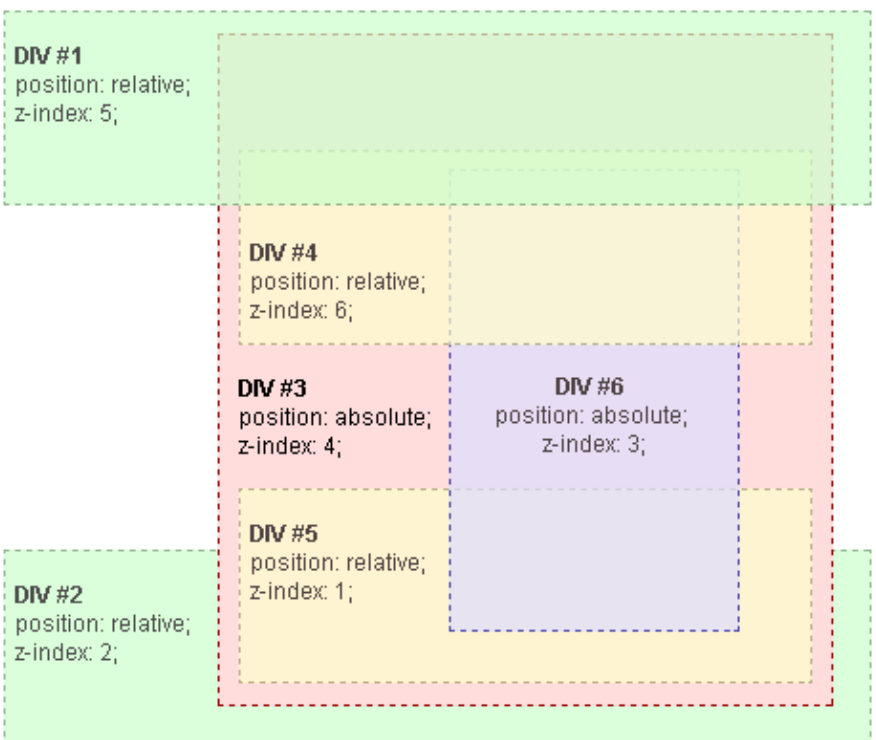
CSS

Bloc englobant et empilements

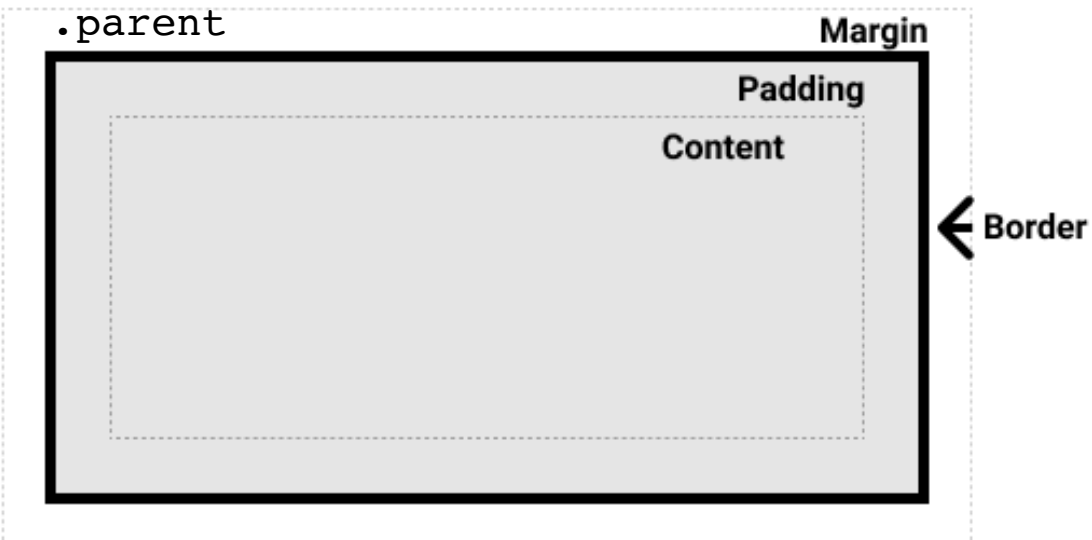


Au programme

1. Bloc englobant (position absolute, relative, etc.)
2. Contexte d'empilement (z-index)

Le Bloc Englobant

référent pour le placement des boîtes

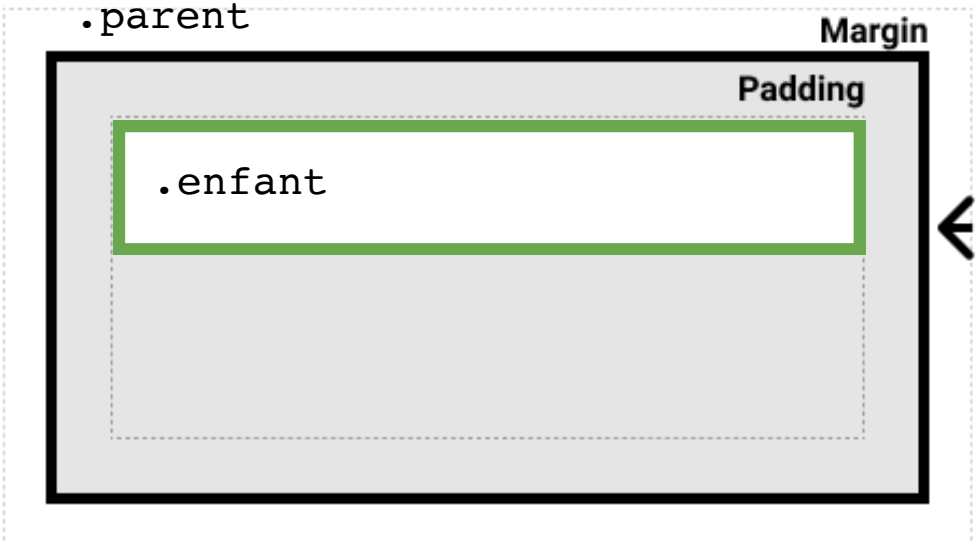


où vais-je me placer dans mon parent ?

.enfant

Le Bloc Englobant

cas des éléments dans le flux

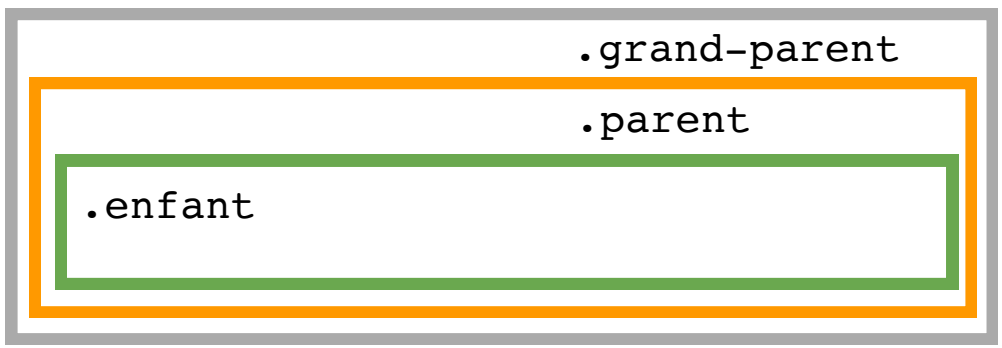



```
1  .parent {
2    display: block;
3  }
4  .enfant {
5    display: block;
6  }
```

Les éléments dans le flux (propriété **position** qui vaut static ou relative) se placent dans la zone de Content, par défaut le plus en haut à gauche

Le Bloc Englobant

cas des éléments absolute



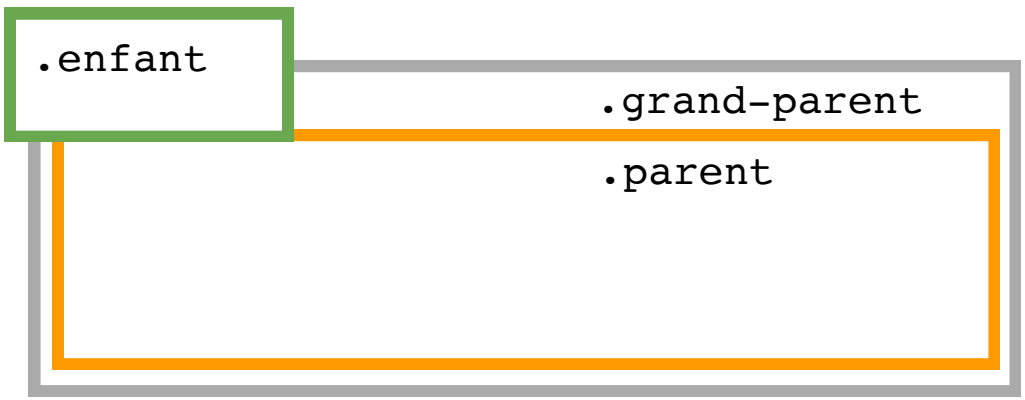

```
1  .grand-parent {
2    display: block;
3  }
4  .parent {
5    display: block;
6  }
7  .enfant {
8    display: block;
9  }
```

↑

Le Bloc Englobant des éléments **positionnés en absolute** est la zone de padding de l'ancêtre le plus proche dont la valeur de position est différente de static (fixed, absolute, relative ou sticky).

