JS

Single Page Application

JS

- 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

JS

Comment renderer
les différentes pages ?

Single Page Application

JS

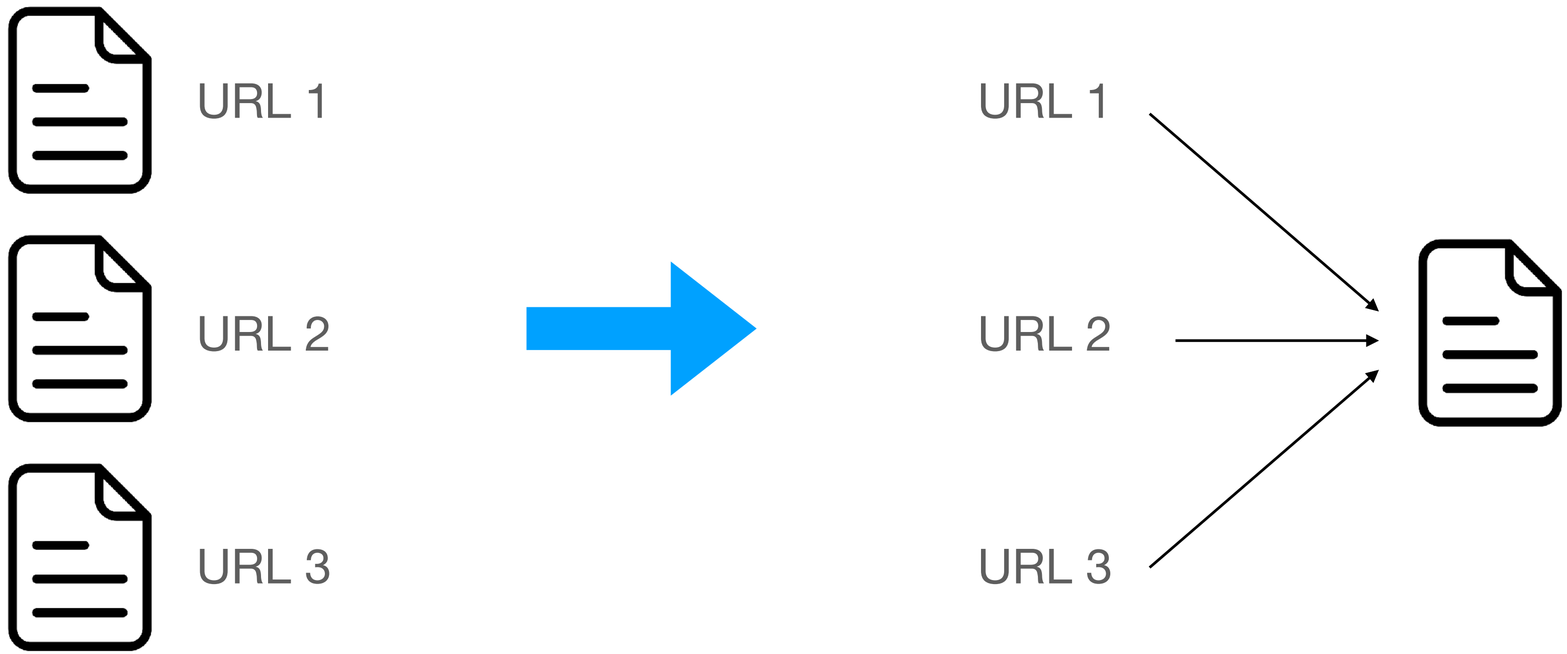
Deux options principales :

