

Web & Mobile UI - Cours 1

Conception d'applications web

Enseignants

Cours

Phase 1

Loris Gavillet

loris.gavillet@heig-vd.ch

Semaine 1 -> Semaine 6

Phase 2

Nicolas Chablotz

nicolas.chablotz@heig-vd.ch

Semaine 7 -> Semaine 12

Objectifs

Cours

- Être capable de réaliser des applications Web progressive (Progressive Web App) en intégrant un design préalablement conçu
- Manipuler différentes API's pour la réalisation d'interfaces web modernes
- Mettre en place des patrons de conception (design patterns)

Contenu

Cours

- De HTML5 vers un standard HTML vivant
- Responsive design, Media Queries
- Data-attributes, custom elements
- API HTML: LocalStorage, History, Service Workers, ...
- Offline mode
- Début templating
- Stack fonctionnelle

Déroulement

Cours

Phase 1 (Semaine 1-6)

- Introduction au cours
- 1/3 Théorie - 2/3 Pratique
- Live coding - Corrections entre les cours
- Setup des outils de base
- Projet sur 8 demi-journées de cours
- Examen sur 4 périodes le 25.03.2024

Technologies utilisées

Cours

Phase 1

- Parcel JS
- HTML / CSS / JS
- API HTML
- Templating
- PWA / Service workers
- Intro aux frameworks

Technologies utilisées

Cours

The untold history of web development:

- 1990: HTML invented
- 1994: CSS invented to fix HTML
- 1995: JS invented to fix HTML/CSS
- 2006: jQuery invented to fix JS
- 2010: AngularJS invented to fix jQuery
- 2013: React invented to fix AngularJS
- 2014: Vue invented to fix React & Angular
- 2016: Angular 2 invented to fix AngularJS & React
- 2019: Svelte 3 invented to fix React, Angular, Vue
- 2019: React hooks invented to fix React
- 2020: Vue 3 invented to fix React hooks
- 2020: Solid invented to fix React, Angular, Svelte, Vue
- 2020: HTMX 1.0 invented to fix React, Angular, Svelte, Vue, Solid
- 2021: React suspense invented to fix React, again
- 2023: Svelte Runes invented to fix Svelte
- 2024: jQuery still used on 75% of websites

Repository et informations générales

Cours

<https://github.com/lgavillet/webmobui>

Projet

Concept Projet



Spotlified

Simplified spotify

Prototype

Projet

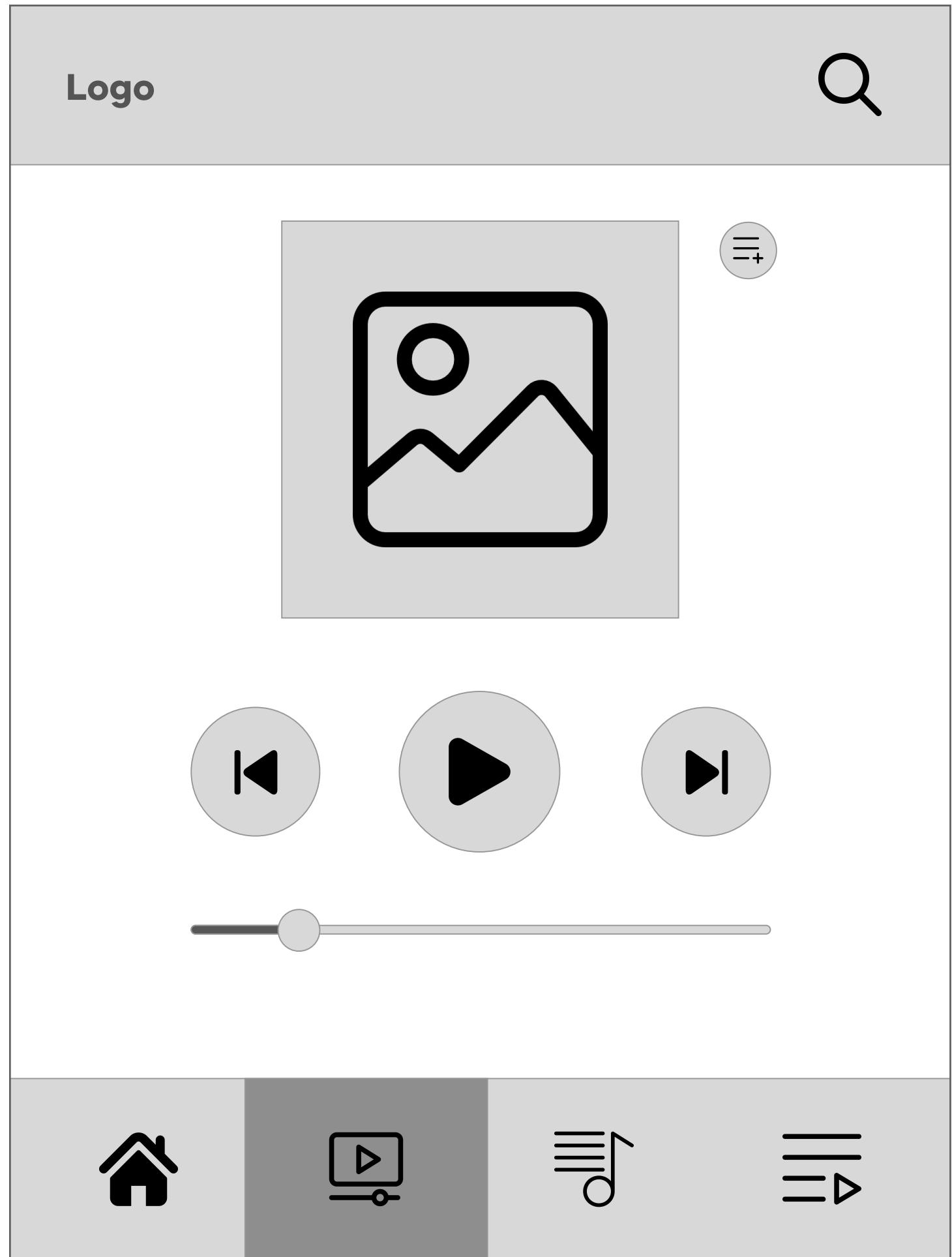
<https://invis.io/SZ13FGNDRJQP>

Concept Projet

Spotlified version simplifiée

3 modèles :

- Chansons
- Artistes
- Favoris



Architecture

Projet



Features

Projet

- Homepage
- Liste des artistes
- Liste des chansons par artiste
- Lecteur audio
- Champ de recherche
- Gestion online/offline
- Gestion des favoris
- Paroles
- Caching
- Installation de l'application

Composants ?

Projet

- Projet Parcel vide
- Squelette HTML
- Styles CSS structurels
- Icônes
- Composants rendering
- Routeur pour les pages web
- Client pour l'API JSON
- Lecteur audio
- Local storage pour les favoris
- Détection online/offline
- Manifest PWA
- Caching
- Service worker
- Notifications Push

Truth be told

Projet

Dad: Why are your eyes red son?

Son: I smoke weed

**Dad: Don't lie, you're crying because
you have been coding in JavaScript**



Packages

What is it?

Packages



Définition

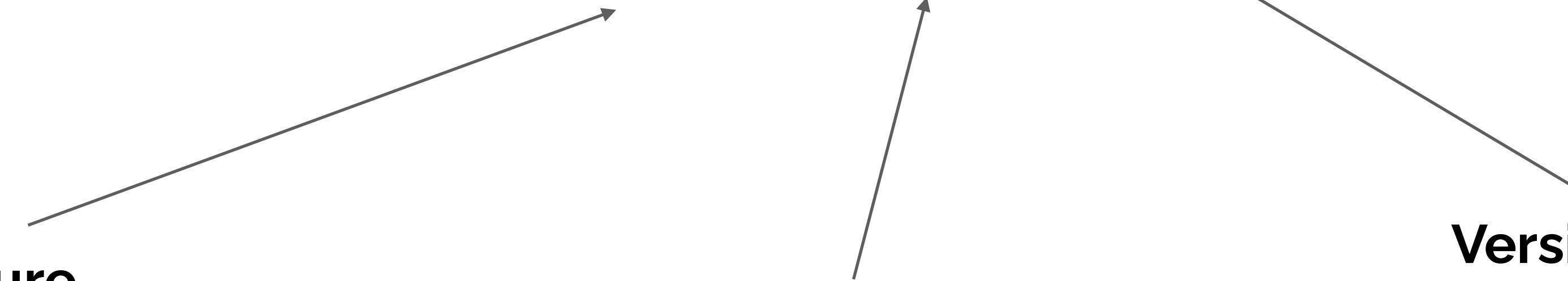
Package

- Bout de code réutilisable
- Expose des fonctionnalités
- Intégrable dans une application
- Versionable

Versioning - Semantic versioning

Package

v16.2.6



Version majeure

Le numéro de version MAJEUR quand il y a des changements non rétrocompatibles,

Version mineure

Le numéro de version MINEUR quand il y a des ajouts de fonctionnalités rétrocompatibles

Version patch

Le numéro de version de CORRECTIF quand il y a des corrections d'anomalies rétrocompatibles

Choix d'un package

Package

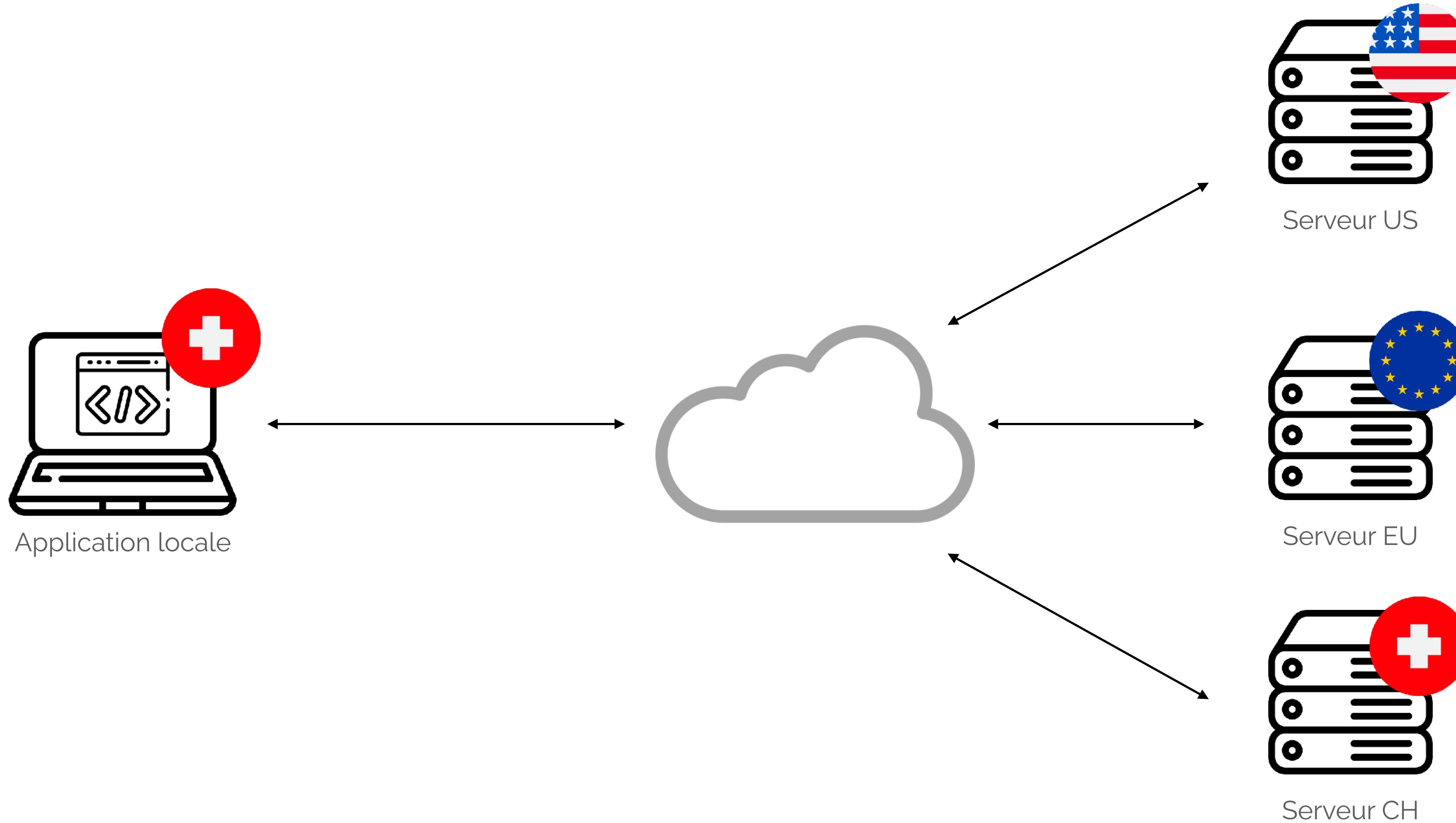
- Vérifier la version -> v0.x.x -> à bannir
- Nombre d'utilisateurs ?
- Maintenu ? Date dernier patch ?
- Dépendances ?

Intégration Package

1. Copy-paste old school / Téléchargement
2. CDN - Content Delivery Network
3. Package manager

CDN - Content Delivery Network

Package



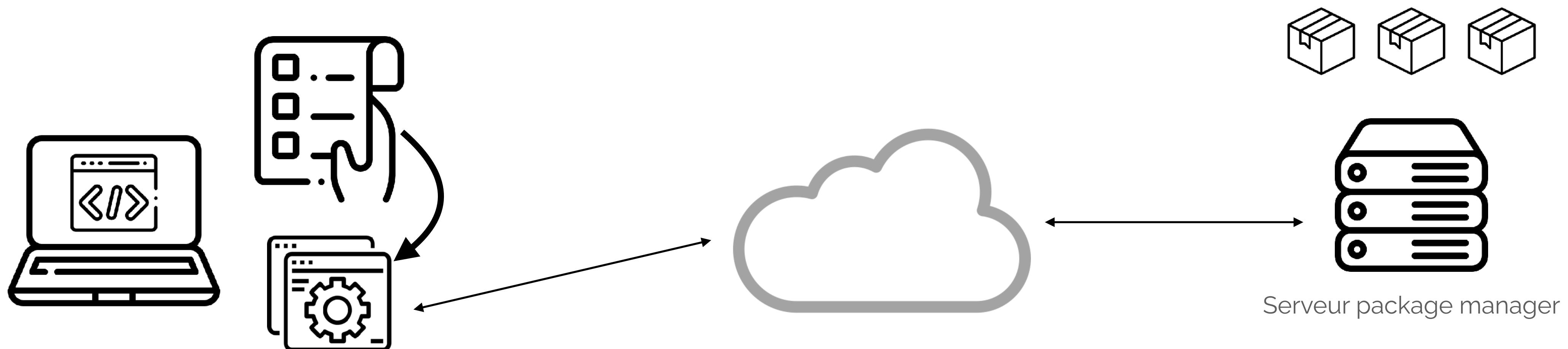
CDN - Content Delivery Network

Package

```
<link
  rel="stylesheet"
  href="https://fonts.googleapis.com/css2?family=Material+Icons"
/>
```

Package manager

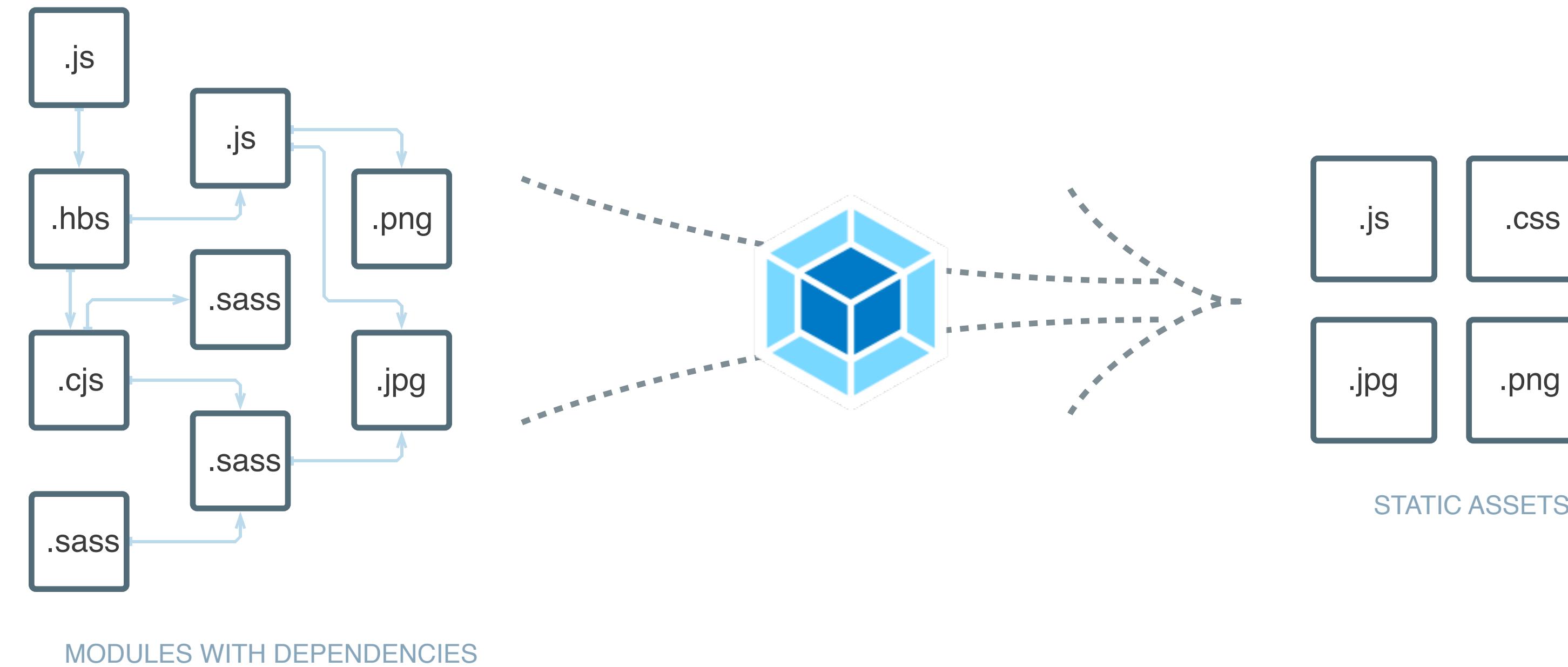
Package



Parcel JS

Concept

Parcel JS



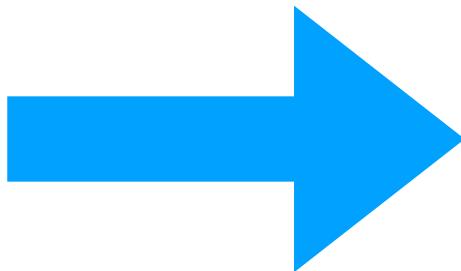
<https://parceljs.org/>

Concept

Parcel JS



Entrypoint



Entrypoint

Concept

Parcel JS

“Parcel JS is a module bundler. Its main purpose is to bundle JavaScript files for usage in a browser, yet it is also capable of transforming, bundling, or packaging just about any resource or asset.”

(Ok... this was taken from <https://webpack.js.org/>)

Concept

Parcel JS

- NodeJS est un langage de programmation backend, écrit en Javascript

Basé sur



- A ne pas confondre avec du Javascript frontend
- Node est livré avec un gestionnaire de package : NPM (**N**ode **P**ackage **M**anager)

Concept

Parcel JS

- Parcel JS est un package de NodeJS
- S'installe comme tout autre package avec

```
> npm install --save-dev parcel
```

ou

```
> yarn add --dev parcel
```

Pour le cours

Parcel JS

- Uniquement packaging en javascript et css
- Serveur web avec Hot reloading
- Pré-configuré et prêt à l'emploi

Structure de base officielle

Parcel JS



Node installé ?