Le Bloc Englobant

cas des éléments absolute

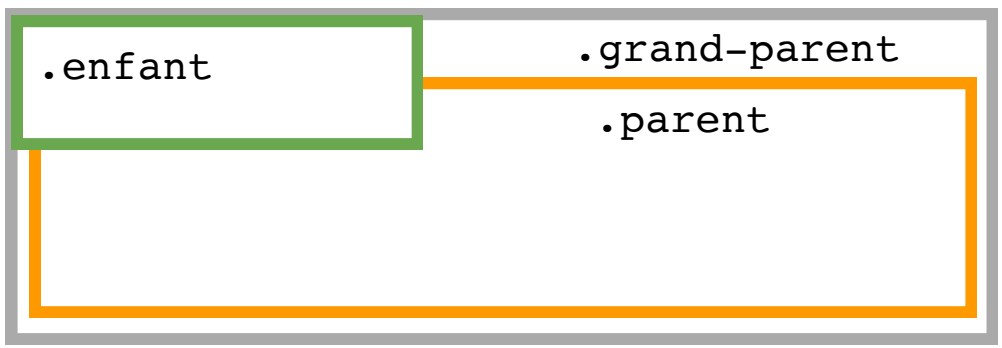



```
1  .grand-parent {
2    display: block;
3  }
4  .parent {
5    display: block;
6  }
7  .enfant {
8    display: block;
9    position: absolute;
10   top: 0;
11   left: 0;
12 }
```

Le Bloc Englobant des éléments **positionnés en absolute** est la zone de padding de l'ancêtre le plus proche dont la valeur de position est différente de static (fixed, absolute, relative ou sticky).

Le Bloc Englobant

cas des éléments absolute

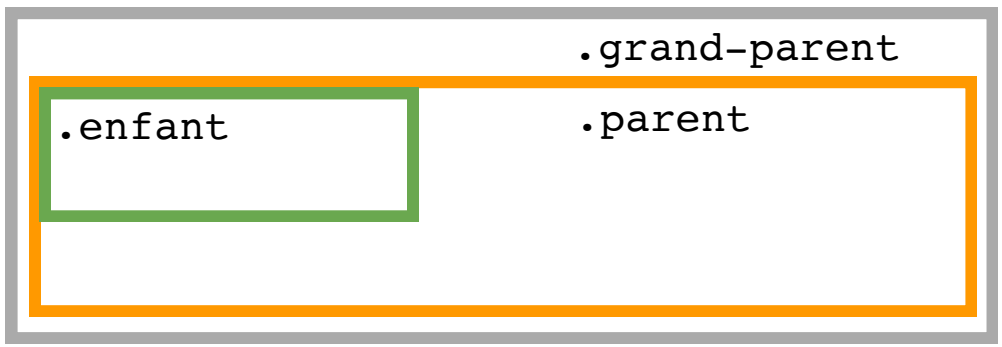



```css
.grand-parent {
  display: block;
  position: relative;
}
.parent {
  display: block;
}
.enfant {
  display: block;
  position: absolute;
  top: 0;
  left: 0;
}
```

Le Bloc Englobant des éléments **positionnés en absolute** est la zone de padding de l'ancêtre le plus proche dont la valeur de position est différente de static (fixed, absolute, relative ou sticky).

Le Bloc Englobant

cas des éléments absolute

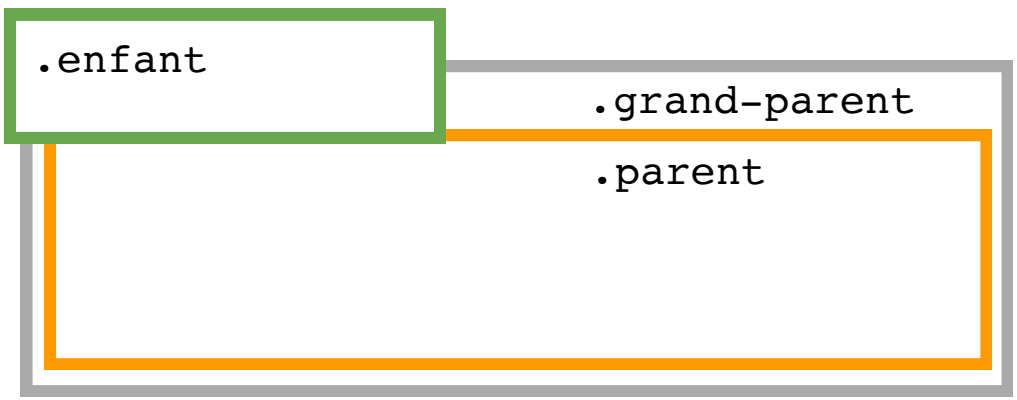



```css
.grand-parent {
  display: block;
  position: relative;
}
.parent {
  display: block;
  position: relative;
}
.enfant {
  display: block;
  position: absolute;
  top: 0;
  left: 0;
}
```

Le Bloc Englobant des éléments **positionnés en absolute** est la zone de padding de l'ancêtre le plus proche dont la valeur de position est différente de static (fixed, absolute, relative ou sticky).

Le Bloc Englobant

cas des éléments fixed

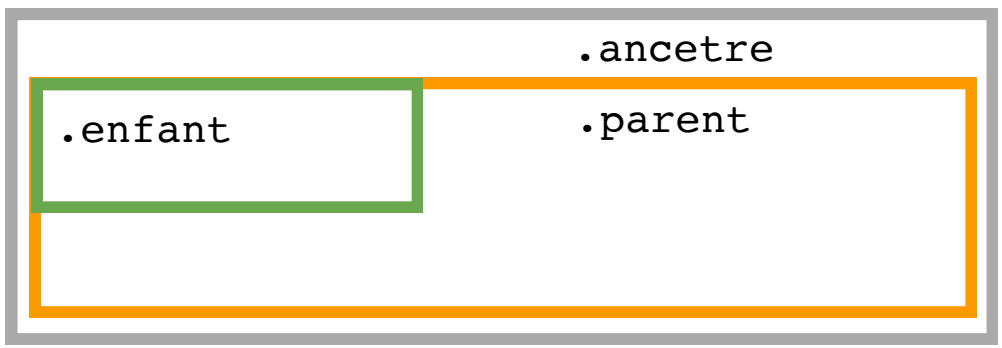


Le Bloc Englobant des éléments **positionnés en fixed** est le Viewport (zone d'affichage de la fenêtre)

```css
.grand-parent {
  display: block;
  position: relative;
}
.parent {
  display: block;
  position: relative;
}
.enfant {
  display: block;
  position: fixed;
  top: 0;
  left: 0;
}
```

Le Bloc Englobant

cas particuliers




```css
.ancetre {
  display: block;
  position: relative;
}
.parent {
  display: block;
  transform: translate(0);
}
.enfant {
  display: block;
  position: absolute;
  top: 0;
  left: 0;
}
```

