BEM

La tête la première !

C'est quoi le plan ?

C'est quoi le plan ?

1. Mes Premiers pas avec BEM
2. Les motivations
3. La FAQ
4. Les conseils pratiques
5. Les astuces Sass
6. Les limites

Avant de commencer...

Les slides sont disponibles :

https://tzi.fr/slides/riviera2018-bem

https://tzi.fr/slides/riviera2018-bem.pdf

Mes Premiers pas avec BEM

Qu'est-ce que c'est ?

~~Une méthodologie~~

Une convention de nommage

Qu'est-ce que c'est ?

B

Block

E

Element

M

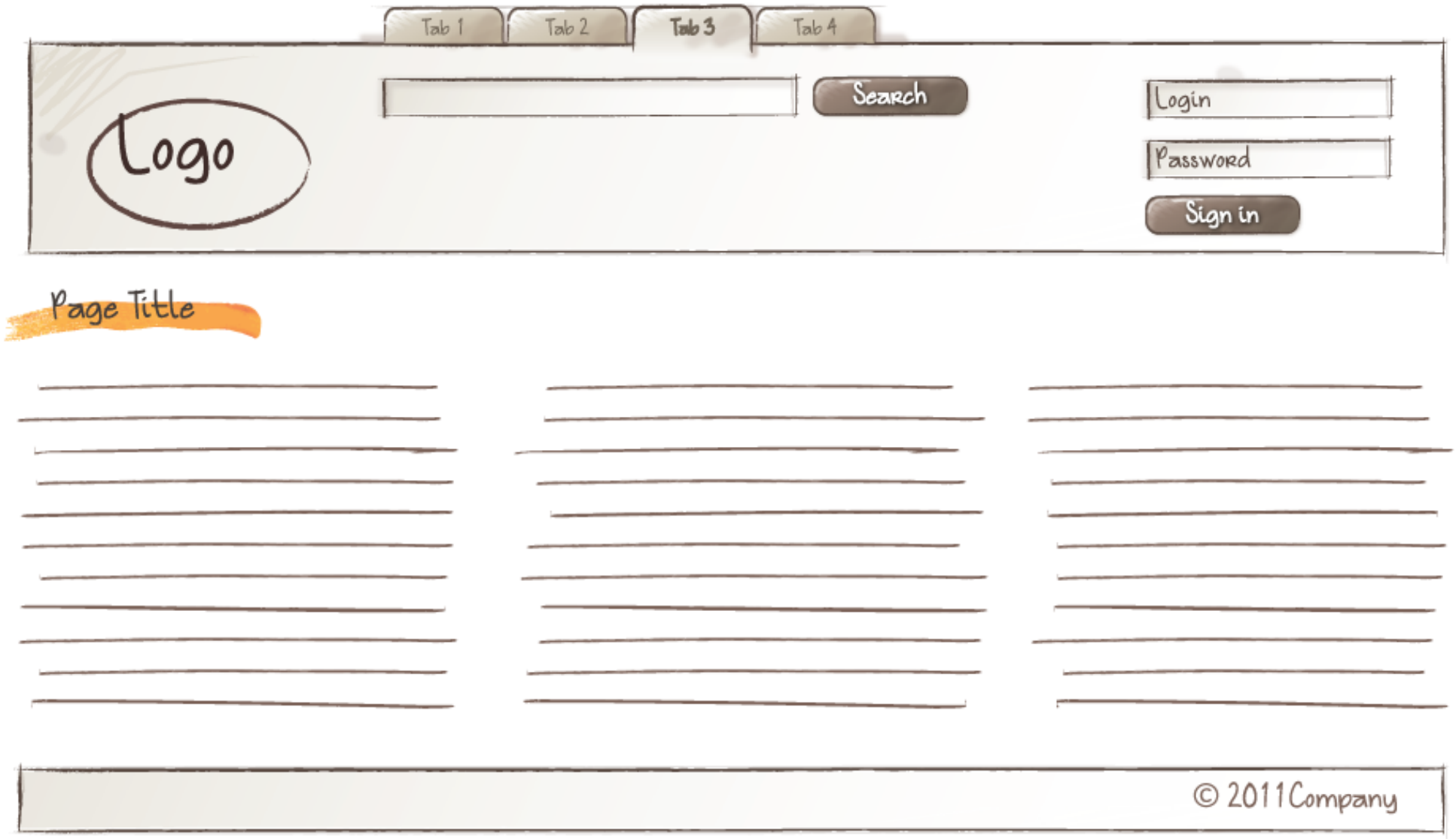
Modifier

Block

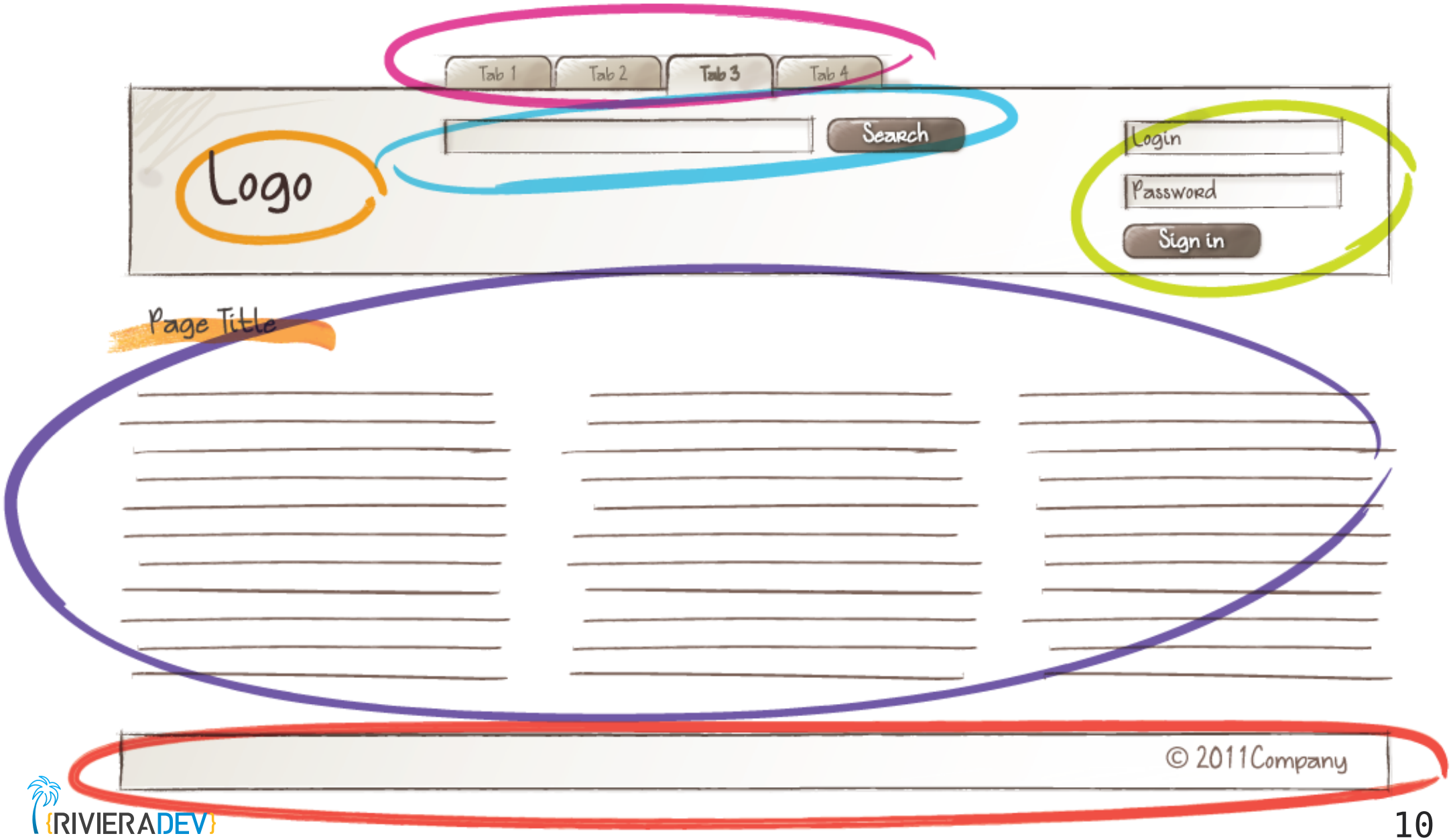
Un block est :

- Une portion d'une page
- Dont le style est indépendant
- Dont le fonctionnel est indépendant

Block


Tab 1
Tab 2
Tab 3
Tab 4
Logo
Search
Login
Password
Sign in
Page Title
© 2011 Company