Parcel JS

```
> node --version
```

Installation

Parcel JS

Installation

- Installer Node (si pas déjà fait)
<https://nodejs.org/en/download/> Version LTS
- Télécharger le projet vide sous:
<https://github.com/lgavillet/webmobui/raw/main/Ressources/projet-vide.zip>
- Ouvrir l'invite de commande dans votre dossier projet
`> npm install`

<https://parceljs.org/getting-started/webapp/>

Utilisation

Parcel JS

Démarrer le serveur de dev

```
> npm start
```

<https://parceljs.org/getting-started/webapp/>

HTML / CSS / JS

W3C vs WHATWG

HTML / CSS / JS

- W3C (**W**orld **W**ide **W**e**C**onsortium)
Acteur historique...
- WHATWG (**W**e**b** **H**ypertext **A**pplication **T**echnology **W**orking **G**roup)
Collaboration non officielle des différents navigateurs web

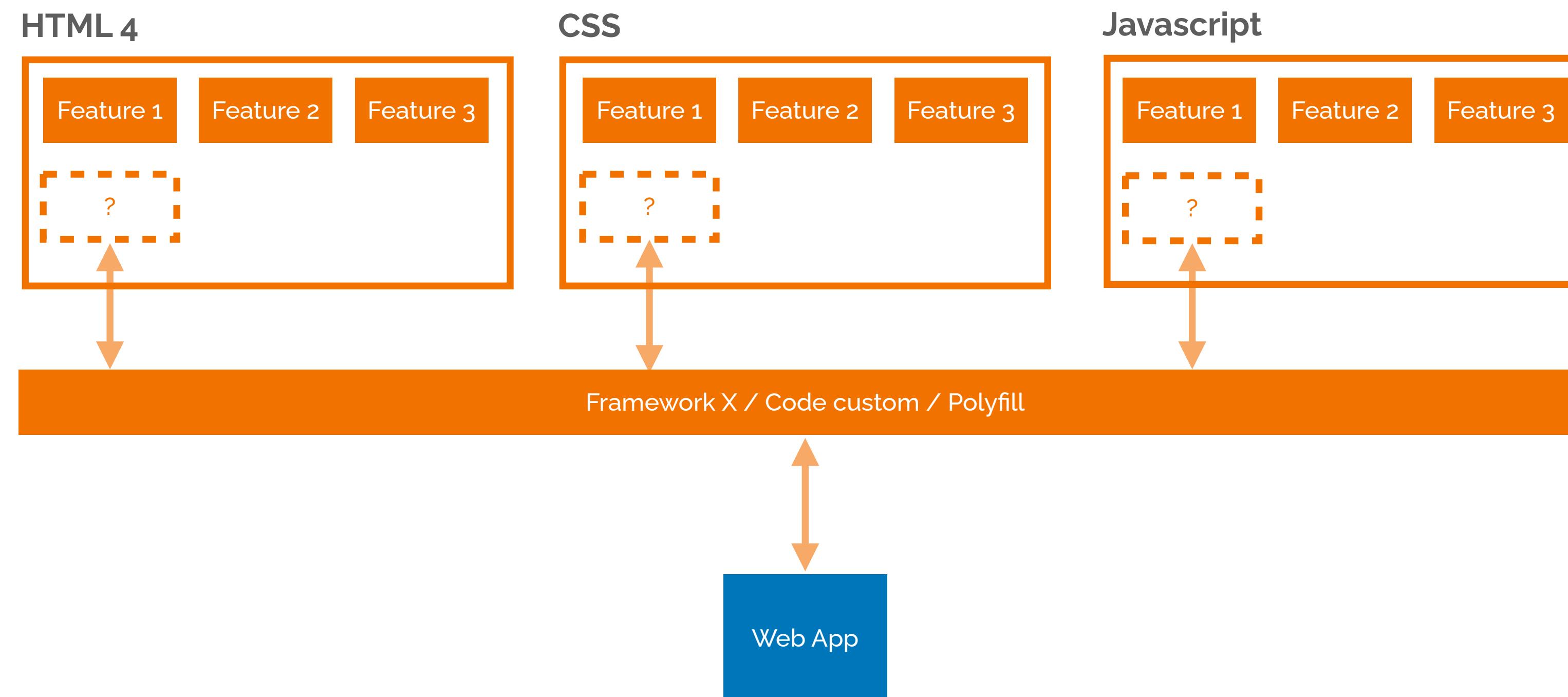
Nouveau standard HTML

HTML / CSS / JS

“Vers une fin des dépendances et une collaboration inter-langages”

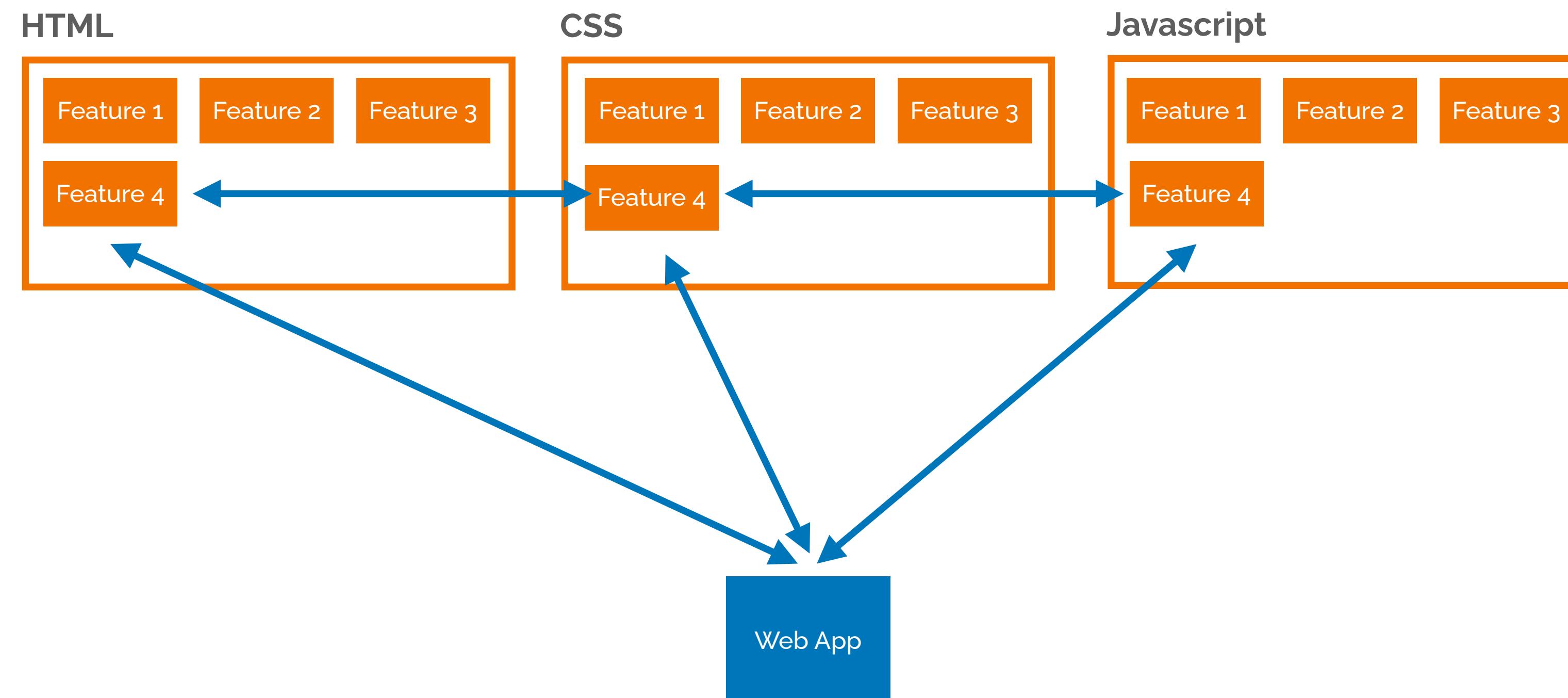
Standardisations des use cases courants

HTML / CSS / JS



Standardisations des use cases courants

HTML / CSS / JS



Polyfill

HTML / CSS / JS

- Un “Polyfill” est le terme générique donné à un bout de code permettant de combler une feature manquante dans un browser
- Permet la retrocompatibilité
- Le plus connu : Babel Polyfill

Standardisations des use cases courants

HTML / CSS / JS

- Validations de formulaires
- Lecteur audio/vidéo
- Eléments complexes (progress bar, ...) -> Shadow DOM !
- Sélectionner des éléments impaires
- Etc...

Nouveau standard HTML

HTML / CSS / JS

- Amélioration de HTML 4
- Fortement lié à Javascript
- Nouveaux éléments sémantiques (header, main, section, article, ...)
- Form inputs améliorés (types d'inputs, claviers mobiles, validations, transformations)
- APIs (géolocalisation, local storage, history, push, service workers, ...)
- Distribution automatique des éléments
- Custom elements

HTML

Image responsives

HTML

```

```

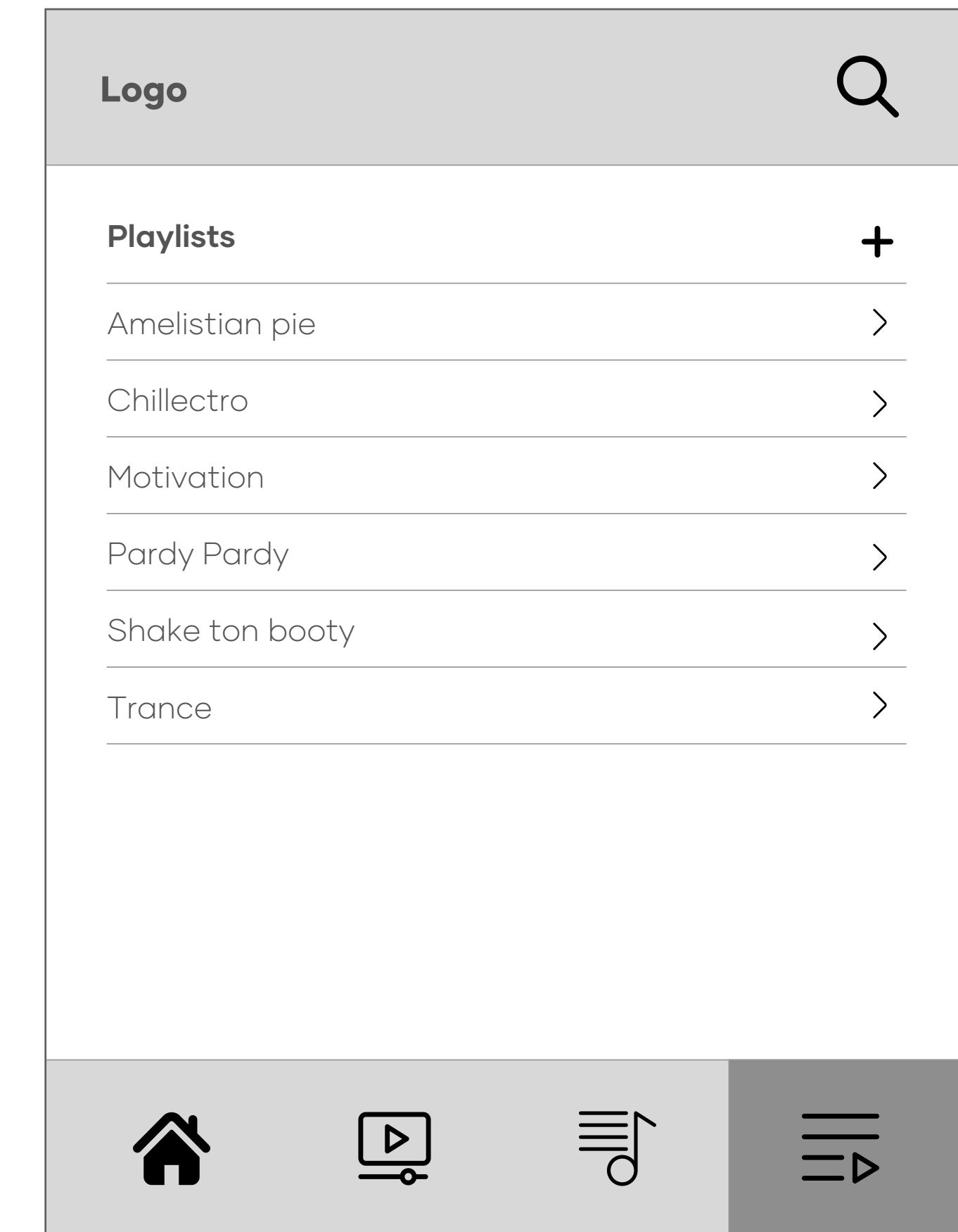
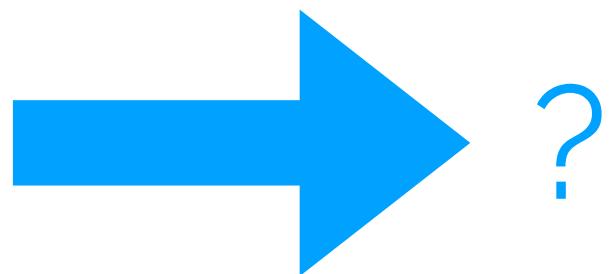
```

```

Eléments sémantiques

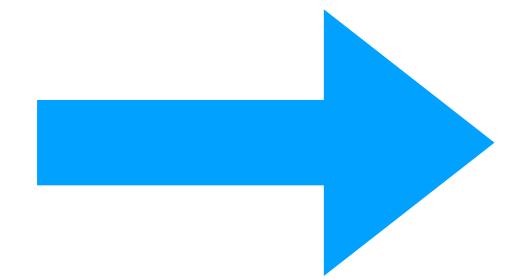
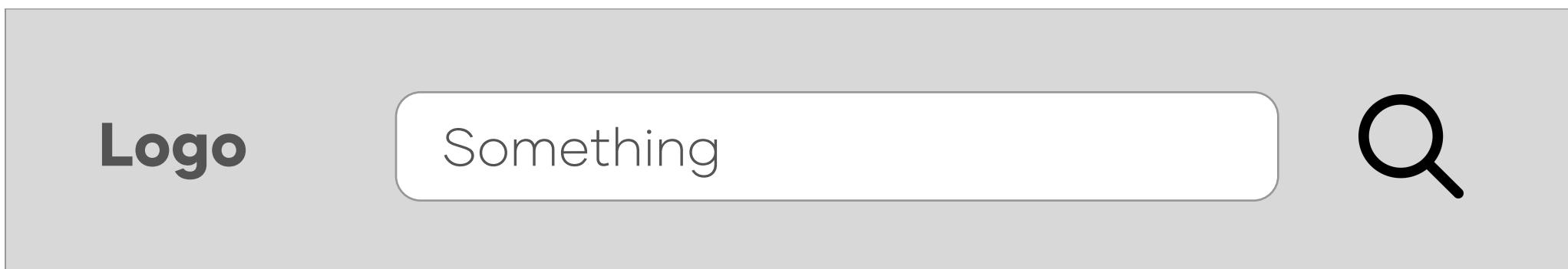
HTML

```
<header />  
<nav />  
<main />  
<section />  
<footer />  
<article />
```



Inputs on steroids

HTML

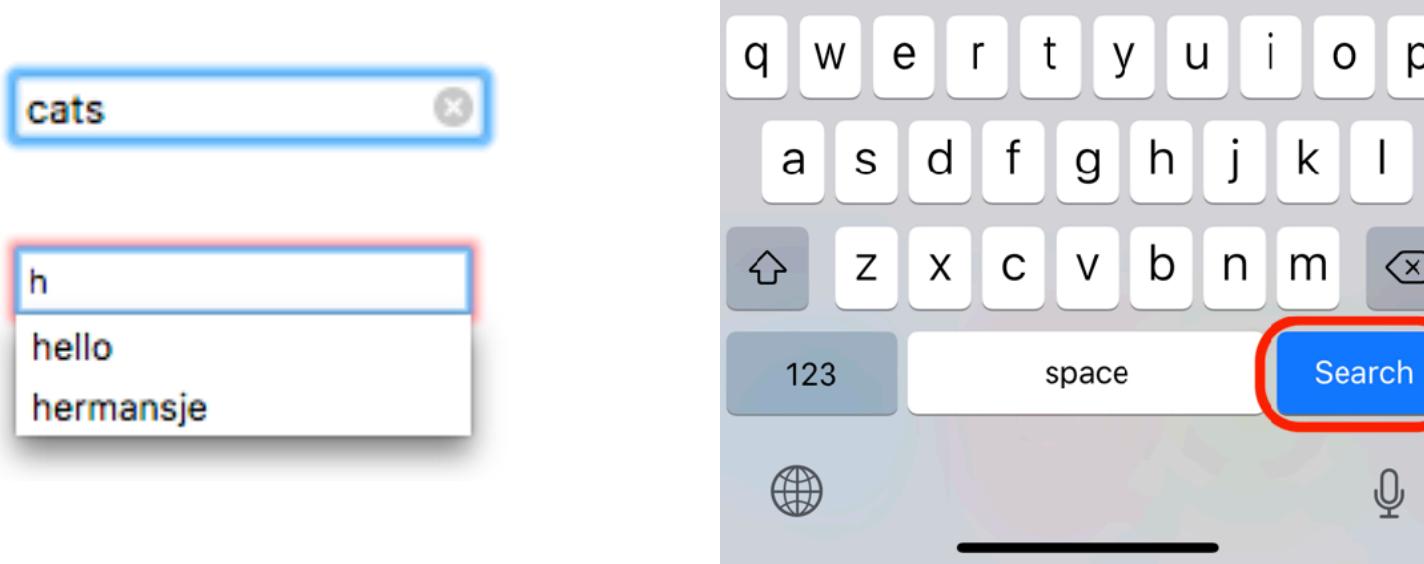


?

```
<input type="search" />
```

Inputs on steroids

HTML



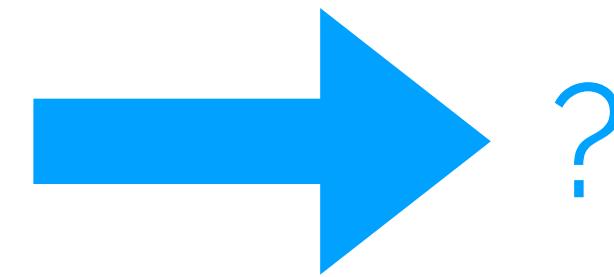
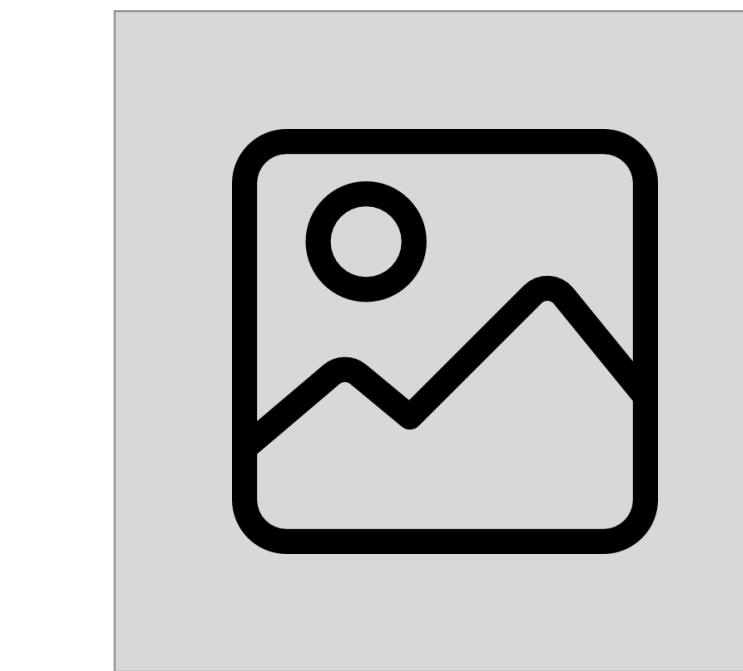
```
<input  
    type="search"  
    spellcheck="false"  
    autocapitalize="false"  
    autofocus  
/>
```



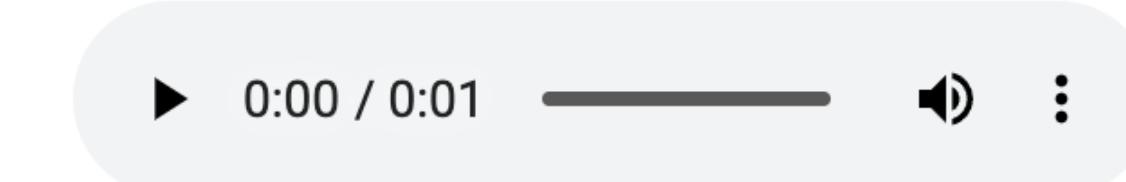
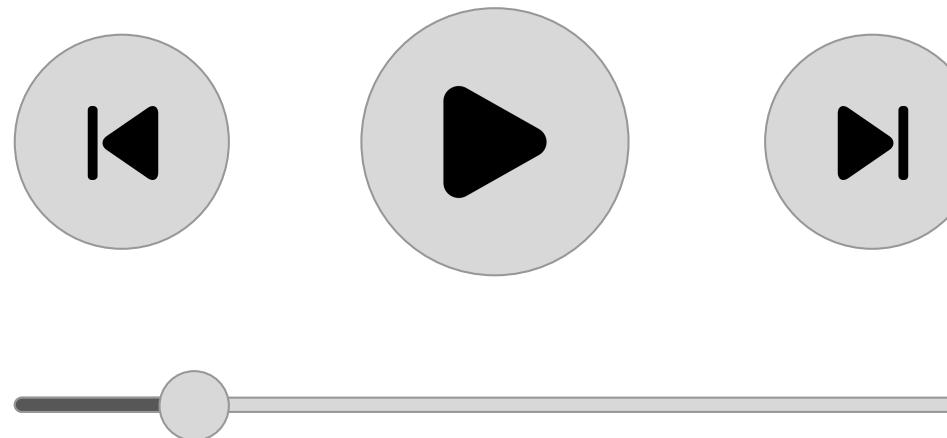
autofocus et autofocus="true" sont sémantiquement équivalents

Audio element

HTML



? <audio src="https://..." />



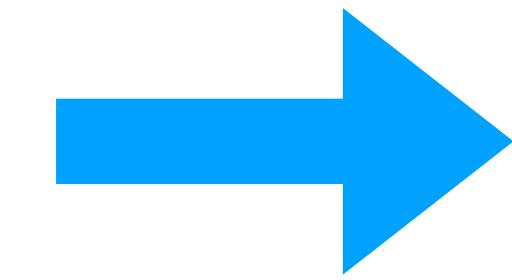
Limitations...

HTML and a bit more...

```
<input type="date" />
```

Progress v1

HTML

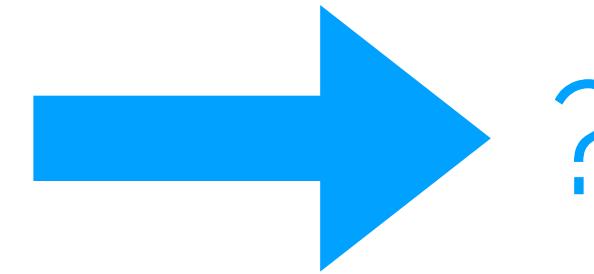


?

```
<progress value="20" max="100" />
```

Progress v2

HTML

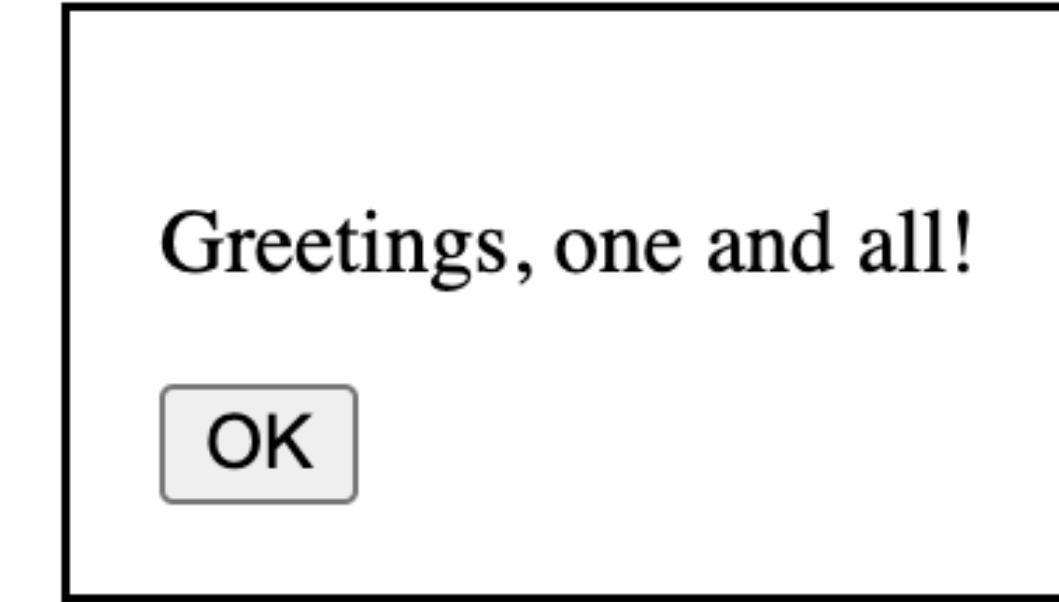
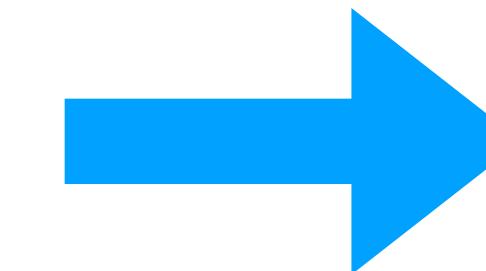


```
<input  
  type="range"  
  value="20"  
  min="0"  
  max="100"  
/>
```

Modal

HTML

```
<dialog open>  
  
  <p>Greetings, one and all!</p>  
  
  <form method="dialog">  
    <button>OK</button>  
  </form>  
  
</dialog>
```



Customer l'arrière plan avec le pseudo-element ::backdrop

css

Browser defaults

CSS

- Chaque browser dispose d'une feuille de style interne par défaut
- Sensé être normé et cross-browser compatible
- En réalité... Pas vraiment.
- Utilisation de feuilles de style CSS reset

CSS reset/reboot/normalize/younameit CSS

- Permet de charger des styles par défaut consistants
- Assurance d'une compatibilité cross-browser à 100%
- Pléthore de versions online
- La plus répandue <https://necolas.github.io/normalize.css/>
(utilisée par Bootstrap, Twitter, GitHub, ...)

