Routing and Architecture

Browser Routing Design Patterns in JS

SoftUni Team

Technical Trainers

Software University

http://softuni.bg

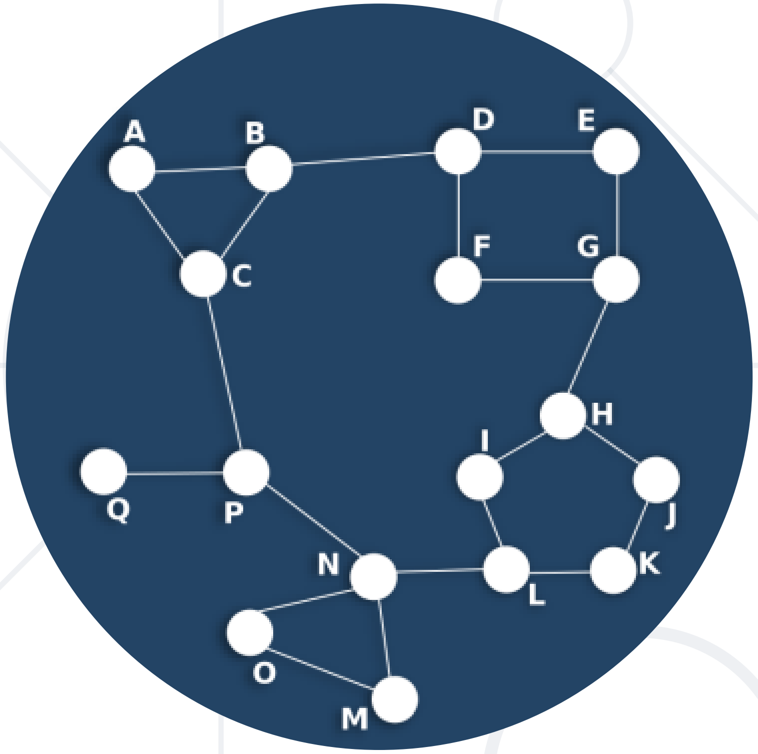
Table of Content

Have a Question?

SoftUni Foundation

[sli.do](sli.do)