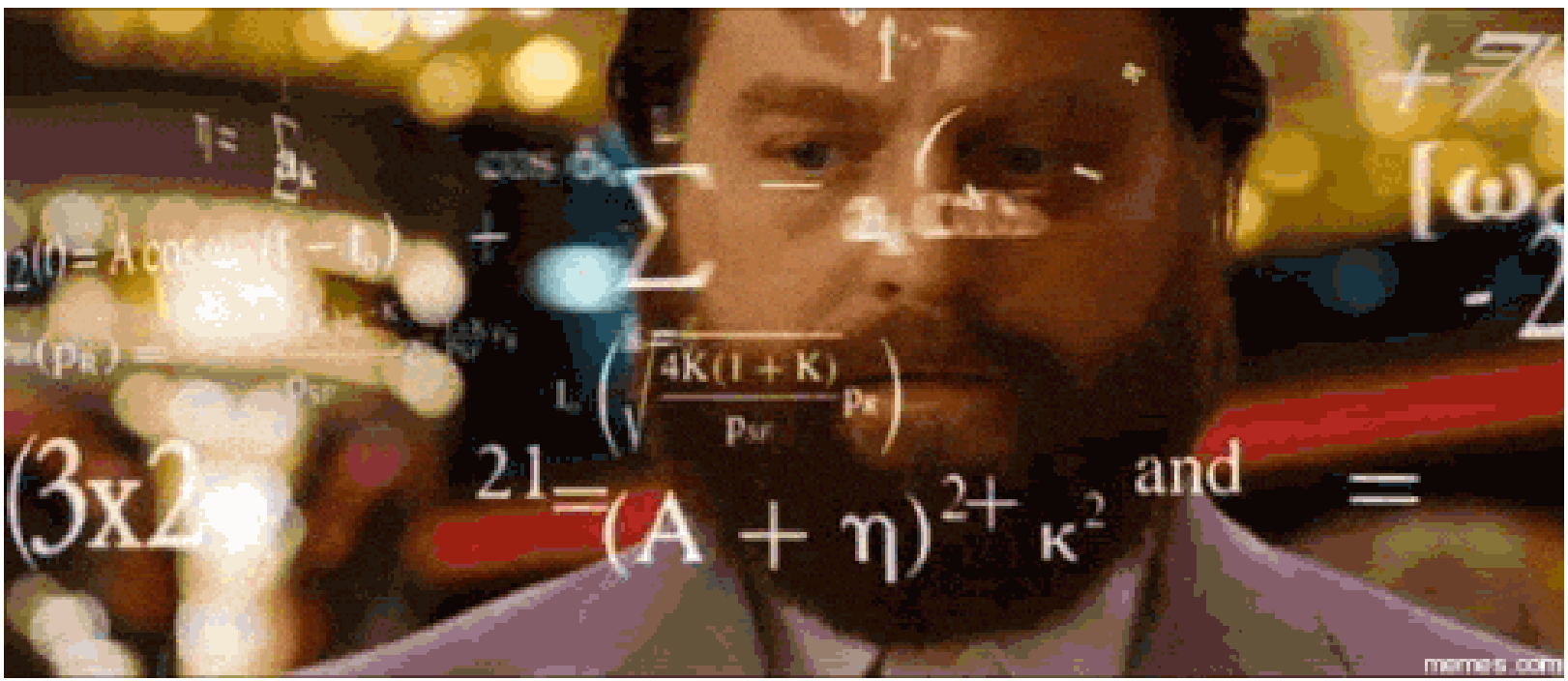
↑

Le Bloc Englobant des éléments **positionnés en absolute ou fixed** peut aussi être l'ancêtre ayant :

- une propriété **transform** ou **perspective**,
- une propriété **will-change** qui vaut transform ou perspective
- une propriété **filter** différente de none
- une propriété **contain** qui vaut paint.

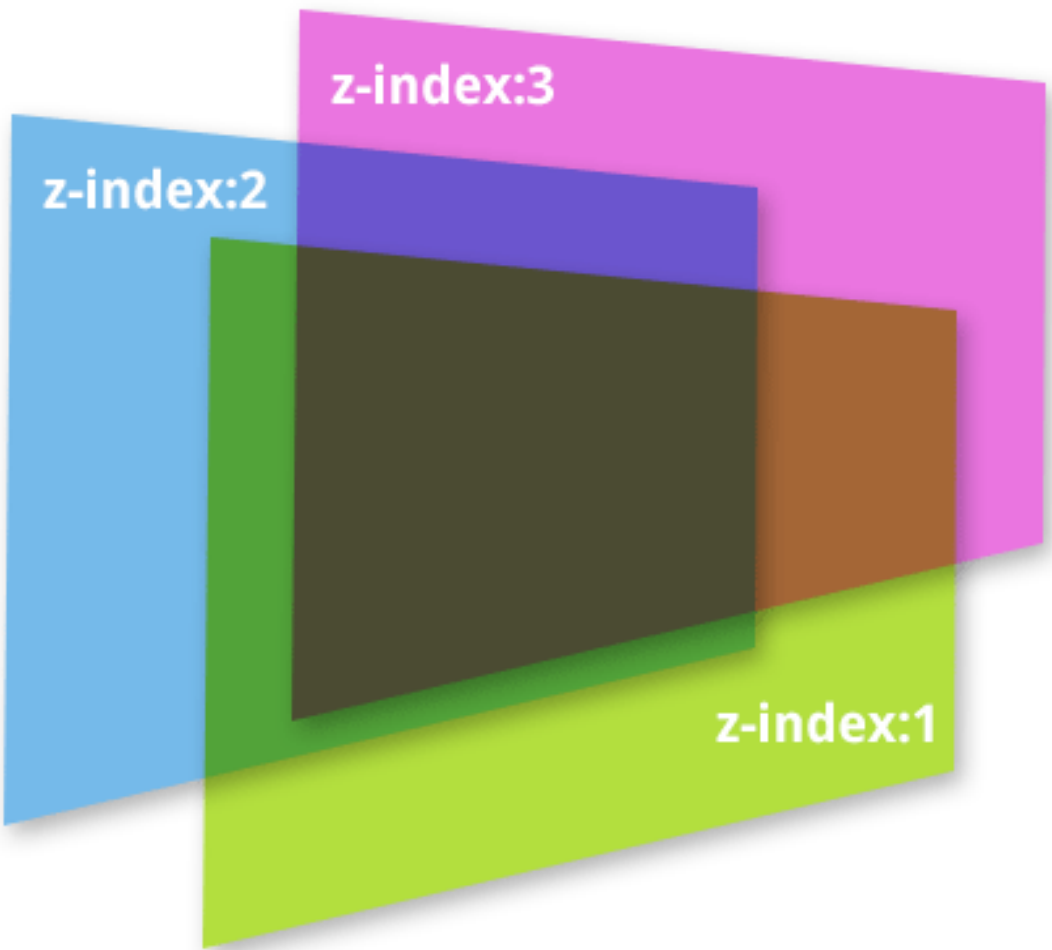
Exercice !

bloc englobant



Le Contexte d'empilement

Hello z-index: 9999, my old friend



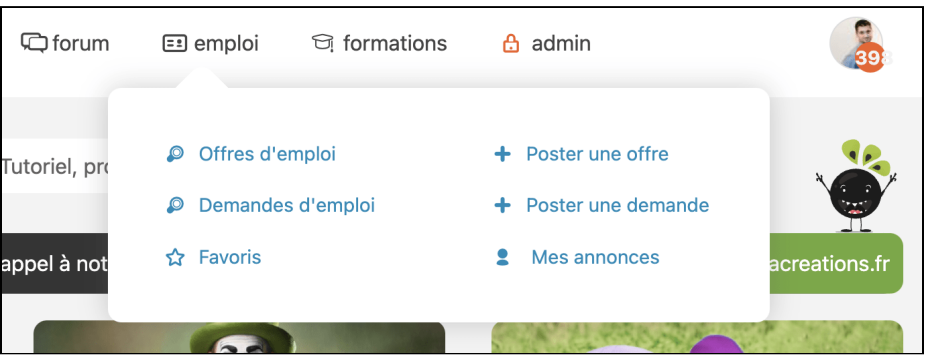
“

z-index (n.f. Propriété CSS qui ne fonctionne jamais)
Inventée par le Démon pour répandre le mal dans le monde du Web et souvent évoquée dans certaines incantations de magie noire.

z-index

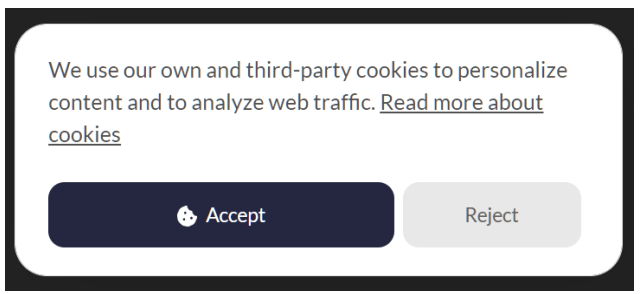
mais... pourquoi faire ?

un Menu déroulant

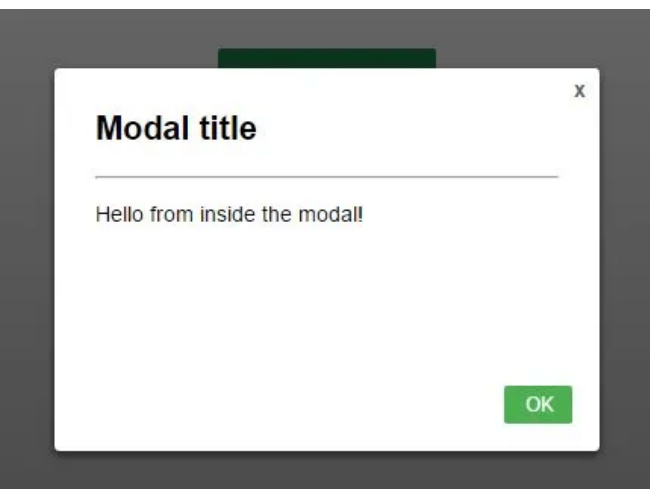


une Tooltip

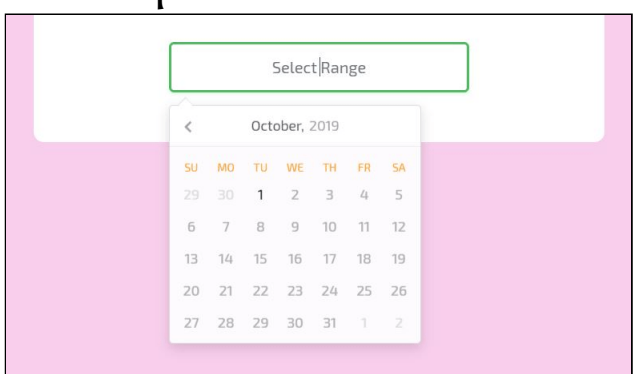
un Bandeau cookies



une Modale



un Datepicker



pour briller en société...

z-index

1

Plus grande valeur possible ?