Block

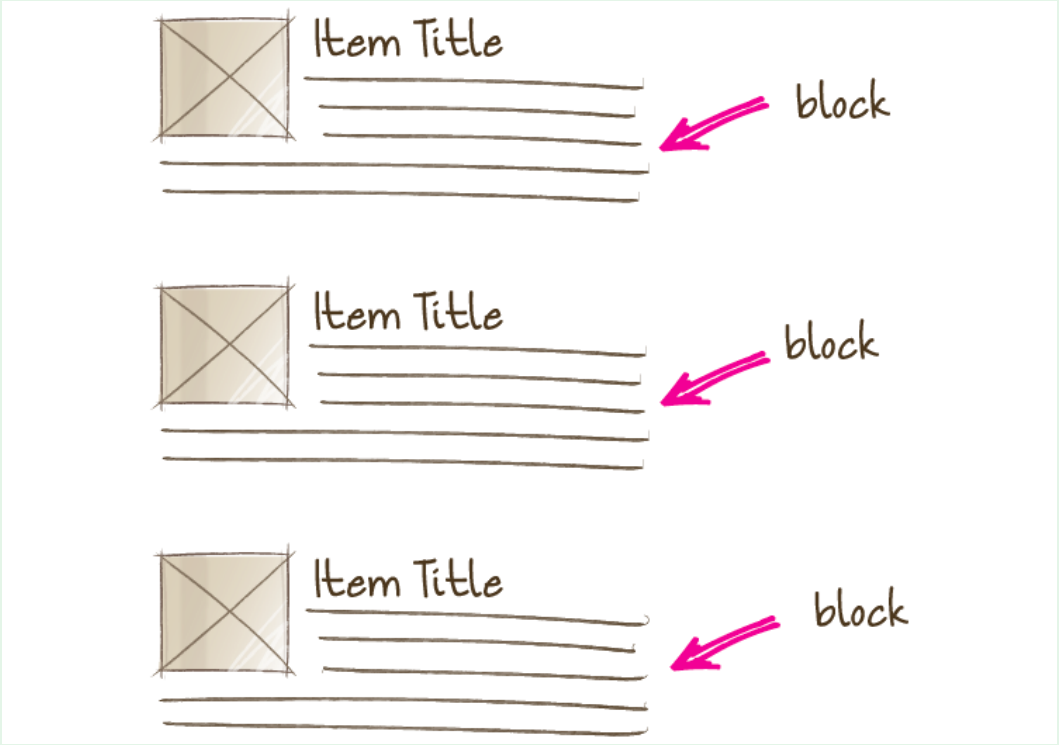

Tab 1
Tab 2
Tab 3
Tab 4
Logo
Search
Login
Password
Sign in
Page Title
© 2011 Company

