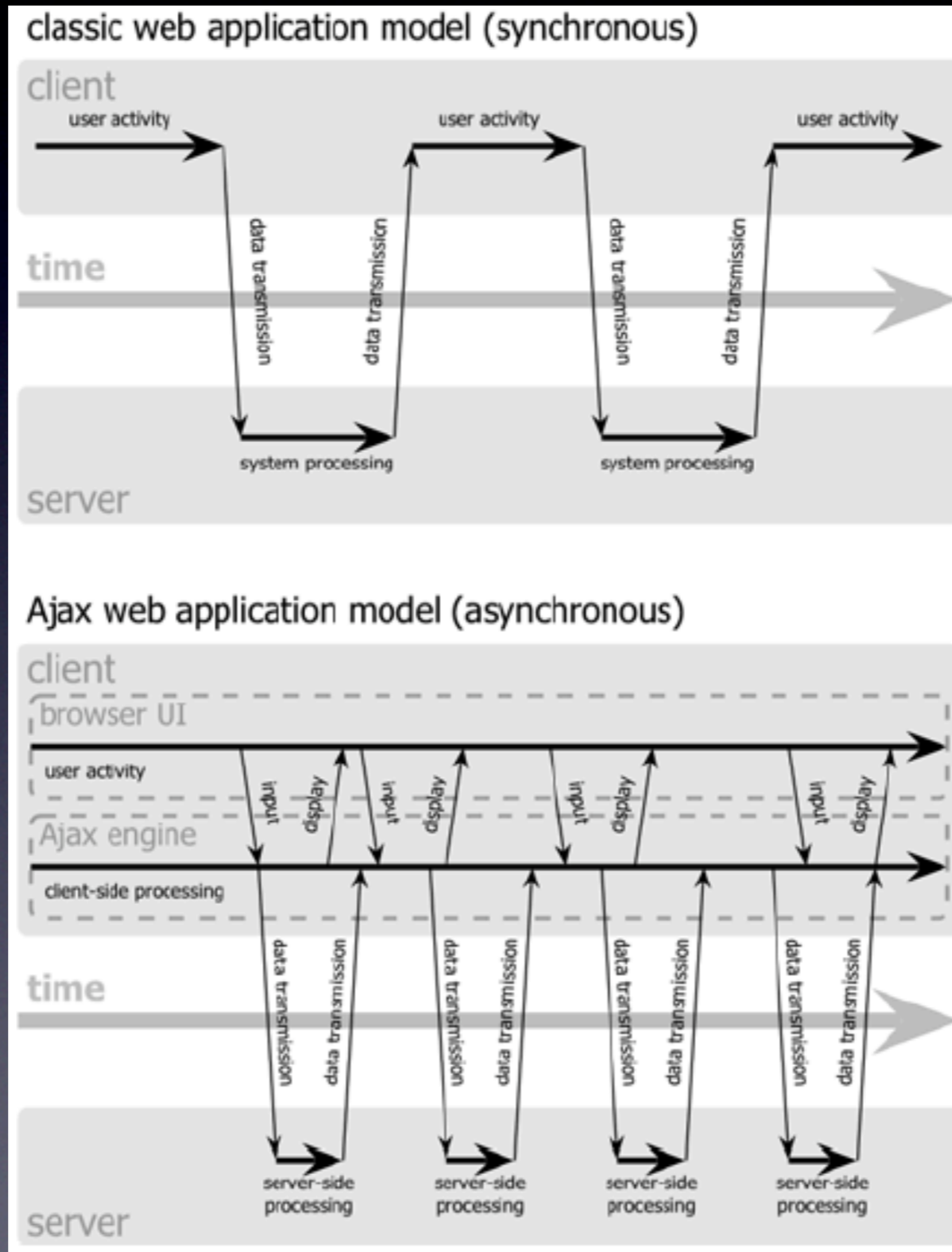


AJAX on Rails

AJAX

Asynchronous JavaScript and XML

Web aplikāciju modeļi



Pielietošanas piemēri

- Pievienot jaunu ierakstu un pievienot attēlotajam sarakstam
- “in-place” formu rediģēšana
- teksta lauku “autocompletion”
- “drag-and-drop” sakārtojami saraksti
- “dzīvā” meklēšana