Pas de limite dans les spécifications CSS mais en pratique valeur signée de 32 bits *(soit une plage de −2147483648 à +2147483647)*... ou calc(infinity) qui vaut 3.40282e+38

2

Valeurs les plus utilisées ?

Les dix valeurs les plus utilisées au monde sont : 1, 2, 10, 0, 100, 1000, 9999, -1, 999 et 3.

3

Conventions de nommage ?

Les trois conventions les plus utilisées sont :
1- $zindex-modal, $zindex-tooltip (sémantique à la Bootstrap)
2- z-20, z-30 (utilitaire à la Tailwind)
3- z-null, z-lowest, z-low.

4

Valeur par défaut ?

La valeur par défaut n'est pas un nombre, mais la valeur "auto".

5

Mais pourquoi le nom "z-index" ?

Le W3C reconnaît qu'il aurait dû nommer la propriété "z-order" plutôt *(source)*

Stacking Context

Contexte d'Empilement

Périmètre permettant d'organiser
l'empilement des éléments qui y sont situés

← OK super, et donc comment est créé un Contexte ?

Stacking Order

Niveau d'Empilement

Niveau d'empilement sur l'axe Z au sein d'un
Contexte d'Empilement

← D'accord j'ai compris, mais comment modifier l'Ordre ?

Principe de base : les éléments sont systématiquement empilés selon leur *Niveau d'Empilement* et uniquement au sein d'un même *Contexte d'empilement*.

Stacking Order

Niveau d'Empilement

Stacking Order

en résumé

> Trois règles définissent le niveau d'empilement d'un élément en cas de superposition sur l'axe de la profondeur (z)

Selon une croyance ancestrale, seule la propriété z-index permet de modifier l'ordre d'empilement sur l'axe Z.
C'est faux.

Stacking Order (Niveau d'Empilement)

Règles

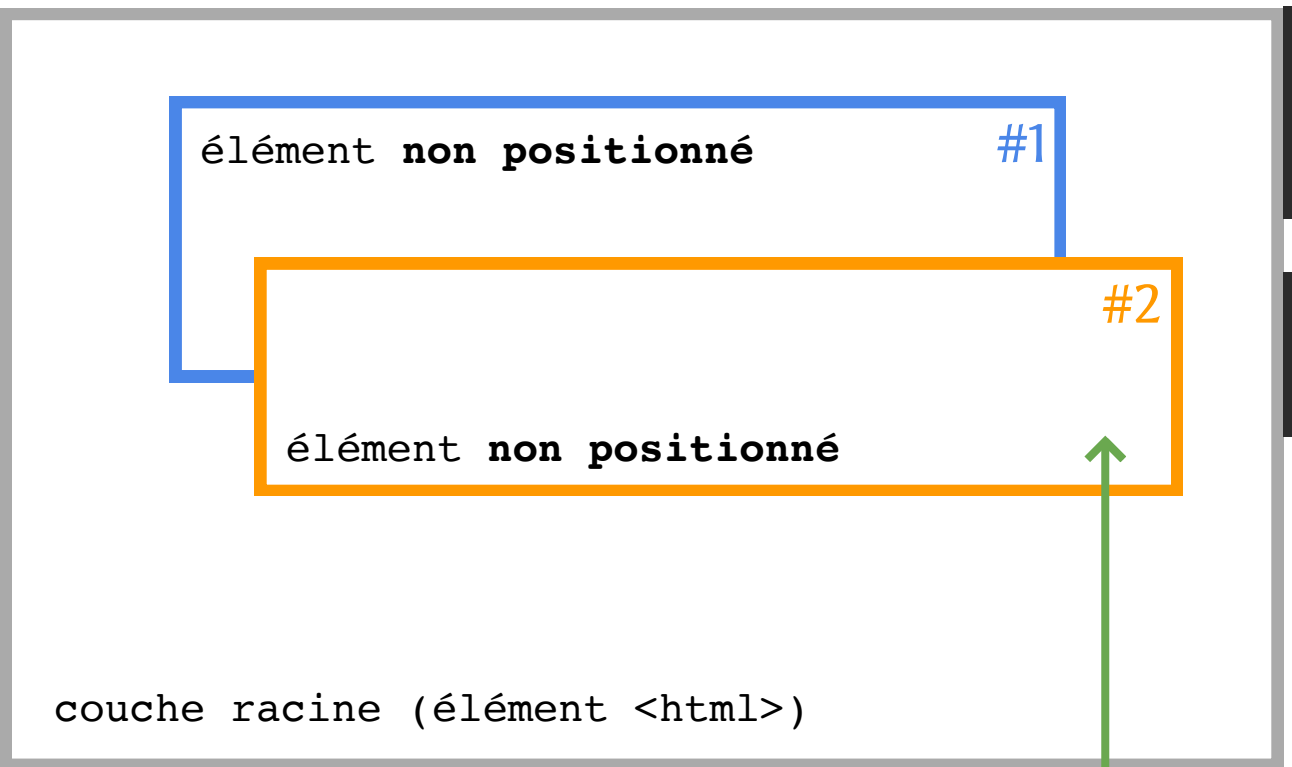
1

Cas des Éléments "non positionnés"

Les éléments non positionnés se placent (et s'empilent) dans leur ordre d'apparition dans le HTML (DOM)

Stacking order

1) Éléments non positionnés




```
1 <html>
2   <div #1>
3   <div #2>
4 </html>
```

```
1 #2 {
2   margin-top: -100px;
3 }
```

Les éléments non positionnés se placent (et s'empilent) dans leur ordre d'apparition dans le HTML (DOM)

Stacking Order (Niveau d'Empilement)

Règles

1

Cas des Éléments "non positionnés"

Les éléments non positionnés se placent (et s'empilent) dans leur ordre d'apparition dans le HTML (DOM)

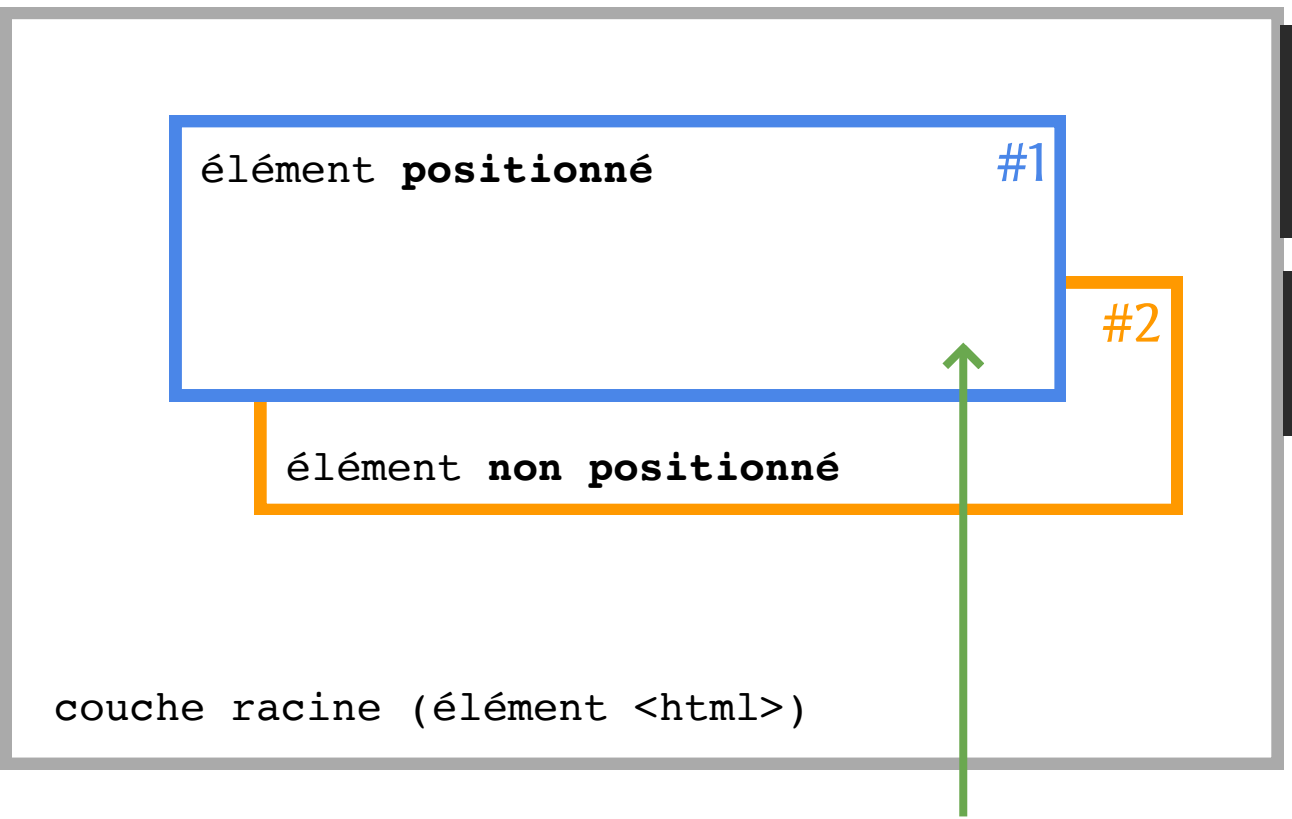
2

Cas des Éléments "positionnés"

Les éléments *positionnés* (ou *transformés)*, se placent au-dessus des éléments non positionnés.