Block

Ils sont indépendants

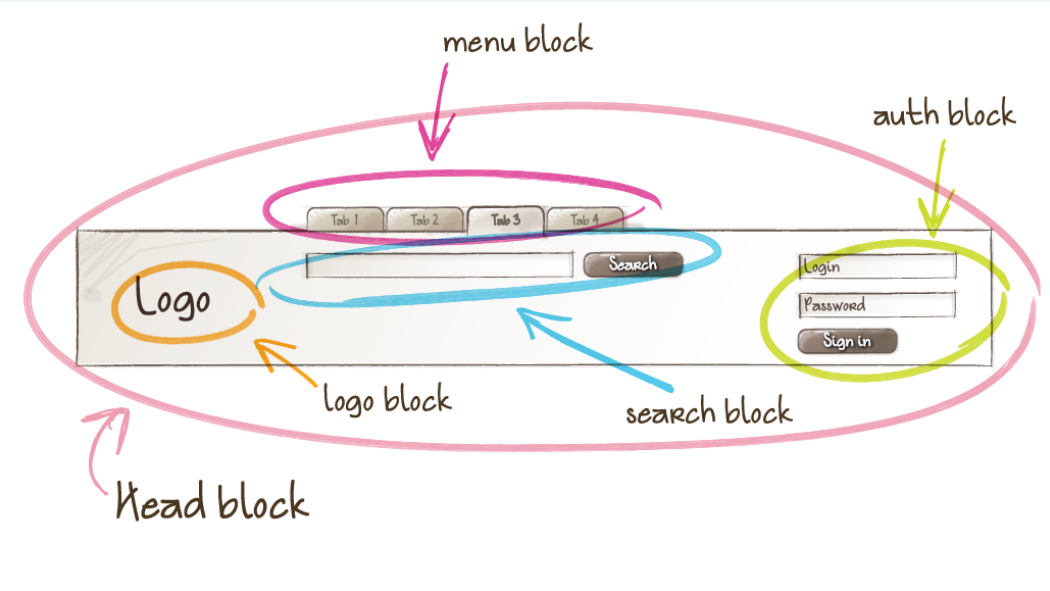
Block

Ils peuvent être réutilisés



Block

Ils peuvent être imbriqués



Block

```
/* syntax */
.block-name { ... }

/* example */
.top-menu { ... }
```

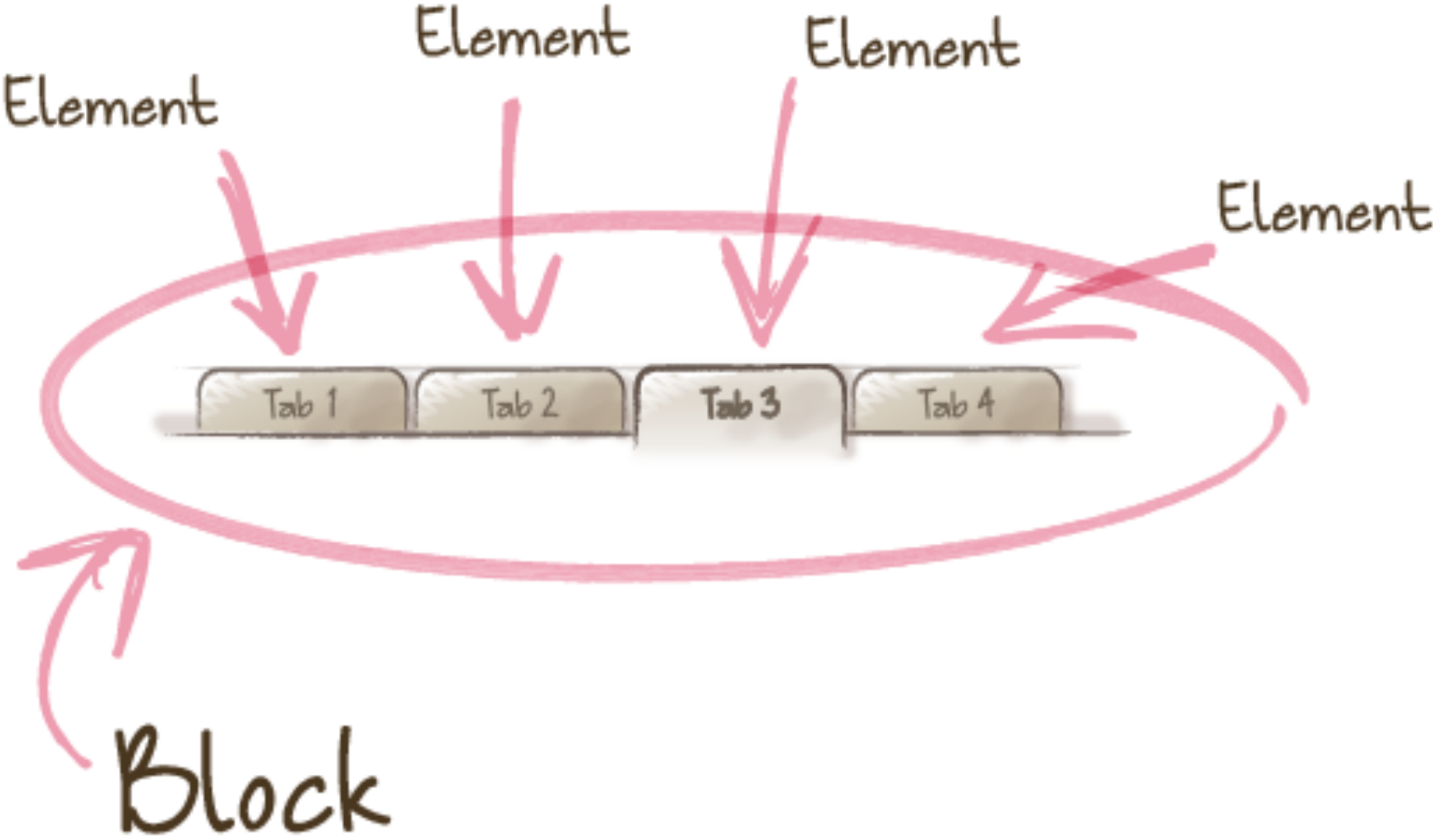
Element

Un élément est :

- Une portion d'un block
- Il est dépendant de son block
- Il ne peut pas être utilisé sur un autre block

