

Routing

Single Page Application

Routing

- Une Single Page Application est une application web qui réside sur une seule page HTML
- Le javascript va gérer l'affichage des différents éléments et non les pages HTML
- Typiquement utilisé pour des applications PWA

Single Page Application

Routing

Comment afficher
les différentes pages ?

Single Page Application

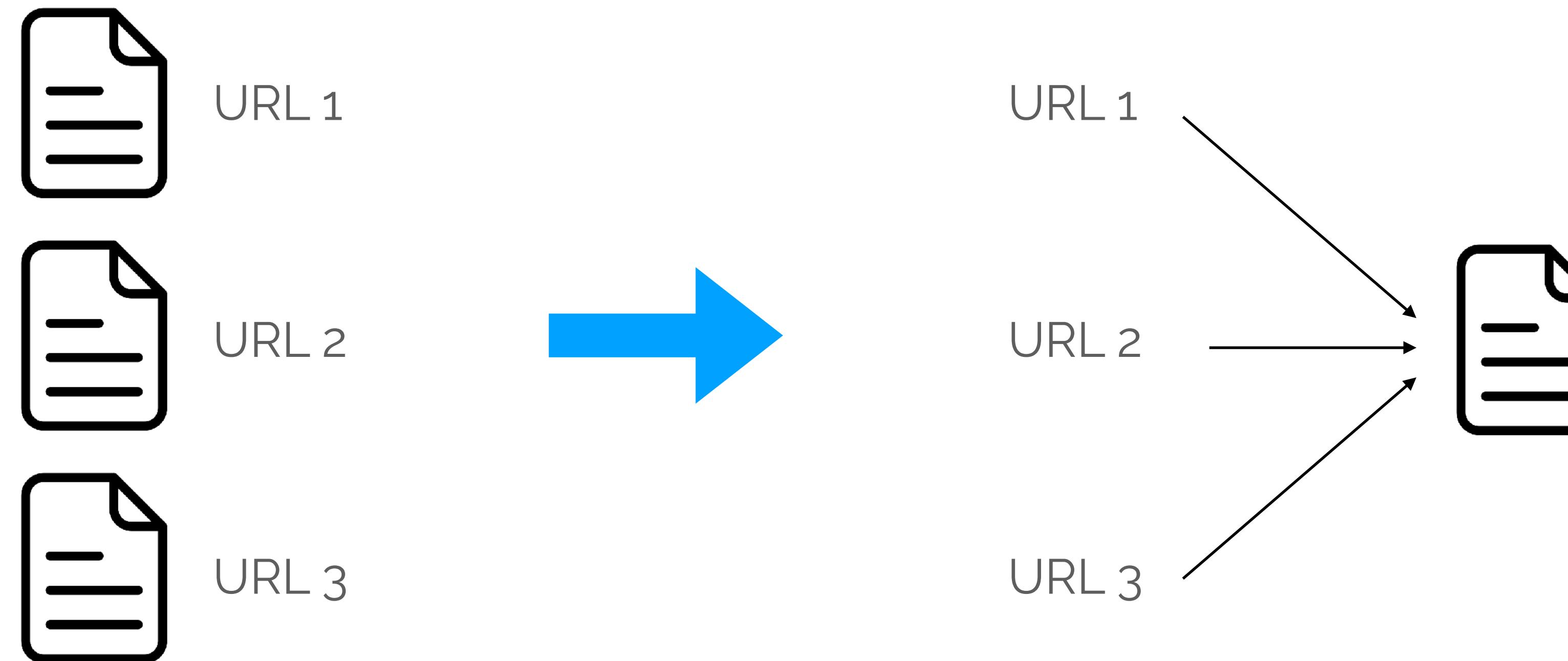
Routing

Deux options principales :

- Les vues ou pages sont existantes dans le DOM et sont affichées/ cachées par CSS (`display: none`)
- Les vues ou pages sont à chaque fois renderée au complet par le javascript. Typiquement un custom element ou une classe javascript avec une méthode render

Single Page Application - Routing

Routing



Single Page Application - Routing

Routing

Comment gérer l'état
de la page en cours?