Stacking order

2) Éléments positionnés et non positionnés




```
1 <html>
2   <div #1>
3   <div #2>
4 </html>
```

```
1 #1 {
2   position: relative;
3 }
```

Les éléments positionnés (ou transformés), se placent au-dessus des éléments non positionnés.

Stacking Order (Niveau d'Empilement)

Règles

1

Cas des Éléments "non positionnés"

Les éléments non positionnés se placent (et s'empilent) dans leur ordre d'apparition dans le HTML (DOM)

2

Cas des Éléments "positionnés"

Les éléments *positionnés* (ou *transformés)*, se placent au-dessus des éléments non positionnés.

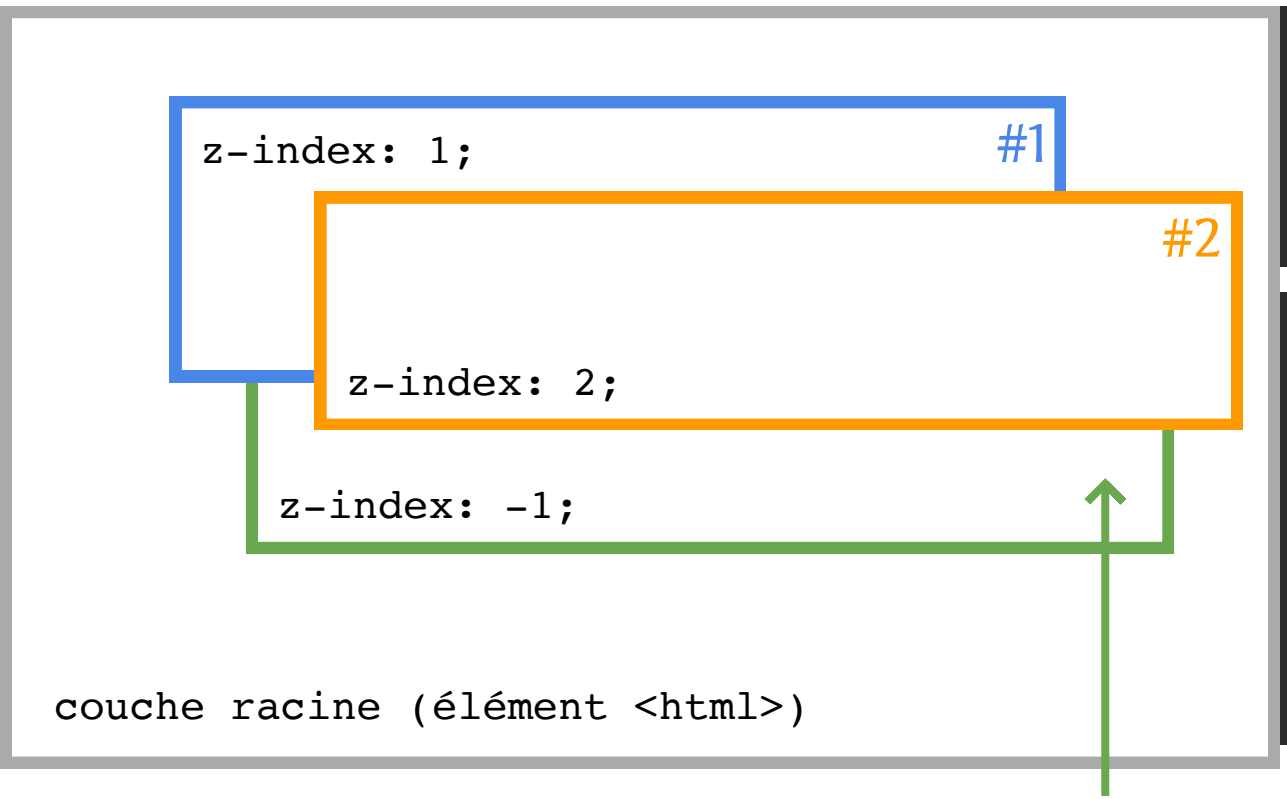
3

Cas des Éléments ayant un z-index

La propriété *z-index* modifie l'ordre d'empilement des éléments positionnés selon sa valeur. Plus la valeur est grosse, plus l'élément est au-dessus de la pile.

Stacking order

3) Éléments positionnés et z-index




```html
<html>
  <div #1>
  <div #2>
  <div #3>
</html>
```

```css
#1 {
  z-index: 1;
}
#2 {
  z-index: 2;
}
#3 {
  z-index: -1;
}
```

La propriété z-index modifie l'ordre d'empilement des éléments positionnés (au sein du Contexte d'Empilement en cours).

aparté

Qui peut bénéficier de z-index ?

Seuls **deux** types d'éléments peuvent avoir un z-index.

1. **Élément positionné :**
   un élément dont la propriété CSS *position* a pour valeur *relative*, *absolute*, *fixed* ou *sticky*.
   Par défaut, les éléments d'une page ne sont pas positionnés (ils sont en *position: static*).

2. **Flex item ou Grid item:**
   un élément enfant direct d'un flex-container ou d'un grid-container

Stacking order

récapitulatif

1

Trois règles définissent le niveau d'empilement :

1. élément non positionné
2. élément positionné (ou transformé)
3. élément avec z-index

2

z-index modifie le niveau d'empilement :
plus la valeur est grosse, plus l'élément est au-dessus de la pile.

3

z-index ne s'applique pas à tout le monde :
l'élément doit être positionné, ou être flex-item ou grid-item.
c'est tout. vraiment.

Stacking context

Contexte d'empilement

Stacking context

en résumé

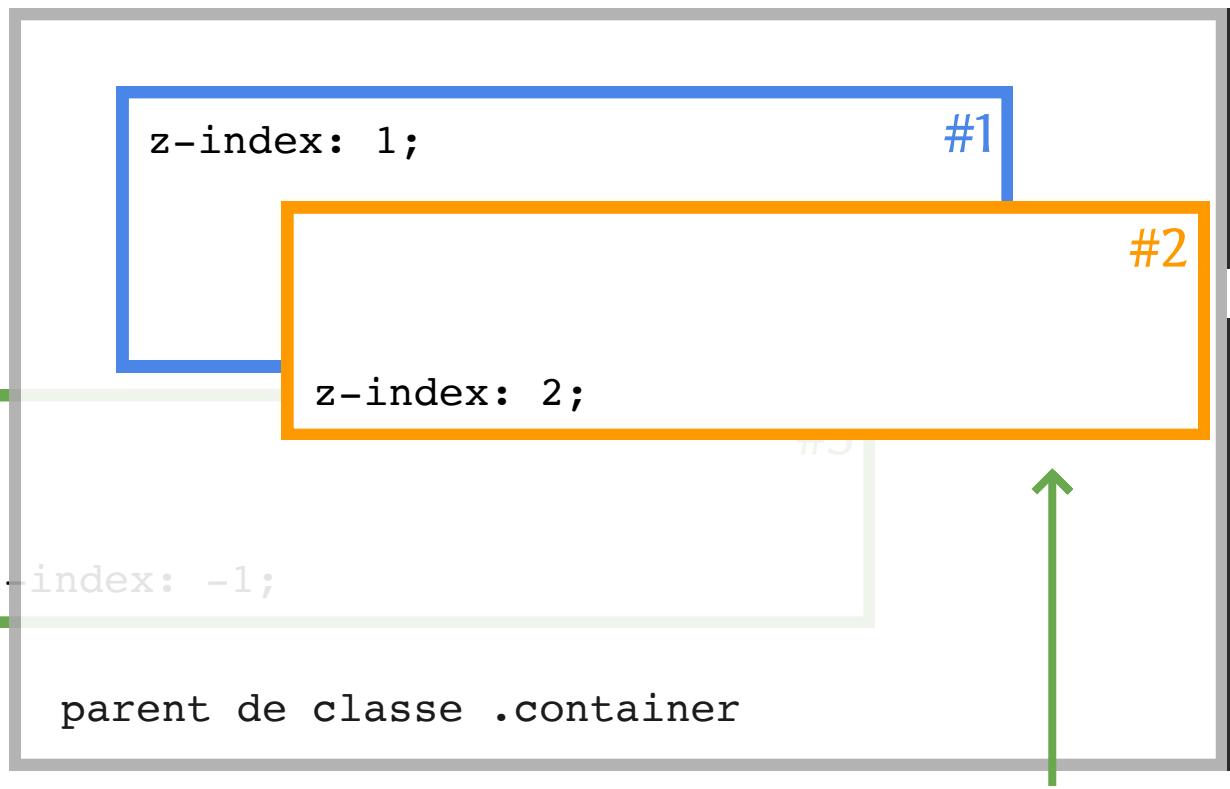
> Le Contexte d'Empilement définit le périmètre des règles d'empilement qu'on vient de voir. (oui, fallait suivre un peu)
>
> Un élément peut créer un Contexte d'Empilement pour gérer l'empilement de ses descendants.

