

Browser Routing

Navigation for Single-Page Applications



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

sli.do

#js-advanced

Table of Contents

1. **Routing** Concepts
2. Client-Side **Routing**
3. **Navigation** and **History**
4. Overview of **page.js**

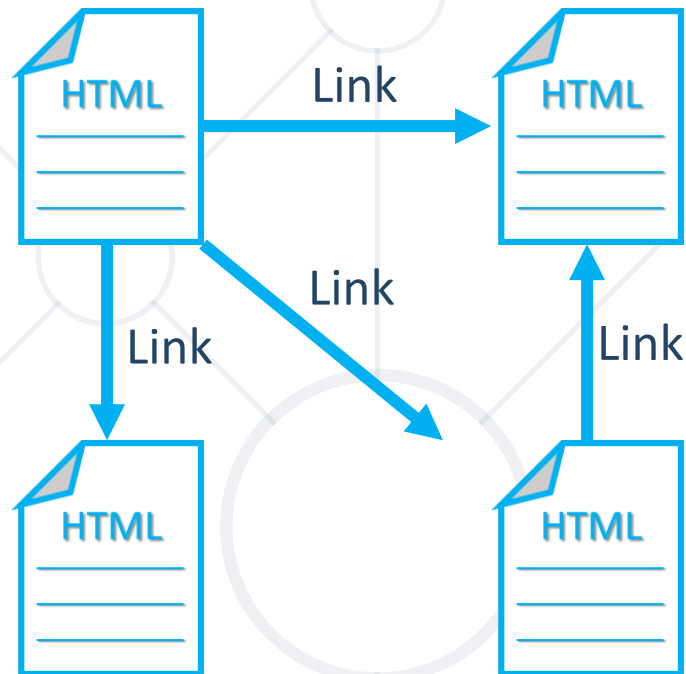




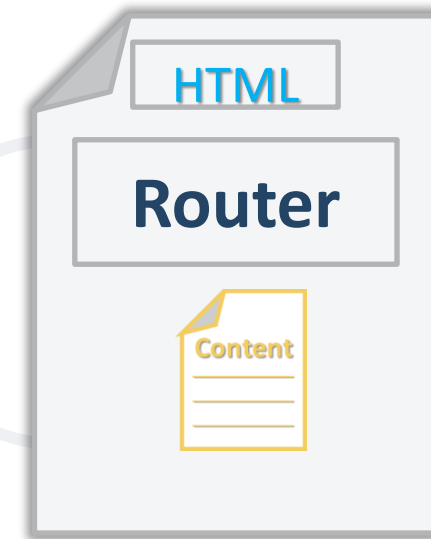
Client-Side Routing

Navigation for Single Page Apps

- Standard Navigation (Back-end routing)



- Client-Side Routing** – navigation **without reloading** the page



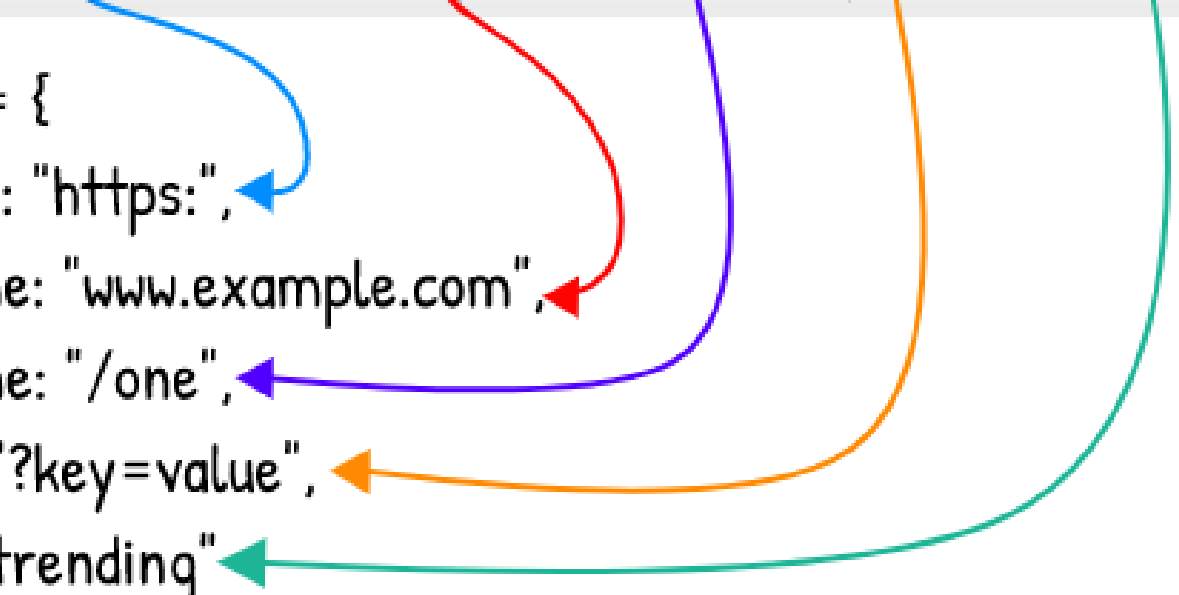
How Routers Work

- A **Router** signals the application 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**



← → https://www.example.com/one?key=value#trending

```
location = {  
  protocol: "https:",  
  hostname: "www.example.com",  
  pathname: "/one",  
  search: "?key=value",  
  hash: "#trending"  
}
```



- Allow for application **state** to be **serialized** into the URL
- Represented by a series of **key-value pairs**, separated by **&**
 - Example: `search=js+advanced&opCourses=true`
 - First parameter
 - Second parameter
- Common **use cases**
 - Representing the current page number in a paginated collection
 - Filter criteria
 - Sorting criteria