#JS-CORE

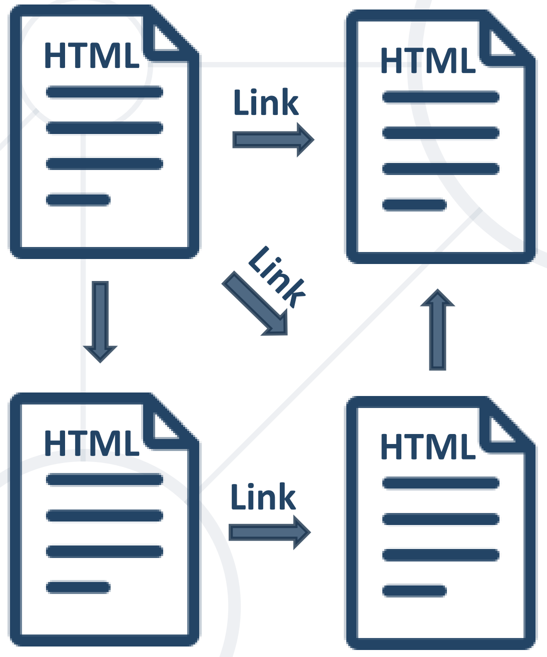


Routing Concepts

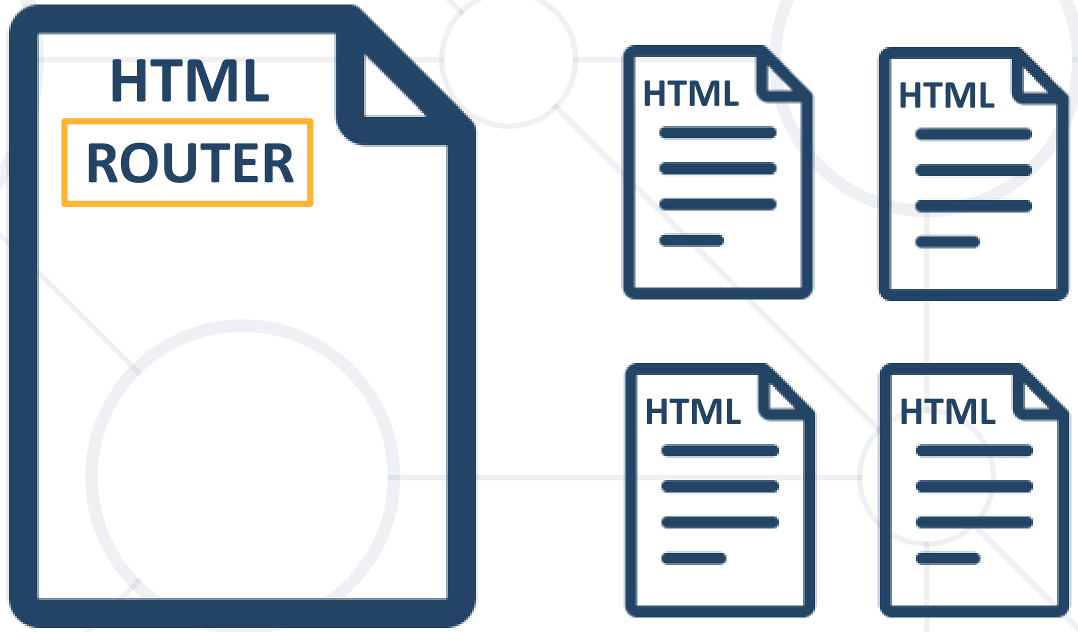
Navigation for Single Page Applications

What is Routing?

- Allows navigation, **without reloading** the page
- Pivotal element of writing **Single Page Applications**



Standard Navigation



Navigation using Routing

Single-page applications (SPA)

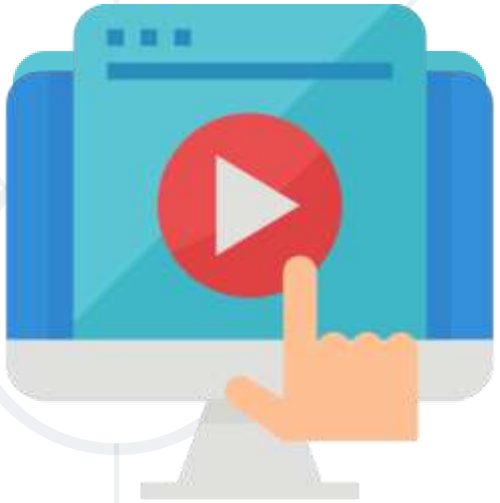
- **One web page** that you visit which then loads all other content using JavaScript
- Does **not require** page **reloading during use**
- Request just the peace you need
- Take **advantage** of the **repetition**
- Can use **state** from an **external source** or **track state internally**

How Routers Work

- A **Router** loads the appropriate content when the **location changes**
  - E.g. when the user manually **enters an address**
- Conversely, a change in content is reflected in the address bar
  - E.g. when the user **clicks on a link**
- Benefits:
  - Load all scripts **only once**
  - **Maintain state** across multiple pages
  - Browser **history** can be used
  - Build User Interfaces that **react quickly**

How Routers Work

- Hash-based routing
- Using the **#hash** part of the URL to simulate different content
- The routing is possible because changes in the hash **don't trigger page reload**



How Routers Work (2)

```
var url = null;
var getCurrent = function () {
  return window.location.hash;
};
var listen = function () {
  var current = getCurrent();
  if (current !== url) {
    console.log('URL changed to' + current);
    url = current;
  }
  setTimeout(listen, 200);
};
Listen();
```

The pushState() method

- **pushState()** takes three parameters: a **state object**, a **title** and a **URL**
  - **State** - object which is associated with the new history entry
  - **Title** - browsers currently ignore this parameter
  - **URL** - The new history entry's URL is given by this parameter

The replaceState() method

- **history.replaceState()** - **modifies the current history entry** instead of creating a new one
- It is useful when you want to update the **state object** or **URL**

```
var stateObj = { facNum: "56789123" };
history.pushState(stateObj, "", "student.html");

history.replaceState(stateObj, "", "newStudent.html");
```