- 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

JS



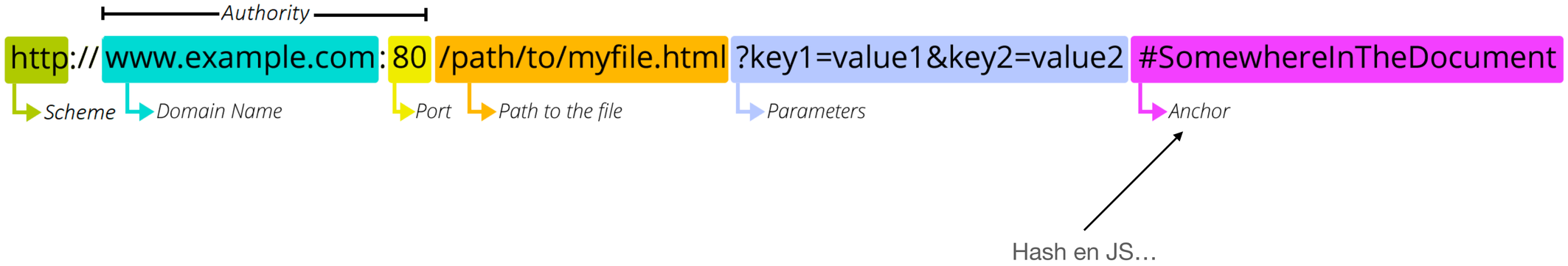
Single Page Application - Routing

JS

Comment géré l’état
de la page en cours?

Single Page Application - Think RESTful…

JS



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

Single Page Application - Think RESTful…

JS

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

JS

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 (<a />) sont des ancres 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

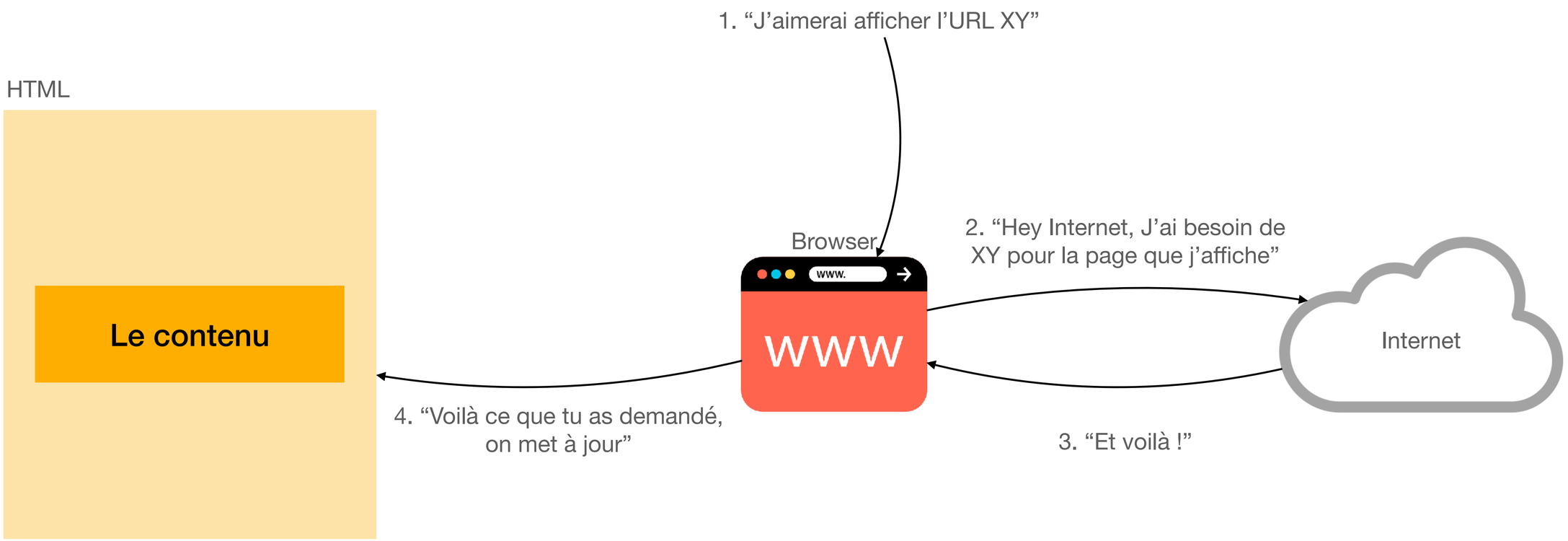
JS

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

Avantages ? Inconvénients ?