Single Page Application - Routing

Routing

- Les URLs servent à identifier des **ressources** de manière **unique**
- Une ressource peut être de différents types (image, css, html,...)
- Une ressource peut également être un contenu de l'application:
Un artiste, une chanson, un album, ...

Single Page Application - Routing

Routing

- REST s'inscrit aujourd'hui comme le standard en terme de structure d'URL
- Principalement utilisé dans les frameworks web
- Il indique quel verbe HTTP utiliser et quel type de chemin

Single Page Application - Routing

Routing

Exemples :

- Afficher l'utilisateur avec l'id 1 : GET /users/1
- Modifier l'utilisateur avec l'id 1 : DELETE /users/1
- Récupérer ses photos : GET /users/1/photos
- Les photos de tous les utilisateurs : GET /users/photos

Single Page Application - Routing

Routing

De manière générale :

- Une ressource est identifiée par son id, suivant son type (/users/1)
- Une ressource peut avoir des ressources enfants (/users/1/photos)
- Des ressources peuvent également avoir des ressources enfants (/users/photos)

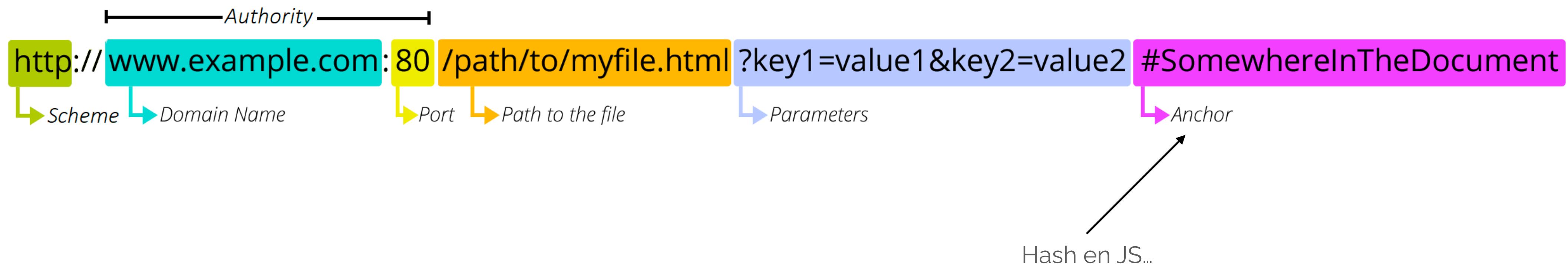
Single Page Application - Routing

Routing

- Une URL bien structurée est dite "**stateful**"
- Elle contient toutes les informations nécessaires pour représenter l'application à un instant **T**
- Exemple:
`/users/1/statistics?start_date=2020&end_date=2025&order=desc`

Single Page Application - Think RESTful...

Routing



https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL

Single Page Application - Think RESTful...

Routing

http://localhost:8080/hello?something=bonjour#quelquechose

> window.location

```
=>
hash => "#quelquechose"
host => "localhost:8080"
hostname => "localhost"
href => "http://localhost:8080/hello?something=bonjour#quelquechose"
origin => "http://localhost:8080"
pathname => "/hello"
port => "8080"
protocol => "http:"
search => "?something=bonjour"
```

Single Page Application - Routing

Routing

Deux options principales :

- Utiliser les Anchors ou Hash
- Utiliser l'API history et la réécriture d'URL sur le path

Single Page Application - Anchors/Hash JS

- Historiquement, les anchors (<>) sont des ancrés dans la page
- Une manière de mettre des liens internes à la page pour avancer dans le contenu
- Single page by design -> ne refresh pas la page lorsqu'on clique dessus
-> parfait pour notre app !