Rails un AJAX

- Prototype bibliotēka
AJAX pieprasījumu veidošana un
DOM manipulēšana
- script.aculo.us bibliotēka
vizuālie efekti, autocompletion,
drag-and-drop
- RJS šabloni
JavaScript ģenerēšana ar Ruby

PrototypeHelper piemēri

```
link_to_remote(name, options = {}, html_option = {})  
periodically_call_remote(options = {})
```

```
form_remote_tag(options = {}, &block)  
remote_form_for(object_name, *args, &proc)  
submit_to_remote(name, value, options = {})
```

```
remote_function(options)
```

```
observe_field(field_id, options = {})  
observe_form(form_id, options = {})
```

ScriptaculousHelper piemēri

```
visual_effect(name, element_id = false, js_options = {})  
sortable_element(element_id, options = {})  
draggable_element(element_id, options = {})  
drop_receiving_element(element_id, options = {})
```

AJAX saišu un formu piemēri

```
link_to_remote("Destroy", :url => {:action => 'destroy', :id => item},  
  :confirm => "Are you sure?")
```

```
link_to_function "Cancel", "$('create_form').hide();"
```

```
link_to_function "Cancel" do |page|  
  page[object.dom_id("rate_link")].show  
  page[object.dom_id("rate")].hide  
end
```

```
<% form_remote_tag :url => {:action => 'update'} do %>  
  <%= hidden_field_tag "prompt[id]", @prompt.id %>  
  <%= render :partial => 'form', :locals => {:mode => 'edit'} %>  
  <%= submit_tag "Edit" %>  
<% end %>
```

RJS

- Rails API, lai ģenerētu JavaScript kodu, kas tiek nosūtīts uz pārlūku un tiek uz tā izpildīts
- Var izmantot gan render metodē norādot :update parametru, vai arī var veidot .rjs šablonus
- RJS ir īpaši noderīgs, ja vienlaicīgi jāmaina vairākus lapas elementus

Demo

RJS piemērs

```
<!-- In index.rhtml: -->  
<% form_remote_tag :url => { :action => :add_to_cart, :id => product } do %>  
<%= submit_tag "Add to Cart" %>  
<% end %>
```

```
# The action:
```

```
def add_to_cart  
  product = Product.find(params[:id])  
  @current_item = @cart.add_product(product)  
  redirect_to_index unless request.xhr?  
end
```

```
# The RJS template add_to_cart.rjs:
```

```
page.select("div#notice").each { |div| div.hide }  
page.replace_html("cart", :partial => "cart", :object => @cart)  
page[:cart].visual_effect :blind_down if @cart.total_items == 1  
page[:current_item].visual_effect :highlight,  
  :startcolor => "#88ff88",  
  :endcolor => "#114411"
```

```
# Position argument is one of :before, :top, :bottom, :after
page.insert_html :bottom 'todo_list', "<li>#{todo.name}</li>"
page.replace_html 'flash_notice', "Todo added: #{todo_name}"
page.replace 'flash_notice', :partial => 'flash', :object => todo
page[:flash_notice].remove|show|hide|toggle # page[:flash_notice] <=> $('flash_notice')
page.alert "The form contains the following errors: #{errors.join(", ")}"
page.redirect_to :controller => 'blog', :action => 'list'
page.assign 'cur_todo', todo.id # Assign a JavaScript variable
page.call 'show_todo', todo.id # Invoke a JavaScript method
page << "alert('Hello there')" # Append raw JavaScript to be executed

# Available effects: :fade, :appear, :blind_up/down, :slide_up/down, :highlight,
# :shake, :pulsate, :fold etc.
page.visual_effect :pulsate, 'flash_notice'

page.delay(3) do
  page.visual_effect :fade, 'flash_notice'
end

page.select('p.welcome b').first.hide

page.select('#items li').each do |value|
  value.hide
end
```