Element

Element



Element

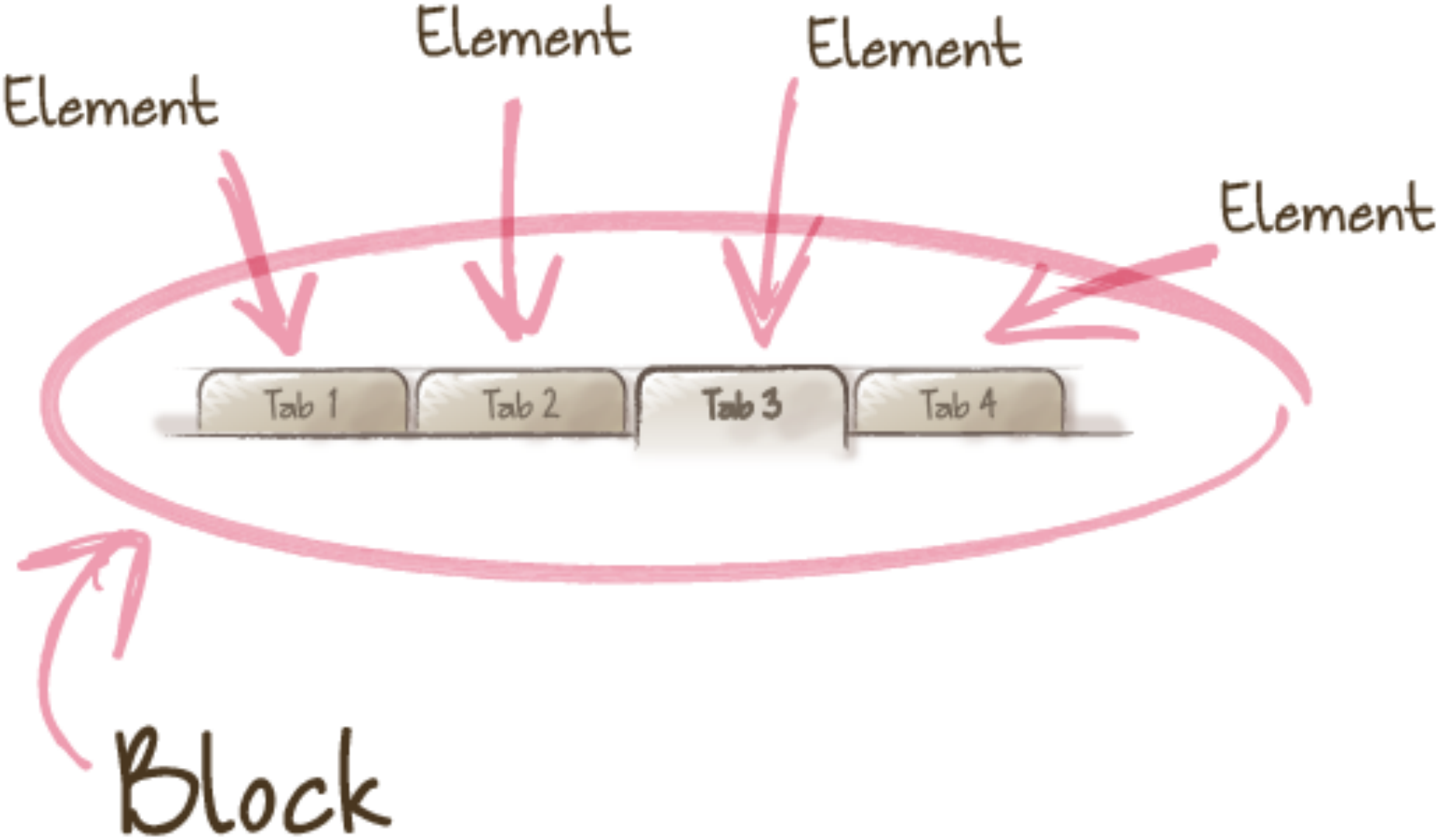
```
/* syntax */
.block-name__element-name { ... }

/* example */
.top-menu__tab { ... }
```

Modifieur

Un modifieur :

- modifie les propriétés d'une entité (block ou élément)
- s'utilise de pair avec une entité
- une entité peut avoir plusieurs modifieurs

Modifieur



Modifieur

Modifieur

```
/* syntax */
.block-name--modifier-name { ... }
.block-name__element-name--modifier-name { ... }

/* example */
.top-menu__tab--active { ... }
```

Côté HTML

```
<nav class="top-menu">
  <a class="top-menu__tab">Tab 1</a>
  <a class="top-menu__tab">Tab 2</a>
  <a class="top-menu__tab top-menu__tab--active">Tab 3</a>
  <a class="top-menu__tab">Tab 4</a>
</nav>
```

Astuce de papa !

Où dois-je appliquer cette classe ?

- Les `___` (tirets du bas) vont sous leur block.
- Les `- -` (tirets du milieu) vont à côté de leur entité.