Single Page Application - Anchors/Hash Routing

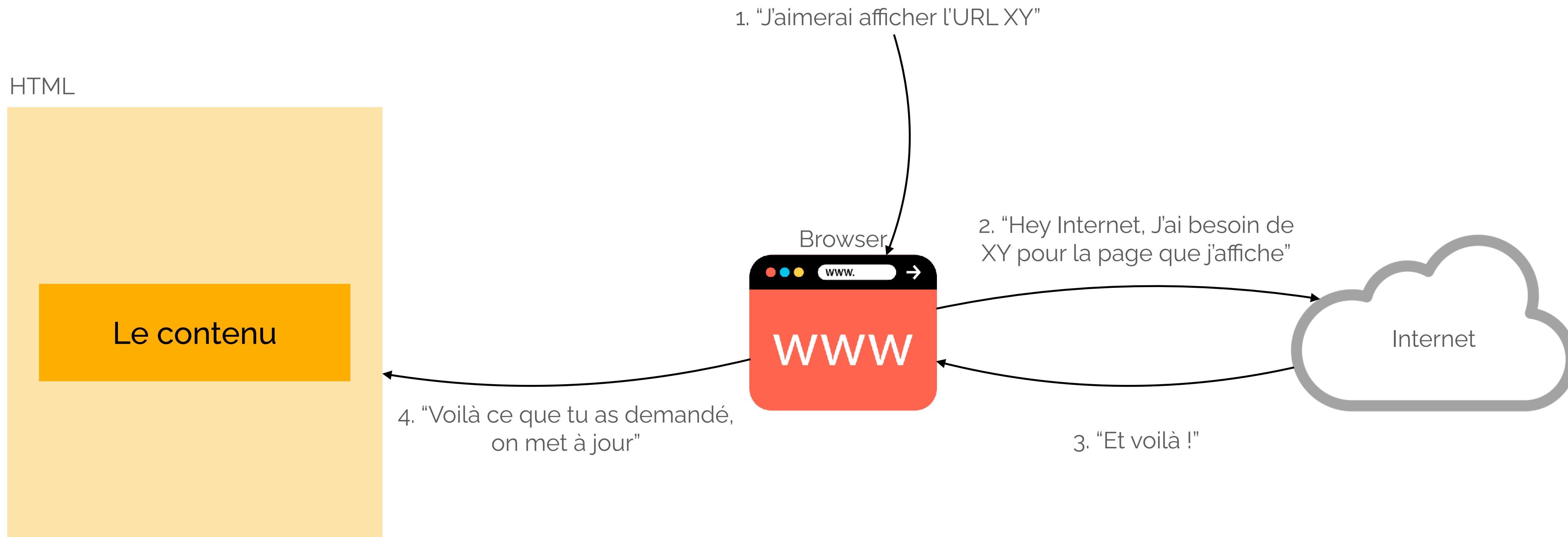
- Possible par exemple de prendre la structure d'URL suivante :
 - `/#home`
 - `/#player`
 - `/#artists`
 - `/#artists/12`



Avantages ? Inconvénients ?

Changement de section - Chargement de la page

Routing



Changement de section - Lien classique

Routing

HTML



1. "J'ai été cliqué !
Voilà ce que je demande..."

5. "Voilà ce que tu as demandé,
on met à jour"



2. "Hey Internet, J'ai besoin de
XY pour la page que j'affiche"

3. "Et voilà !"



4. "On a changé de place,
j'affiche une autre URL"

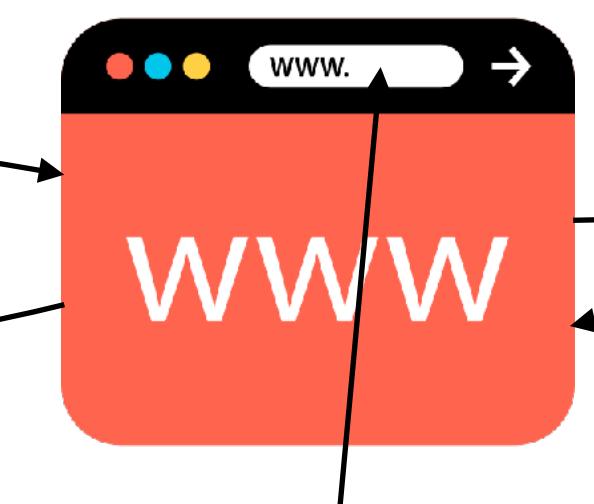
Changement de section - Lien avec hash

Routing

HTML



1. "J'ai été cliqué !
Voilà ce que je demande..."