Pour bien nous embrouiller, z-index modifie l'Ordre d'Empilement... mais crée également un nouveau Contexte d'Empilement. Et c'est pas tout...

Stacking context (contexte d'empilement)

1. Un Contexte d'Empilement s'applique à un conteneur pour gérer l'empilement de ses descendants
2. z-index (mais pas que lui) définit l'Ordre d'Empilement des éléments au sein d'un Contexte d'Empilement
3. L'élément <html> constitue le premier Contexte d'Empilement
4. Certaines propriétés CSS créent un nouveau Contexte d'Empilement

Stacking context (contexte d'empilement)

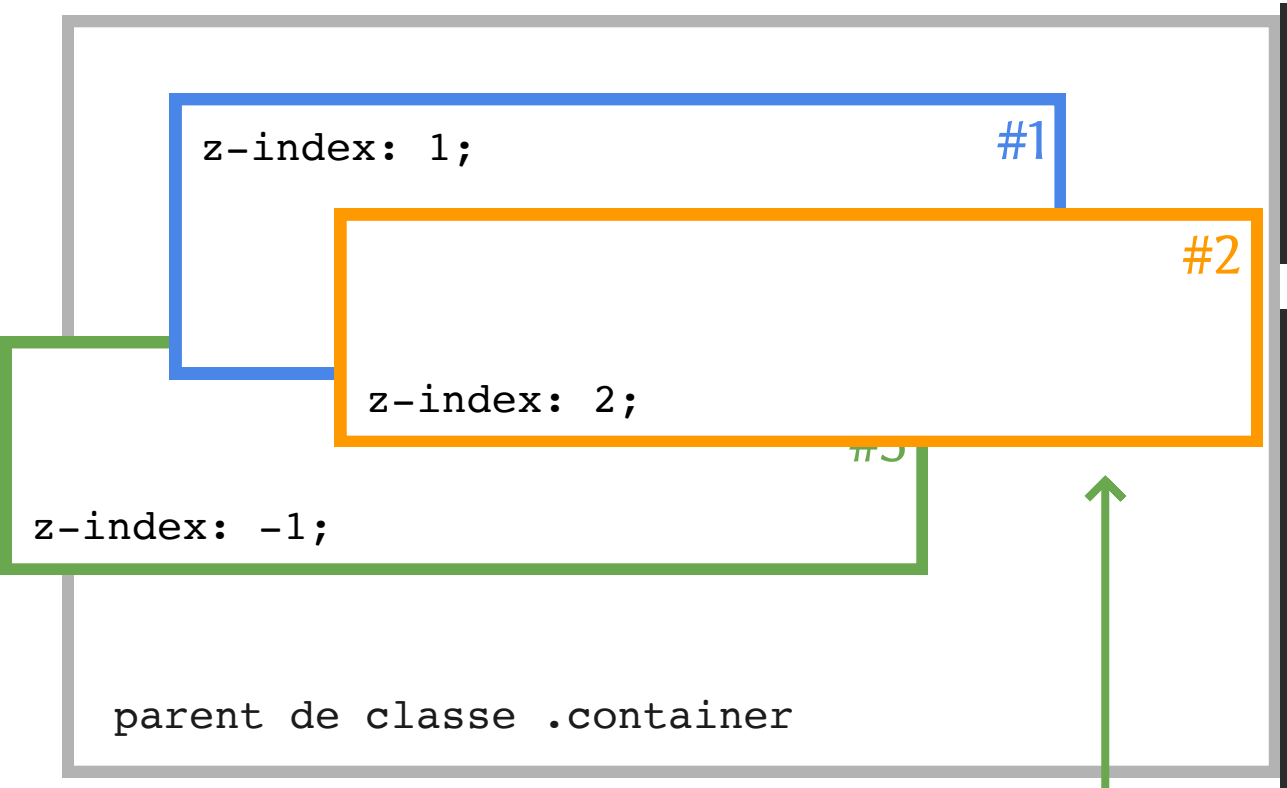



```html
<div .container>
  <div #1>
  <div #2>
  <div #3>
</div>
```

```css
#1 {
  z-index: 1;
}
#2 {
  z-index: 2;
}
#3 {
  z-index: -1;
}
```

La portée de z-index est celle de **son Contexte d'Empilement**
(Ici, le parent .container ne crée pas de Contexte d'Empilement et div#3 se place en dessous)

Stacking context (contexte d'empilement)

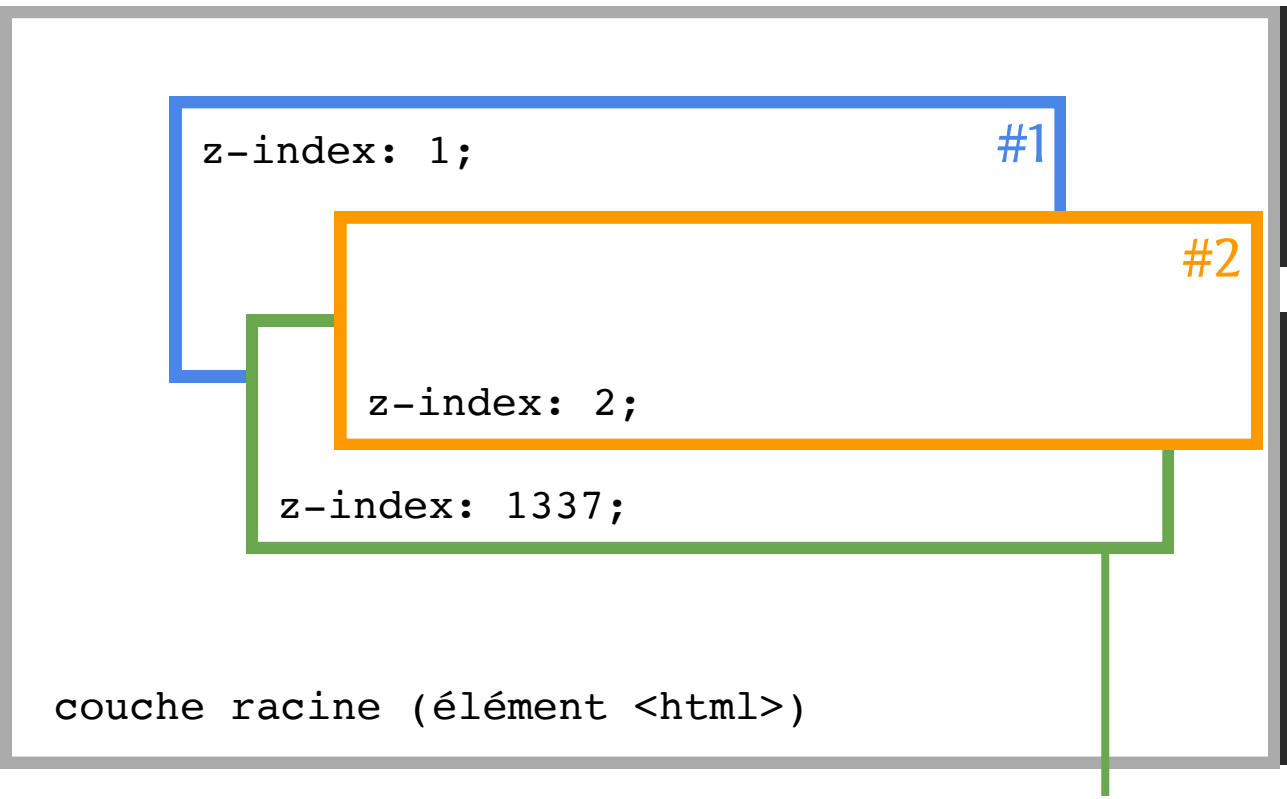



```html
<div .container>
  <div #1>
  <div #2>
  <div #3>
</div>
```

```css
.container {
  z-index: 1;
}
#1 {
  z-index: 1;
}
#2 {
  z-index: 2;
}
#3 {
  z-index: -1;
}
```

La portée de z-index est celle de **son Contexte d'Empilement**
(Ici, le parent .container crée un Contexte d'Empilement)

Stacking context (contexte d'empilement)




```html
<html>
  <div #1>
    <div #3>
  <div #2>
</html>
```

```css
#1 {
  z-index: 1;
}
#2 {
  z-index: 2;
}
#3 {
  z-index: 1337;
}
```

La portée de z-index est celle de **son Contexte d'Empilement**
(Ici, #3 est contenu dans #1 et chacun crée un Contexte d'Empilement)

Stacking context (contexte d'empilement)

1. Un Contexte d'Empilement s'applique à un conteneur pour gérer l'empilement de ses descendants
2. z-index définit l'Ordre d'Empilement des éléments au sein d'un Contexte d'Empilement
3. L'élément <html> constitue le premier Contexte d'Empilement
4. Certaines propriétés CSS créent un nouveau Contexte d'Empilement

aparté

qui crée un Contexte d'Empilement ?