CSS icons

CSS

- Actuellement Google Material icons
- Intégré en mode CDN
- <https://fonts.google.com/icons>
- Exemple : <face>
- Libre à vous !

En parlant de CDN...

Tips & tricks

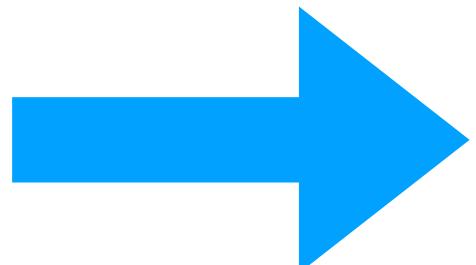
Super CDN pour des images **placeholder**, peu importe votre projet :

placekitten.com

Usage: [http://placekitten.com/\[width\]/\[height\]](http://placekitten.com/[width]/[height])

```

```



Unités CSS

CSS

Deux types d'unités :

- Unités absolues - px, cm, pt, in, ...
- Unités relatives - %, em, rem, vw, vh, ...

Unités CSS – em

CSS

- 1 em = Taille de police à 100% de l'élément parent
- Si pas d'élément parent, 1 em est égal à la hauteur d'une lettre en taille standard, selon la résolution d'écran

Unités CSS - em

CSS

Pas de parent ? Résolution par défaut. Exemple: 16px

```
<body>  
<div>  
  <p>Hello</p>  
</div>  
</body>
```

```
body {  
  font-size: 1em;  
}  
  
div {  
  font-size: 0.75em;      3/4 du parent -> 9 pixels  
}  
  
p {  
  font-size: 0.75em;  
}
```

Unités CSS - rem

CSS

- 1 rem = Taille de police à 100% de l'élément root (**Root EM**)
- Si pas d'élément parent, 1 rem est égal à la hauteur d'une lettre en taille standard, selon la résolution d'écran

Unités CSS - rem

CSS

Pas de parent ? Résolution par défaut. Exemple: 16px

```
<body>  
  <div>  
    <p>Hello</p>  
  </div>  
</body>
```

```
body {  
  font-size: 1rem;  
}  
  
div {  
  font-size: 0.75rem;      3/4 du root -> 12 pixels  
}  
  
p {  
  font-size: 0.75rem;  
}
```

Unités CSS – rem

CSS

- Le REM est beaucoup plus utilisé, typiquement pour des design responsive ou dans des frameworks CSS
- En définissant toutes les tailles de polices et marges en rem, d'après l'élément root, une seule adaptation du CSS met à jour l'entier du document

Unités CSS - rem

CSS

```
body {  
    font-size: 16px;  
}  
  
h1 {  
    font-size: 2.5rem;  
    margin-bottom: 1rem;  
}  
  
h2 {  
    font-size: 2rem;  
    margin-bottom: 0.75rem;  
}  
...
```

Unités CSS - vw, vh

CSS

- vw = viewport width
- vh = viewport height
- Très utile pour donner la largeur ou la hauteur de l'écran à un élément, sans devoir chaîner des **height: 100%** sur tous les parents

Variables CSS

CSS

- CSS dispose aussi de variables, comme Javascript par exemple
- Une variable commence toujours par "--"
- Ce sont en fait des custom properties... mais utilisées comme variables !
- Une variable peut être déclarée sur n'importe quel déclaration, mais ne sera accessible que par l'élément et ses enfants

Variables CSS

CSS

Déclaration

```
body {  
  --text-color: #00ff00;  
}
```

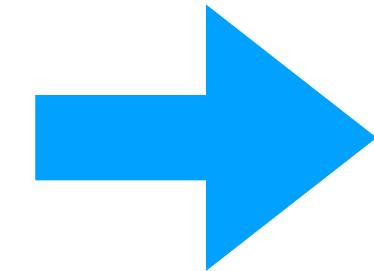
Utilisation

```
h1 {  
  color: var(--text-color);  
}
```

Variables CSS – Portée CSS

Exemple

```
header {  
  --text-color: #00ff00;  
}
```



```
footer {  
  /* rien. */  
}
```

```
header h1 {  
  color: var(--text-color); /* Yes! */  
}  
  
footer h1 {  
  color: var(--text-color); /* Nope. Invalide */  
}
```

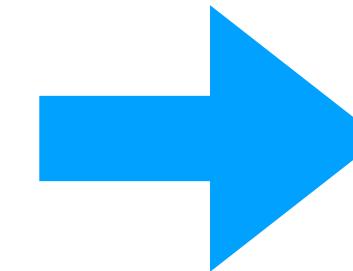
Variables CSS - Déclaration idéale

CSS

```
/* Utiliser le pseudo element :root */  
  
:root {  
  --text-color: #00ff00;  
}
```

Variables CSS - Variable d'une variable CSS

```
:root {  
  --primary-color: #00ff00;  
  
  --link-color: var(--primary-color)  
}
```



--primary-color => #00ff00
--link-color => #00ff00

Pseudo-classes

CSS

- Beaucoup de pseudo-classes...
- Les historiques :**hover**, :**active**, :**visited**
- Les game-changes :**has**, :**not**, :**last-child**, :**valid**, :**user-valid**, :**checked**, ...

Pseudo-classes

CSS

```
a {  
    color: #000000;  
}
```

Lien

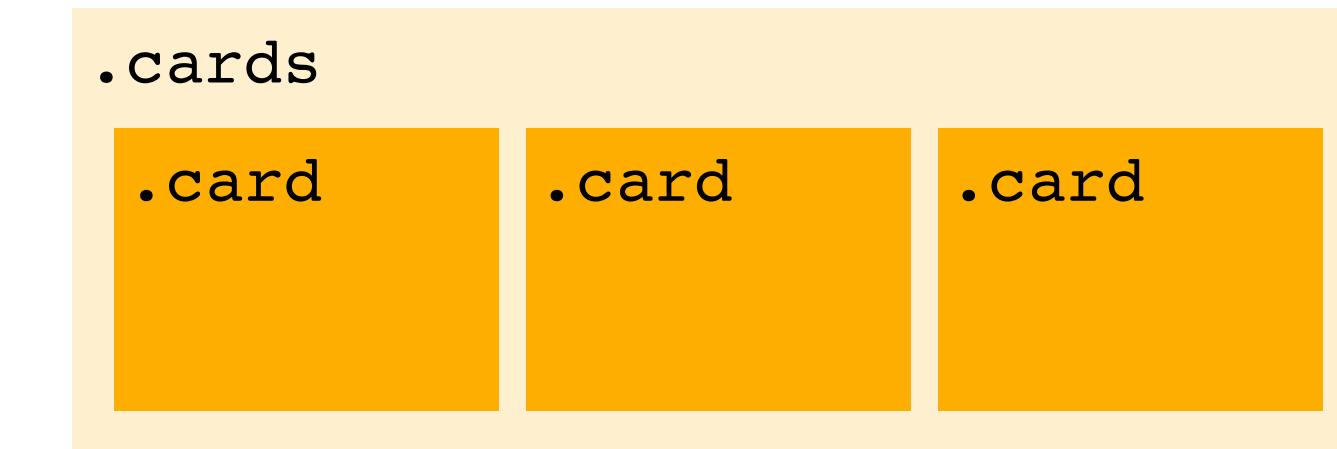
```
a:hover {  
    color: #00ff00;  
}
```

Lien 

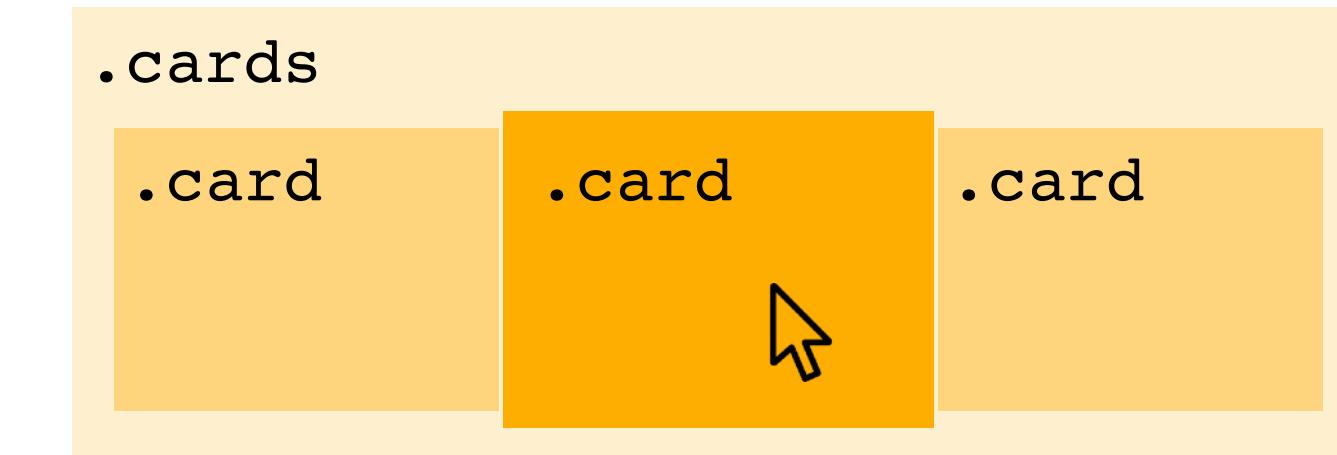
Pseudo-classes

CSS

```
.cards .card:hover {  
    transform: scale(1.1);  
}
```



```
.cards:has(.card:hover) .card:not(:hover) {  
    opacity: 0.4;  
}
```



Transitions CSS

CSS

- Tiens sur une ligne ! Animations du flemmard !
- Il suffit de dire :
 - Quoi
 - Combien de temps
 - Comment

Transitions CSS

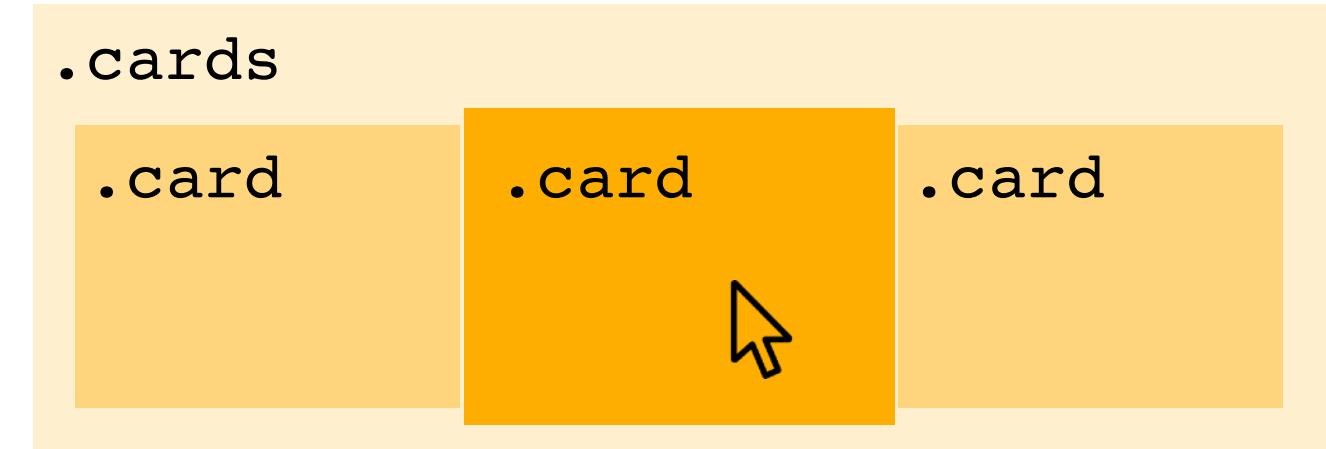
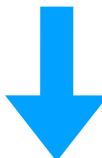
CSS

- Tiens sur une ligne ! Animations du flemmard !
- Il suffit de dire :
 - Quoi
 - Combien de temps
 - Comment

Transitions CSS

CSS

```
.cards .card {  
    transition: all 500ms ease-in-out;  
    // ou  
    transition: transform 500ms ease-in-out, opacity 500ms ease-in-out;  
}  
  
.cards .card:hover {  
    transform: scale(1.1);  
}  
  
.cards:has(.card:hover) .card:not(:hover) {  
    opacity: 0.4;  
}
```

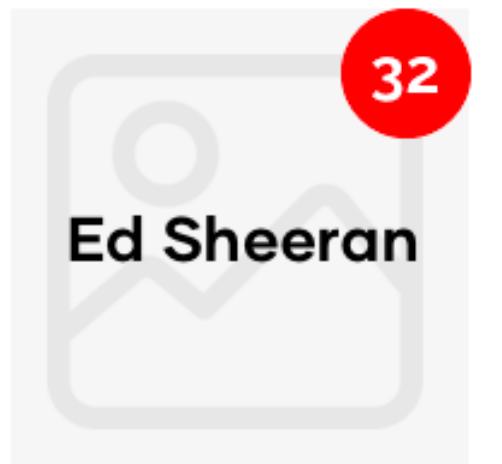


Data attributes

CSS

```
<div  
  data-id="1"  
  data-name="Ed Sheeran"  
  data-count="32"  
  data-image="https://..."  
>  
</div>
```

```
div {  
  /*__/_!\\ Bientôt...__*/  
  background-image: attr(data-image);  
  
  /*__Possible! Content only __*/  
  content: attr(data-name);  
}  
  
div:before {  
  /*__Possible! Content only __*/  
  content: attr(data-count);  
  
  position: absolute;  
  top: 0;  
  right: 0;  
  border-radius: 50%;  
  background-color: #ff0000;  
  color: #fff;  
  text-align: center;  
}
```



32

Ed Sheeran

Viewport

CSS

- Le viewport est la zone de la fenêtre dans laquelle le contenu web peut être vu
- Le viewport est souvent plus grand que la zone affichée par le navigateur ->
La view

Viewport CSS



https://developer.mozilla.org/fr/docs/Web/HTML/Viewport_meta_tag

Viewport - Déjà vu ? Vue compactée ?

CSS



https://developer.mozilla.org/fr/docs/Web/HTML/Viewport_meta_tag

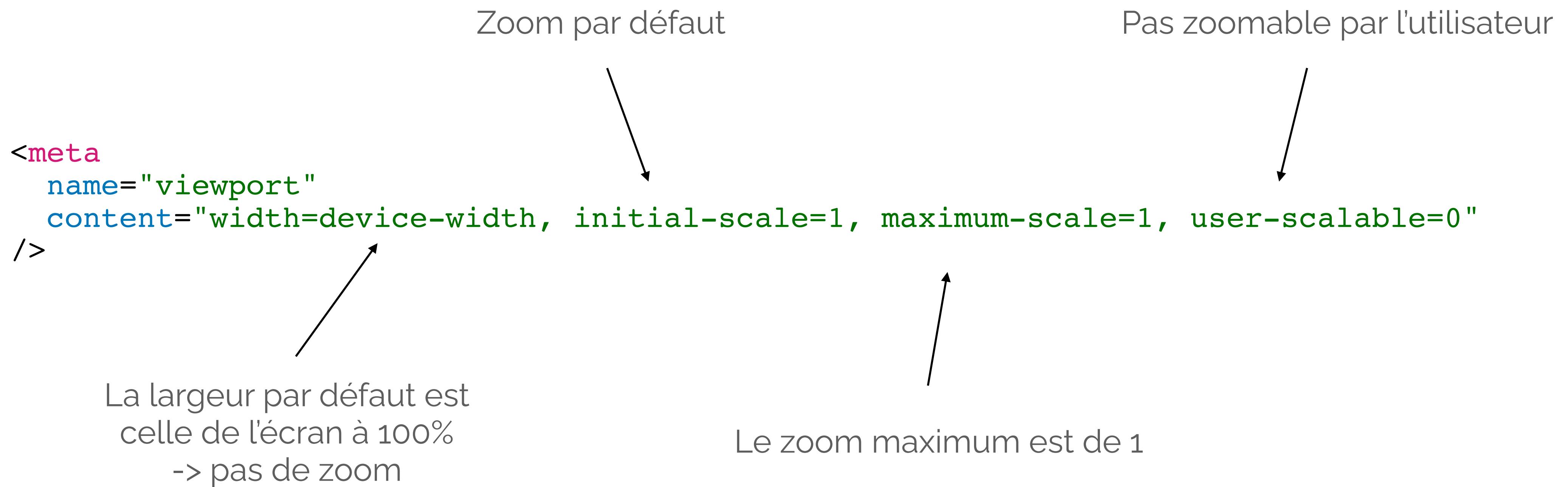
Viewport – Autozoom

CSS

- Les résolutions des versions mobiles sont de plus en plus précises et on ne distingue plus les pixels
- Un mécanisme d'autozoom est utilisé par les navigateurs pour renderer une version plus large de la page et la redimensionner à une résolution plus petite
- Par exemple, si l'écran d'un téléphone mobile a une largeur de 640 pixels, les pages peuvent être affichées dans une fenêtre virtuelle de 980 pixels, puis réduites pour tenir dans l'espace de 640 pixels.

Viewport - La balise meta viewport

CSS



Media queries

CSS

- Design responsive
- Blocs CSS conditionnels liés au type de display ou à sa taille
- Applicable à certains tags HTML via l'attribut `media=`
- Utilisable via Javascript pour tester et surveiller

Media queries – Blocs CSS

```
@media [not|only] mediatype and (mediafeature [and|or|not] mediafeature) {  
    ...  
    CSS-Code;  
    ...  
}
```

mediatype = all | print | screen | speech

mediafeature = [max-width | min-height | orientation | ...]: value

Media queries – Blocs CSS

Exemple 1

```
.container {  
  display: flex;  
  flex-direction: row;  
  ...  
}  
  
@media (max-width: 767px) {  
  .container {  
    flex-direction: column;  
  }  
}
```

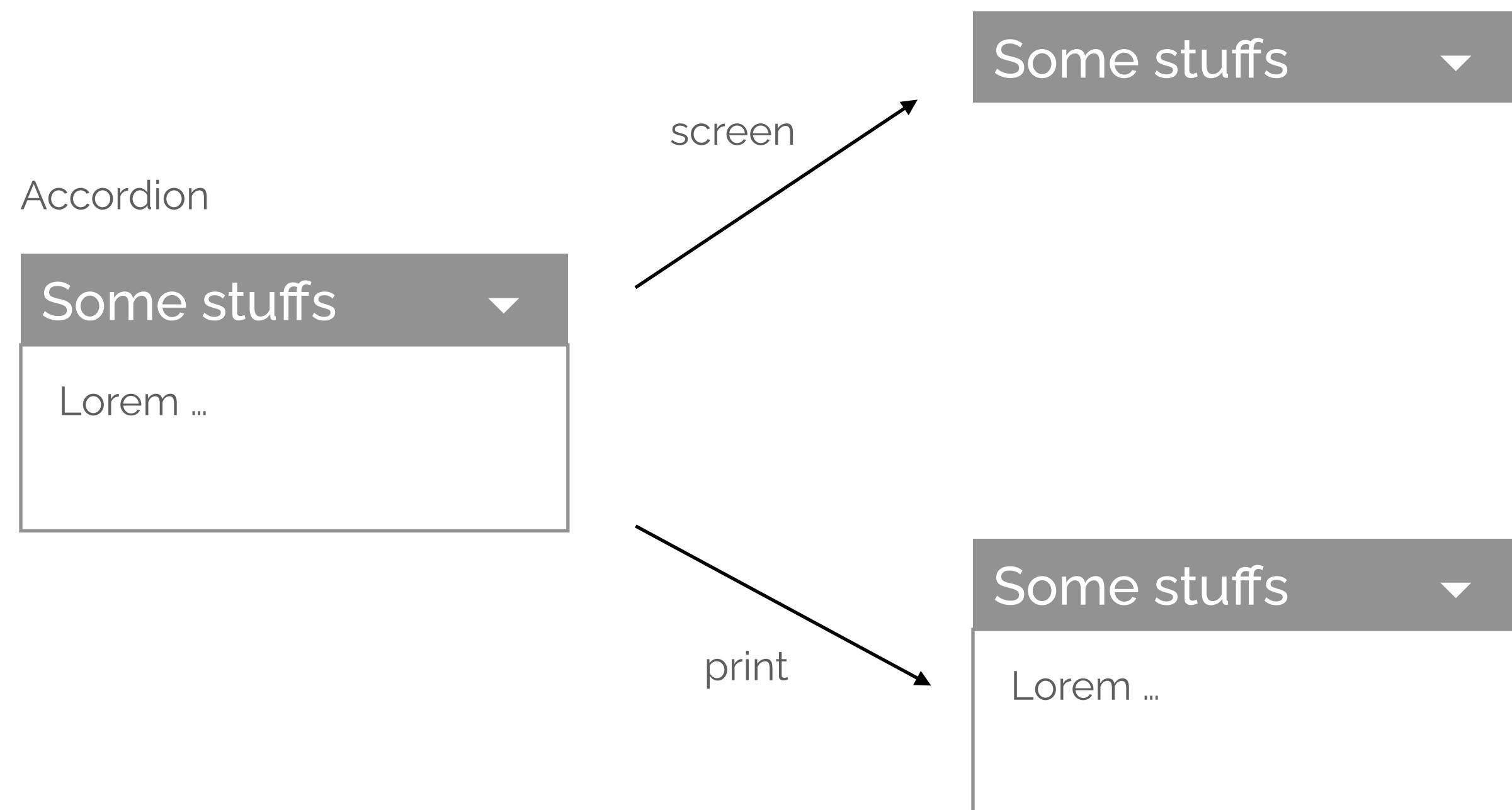
Exemple 2

```
body {  
  background-color: white;  
  ...  
}  
  
@media screen and (prefers-color-scheme: dark) {  
  body {  
    background-color: black;  
  }  
}
```