2. "Ah, c'est un hash,
rien besoin de charger"

3. "On a changé de place,
j'affiche une autre URL"

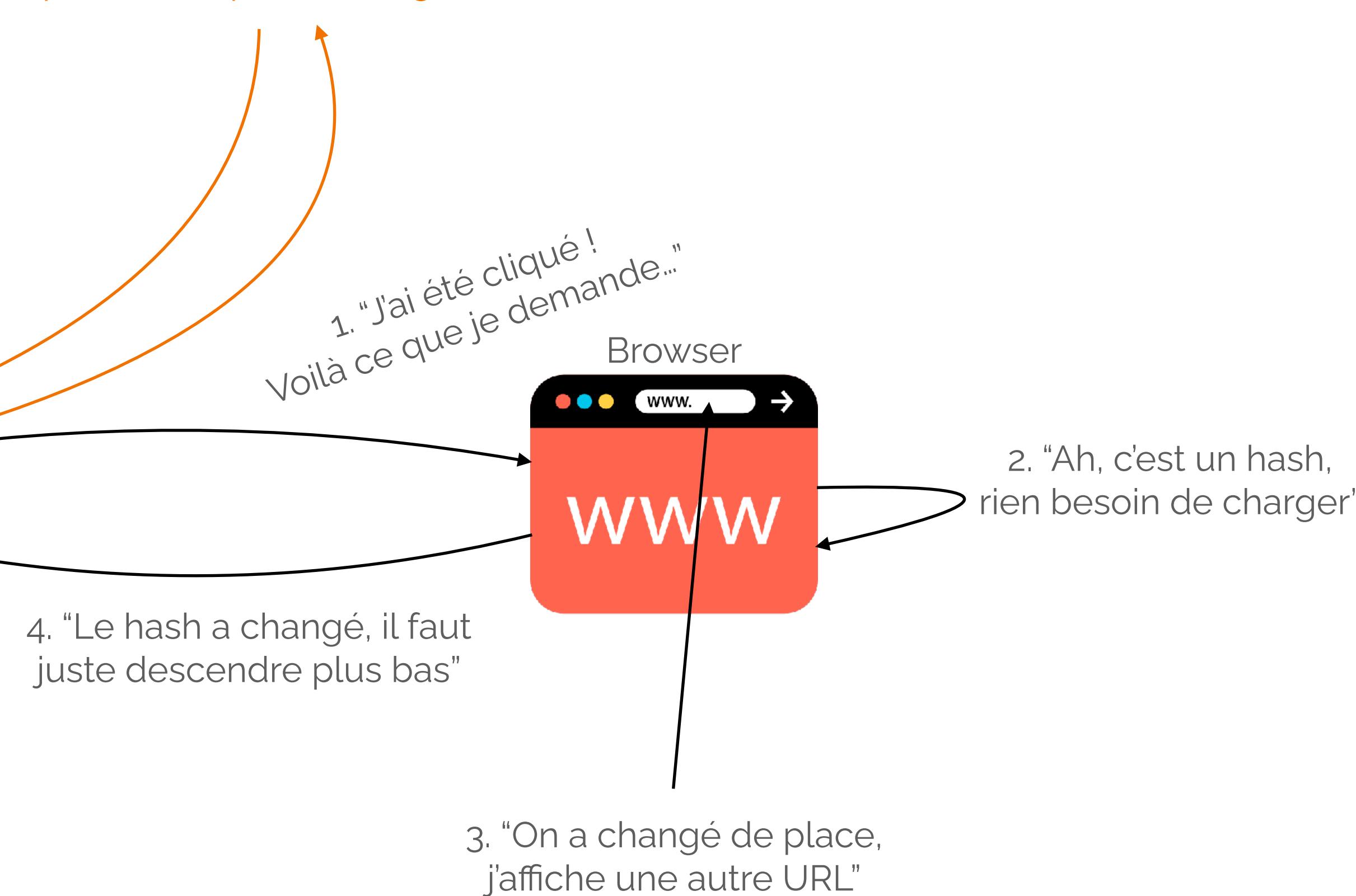
4. "Le hash a changé, il faut
juste descendre plus bas"