Un nouveau Contexte d'Empilement (Stacking Context) est créé pour chaque élément correspondant à l'un de ces critères :

- un élément racine <html>
- un élément avec z-index autre que auto
- un élément en position fixed ou sticky

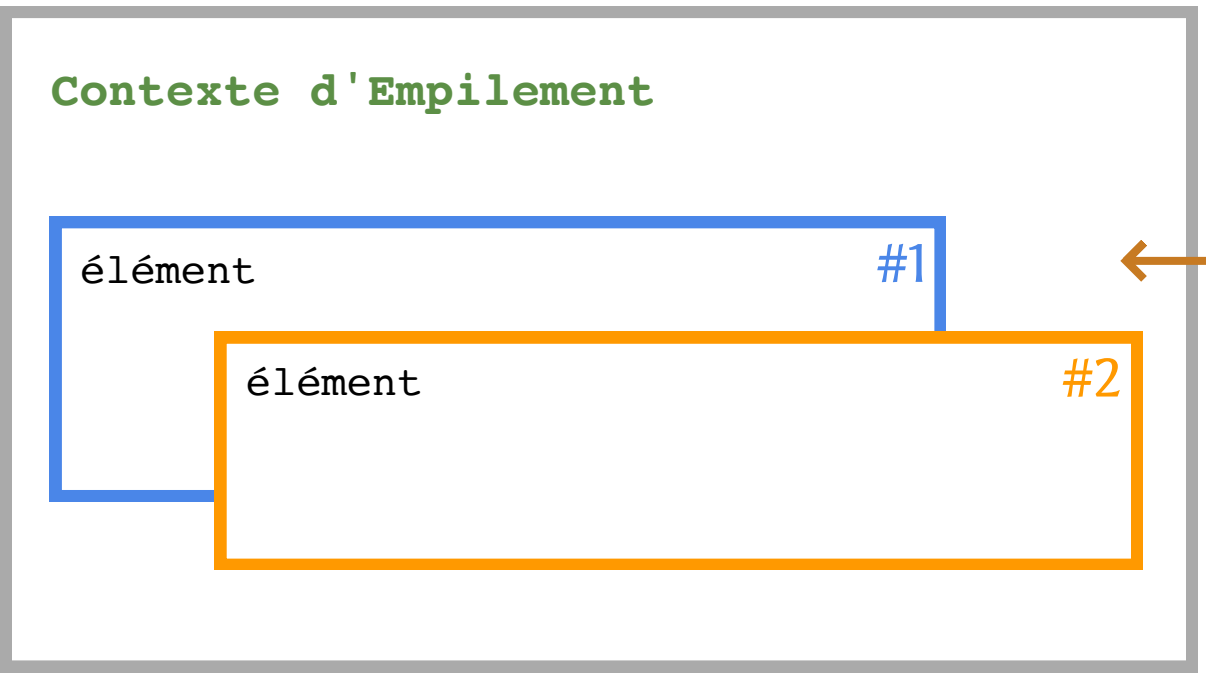
- un élément avec opacity inférieur à 1
- un élément avec transform autre que none
- un élément avec filter autre que none
- un élément avec will-change (de valeur transform ou filter)
- un élément avec isolation (de valeur isolate)
- un élément avec clip-path
- un élément avec mask

Il existe 17 façons de créer un Stacking Context;
Il n'en existe aucune pour en annuler un.

liste complète sur MDN

(alors n'en mets pas partout au départ)

Récap général



Ordre d'Empilement

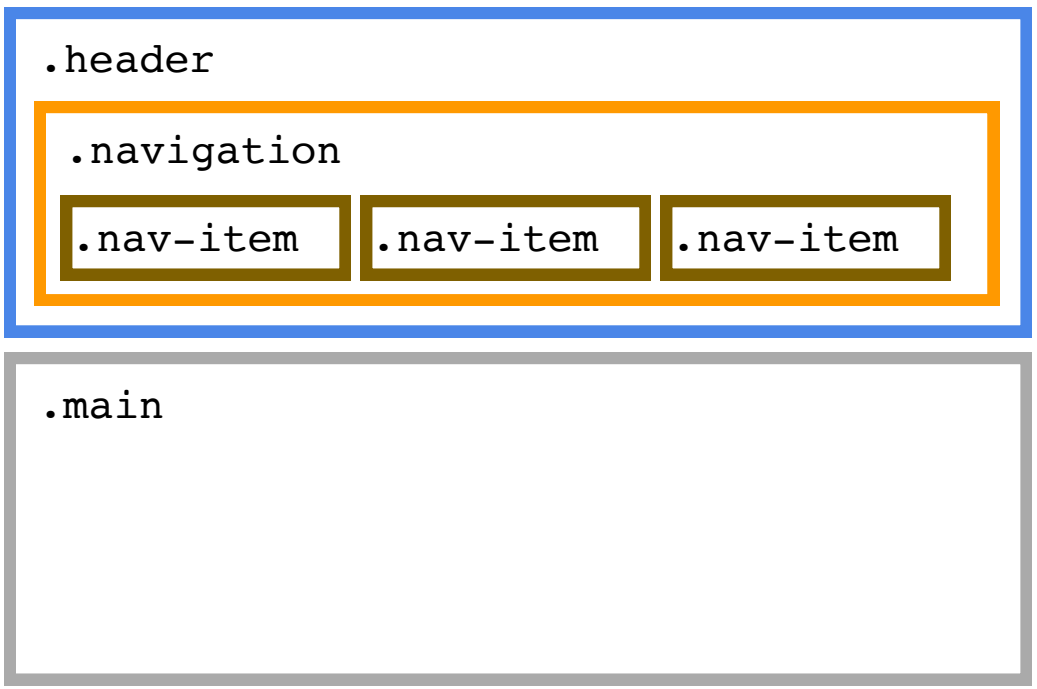
1. dans le Flux naturel
2. élément positionné ou transformé
3. z-index non auto

uniquement si positionné ou flex-item ou grid-item

Exemples d'éléments :

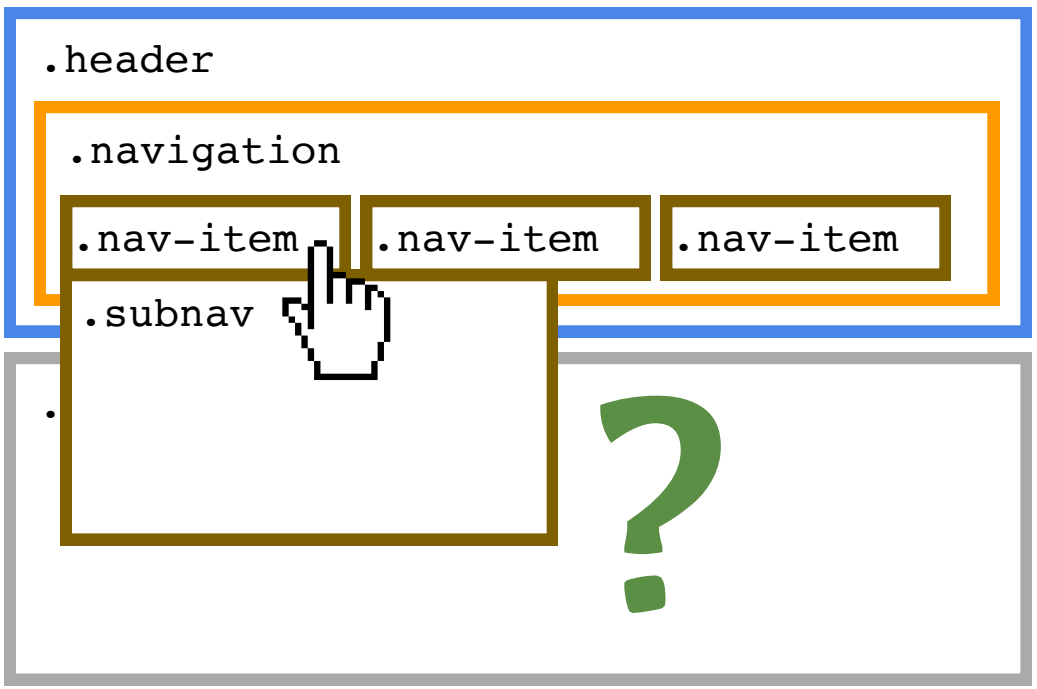
- z-index,
- en fixed, en sticky,
- transformé,
- filter,
- avec opacity <1,
- avec isolation,
- etc.

t'as compris ?