Media queries – Blocs CSS

Exemple 3

```
.accordion .accordion-content {  
    display: none;  
}  
...  
  
@media print {  
    .accordion .accordion-content {  
        display: block;  
    }  
}
```



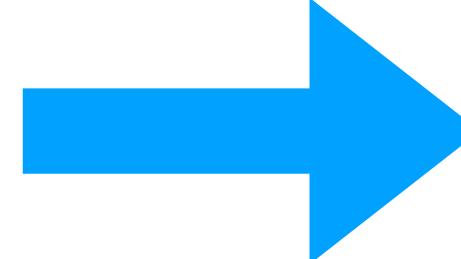
Media queries – Blocs CSS



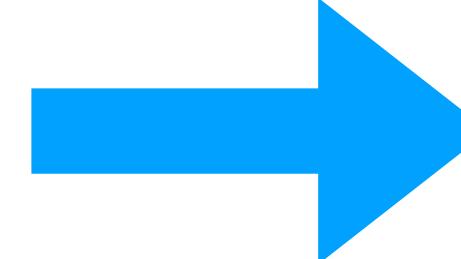
Cumul des règles par qualification des sélecteurs

```
.container {  
  display: flex;  
  flex-direction: row;  
  
  ...  
}  
  
@media (max-width: 767px) {  
  .container {  
    flex-direction: column;  
  }  
}
```

Sélecteur interprété
comme



```
.container {  
  display: flex;  
  flex-direction: row;  
  
  ...  
}  
  
@media (max-width: 767px) .container {  
  flex-direction: column;  
}
```



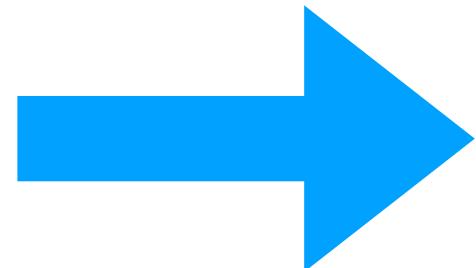
Media queries – Blocs CSS



Cumul des règles par qualification des sélecteurs

```
.container {  
  display: flex;  
  flex-direction: row;  
}  
...  
  
@media (max-width: 767px) .container {  
  flex-direction: column;  
}
```

Comparable à une sur-qualification
du type

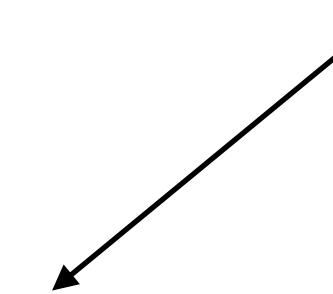


```
.container {  
  display: flex;  
  flex-direction: row;  
}  
...  
  
body .container {  
  flex-direction: column;  
}
```

Media queries – Tags HTML

CSS

Accepte des media queries



```
<link rel="stylesheet" media="screen and (min-width: 1201px)" href="widescreen.css">
<link rel="stylesheet" media="screen and (max-width: 600px)" href="smallscreen.css">
```



Avantages ? Inconvénients ?

Media queries – Breakpoints

CSS

- **320px – 480px** : Mobile devices
- **481px – 768px** : iPads, Tablets
- **769px – 1024px** : Small screens, laptops
- **1025px – 1200px** : Desktops, large screens
- **1201px and more** : Extra large screens, TV

Media queries – Javascript CSS

Tester une media query

```
const mql = window.matchMedia("(orientation: portrait)");

if (mql.matches) {
  /* La zone d'affichage/viewport est en portrait */
} else {
  /* La zone d'affichage/viewport est en paysage */
}
```

Media queries – Javascript CSS

Ajouter un listener sur une media query

```
const mql = window.matchMedia("(orientation: portrait)");
mql.addListener(handleOrientationChange);

function handleOrientationChange(mql){
    if (mql.matches) {
        /* La zone d'affichage/viewport est en portrait */
    } else {
        /* La zone d'affichage/viewport est en paysage */
    }
}
```

Flexboxes

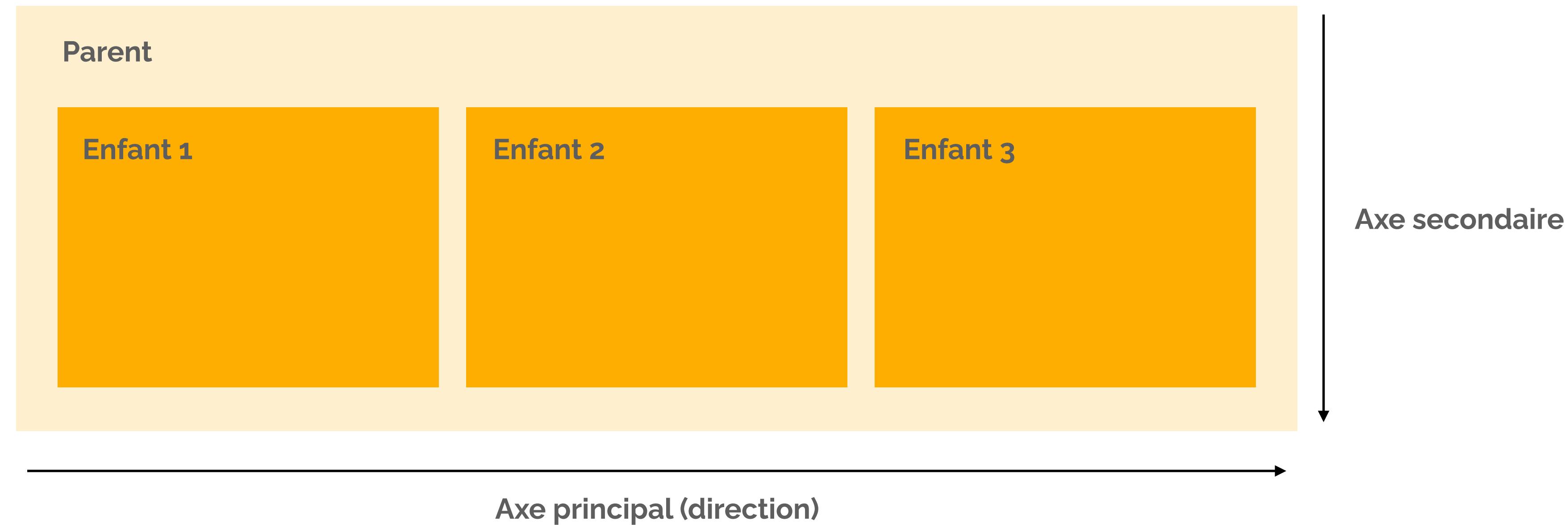
CSS

- Permet de distribuer l'espace entre des éléments, au sein d'un parent
- Unidimensionnel - Ligne ou colonne
- Gère également l'alignement et le dimensionnement
- Comparable à des “float” gérées par le parent
- Très fortement lié au concept de “responsive design”



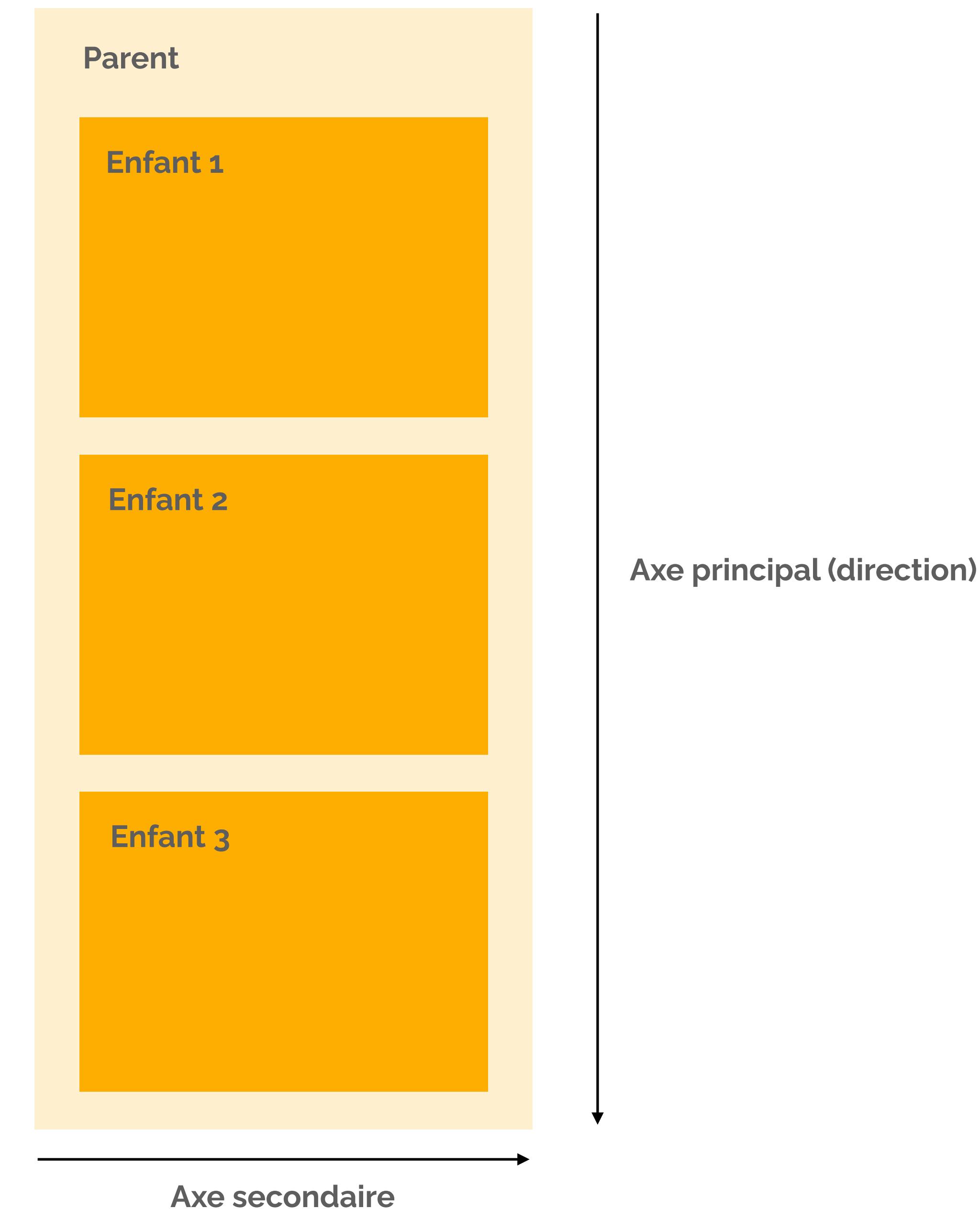
Flexboxes

CSS



Flexboxes

CSS



Flexboxes – Propriétés

CSS

Propriétés du parent

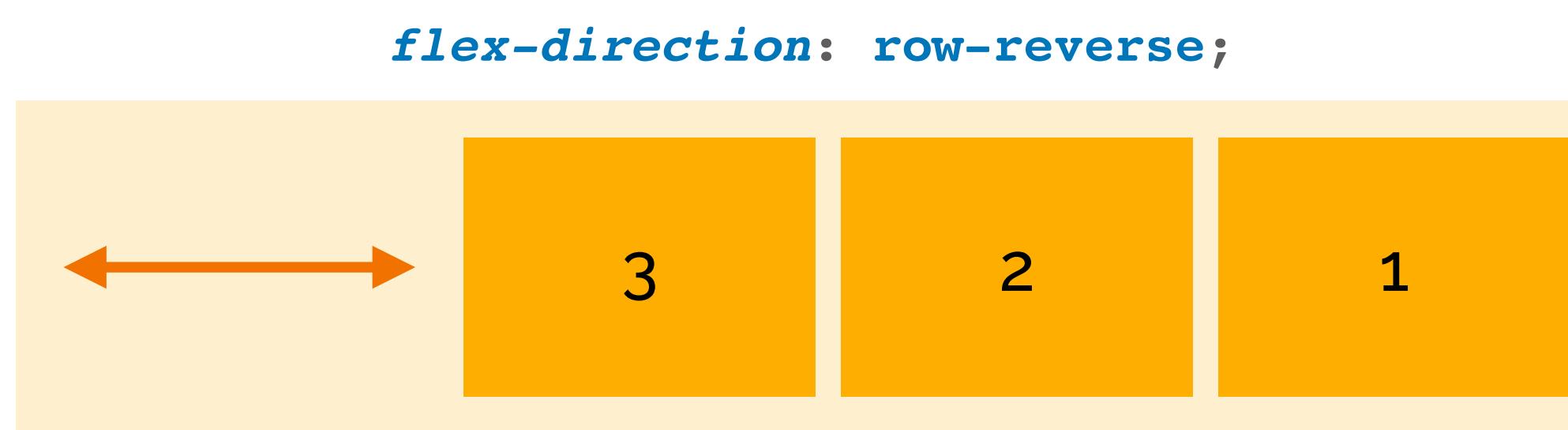
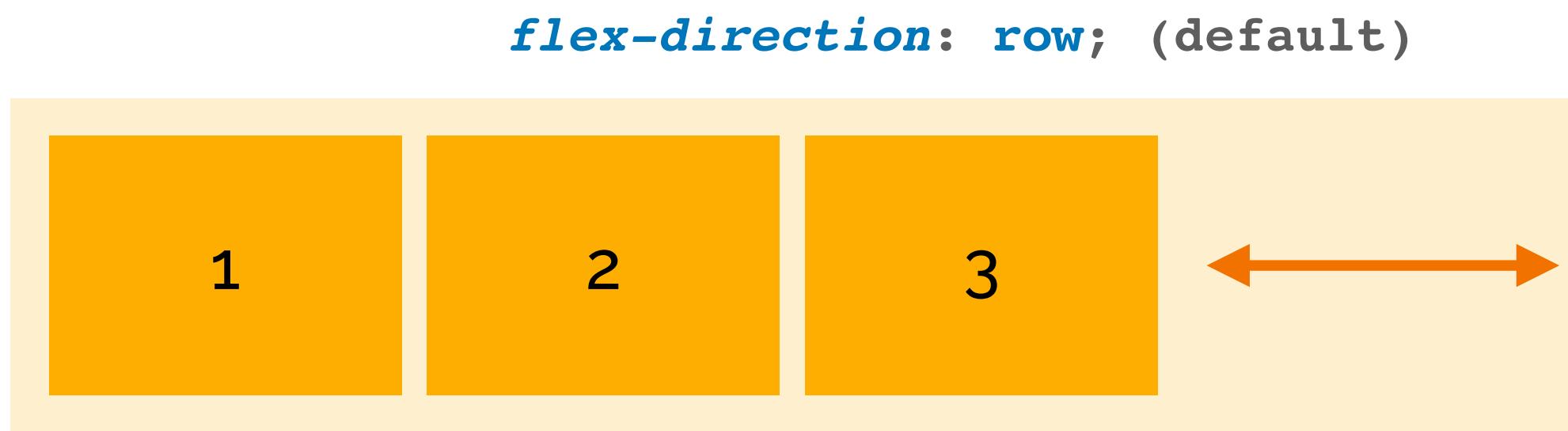
- Direction (`flex-direction`)
- Alignement axe principal (`justify-content`)
- Alignement axe secondaire (`align-items`)
- Multi-lignes (`flex-wrap`)
- Alignement multi-lignes (`align-content`)
- Ecartement (`gap`)

Propriétés des enfants

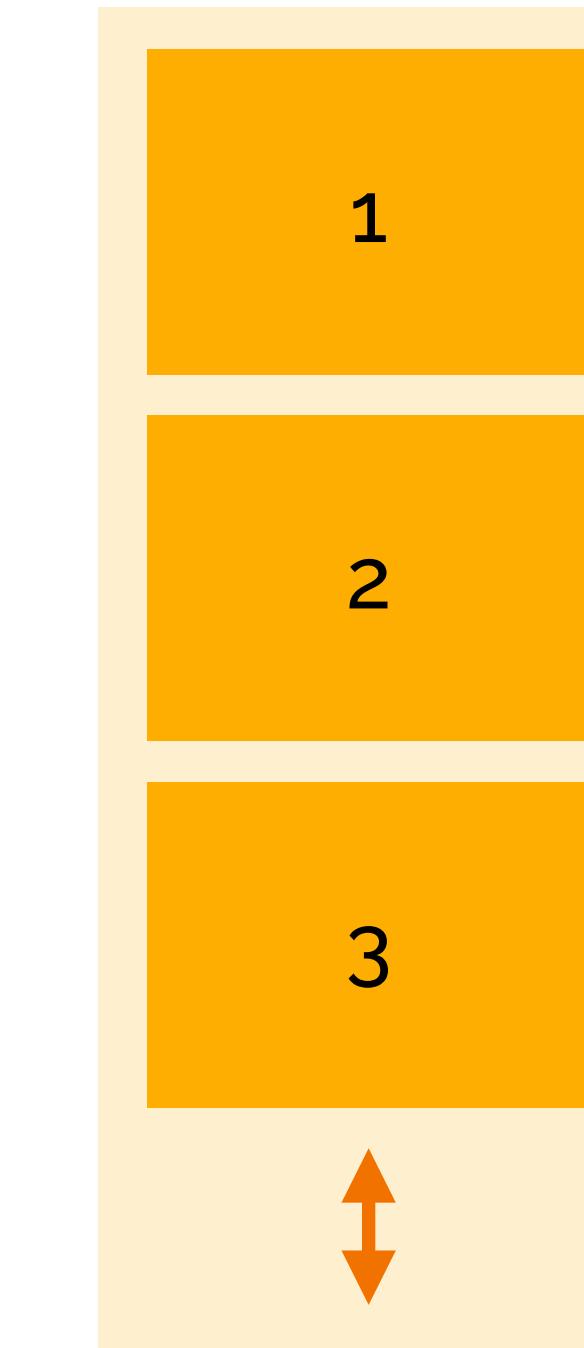
- Taille de base (`flex-basis`)
- Taille du rétrécissement (`flex-shrink`)
- Taille de l'agrandissement (`flex-grow`)
- Overrides des propriétés du parent (`...-self`)

Flexboxes - Direction (flex-direction)

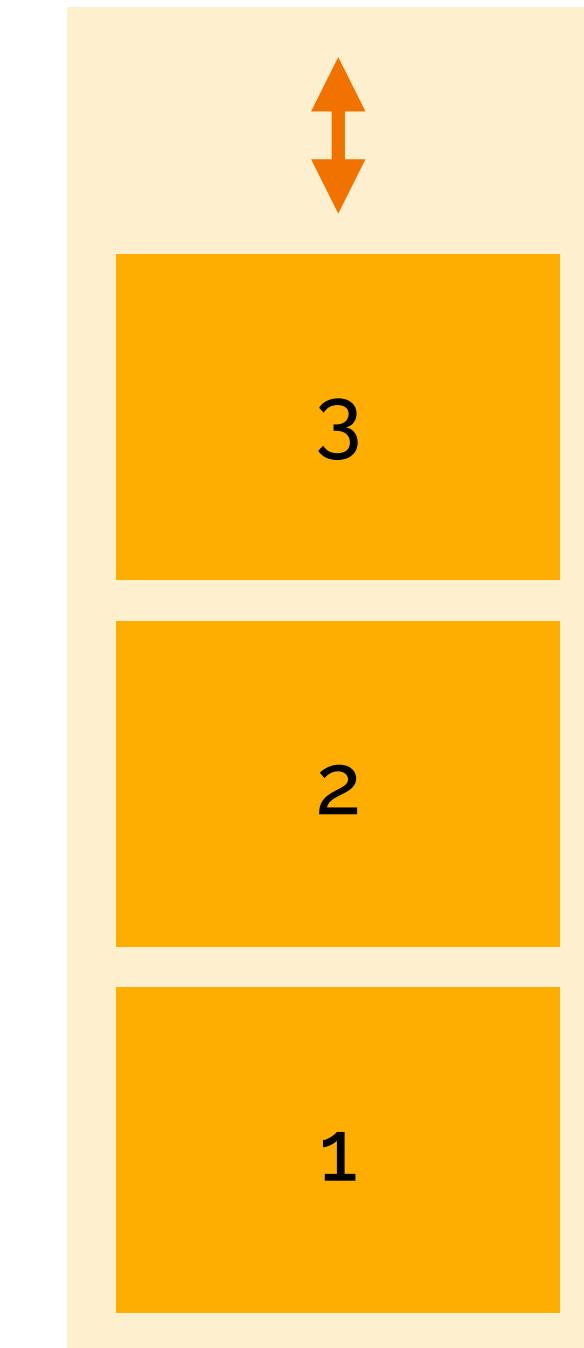
CSS



flex-direction: column;



flex-direction: column-reverse;