Changement de section - Comment ? V1

Routing

0. On intercepte avec un listener 'click' et on remplace l'action par défaut, par l'affichage de la section

HTML



Changement de section - Comment ? V1

Routing

Manipulation des URLs

- Le browser nous permet de modifier l'historique de navigation manuellement
- Possible de le contrôler (précédent/suivant)
- Ajouter une entrée dans l'historique
- Remplacer une entrée
- Être informé d'un changement d'état

Changement de section - Comment ? V1

Routing

L'API History

- `history.go(entier)` ou `history.forward()`/`back()`
- `history.pushState(état, titre, url)`
- `history.replaceState(état, titre, url)`
- L'événement `popstate` sur `window`

Changement de section - Comment ? V2

Routing

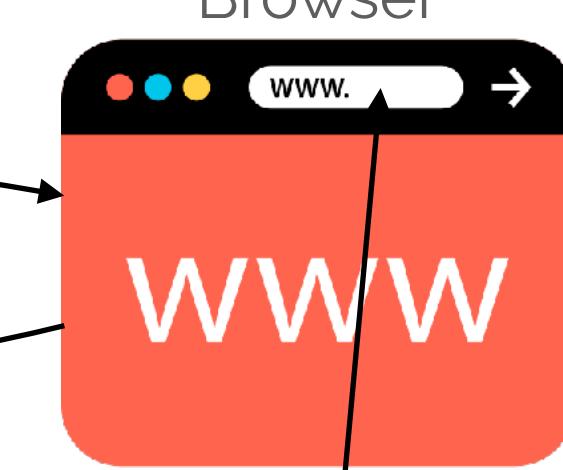
HTML



1. "J'ai été cliqué !
Voilà ce que je demande..."

4. "Le hash a changé, il faut
juste descendre plus bas"

6. "Ah! Le hash a changé ? Alors on affiche la section"



Browser

2. "Ah, c'est un hash,
rien besoin de charger"

3. "On a changé de place,
j'affiche une autre URL"

V1

https://developer.mozilla.org/fr/docs/Web/API/History_API

history...

V2

<https://developer.mozilla.org/fr/docs/Web/API/WindowEventHandlers/onhashchange>

```
window.addEventListener( "hashchange" , () => { ... } )
```

Changement de section - Comment ?