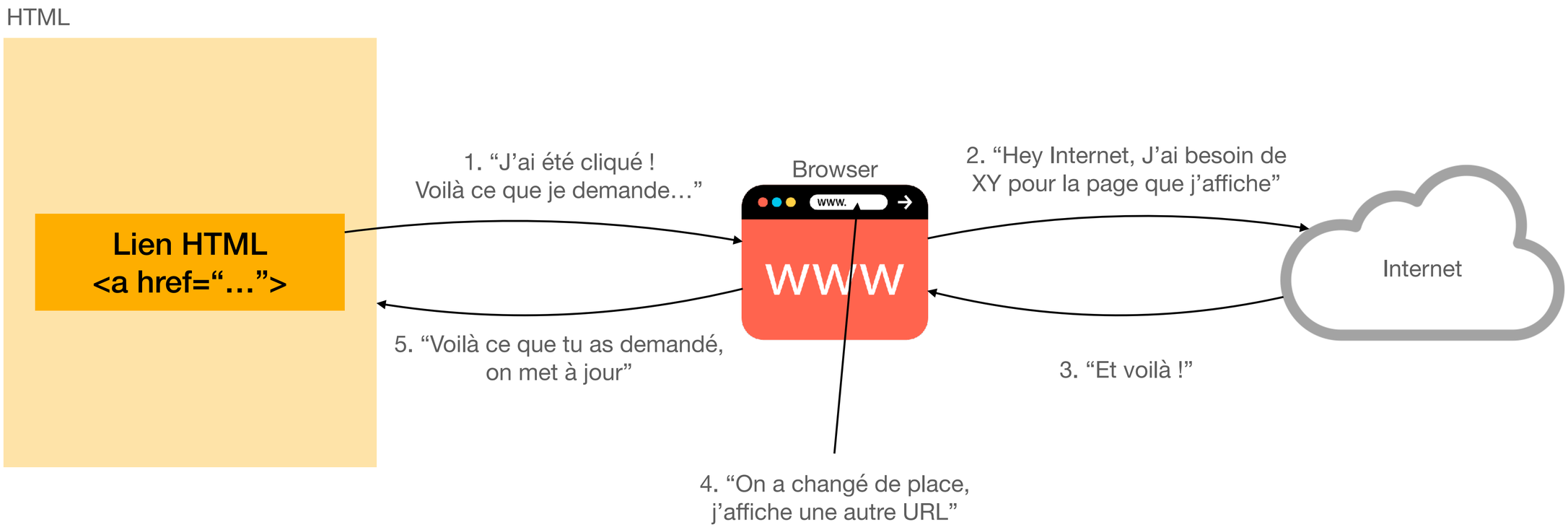
Changement de section - Chargement de la page