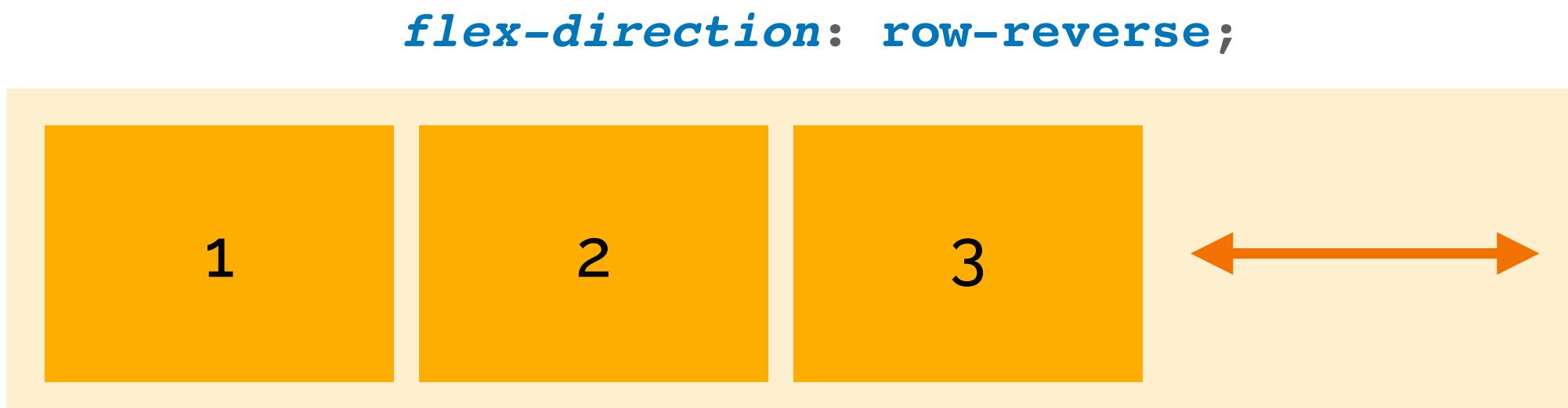
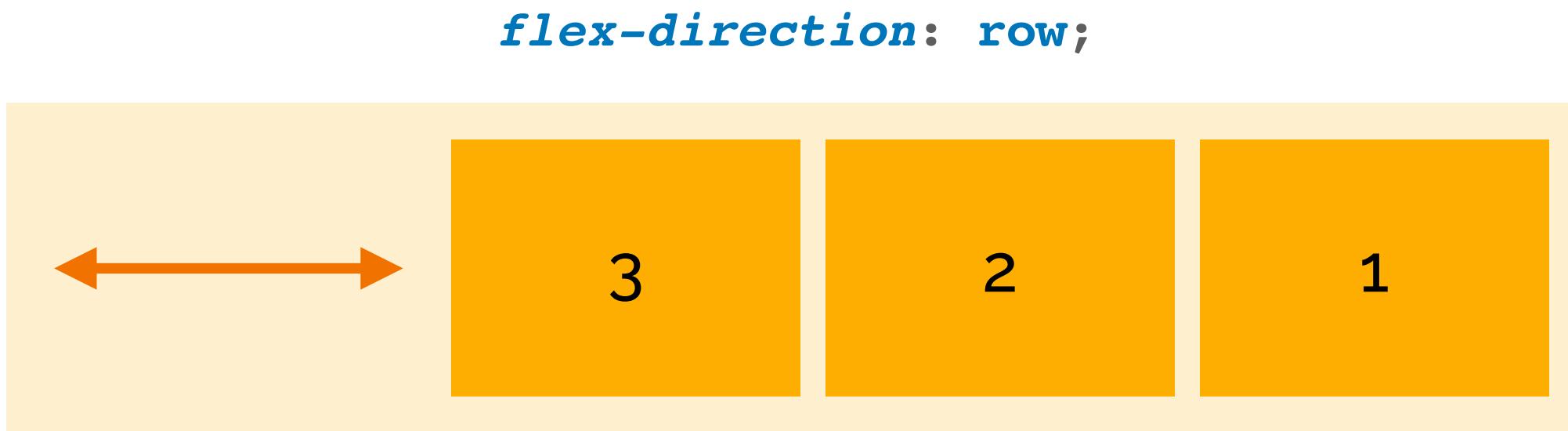


Flexboxes – Direction rtl

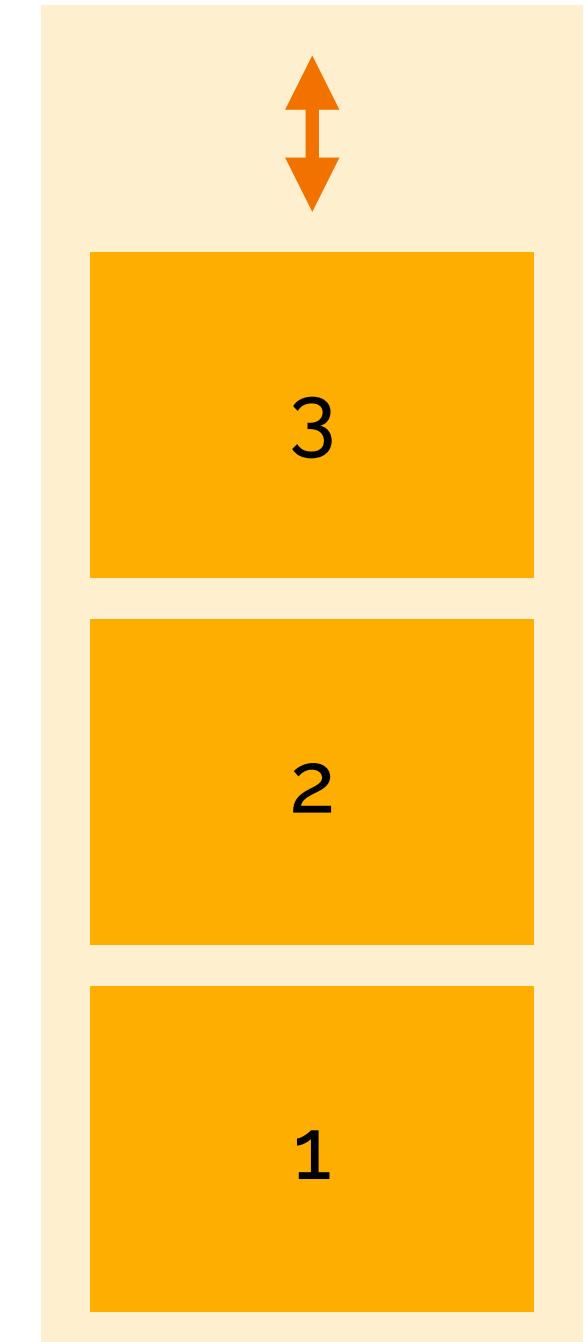
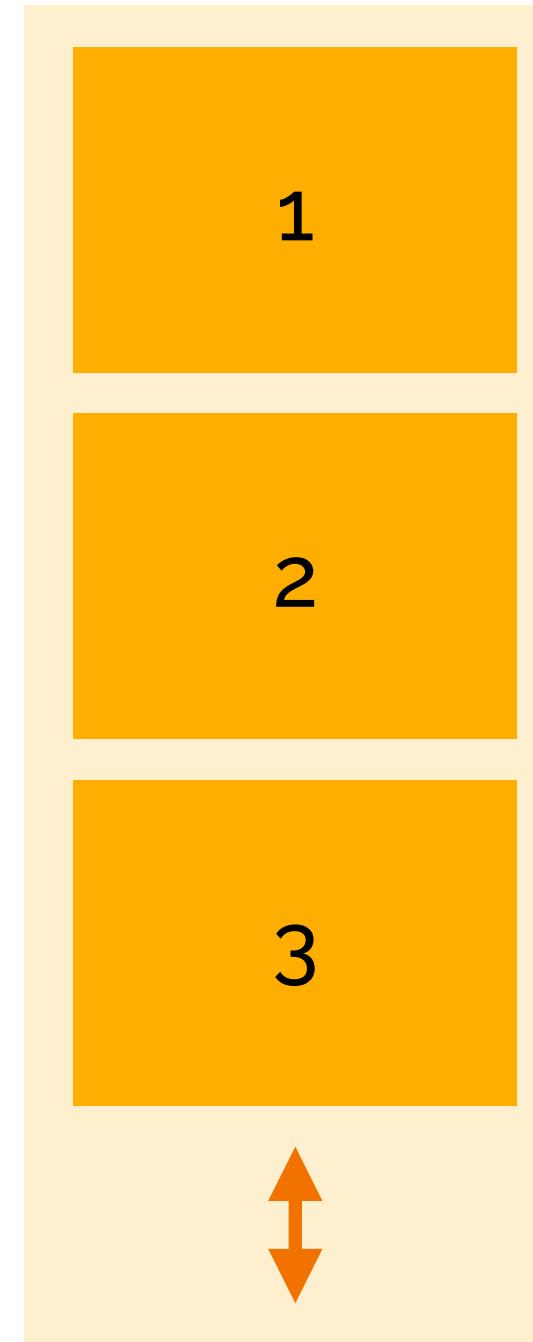
CSS



En écriture droite-gauche (arabe) -> **direction: rtl;**



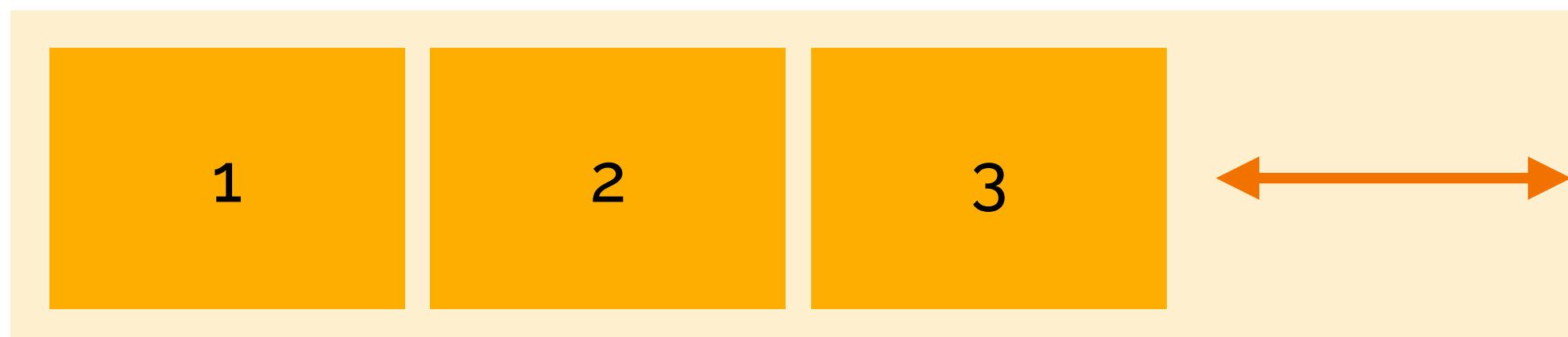
flex-direction: column; **flex-direction: column-reverse;**



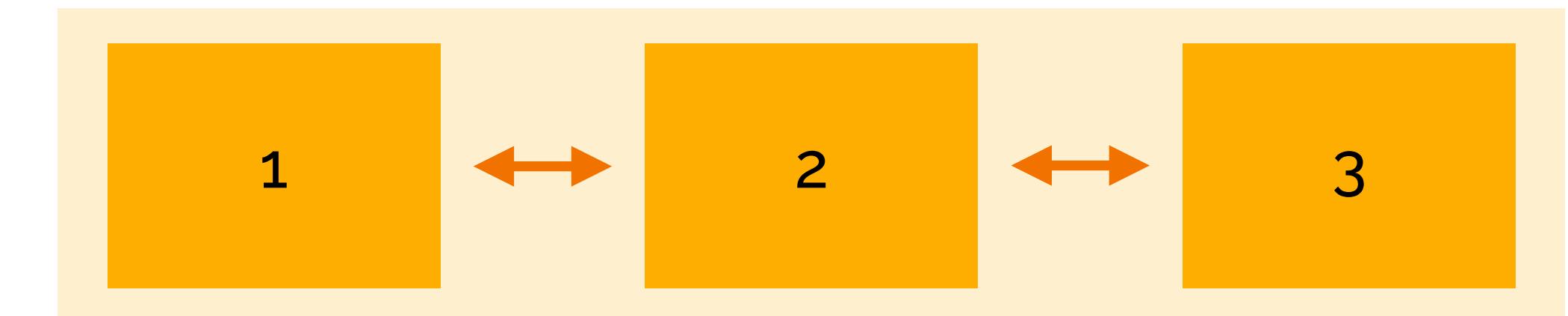
Flexboxes - Alignement axe principal (justify-content)

CSS

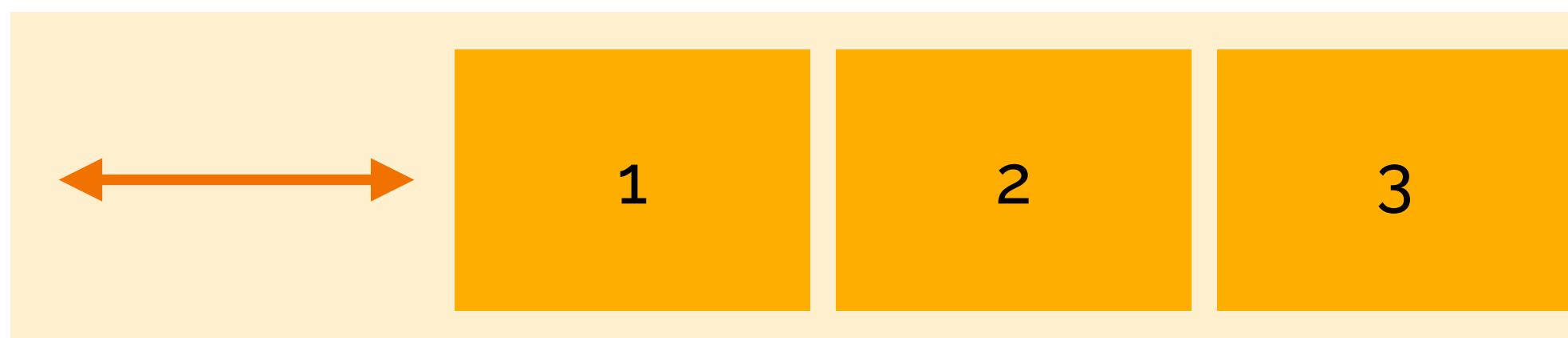
justify-content: flex-start; (default)



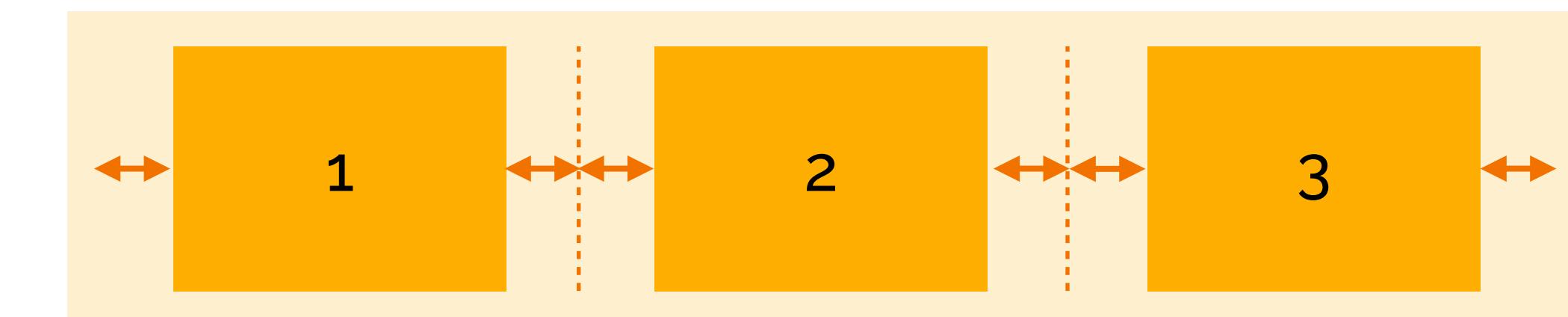
justify-content: space-between;



justify-content: flex-end;



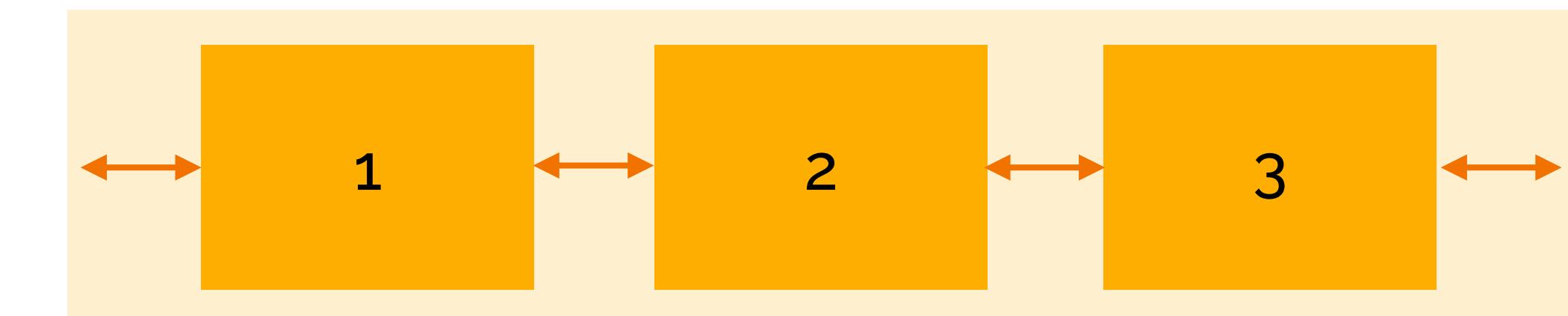
justify-content: space-around;



justify-content: center;



justify-content: space-evenly;



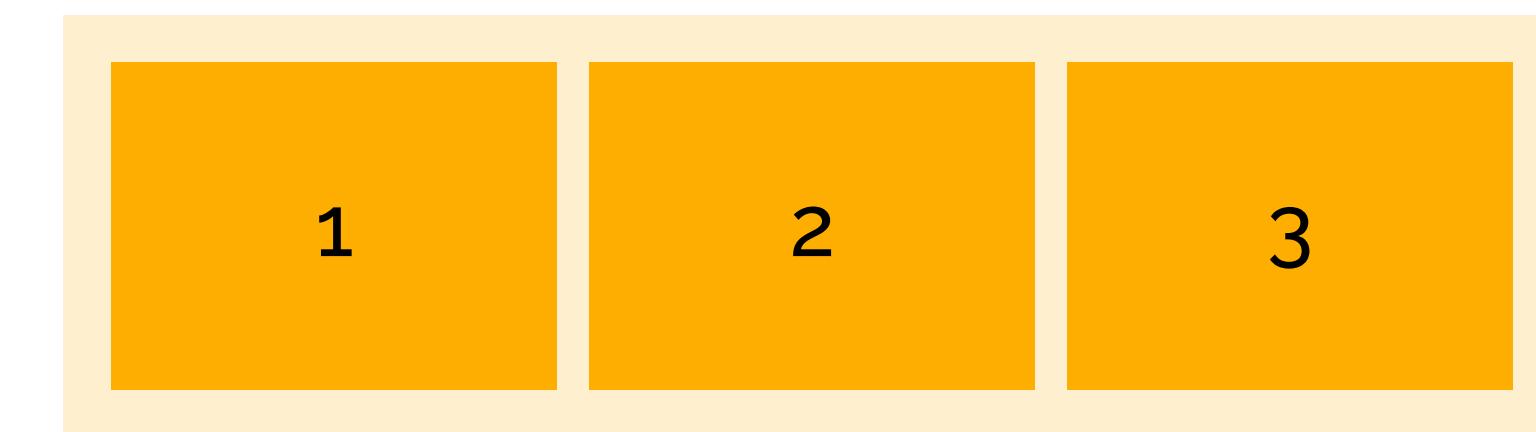
Flexboxes – Alignement axe secondaire (align-items)