The popstate event

- A **popstate** event is dispatched to the window every time the active history entry changes
- If the history entry being activated:
  - was created by a call to **pushState**
  - affected by a call to **replaceState**
- The **popstate** event's **state property** contains a copy of the history entry's state object

Live Demo

Routing with Sammy.js

Overview and Examples

Sammy.js Overview

- Sammy is a lightweight **routing library**
- Modular design with **plugins** and **adapters**
- **Requires** jQuery
- Many **additional features**

```
const app = Sammy('#main', function() {
  this.get('#/index.html', () => {
    this.swap('Index');
  })
});
```

Installation

- Download Sammy, by using WebStorm's terminal

```
npm install --save sammy
```

- Or download from sammyjs.org
- Browser builds will be located in:

```
node_modules/sammy/lib/
```

- It's best if your project has a `package.json` file

Application Initialization

- Create a Sammy instance to initialize your application

Invoke library

Element selector

```
const app = Sammy('#main', function () {
  // Define routes and other logic here
});

$(() => app.run()); // Activate
```

- You may have **multiple apps** running
- Each selector can only hold one app
  - If you refer to it again, it **extends the functionality**

Creating Routes

SoftUni Foundation

- The main **building block** of Sammy is the **route**
  - Defined by **method** and **address** (URI)
- Place this block inside a **Sammy initializer**:

Route **Method**

Route **address**

```
this.route('get', '#/about', function() {
  this.swap('<h2>Contact Page</h2>');
});
```

- A note on using **this**: it holds a **reference** to the **router object**, but may not work correctly in an **arrow function**

Route Aliases

- Each method has an **alias** for shorter code

```
this.get('#/catalog', loadBooks);
```

```
this.post('#/login', userLogin);
```

```
this.put('#/catalog/:bookId', updateBook);
```

```
this.del('#/catalog/:bookId', deleteBook);
```

URL Parameters

- Parameters allow for dynamic routes
  - E.g. products in a catalog will load the same page

Parameter name

Receive context

```
this.get('#/catalog/:productId', (context) => {
  console.log(context.params.productId);
});
```

Access passed in value

- You can get the whole path using **this.path**

Hello Sammy

index.html

```html
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <title>Hello Sammy</title>
  <!-- Include jQuery and Sammy distributions -->
</head>
<body>
  <header>
    <h1>Hello Sammy</h1>
    <a href="#/index.html">Home</a>
    <a href="#/about">About</a>
    <a href="#/contact">Contact</a>
  </header>
  <div id="main"></div>
</body>
</html>
```

Hello Sammy (2)

app.js

```
const app = Sammy('#main', function () {
    this.get('#/index.html', () => {
        this.swap('<h2>Home Page</h2>');
    });
    this.get('#/about', () => {
        this.swap('<h2>About Page</h2>');
    });
    this.get('#/contact', () => {
        this.swap('<h2>Contact Page</h2>');
    });
});

$(() => {
    app.run();
});
```

Handling Forms

Forms inside the main element are automatically handled

Route address

Route method

```
<form action="#/login" method="post">
  User: <input name="user" type="text">
  Pass: <input name="pass" type="password">
  <input type="submit" value="Login">
</form>
```

Names of inputs

```
this.post('#/login', (context) => {
  console.log(context.params.user);
  console.log(context.params.pass);
});
```

Integrating Handlebars

- Download and include the Handlebars source in your HTML
- Include **sammy.handlebars.js** (found under **lib/plugins**)
- Load the plugin inside a Sammy initializer:

```
this.use('Handlebars', 'hbs');
```

Template file **extension**

- Create a **RenderContext** with **render**, **load** or **partial**

Using Handlebars

greeting.hbs

```
<h1>{{title}}</h1>
<p>Hello, {{name}}!</p>
```

app.js

```
const app = Sammy('#main', function () {
  this.use('Handlebars', 'hbs');
  this.get('#/hello/:name', function() {
    this.title = 'Hello!'
    this.name = this.params.name;
    this.partial('greeting.hbs');
  });
});
$(() => app.run());
```