JS



Changement de section - Lien classique

JS

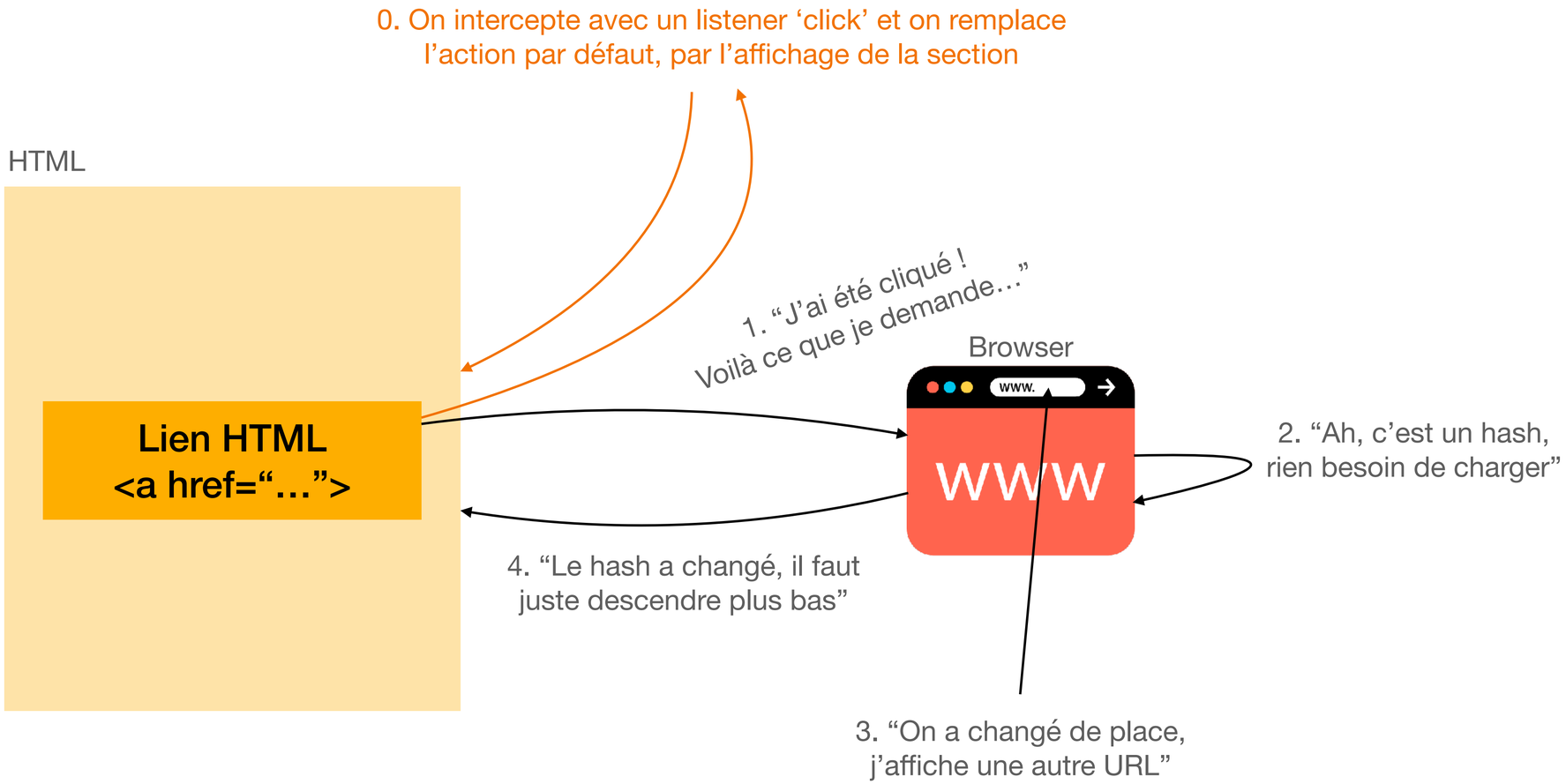


Changement de section - Lien avec hash

JS

Changement de section - Comment ? V1

JS



Changement de section - Comment ? V1

JS

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

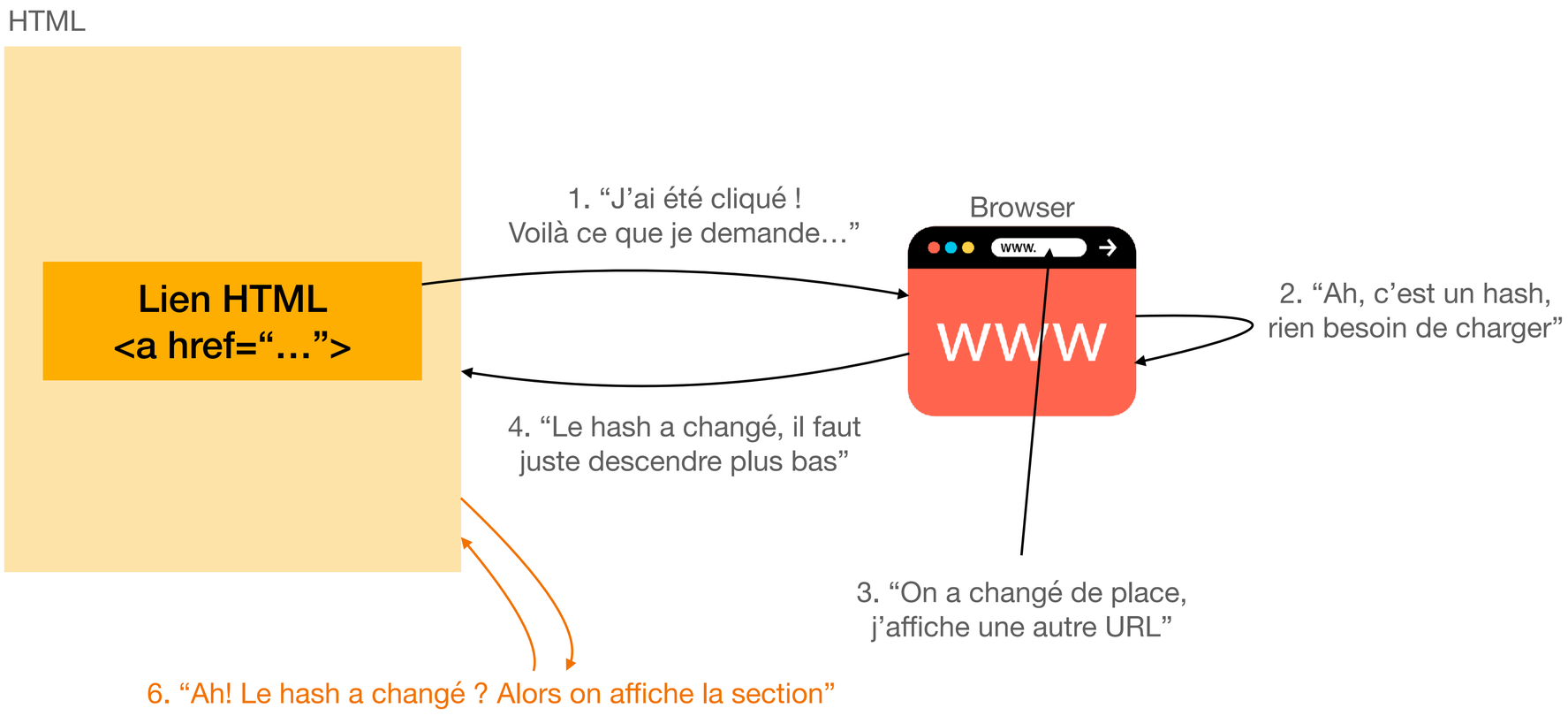
JS

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

JS



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", () => { … })
```

ou

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

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

GO !

Architecture de code

Modularité et responsabilité

Architecture de code

- Une architecture est dite “modulaire” quand elle est séparée en plusieurs modules, avec des responsabilités précises

- Chaque module a une implémentation privée qui lui est propre et une implémentation publique pour interagir avec les autres modules

- Chaque module a des voisins directs, avec qui il a une forte interaction, et d’autres voisins indirects avec lesquels il échange par le biais d’autres modules

- Cela s’applique aussi bien à une vue globale qu’à une vue détaillée

Modularité et responsabilité - Exemple

Architecture de code

- Le CSS est responsable de la mise en page
- Il offre le langage CSS comme implémentation publique (on lui dit quoi faire)
- Il a comme voisin direct le HTML, car il met en page des éléments du langageHTML

- Le HTML est responsable de la structure de la page
- Il offre le langage HTML comme implémentation publique
- Il a comme voisin direct le CSS, car il lui fournit les éléments à mettre en page