CSS

`align-items: flex-start;`



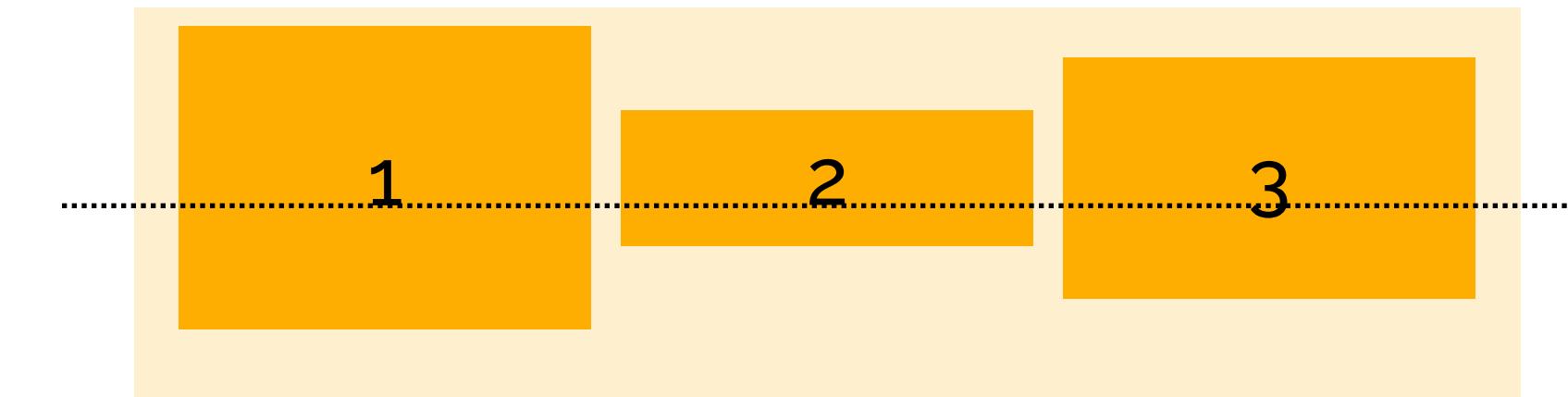
`align-items: stretch; (default)`



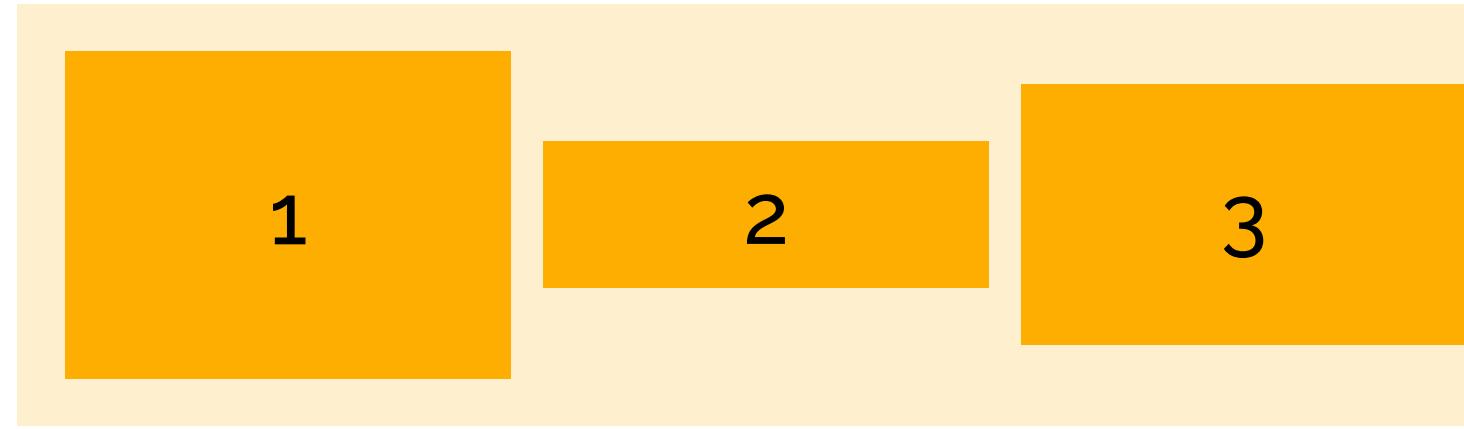
`align-items: flex-end;`



`align-items: baseline;`



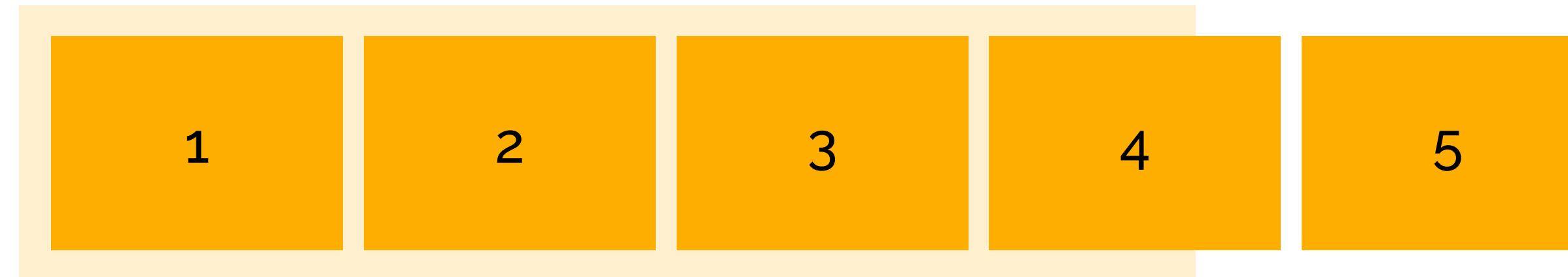
`align-items: center;`



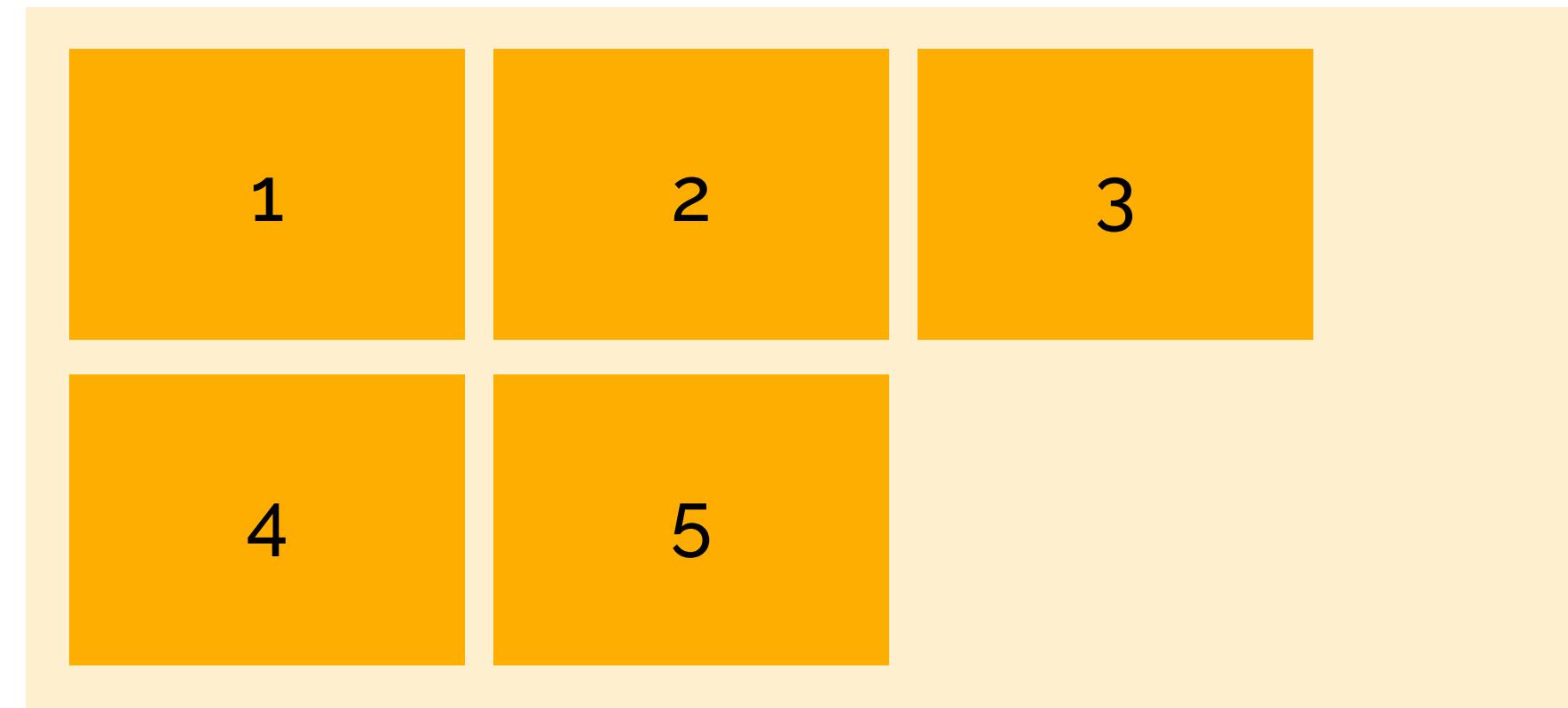
Flexboxes – Multi-lignes (wrap)

CSS

`flex-wrap: nowrap; (default)`



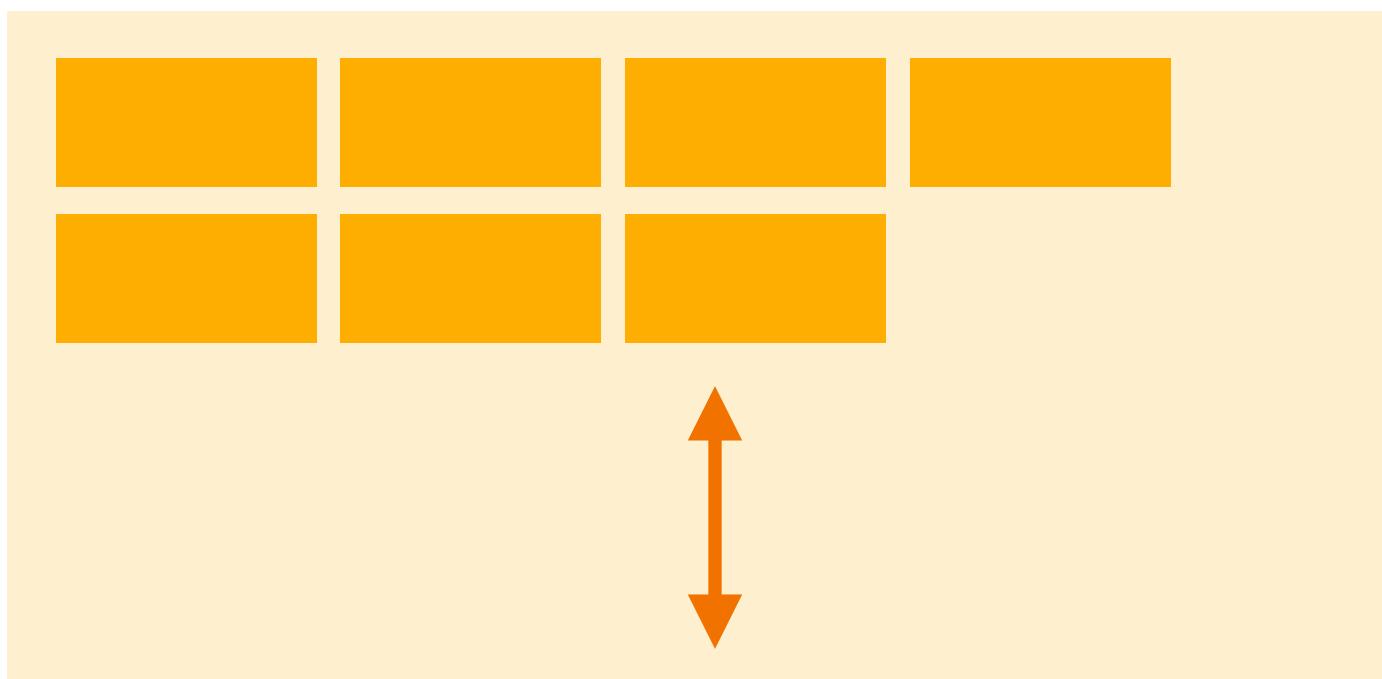
`flex-wrap: wrap;`



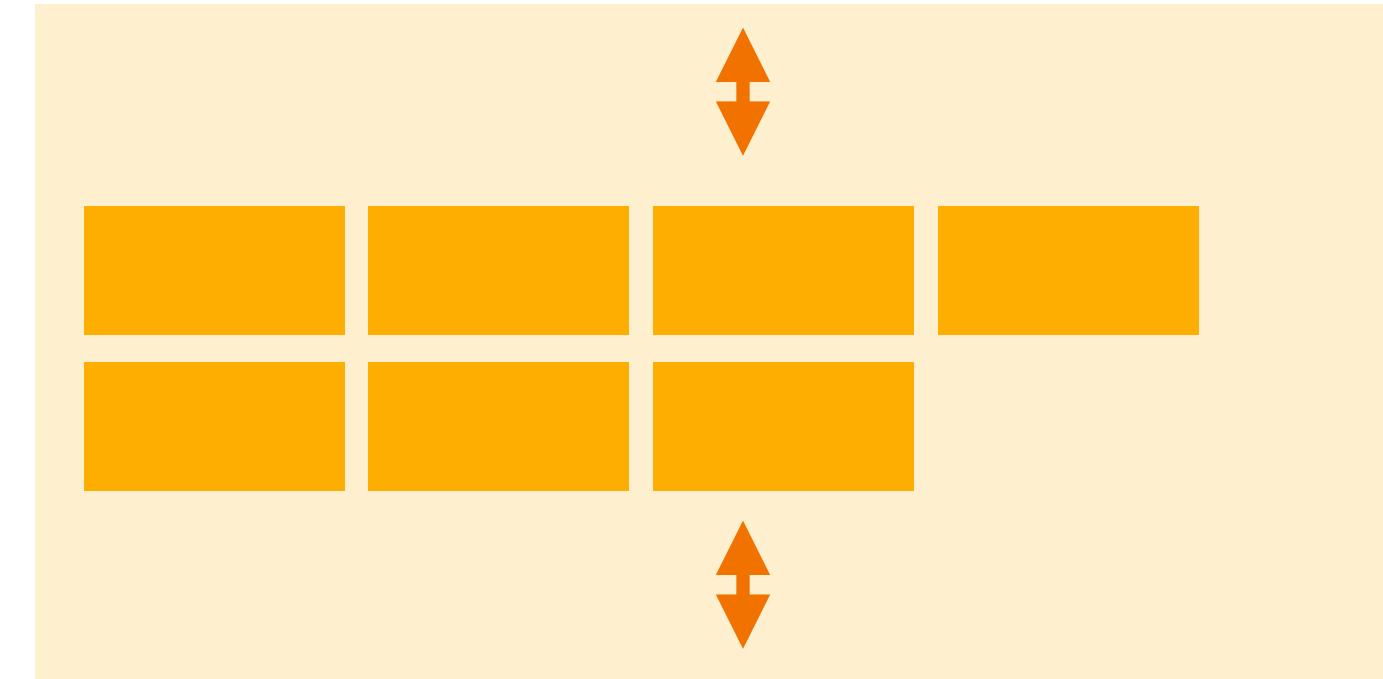
Flexboxes - Alignement multi-lignes (align-content)

CSS

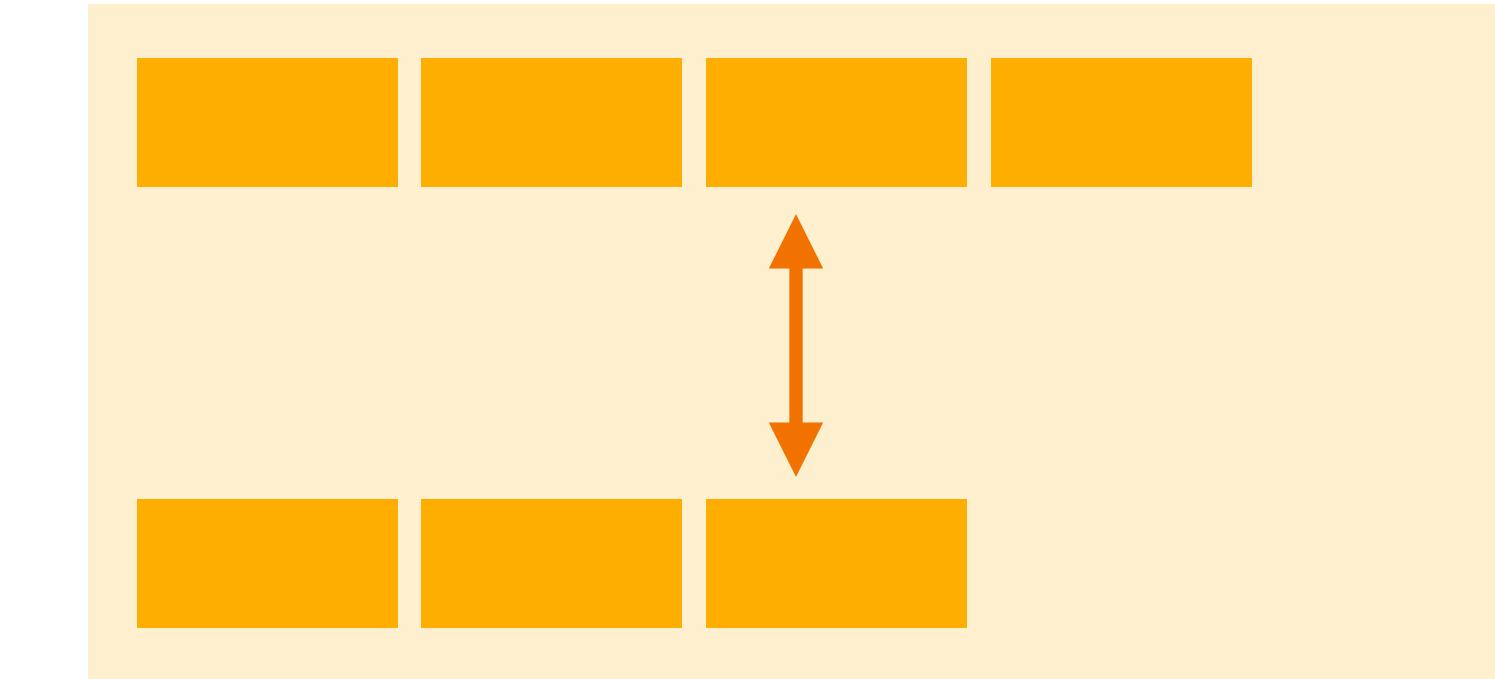
align-content: flex-start;



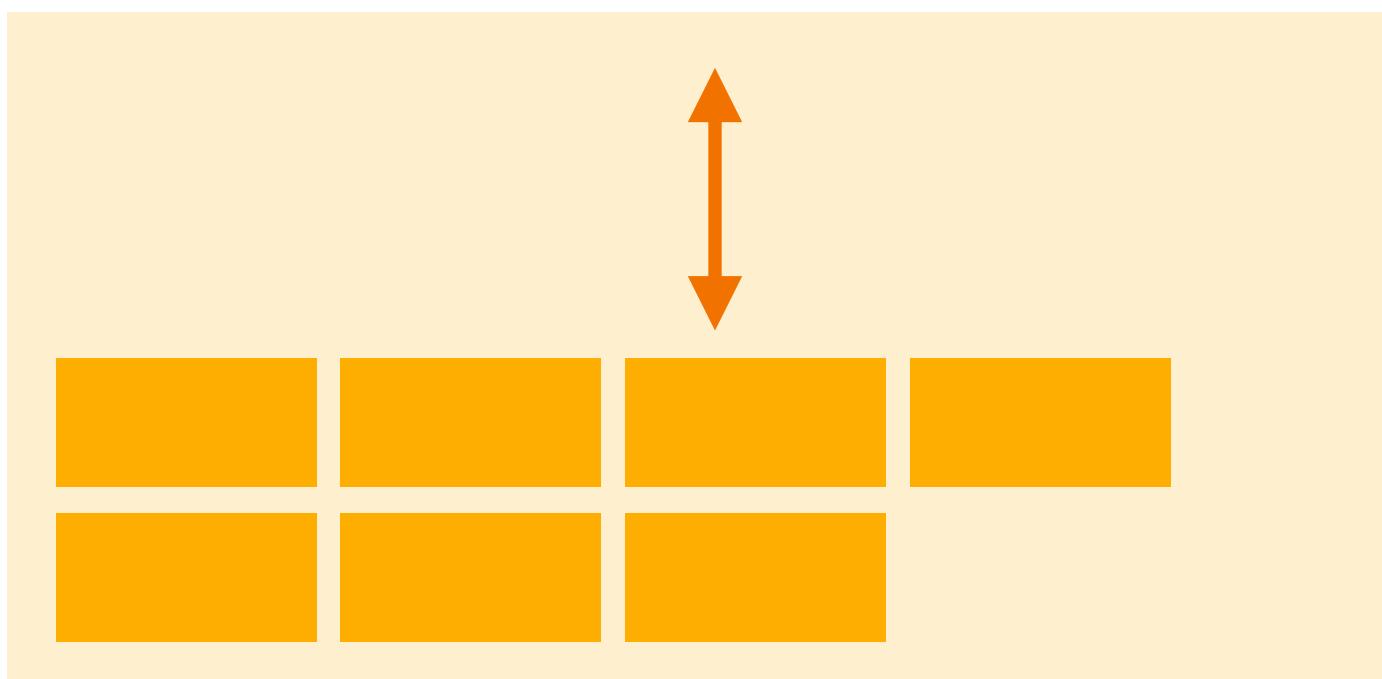
align-content: center;



align-content: space-between;



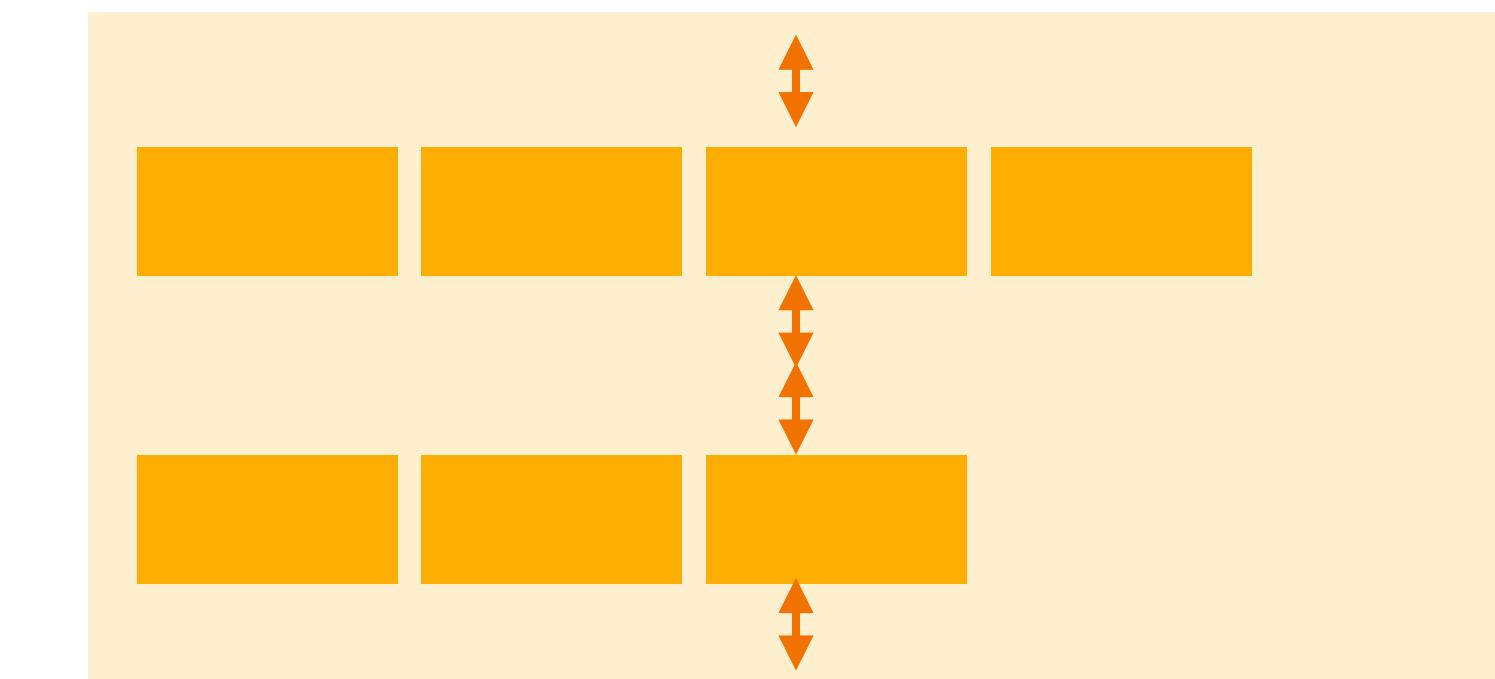
align-content: flex-end;



align-content: stretch;