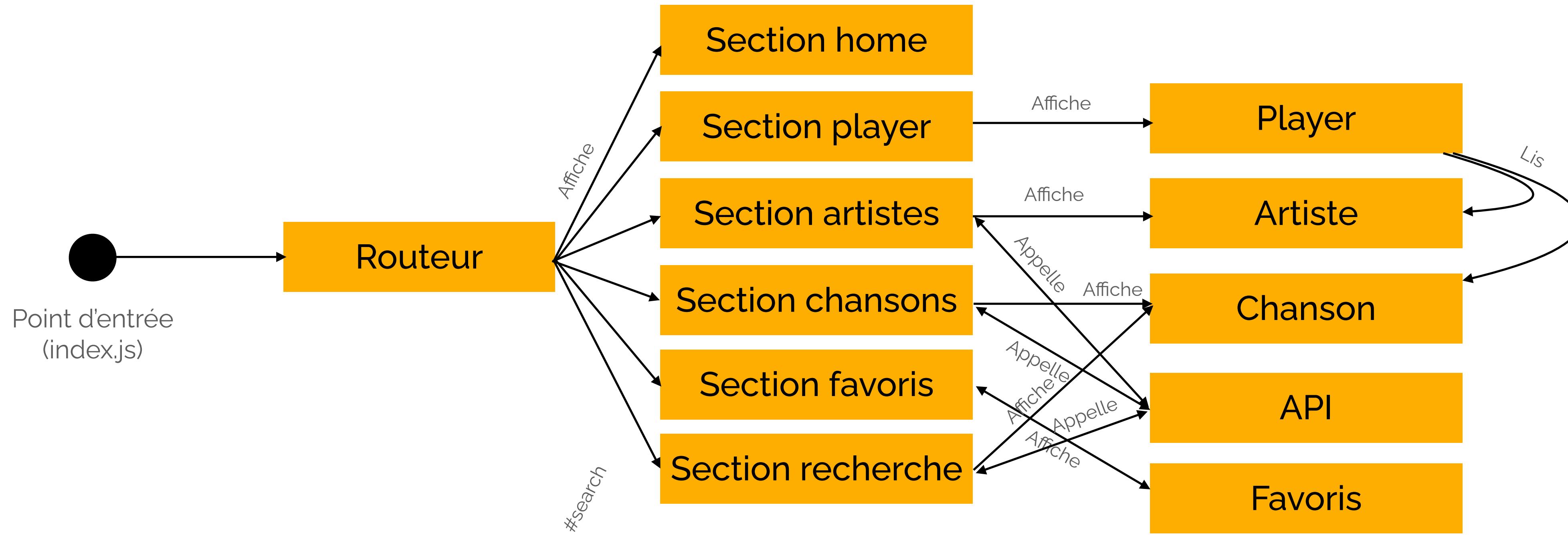
JS

- Garder en tête que les boutons précédent/suivant du navigateur doivent fonctionner
- Ouvrir le lien dans un nouvel onglet doit fonctionner également
- `window.location` vous donne toutes les infos sur l'URL en cours

Put it together

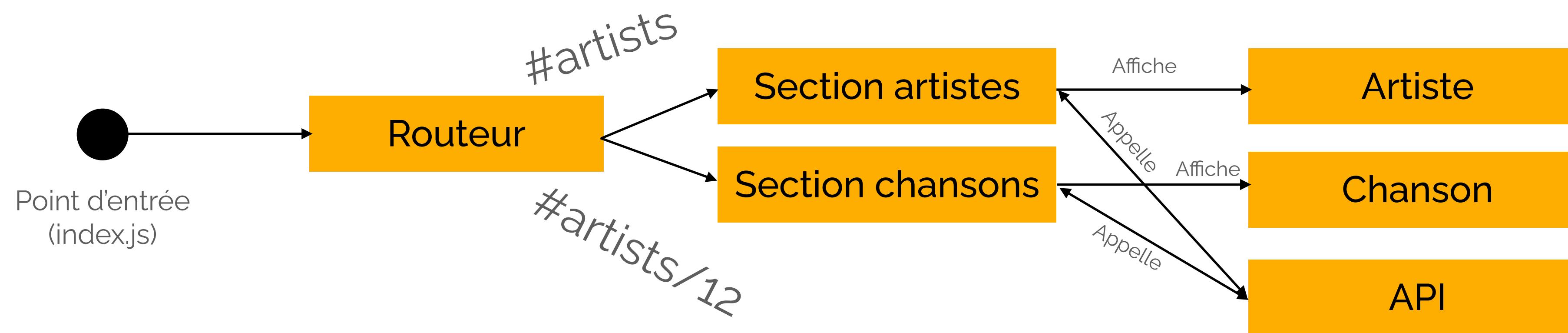
Vue globale

Put it together



Vue globale

Put it together



Routeur

Put it together

```
// S'il n'y a pas de hash (par ex, on est sur "localhost:1234/"), le défaut devient '#home'  
const hash = window.location.hash || '#home'  
  
if(hash == '#home') {  
    afficherLaHome()  
} else if (hash == '#artists') {  
    afficherLesArtistes()  
}
```