- Le JS comme autre voisin direct, car il interagit avec les éléments HTML

- Le CSS est responsable de la dynamique de la page
- Il offre le langage JS comme implémentation publique
- Il a comme voisin direct le HTML, car il utilise les éléments HTML pour les rendre dynamique
- Le CSS n’est pas un voisin direct, car le JS va d’abord utiliser un élément HTML pour y ajouter des styles

Loi de Déméter - Principe de connaissance minimale

Architecture de code

« Ne parlez qu'à vos amis immédiats ».

Loi de Déméter - Principe de connaissance minimale

Architecture de code

- La loi de Déméter vise à limiter les connaissances de chaque module
- Le but est de diminuer les dépendances et donc la complexité
- Cela permet une plus grande flexibilité et une notion d’agilité

Loi de Déméter - Principe de connaissance minimale

Architecture de code

- Un client qui aurait besoin d'un conseille au sein d'une entreprise, appelle la réception qui va l'aiguiller vers la personne adaptée à sa demande

- Le client ne se soucie pas de la liste des employés, leur présence, changements, etc…

Loi de Déméter - Dans la pratique

Architecture de code

On souhaite afficher une <section />


```css
section {
  /* display: flex */
  display: none;
}
```

```html
<section id="ma-section">
  …
</section>
```

```js
const section = document.querySelector('#ma-section')
section.style.display = 'flex'
```

Pas très “Déméter”… Le JS va directement modifier le CSS.
Que se passe-t-il si cette section devient un block et plus un flex ?

Loi de Déméter - Dans la pratique

Architecture de code

On souhaite afficher une <section />