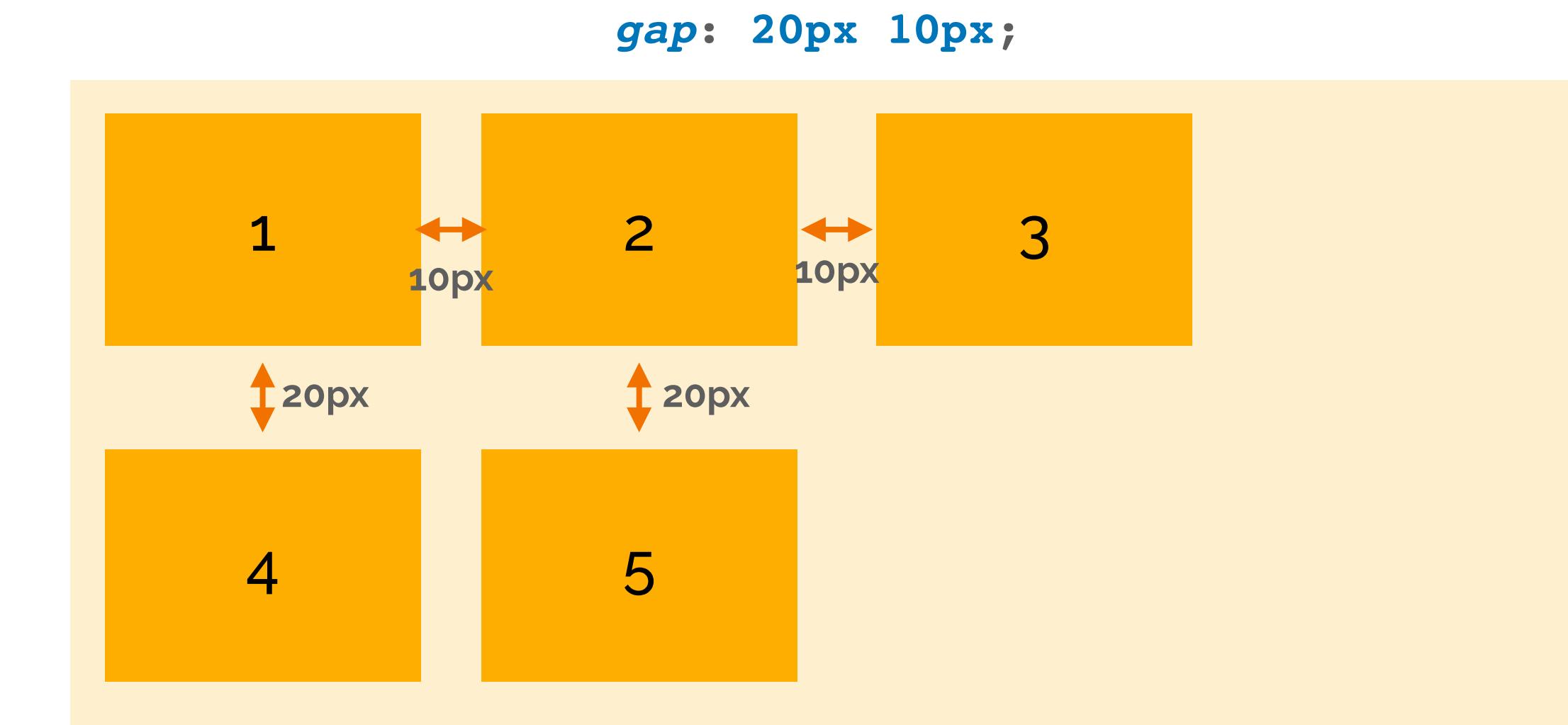
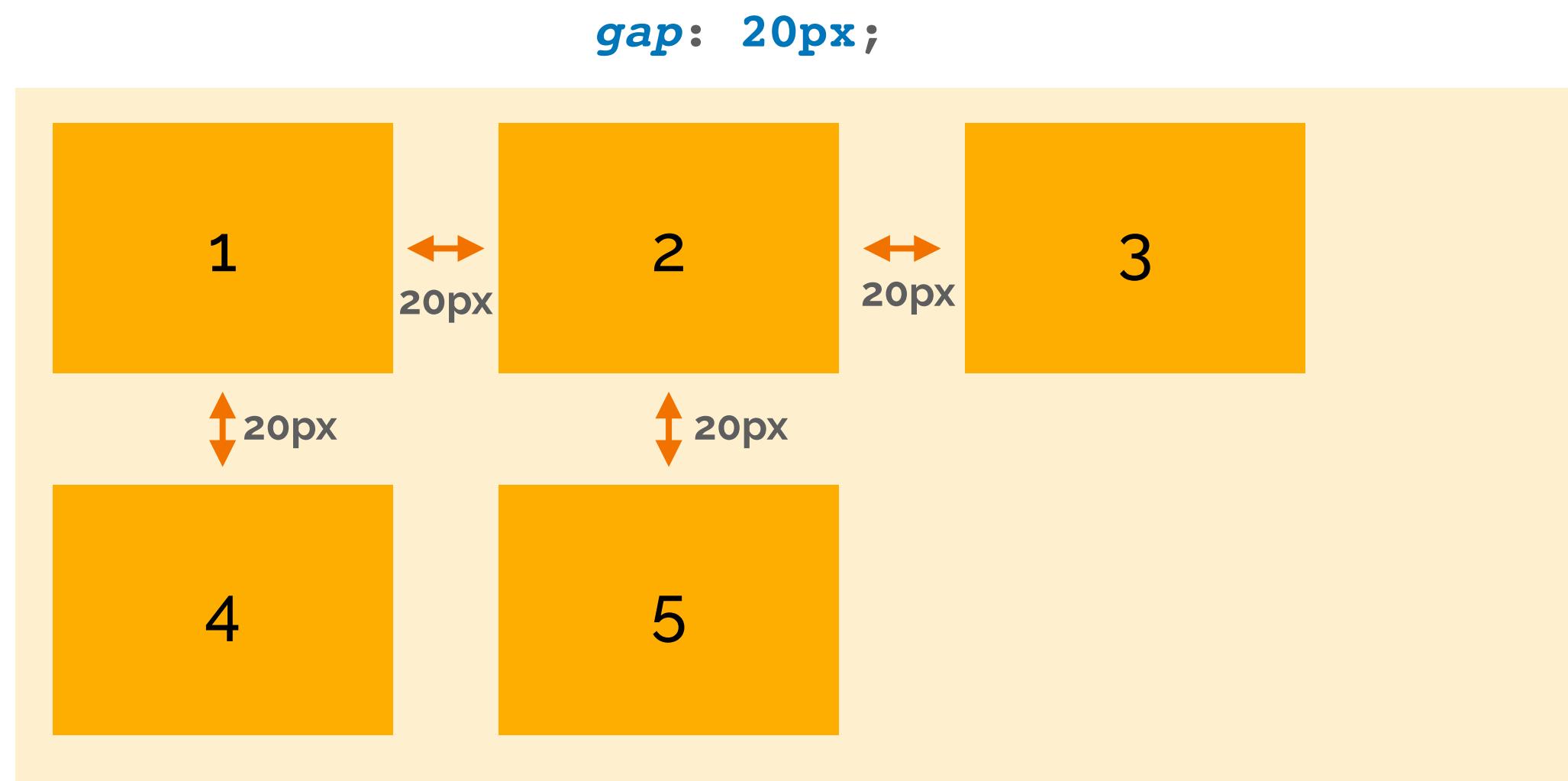


align-content: space-around;



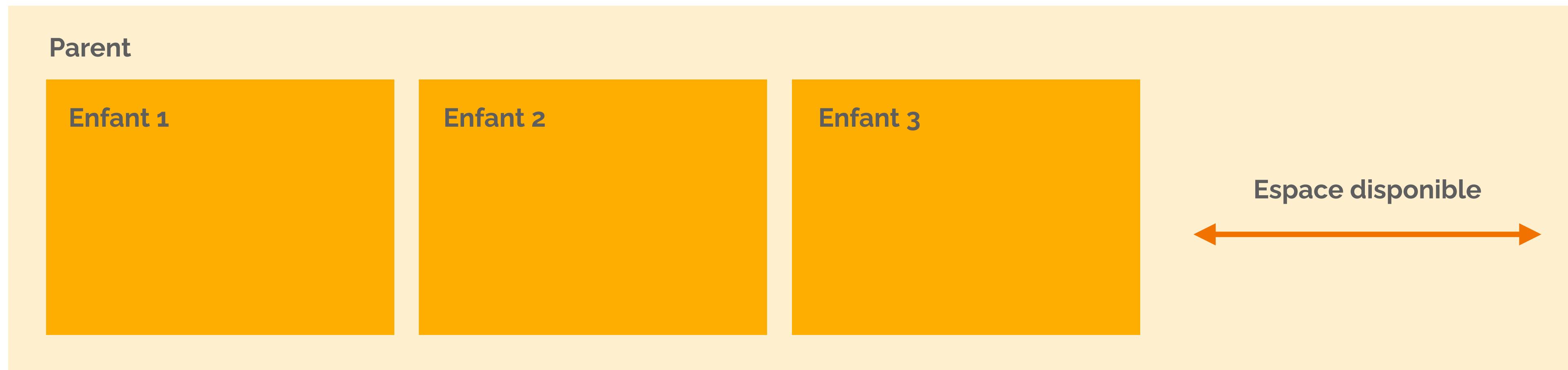
Flexboxes – Ecartement (gap)

CSS



Flexboxes – Sizing

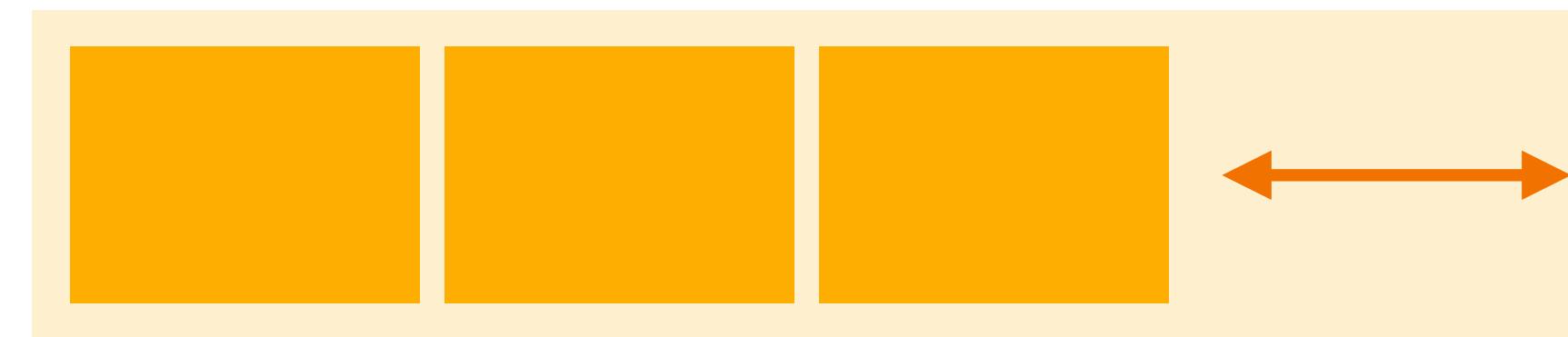
CSS



Flexboxes - Sizing & responsiveness

CSS

flex-basis: [px|%|auto|...];



Rétrécissement

flex-shrink: entier;



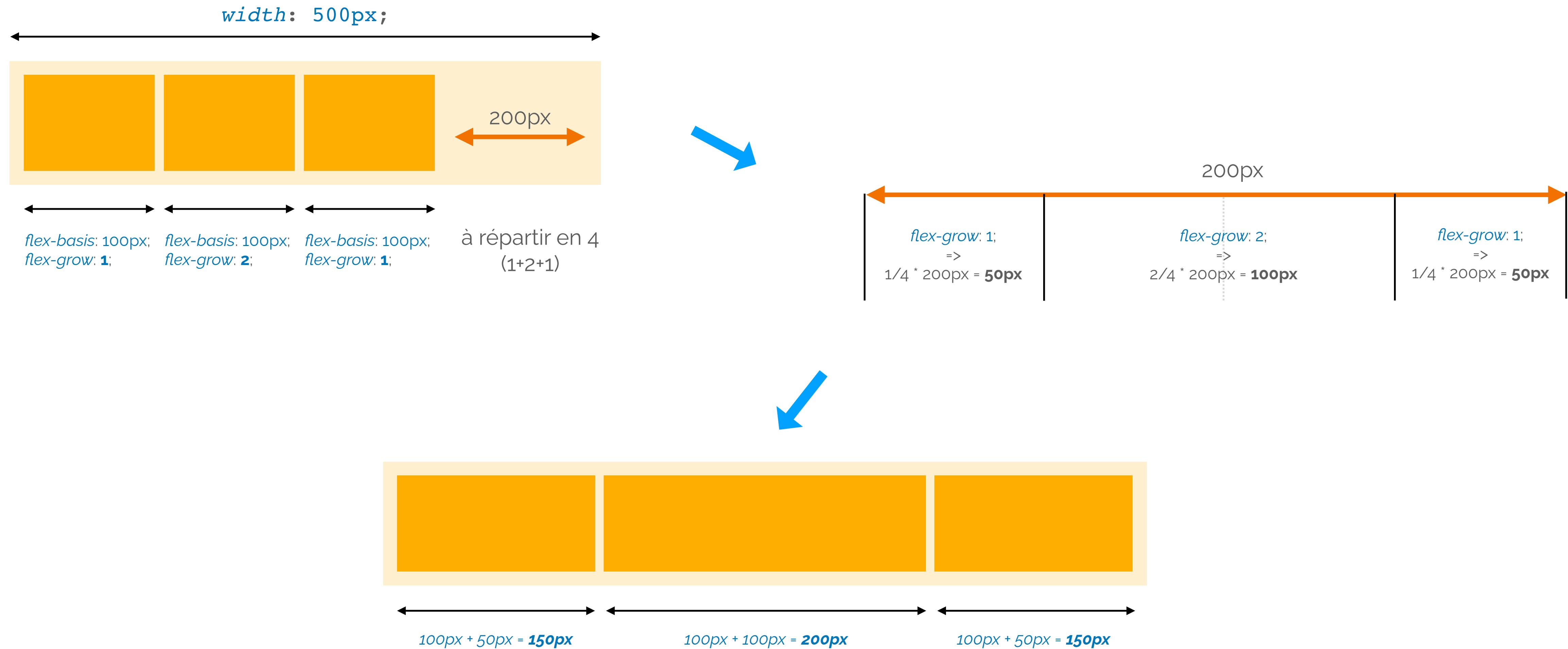
Agrandissement

flex-grow: entier;



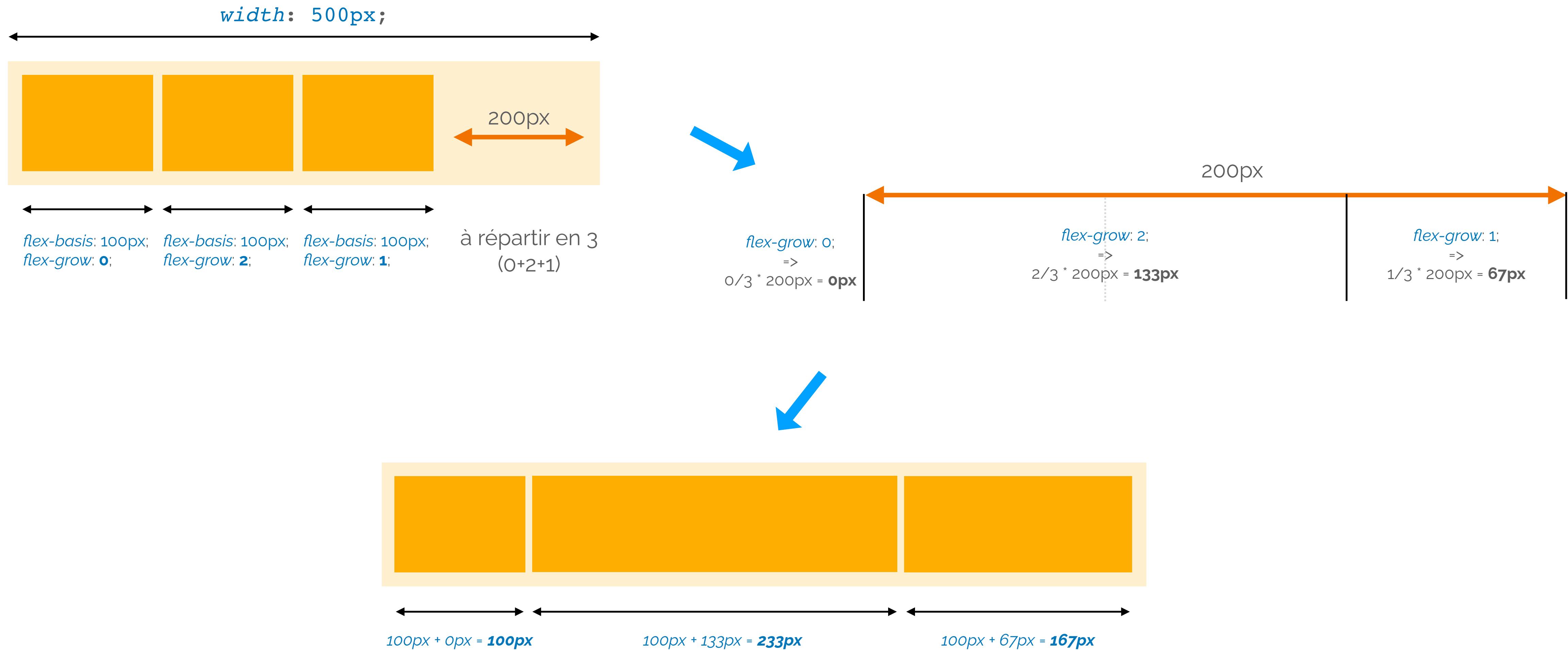
Flexboxes – Sizing flex-grow

CSS



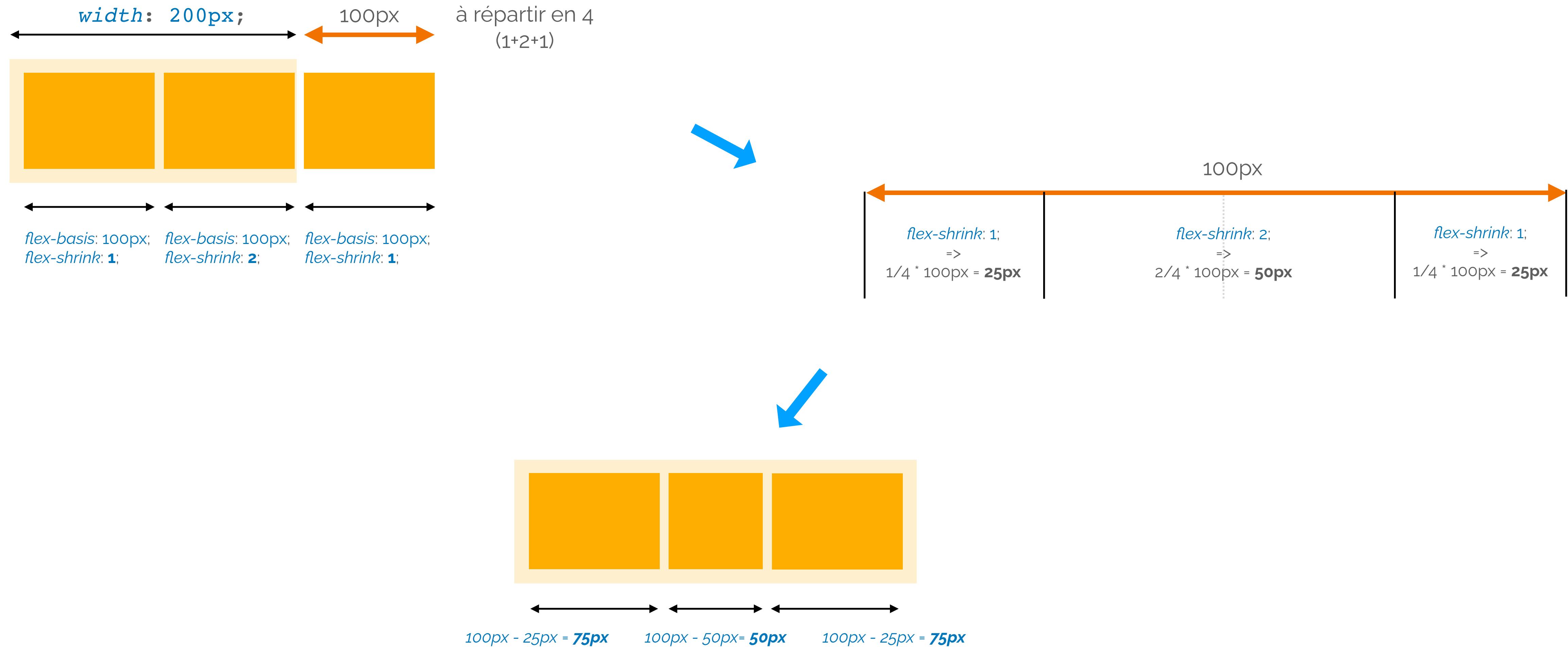
Flexboxes – Sizing flex-grow

CSS



Flexboxes - Sizing flex-shrink

CSS



Flexboxes - Sizing flex-basis & sizing

CSS



Flex-basis hérite du width (si défini) ou "auto" par défaut



Flex-grow vaut 0 par défaut



Flex-shrink vaut 0 par défaut



Par défaut, un élément flex enfant a un min-width défini à 0 et non "auto" comme les autres éléments (resp. height selon direction)

Cela implique que, par défaut, il ne peut être plus petit que son contenu et overflow son parent

Pour autoriser l'overflow, définir "min-width: auto" sur l'enfant

Flexboxes – Overrides parents

CSS

- `align-self` (overrides `align-items`)
- `justify-self` (overrides `justify-content`)

https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox
<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Flexboxes – Order

CSS

```
<div style="display: flex">  
  <div style="order: 2">2</div>  
  <div style="order: 3">3</div>  
  <div style="order: 1">1</div>  
</div>
```