Routeur - Liens enfants (vue artiste)

Put it together

- Comment gérer les liens enfants, type "#artists/12" ?
- Il nous faut différencier :
 - La liste des artistes "#artists"
 - La vue d'un artiste, selon son id "#artists/12"

Routeur - Liens enfants (vue artiste)

Put it together

"#artists/12"



Est-ce qu'il y a un slash ?

Routeur - Liens enfants (vue artiste)

Put it together

- Pléthore de manières de faire... Les deux principales seraient:
 - Chercher le tiret dans la chaîne de caractère et la découper avec des expressions régulières ou des fonctions de recherche...
 - Se dire que la chaîne "#artists/12" n'est en fait qu'une liste d'éléments séparés par des slashes... et les convertir en tableau

Routeur - Liens enfants (vue artiste)

Put it together

- La v2 semble plus simple...
- La fonction “split” vient à la rescousse !
- Elle permet de découper une chaîne en un tableau de sous-chaînes de caractères, selon un caractère donné
- Exemple:
`'une/chaine/de/slashes'.split('/') => ['une', 'chaine', 'de', 'slashes']`

Routeur - Fonction split

Put it together

- Exemple précédent
`'une/chaine/de/slashes'.split('/')` ==> ['une', 'chaine', 'de', 'slashes']
- Que se passe-t-il si pas de slash ?
`'uneChaineSansSlashes'.split('/')` ==> ['uneChaineSansSlashes']
- Que se passe-t-il avec une chaine vide ?
`' '.split('/')` ==> ['']

Routeur - Fonction split

Put it together

- Avec nos artistes :

- Avec slash

```
const hashSplité = '#artists/12'.split('/') ==> ['#artists', '12']
```

- Que se passe-t-il si pas de slash ?

```
'#artists'.split('/') ==> ['#artists']
```

Routeur - Fonction split

Put it together

- Exemple d'implémentation :

```
const hashSplité = window.location.hash.split(' / ')
```

```
// avec '#artists/12' ==> ['#artists', '12']
```

```
// avec '#artists/section' ==> ['#artists', 'section']
```

Routeur - Fonction split

Put it together

- Que se passe-t-il quand on essaie d'accéder à un élément d'un tableau qui n'existe pas ?

Exemple: La cellule 397 du tableau ['a', 'b', 'c'] ?

```
const tableau = ['a', 'b', 'c']
```

```
console.log(tableau[397]) // ==> undefined
```

Routeur - Fonction split

Put it together

Par exemple...

```
const hashSplité = window.location.hash.split('//')

// si le premier élément est artiste, on est dans la gestion des artistes...

if(hashSplité[0] == '#artists') {

    // est-ce que le deuxième élément retourne quelque chose ? Et donc n'est pas undefined ? Oui?
    // Alors il y a un id et on affiche cet artiste

    if(hashSplité[1]) {

        afficherChansonsArtiste(hashSplité[1]) // la fonction prend en argument l'id de l'artiste à afficher
    }

    else {

        afficherArtistes()
    }
}
```

Routeur - Fonction split

Put it together

Par exemple... avec un switch

```
const hashSplité = window.location.hash.split('//')

// si le premier élément est artiste, on est dans la gestion des artistes...

switch(hashSplité[0]) {

    case '#artists':

        // est-ce que le deuxième élément retourne quelque chose ? Et donc n'est pas undefined ? Oui?
        // Alors il y a un id et on affiche cet artiste

        if(hashSplité[1]) {

            afficherChansonsArtiste(hashSplité[1])

        }

        else {

            afficherArtistes()

        }

        break;

    case '#player':

        ...

        break;

}
```