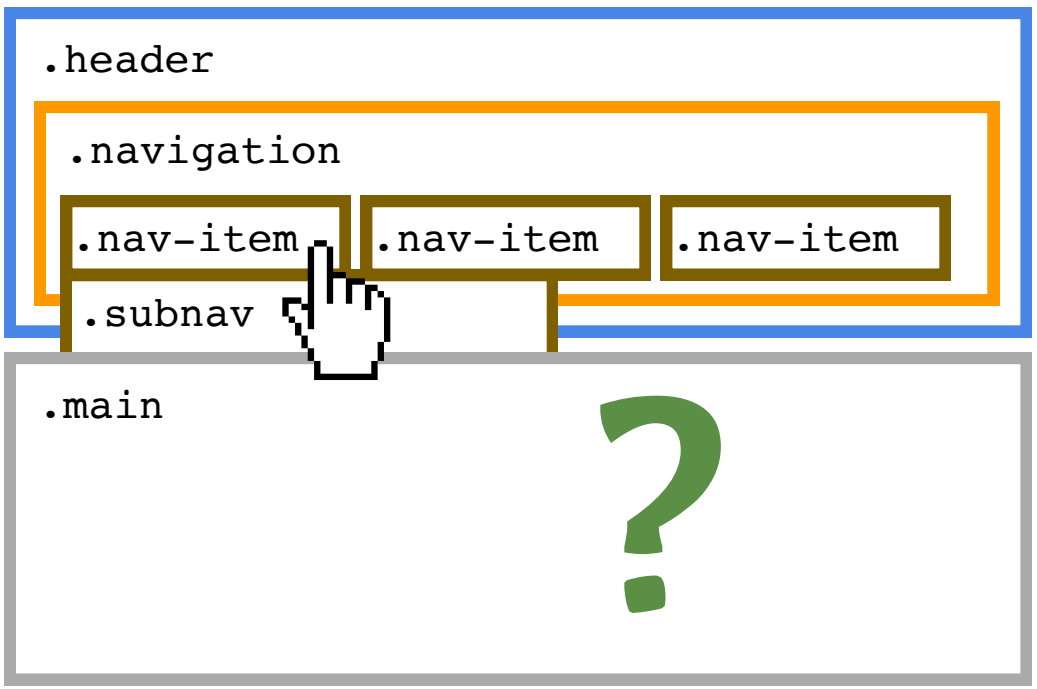

```css
.header {
  position: relative;
}
.main {
  position: relative;
}
```

Hypothèse 1



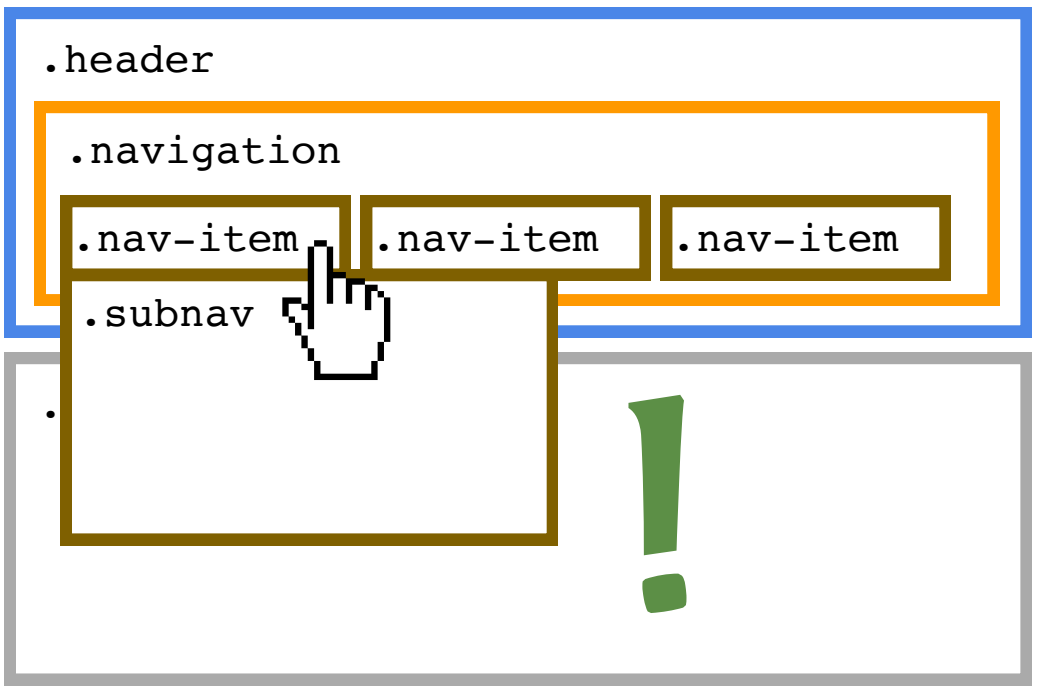

```css
.header {
  position: relative;
}
.main {
  position: relative;
}
.subnav {
  position: absolute;
  z-index: 1337;
}
```

Hypothèse 2




```css
.header {
  position: relative;
}
.main {
  position: relative;
}
.subnav {
  position: absolute;
  z-index: 1337;
}
```

Réponse : 1




```
1  .header {
2    position: relative;
3  }
4  .main {
5    position: relative;
6  }
7  .subnav {
8    position: absolute;
9    z-index: 1337;
10 }
```

1. .header, .main et .subnav sont tous les 3 dans *le même* Contexte d'Empilement (html ?)
2. L'Ordre d'Empilement est donc :
   .subnav (parce que z-index)
   .main (parce que positionné + 2e dans le DOM)
   .header (parce que positionné + 1er dans le DOM)

INTERLUDE

j'ai retenu des trucs ?

INTERLUDE

j'ai retenu des trucs ?

vrai ou faux ?

span { position: relative; }

... relative est la valeur par défaut de position

Faux !

la valeur par défaut de position est static

vrai ou faux ?

span { position: relative; }

... permet d'appliquer un z-index sur l'élément

Vrai !

z-index s'applique sur un élément positionné

vrai ou faux ?

span { position: relative; }

… change le niveau d'empilement

Vrai !

un élément positionné passe au-dessus des éléments non positionnés

vrai ou faux ?

span { position: relative; }

... crée un contexte d'empilement

Faux !

position: relative seul ne suffit pas à créer un stacking context (il faut ajouter z-index par exemple)

vrai ou faux ?

span { opacity: 0.99; }

... permet d'appliquer un z-index sur l'élément

Faux !

z-index s'applique uniquement sur un élément positionné ou un flex-item / grid-item

vrai ou faux ?

span { opacity: 0.99; }

... crée un contexte d'empilement

Vrai !

opacity < 1 crée un stacking context.
donc opacity 0 et 0.99 créent un SC, mais pas opacity 1

vrai ou faux ?

span { opacity: 0.99; }

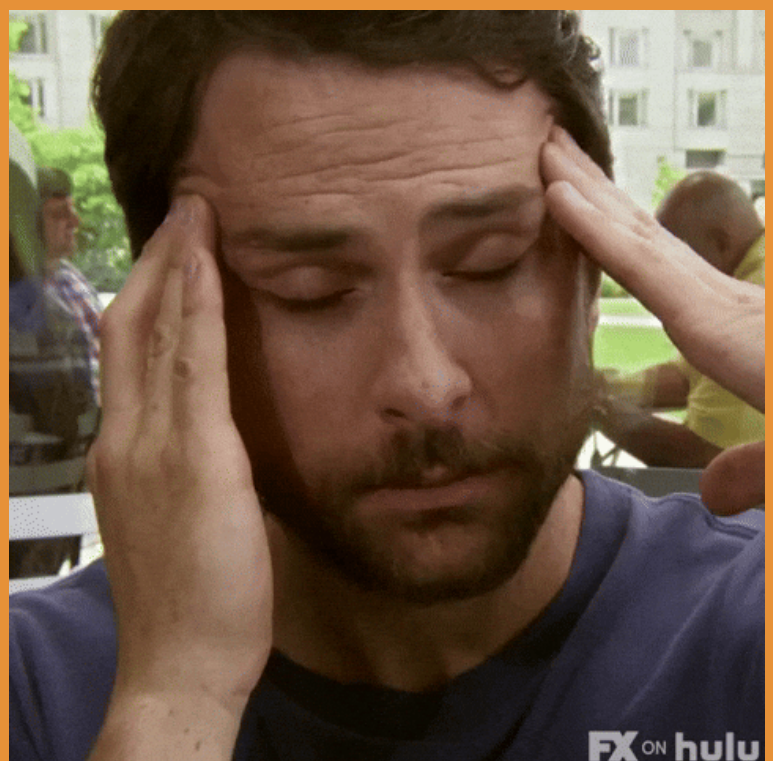
... change le niveau d'empilement

Faux !

le stacking order est modifié par les éléments positionnés, transformés ou ayant un z-index non auto

vrai ou faux ?

j'ai mal à la tête



Vrai !

Fil rouge

Project Wallace

z-index : combien et quelle valeur max ?

Z-indexes

TOTAL 78 UNIQUE 32 (41.0%)

Sort by source order | Sort by number | Sort by count

15×	1	2×	50	1×	200	1×	6000	1×	9999999
7×	10	2×	101	1×	460	1×	7000	1×	-1000
7×	450	2×	150	1×	3000	1×	7100	1×	-10
6×	0	2×	449	1×	3010	1×	7200	1×	9999999999
6×	-1	2×	1000	1×	3020	1×	7300		
3×	5	2×	26000	1×	3050	1×	9999		
3×	1000000	1×	100	1×	5000	1×	999000		

Neuf MILLIARDS

: This value might not work because it is not a valid 32-bit integer.

https://www.projectwallace.com/analyze-css