Formu nově rošána

```
<%= form_tag({:action => "search"}, {:id => 'search_form'}) %>
```

```
...
```

```
<%= observe_form('search_form',  
  :url => {:action => 'search_count'},  
  :frequency => 3,  
  :condition => (@model_name == 'Contact' ?  
    "!$('search_form').submitting && !contact_search_form_empty()" :  
    "!$('search_form').submitting && !outlet_search_form_empty()"),  
  :with => "search_form",  
  :loading => "$('spinner').show();",  
  :complete => "$('spinner').hide();") %>
```

```
<%= observe_field("show_hide_select",  
  :url => { :action => 'toggle_column', :item_count => item_count },  
  :with => "'column_name=' + value") %>
```

```
# This is a search form where we want to display a preview of the number
# of search hits
def search_count
  query = params[:search_form][/^q=(.*)/, 1]
  if form_has_errors?
    render :update do |page|
      page.alert(@form_errors.join("\n"))
    end and return
  end
  ...
  render :update do |page|
    page["search_count_preview"].show
    page["search_count_preview"].replace_html :partial =>
      '/mcm/search/search_count_preview'
    page.visual_effect :highlight, "search_count_preview"
    if @search_count > 0
      page.mcm.set_color("search_count_preview", "green")
    else
      page.mcm.set_color("search_count_preview", "red")
    end
  end
end
end
```

Drag & drop piemērs

```
# This example is about dragging books in list to a shopping cart in the menu bar.
# In index.rhtml
<li class="book" id="book_<%= book.id %>">
  ...book info here...
</li>
<%= draggable_element("book_#{book.id}", :revert => true) %>

# In application.rhtml
<div id="shopping_cart">
  <%= render :partial => "cart/cart" %>
</div>
<%= drop_receiving_element("shopping_cart", :url =>
  { :controller => "cart", :action => "add" }) %>
<% end %>
```

```
# The action
def add
  params[:id].gsub!(/book_/, "")
  @book = Book.find(params[:id])
  if request.xhr?
    @item = @cart.add(params[:id])
    flash.now[:cart_notice] = "Added <em>#{@item.book.title}</em>"
    render :action => "add_with_ajax"
  elsif request.post?
    @item = @cart.add(params[:id])
    flash[:cart_notice] = "Added <em>#{@item.book.title}</em>"
    redirect_to :controller => "catalog"
  else
    render
  end
end

# add_with_ajax.rjs:
page.replace_html "shopping_cart", :partial => "cart"
page.visual_effect :highlight, "cart_item_#{@item.book.id}", :duration => 3
page.visual_effect :fade, 'cart_notice', :duration => 3
```

Autocompletion

```
# In text field view:
<%= text_field 'user', 'favorite_language' %>
<div class="auto_complete" id="user_favorite_language_auto_complete"></div>
<%= auto_complete_field :user_favorite_language,
  :url=>{:action=>'autocomplete_favorite_language'}, :tokens => ',',
  :frequency => 0.5,
  :min_chars => 3 %>
```

```
# The action
```

```
def autocomplete_favorite_language
  re = Regexp.new("^#{params[:user][:favorite_language]}", "i")
  @languages= LANGUAGES.find_all do |l|
    l.match re
  end
  render :layout=>false
end
```

```
# The response view in autocomplete_favorite_language.rhtml
```

```
<ul class="autocomplete_list">
  <% @languages.each do |l| %>
    <li class="autocomplete_item"><%= l %></li>
  <% end %>
</ul>
```

In-place-edit

```
# In the view
<div id="<%= dom_id(:email, :div) %>"><%= @user.email %></div>
<%= in_place_editor dom_id(:email, :div),
  :url => {:action => "set_user_email", :id => @user} %>

# In the controller
class UsersController < ApplicationController
  in_place_edit_for :user, :email
  ...
end

# WARNINGS
# 1) in_place_edit_for does *not* use validation
# 2) in_place_editor quotes the value edited. TODO: details on this
# Options:
# :rows, :cols, :cancel_text, :save_text, :loading_text, :external_control,
# :load_text_url
```

Noderīgi rīki

- Firefox
 - Firebug
 - Web Developer Extension