```css
section {
  display: none;
}
section.active {
  display: flex;
}
```

```html
<section id="ma-section">
  …
</section>
```

```js
const section = document.querySelector('#ma-section')
section.classList.add('active')
```

Mieux ! Le JS ne fait qu'ajouter un attribut HTML (une classe) et il ne s'occupe pas du mode d'affichage.

C'est le CSS qui va gérer le changement -> Les responsabilités sont respectées !

Loi de Déméter - Dans la pratique 2

Architecture de code

Taille du texte en CSS

- S’occupe d’afficher le player
- La taille exacte du texte ne l’intéresse pas !

- S’occupe de gérer la taille du texte pour le monde
- Le player ne l’intéresse pas !

Loi de Déméter - Changement de section

Architecture de code

Plutôt **V1** ou **V2** ?

Structure de l’application

Architecture de code

Ou d’autres dans l’application ?

Schémas UML

Architecture de code

- Les schémas UML (**U**nified **M**odeling **L**anguage) sont des schémas standardisés pour représenter une application

- Il en existe plus d'une vingtaine…

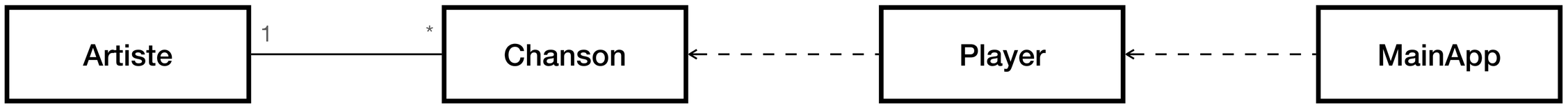
- Nous allons utiliser les diagrammes de classe

https://fr.wikipedia.org/wiki/Diagramme_de_classes

Schémas UML - Diagramme de classe

Architecture de code

- Chaque classe (ou modèle) est représenté par un rectangle avec le nom de la classe
- Optionnellement, la liste de ses attributs et de ses fonctions
- Les éléments sont connectés ensemble par plusieurs types de flèches, représentant le type d'interaction (héritage, composition, …)



https://fr.wikipedia.org/wiki/Diagramme_de_classes

Spotlified

Architecture de code

Quels sont les éléments importants de l’application ?

JSON

What is it?

JSON

“Le JavaScript Object Notation (JSON) est un format standard utilisé pour représenter des données structurées de façon semblable aux objets Javascript.”

https://developer.mozilla.org/fr/docs/Learn/JavaScript/Objects/JSON

What is it?

JSON

- JSON est un format de données, fortement inspiré du JS
- Il dispose de sa propre syntaxe et est indépendant du Javascript
- La plupart des langages de programmation fournissent des librairies pour permettre de le convertir en des objets utilisables ou d'en générer pour l'exporter au sein d'un autre langage
- JSON se stock dans des chaînes de caractères

https://developer.mozilla.org/fr/docs/Learn/JavaScript/Objects/JSON

Quand l’utiliser ?

JSON

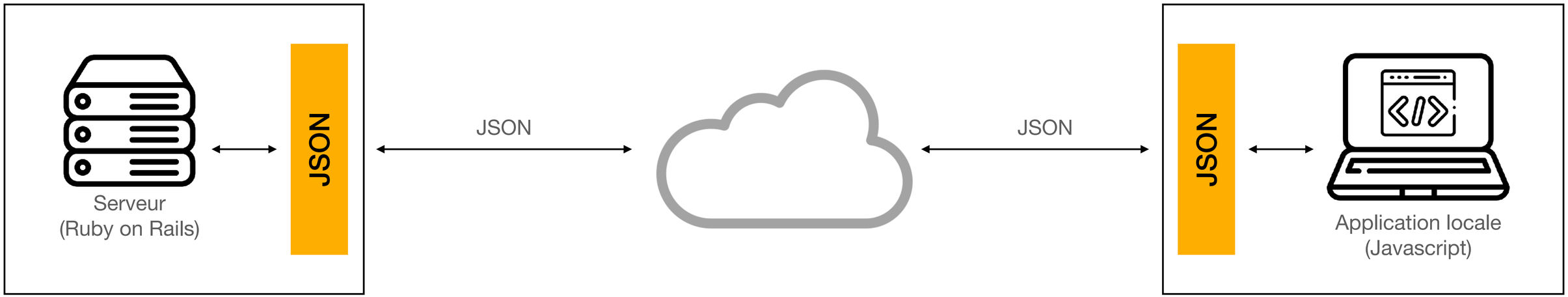
- Idéalement, lorsque deux systèmes séparés doivent s’échanger des informations (par ex. une API entre Javascript et PHP)

- Quand il faut stocker des données structurées en dehors du langage lui-même (par ex. enregistrer un tableau dans un fichier texte)

https://developer.mozilla.org/fr/docs/Learn/JavaScript/Objects/JSON

Usecase

JSON



Exemple

JSON

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
      "Molecule Man",
      "Madame Uppercut"
   ]
}
```