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**



- Extracting the hash from the entire URL

```
let hash = window.location.hash;
```

- Changing the path

```
let changePath = function (hash) {  
    window.location.hash = hash;  
}
```

Processing hashchange events

- Using an event handler:

```
window.onhashchange = funcRef;
```

- Using an HTML event handler:

```
<body onhashchange="funcRef();">
```

- Using an event listener:


```
window.addEventListener("hashchange", funcRef, false);
```

Push-Based Routing

- You can actually surface real **server-side data** to support things like SEO and Facebook Open Graph
- It helps with **analytics**
- It helps fix **hash tag issues**
- You can actually use hash tag for what is was meant for, **deep linking** to sections of long pages



History API


- 
- Provides access to the browser's history through the **history** object
 - **HTML5** introduced the **history.pushState()** and **history.replaceState()**
 - They allow you to add and modify **history entries**
 - These methods work in conjunction with the **popstate** event

The PushState() Method

- Adds new object to the history of the browser
- Takes three parameters:
 - **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
 - It must be of the **same origin** as the current URL



The ReplaceState() Method

- 
- **Modifies the current history entry** instead of creating a new one
 - It is particularly useful when you want to update the **state object** or **URL** of the current history entry

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

The Popstate Event

- Dispatched to the window every time the active history entry changes
- If the history entry being activated was created by a call to **pushState** or affected by a call to **replaceState**,
- The **popstate** event's **state property** contains a copy of the history entry's state object
- You can read the state of the current history entry without waiting for a **popstate** event using the **history.state property**



External Routing Library

Using **page.js** for Single-Page Routing

What is page.js?

- Compact **client-side router**
 - **Small size** – 1.2KB
- **Syntax** inspired by Express (back-end framework)
- Supports:
 - Automatic **link binding**
 - URL glob **matching**
 - **Parameters**
 - Plugins



- **Installation** via **npm** package:

```
npm install page
```

- Direct **import** from online **CDN** (no installation):

```
import page from "https://unpkg.com/page/page.mjs";
```

- Basic Usage:

```
page('/', index);      // Register home route
page('*', notfound);   // Register catch-all (404)
page.start();          // Activate router
```

- Documentation: <https://github.com/visionmedia/page.js>

- Routes are registered via **match pattern** and **callback**

```
page('/catalog', catalogView);
```

- Match pattern can be string, URL glob or RegExp
- The **route handler** (callback) will receive two parameters
 - **context** object with information about **parameters** and **state**
 - **next** callback, used when **chaining** route handlers

```
function catalogView(ctx, next) {  
    // fetch data, render template, handle form, etc.  
}
```

- URL **glob patterns** can match **dynamic** parts of the URL
 - E.g., category name, product ID, user page, etc.

```
page('/catalog/:id', detailsView);  
// match any route, following /catalog
```

- The URL **parameter** can be accessed from the **context**

```
function detailsView(ctx, next) {  
  console.log(ctx.params.id);  
}
```

- **Multiple parameters** can be captured

```
page('/:category/:id', detailsView);
```

- Setup automatic redirect upon visit

```
page.redirect('/home', '/catalog');  
// navigating to /home will be redirected to /catalog
```

- Navigate to a page programmatically

```
page.redirect('/login');
```

- Route handlers can be **chained**

```
page('/catalog/:id', loadData, detailsView);
```

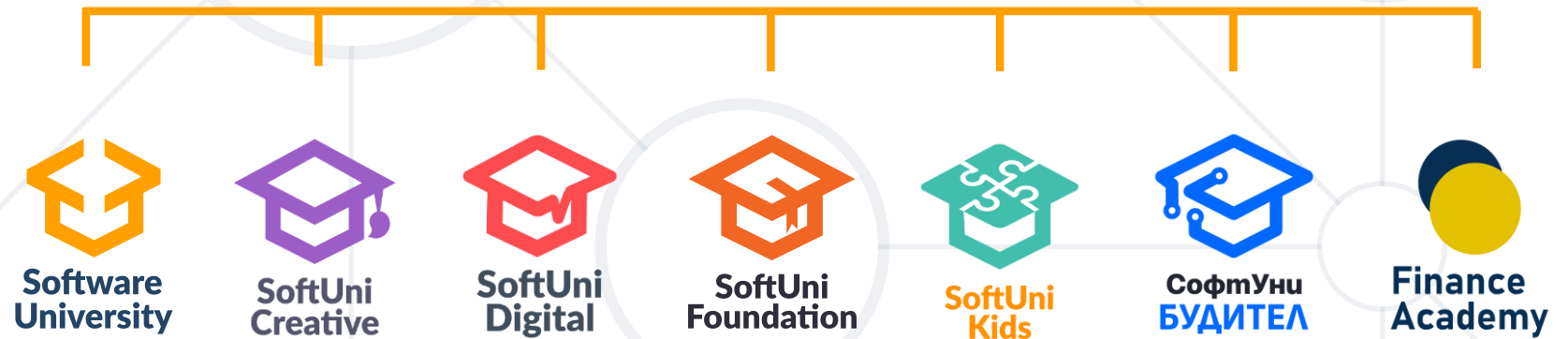
- Practical when **separating concerns**
 - E.g., **fetch** remote data in one handler and **render** in another
- Add values to the **context**, to share them **across handlers**

```
async function loadData(ctx, next) {  
  const data = await fetchProduct(ctx.params.id);  
  ctx.product = data;  
  next();  
}
```

- **Routing** is an association between URL and page content
- Types of client-side routing
 - **Hash-based** for older browser
 - **History API**
- **page.js** is a lightweight routing library
 - Supports History-based routing
 - Has **execution context**
 - Can have **custom middleware**



Questions?



SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg, softuni.org
- Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://softuni.org>
- © Software University – <https://softuni.bg>