Mes Premiers pas avec BEM

On retient :

- Block est le noeud racine
- Element compose un block `__`
- Modifier altère un block ou un élément `--`

Les motivations

Adopter une méthodologie

- CSS est simple à apprendre

Adopter une méthodologie

- CSS est difficile à maîtriser
- CSS est difficile à maintenir

Adopter une méthodologie

> CSS is like a bear cub.
>
> — *Andres Galante*

Adopter une méthodologie

> It's almost a challenge to find a development team [...] where the CSS isn't the most frightening and hated part of that system.
>
> — *Andy Hume*

Adopter une méthodologie

Les symptômes possibles :

- Le temps de développement
- La taille des feuilles de style
- Les aller-retours de bugs
- L'arrivée de nouveau collaborateurs
- Les travaux réservés aux expert

Adopter une méthodologie

Les électrochocs qui poussent à changer :

- Le responsive
- La marque-blanche
- Les performances front-end
- Le temps de développement

Donner du sens aux classes

> Underscores and Meaningful Dashes
>
> — *introduced by Nicolas Gallagher*

Donner du sens aux classes

Les noms de classe peuvent être ambigus

```
<form class="search full">
    <input class="input dropdown">
    <button class="button">
        Search
    </button>
</form>
```

Donner du sens aux classes

BEM explicite le rôle de chacun

```
<form class="search-form c-search-form--full">
    <input class="search-form__input / js-dropdown">
    <button class="button">
        Search
    </button>
</form>
```

Donner du sens aux classes

BEM explicite aussi les problèmes de jointures

```
<form class="search-form c-search-form--full">
    <input class="search-form__input / js-dropdown">
    <button class="button">
        Search
    </button>
</form>
```

Donner du sens aux classes

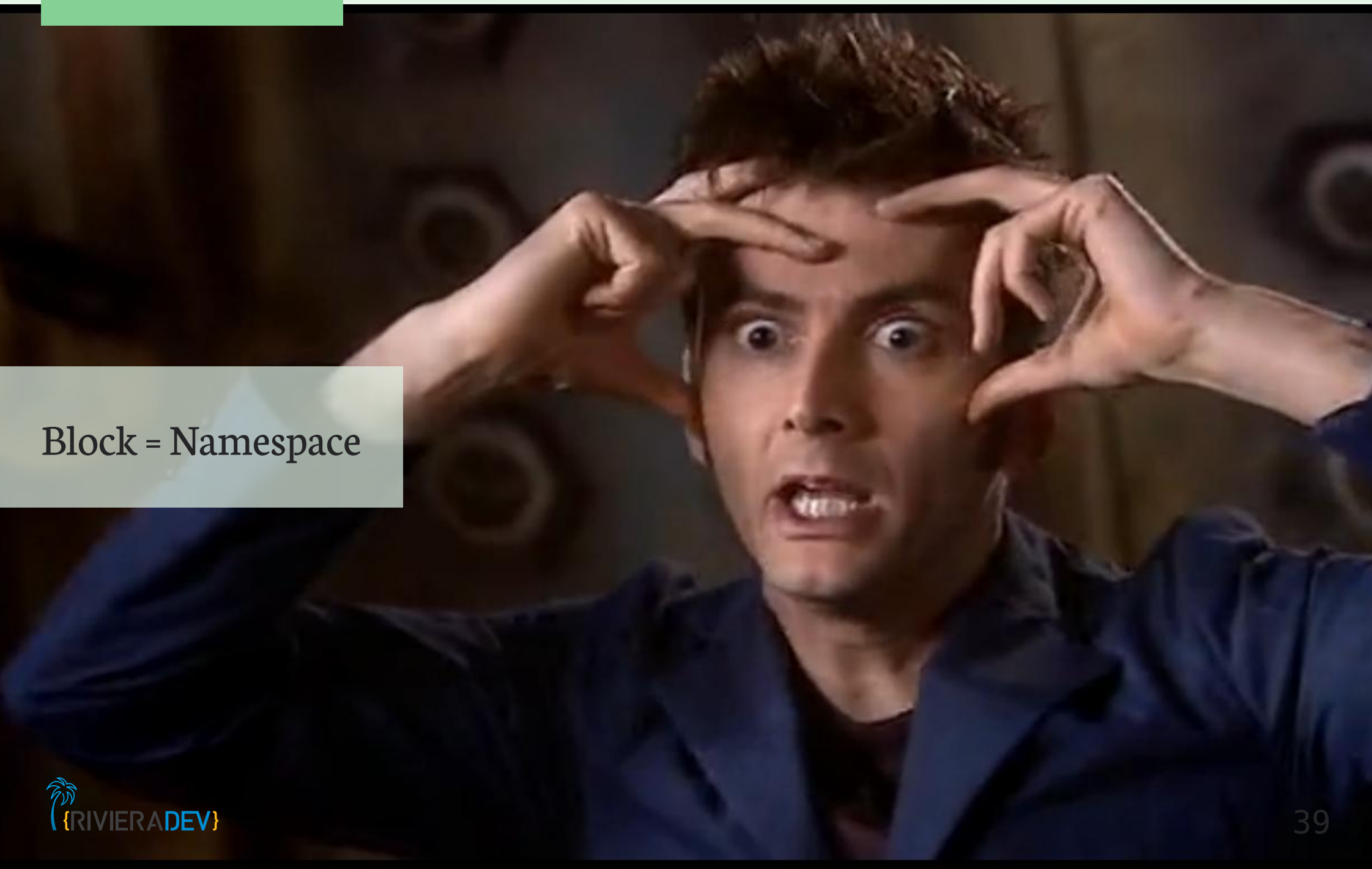
```
<form class="search-form c-search-form--full">
    <input class="search-form__input / js-dropdown">
    <span  class="search-form__action">
        <button class="button">
            Search
        </button>
    </span>
</form>
```