https://developer.mozilla.org/fr/docs/Learn/JavaScript/Objects/JSON

Comment l’utiliser

JSON

- JSON supporte les types de données standards, comme:
  - Chaîne de caractères
  - Nombres
  - Null
  - Booléen

- Il y a également deux types de données structurées :
  - Tableau
  - Objet (clé/valeur, comme en js)

https://developer.mozilla.org/fr/docs/Learn/JavaScript/Objects/JSON

Comment l’utiliser

JSON

Deux méthodes principales :

- `JSON.parse("...")`
  Converti une chaîne de caractères JSON en un objet javascript

- `JSON.stringify(unevariable)`
  Converti une variable Javascript en un objet JSON

https://developer.mozilla.org/fr/docs/Learn/JavaScript/Objects/JSON

Comment l’utiliser - JSON.parse

JSON

```
const monJson = '{ "cours": "WebmobUI", "lieu": "HEIG-VD"}'

const monJsonparsé = JSON.parse(monJson)

console.log(monJsonparsé.cours) => "WebmobUI"

console.log(monJsonparsé.lieu) => "HEIG-VD"
```

https://developer.mozilla.org/fr/docs/Learn/JavaScript/Objects/JSON

Comment l’utiliser - JSON.stringify

JSON

```
const monObjet = { cours: 'WebmobUI', lieu: 'HEIG-VD' }

console.log(monObjet.cours) => "WebmobUI"

console.log(monObjet.lieu) => "HEIG-VD"

const monJson = JSON.stringify(monObjet)

console.log(monJson) => '{ "cours": "WebmobUI", "lieu": "HEIG-VD"}'
```

https://developer.mozilla.org/fr/docs/Learn/JavaScript/Objects/JSON

L’API Spotlified

Concept

L’API Spotlified

- L’API Spotlified tourne sur un serveur distant: https://webmob-ui-22-spotlified.herokuapp.com/

- Tous les endpoints retournent du JSON qui sera à “parser” par vos soins

- Les URLs sont dites “REST” pour plus de clarté

Endpoints

L’API Spotlified

3 endpoints principaux :

- Lister les artistes
- Listes les chansons d’un artiste
- Rechercher une chanson par texte libre

Endpoints - Lister les artistes

L’API Spotlified

URL: https://webmob-ui-22-spotlified.herokuapp.com/api/artists

Response: Tableau d’artistes

Exemple:

```
[
  { "id": 2, "name": "Alan Walker", image: "https://...." },
  { "id": 3, "name": "Dynoro", image: "https://...." }
]
```

Endpoints - Lister les chansons d’un artiste

L’API Spotlified

URL: https://webmob-ui-22-spotlified.herokuapp.com/api/artists/:id/songs

Response: Tableau de chanson, avec l’artiste dans la chanson

Exemple:


```
[
  { "id": 2, "title": "Faded", audio_url: "https://....", "artist": { … }},
  { "id": 3, "title": "Spectre", audio_url: "https://....", "artist": { … }}
]
```

Endpoints - Rechercher une chanson par texte libre

L’API Spotlified

URL: https://webmob-ui-22-spotlified.herokuapp.com/api/songs/search/:query

Response: Tableau de chanson, avec l’artiste dans la chanson

Exemple:

```
[
  { "id": 2, "title": "Faded", audio_url: "https://....", "artist": { … }},
  { "id": 3, "title": "Spectre", audio_url: "https://....", "artist": { … }}
]
```

API Client

L’API Spotlified

Comment structurer l’utilisation de l’API en mode Déméter ?