Load and swap in the template

Using Handlebars

- Load a list of **partial templates** (inside a **route** definition):

```
this.loadPartials({
  firstPartial: 'path-to/first.hbs',
  secondPartial: 'path-to/second.hbs',
  thirdPartial: 'path-to/third.hbs'
}).then(function(context) => {
  console.log(context.partials);
  this.partial('pageTemplate.hbs');
});
```

- The **callback** will be executed once all partials are loaded
- Templates are **cached** - there's no need to manually cache them

Additional Features

- Redirect

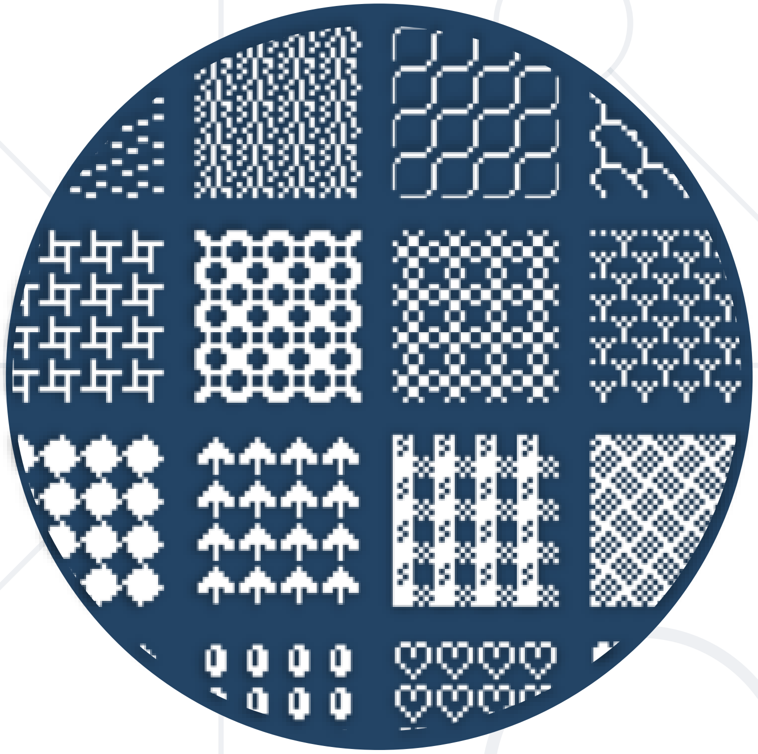
```
this.redirect('#/other/route');
this.redirect('#', 'other', 'route');
```

- Custom events

```
// Register event handler
this.bind('event-name', eventHandlerFunction);

// Raise event
this.trigger('event-name', data);
```

- Useful plugins (found under **lib/plugins**):
  - Storage and Session
  - OAuth2



Routing with Sammy.js

Overview and Examples

What are Design Patterns

- **Design Patterns** are general approaches to solving **commonly occurring** problems
- They provide:
  - Tested, **proven** programming paradigms
  - Guidelines to **organizing** our code
  - A **common vocabulary** between developers
- Using a pattern may **increase** complexity - misuse often creates more problems than it solves

Splitting Your Code

- Splitting your code aims to **separate concerns** (only change the parts that need to be changed)
- Sample code organization
  - Main script
  - Requester (Remote API)
  - Authenticator
  - Router
  - View Controllers

Live Exercise

Summary

- Browser **Routing** allows SPAs to use **history**
- **Sammy.js** is a simple routing library

```
const app = Sammy('#main',
function () {
  this.get('index.html', () => {
    this.swap('<h1>Index
Page</h1>');
  })
});
```

- **Modular code** is more maintainable

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

SoftUni Foundation

xssoftware

SBTech we know sports

telenor

SoftwareGroup doing it right

NETPEAK

SmartIT

Postbank Решения за твоето утре

SUPER HOSTING .BG

INDEAVR Serving the high achievers

INFRAGISTICS

LIEBHERR

æternity

SoftUni Organizational Partners

SoftUni Foundation

IS ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

Lukanet.com

WORLD OF MYTHS

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License