Isolation

BEM permet de regrouper les styles par block.

- Un fichier par block
- Qui contient tous ses éléments et modifieurs
- Qui contient que ses éléments et modifieurs

Isolation



Block = Namespace

Isolation

Si on ne connaît pas toutes les règles d'un fichier :

- On rajoute à la fin du fichier
- On duplique
- On introduit des incohérences
- Les nouvelles ressources pleurent

Réduire la spécificité

Si deux règles sont en conflits,

celle avec la pus grande spécificité gagne.

Réduire la spécificité

Plus l'application grandit,

plus il y a de risque de conflits

Réduire la spécificité

```
#contact-page { .. } /* count 1000 by id */
.button,
[type="submit"],
:hover { ... }       /* count 1 by class, attribute, state */
a,
::before { ... }     /* count 0.1 by element */
```

IT'S GAME TIME!

GAME : Quelle spécificité ?

Niveau facile

```
.header a { ... }
```

GAME : Quelle spécificité ?

Niveau moyen

```
#nav .selected > a:hover { ... }
```

GAME : Quelle spécificité ?

Niveau avancé

```
li:first-child h2 .title::before { ... }
```

GAME : Quelle spécificité ?

Niveau expert

```
:not(:not(.title), :not(.important)) { ... }
```

GAME : Quelle spécificité ?

Ces deux sélecteurs sont équivalents

```
:not(:not(.title), :not(.important)) { ... }
.title.important { ... }
```

... mais n'ont pas la même spécificité !

GAME : Quelle spécificité ?

Réduire la spécificité

```
.news .title { ... }
```

Réduire la spécificité

```
.news .title { ... }
/* ... */
#sidebar .news .title { ... }
```

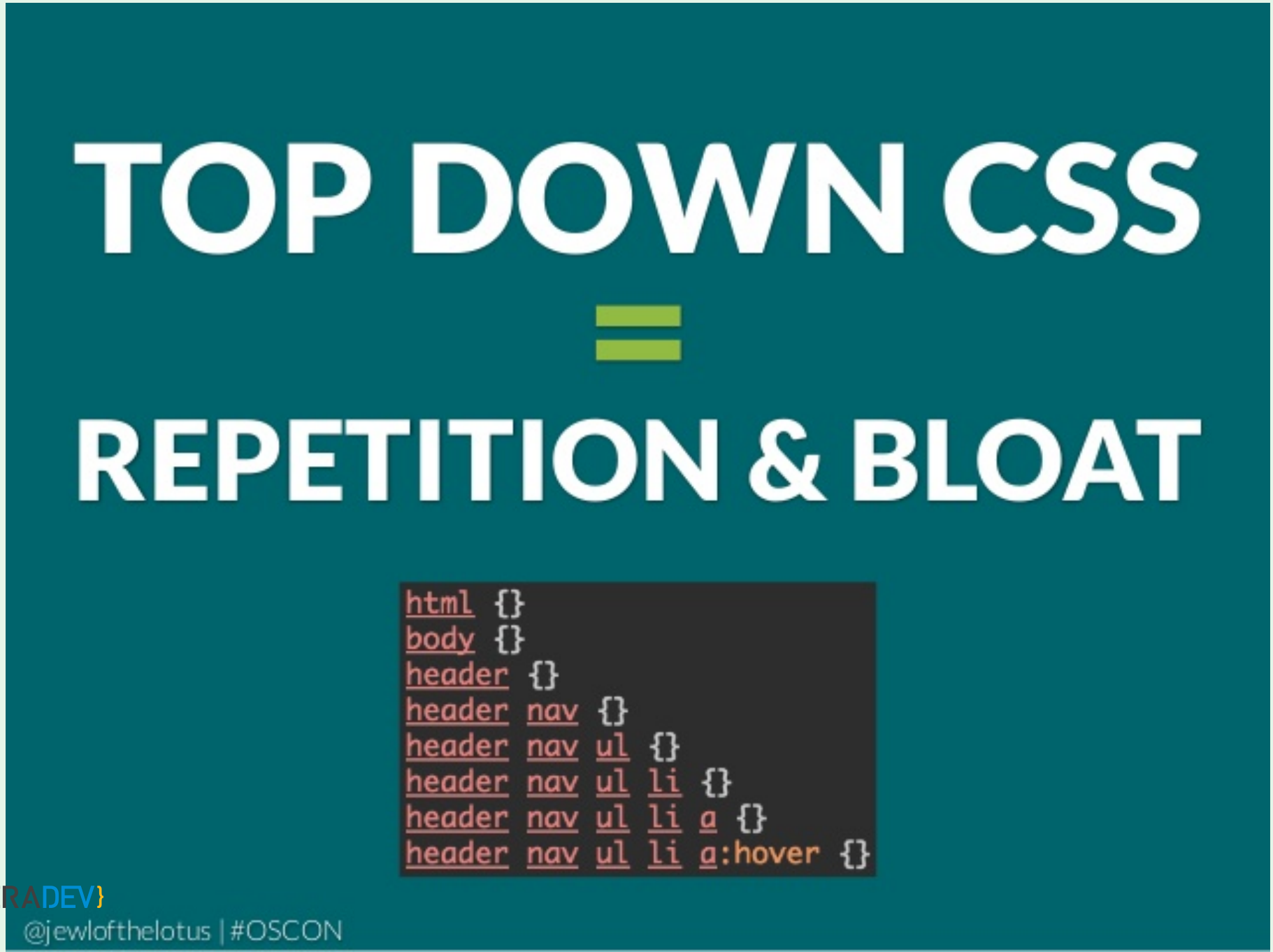
Réduire la spécificité

```
.news .title { ... }
/* ... */
#sidebar .news .title { ... }
/* ... */
.widget .news .title { ... }
```