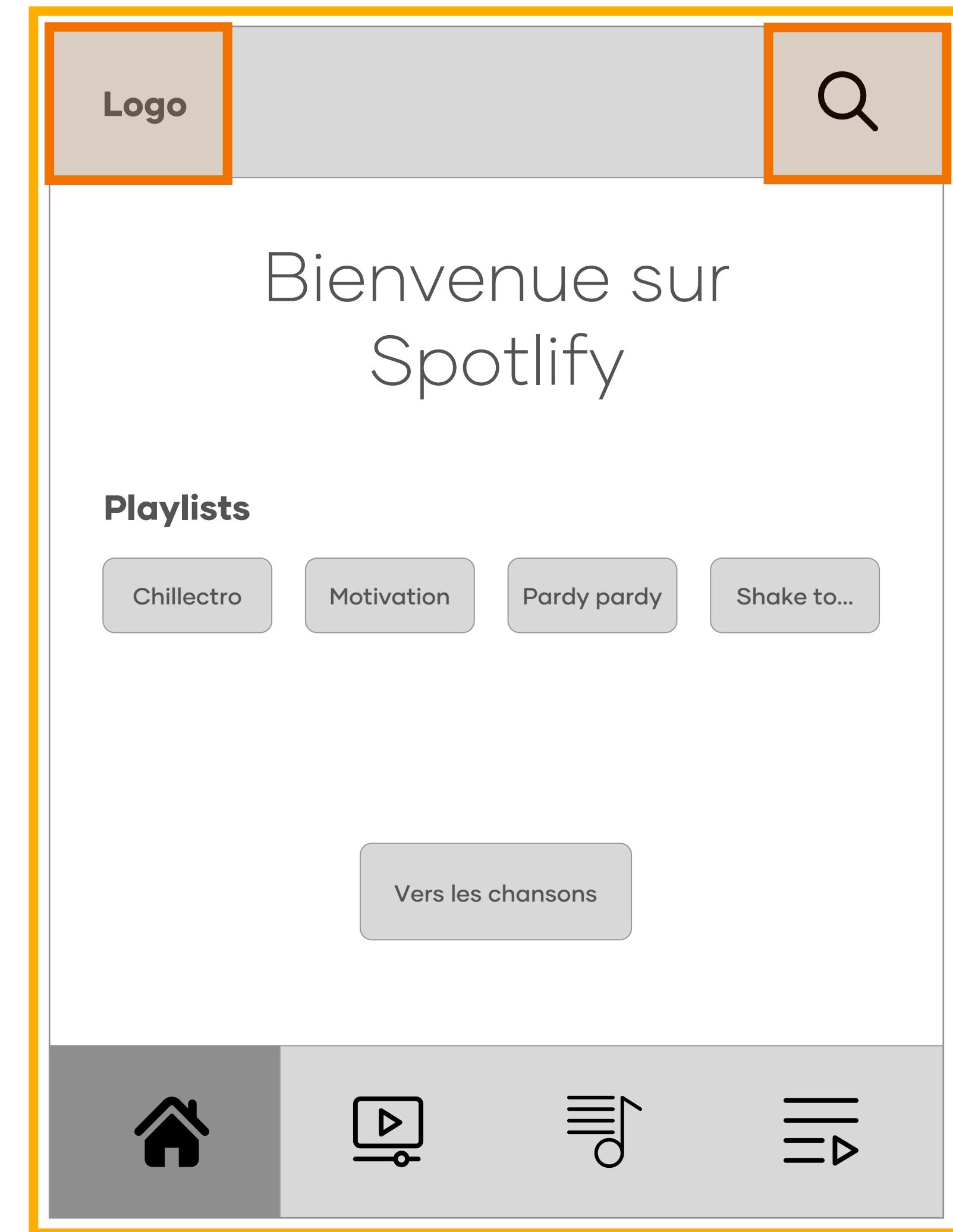


Flexboxes dans le projet

CSS

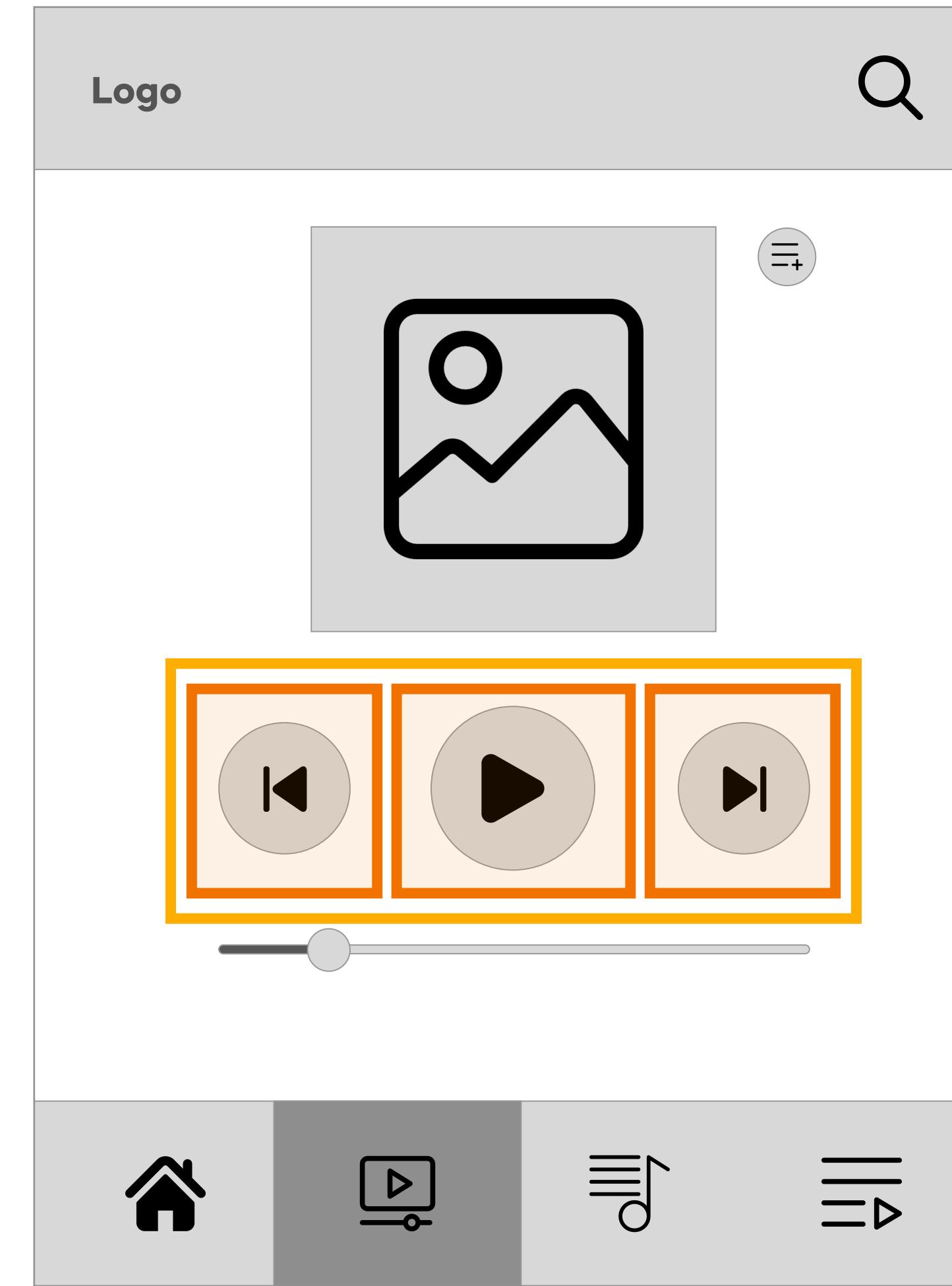


S CSS



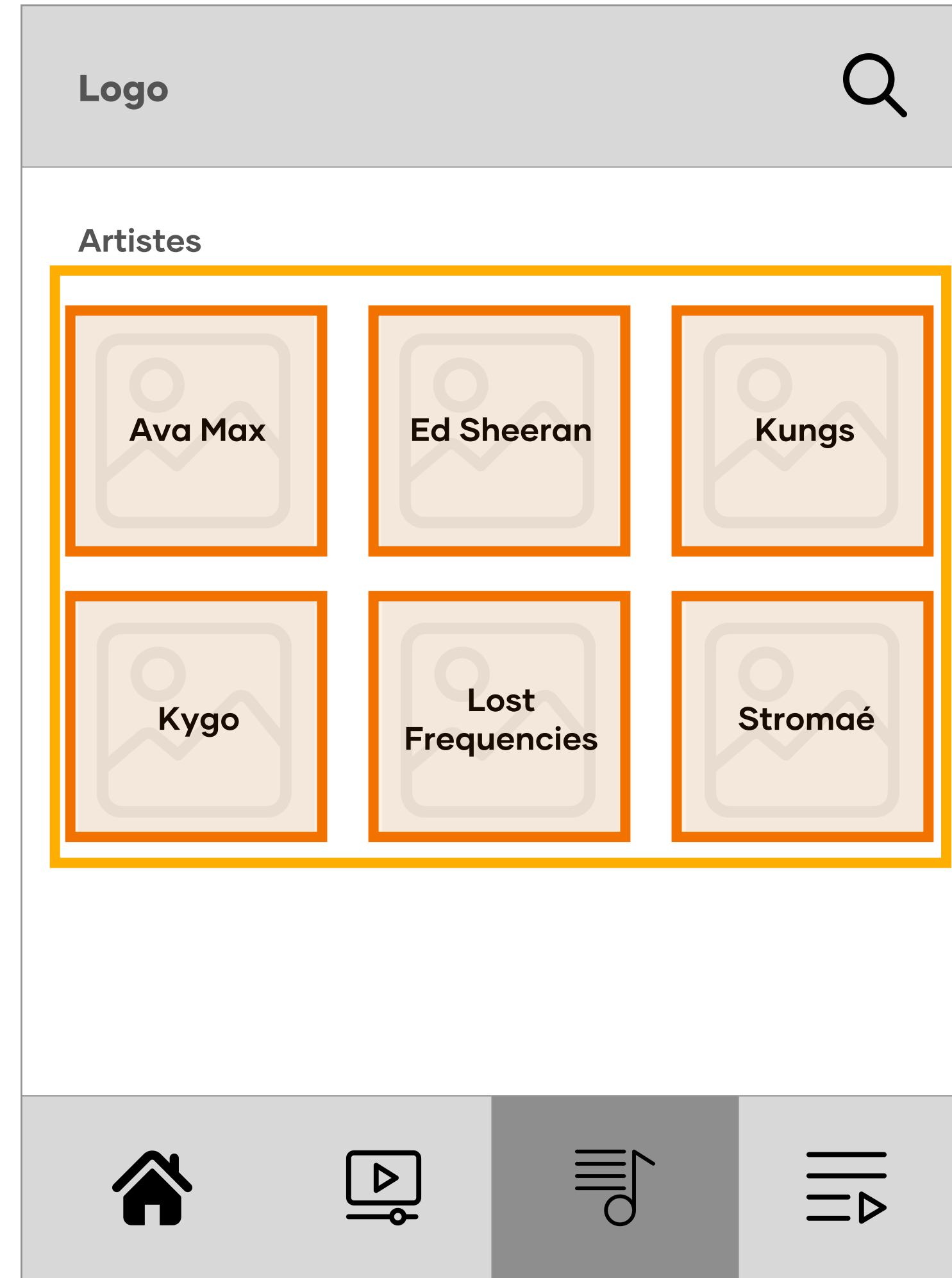
Flexboxes dans le projet

CSS



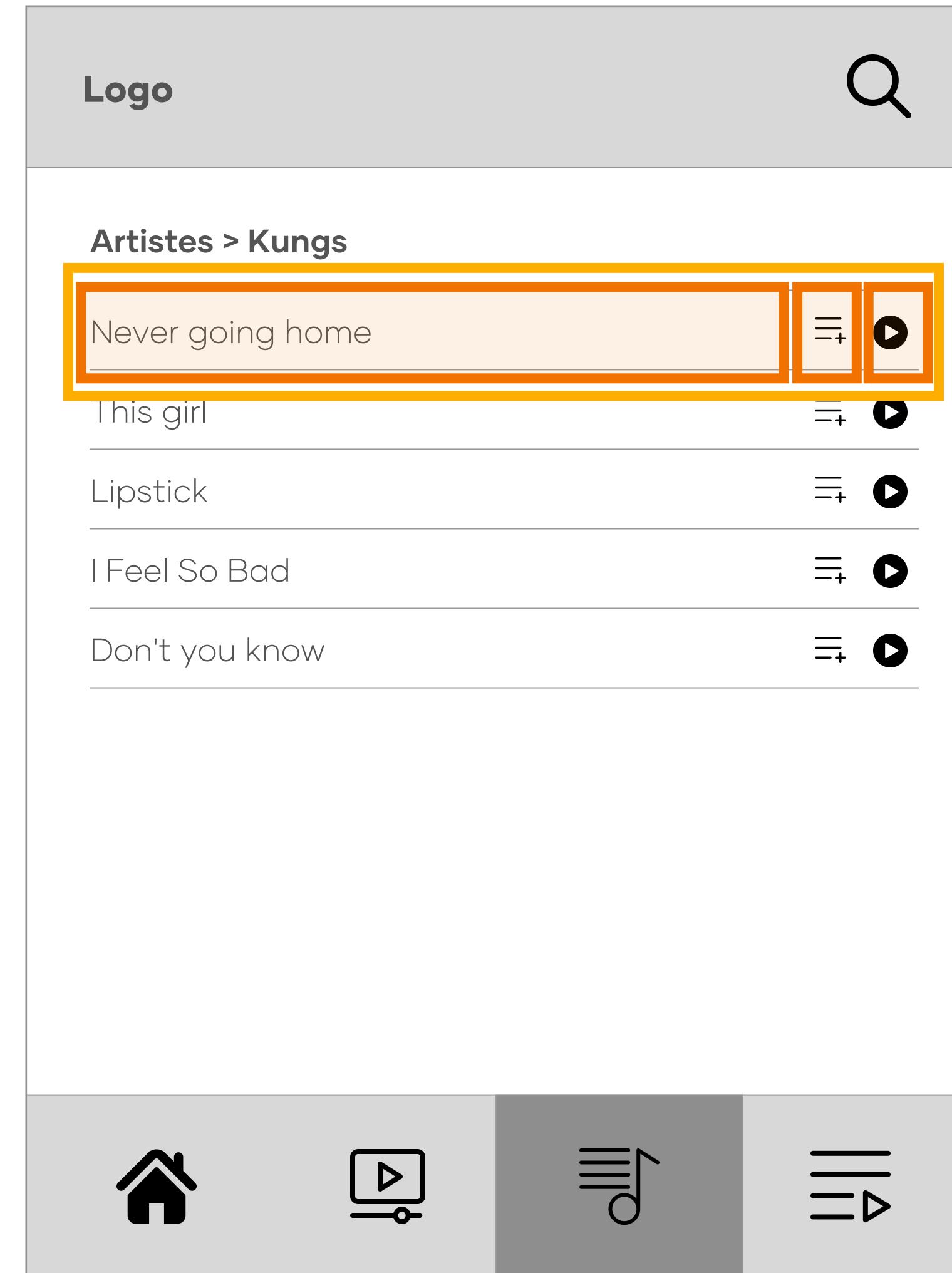
Flexboxes dans le projet

CSS



Flexboxes dans le projet

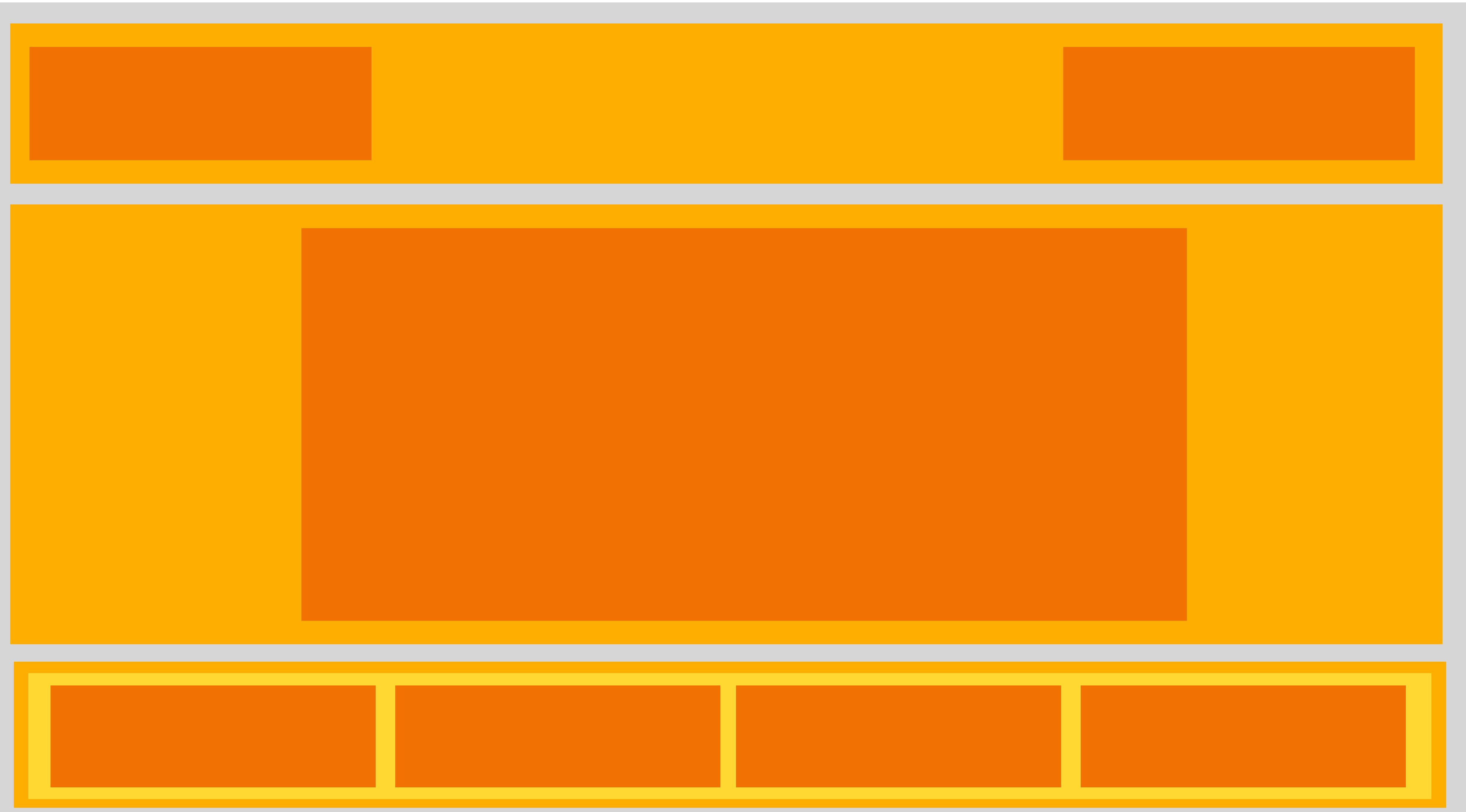
CSS



Goal

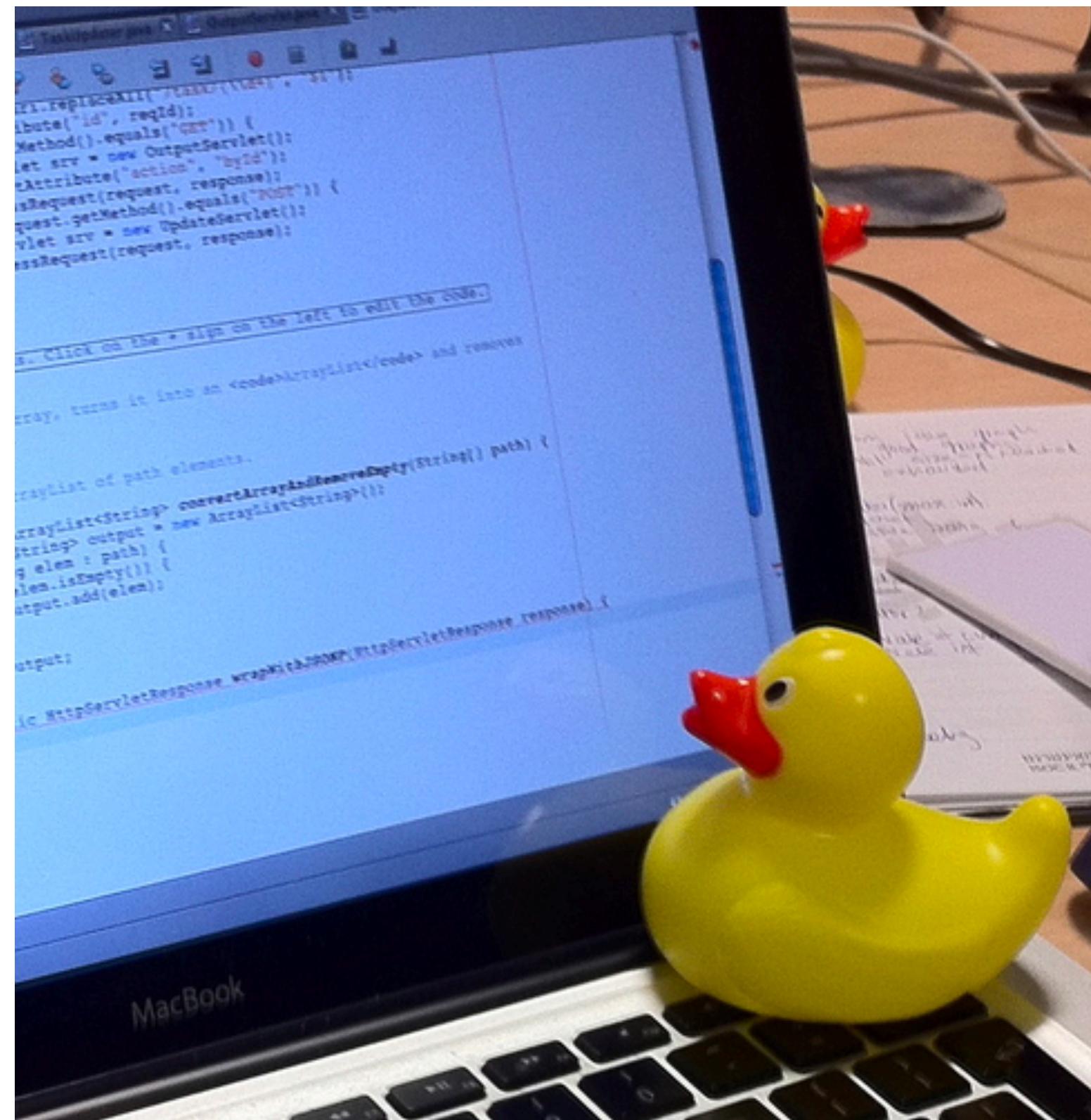
HTML/CSS

- Grâce aux flexboxs, réaliser le squelette suivant (next slide)
- Pas besoin de respecter les couleurs, juste là à titre d'information
- La structure doit ressembler à la home page du prototype



Rubber duck debugging

Tips & tricks



https://fr.wikipedia.org/wiki/M%C3%A9thode_du_canard_en_plastique