Réduire la spécificité

```
.news .title { ... }
/* ... */
#sidebar .news .title { ... }
/* ... */
.widget .news .title,
#sidebar .widget .news .title { ... }
```

Réduire la spécificité



Réduire la spécificité

BEM limite l'effet boule de neige :

```
.news__title { ... }
/* ... */
.news__title--sidebar { ... }
/* ... */
.news__title--widget { ... }
```

Les motivations

On retient :

- Adopter une méthodologie est indispensable.
- BEM donne du sens aux classes
- BEM Isole les styles, réduit les zones de travail
- BEM Limite la spécificité des sélecteurs, une seule classe !

L'anecdote de papa !

BEM a été inventé en 2011,

A l'époque, l'avenir c'était les web components !

FAQ

Question n°1

C'est bien de préfixer ses classes BEM avec `c-` ?

```
<nav class="c-menu"></nav>
```

Réponse n°1

```
<nav class="c-menu"></nav>
```

Oui.

1. On explicite l'usage de BEM même sur les blocks
2. On peut isoler des styles existants, migration douce
3. On peut s'isoler des feuilles externes

L'anecdote de papa !

Lorsque votre projet grandit,

vous pouvez aussi ajouter des préfixes !

L'anecdote de papa !

L'anecdote de papa !

La home Du Figaro, c'est :

- 4 525 sélecteurs
- 3 492 noeuds HTML

Question n°2

On peut enlever la classe si elle est redondante avec l'élément ?

```
<button class="c-button"></button>
/* or */
<button></button>
```

Réponse n°2

```
<button></button>
```

Non.

1. On explicite l'usage de BEM
2. On permet la réutilisation des styles des boutons ailleurs
3. On permet l'utilisation d'autres styles sur des boutons

Question n°3

On peut enlever le nom du block des modifieurs ?

```
<div class="c-search-form c-search-form--full"></div>
/* or */
<div class="c-search-form full"></div>
```

Réponse n°3

```
<div class="c-search-form full"></div>
```

Non.

1. On explicite le sens des modifieurs
2. On uniformise les sélecteurs
3. On s'isole du reste du monde

Question n°4

On peut n'utiliser que le modifieur ?

```
<div class="c-search-form c-search-form--full"></div>
/* or */
<div class="c-search-form--full"></div>
```

Réponse n°4

```
<div class="c-search-form--full"></div>
```

Non.

1. On permet l'utilisation de plusieurs modifieurs
2. On simplifie l'ajout et la suppression via JavaScript

Question n°5

Est-ce qu'un modifieur de block peut affecter un élément ?

```
.c-search-form--full .c-search-form__input { … }
```

Réponse n°5

```
.c-search-form--full .c-search-form__input { … }
```

Oui.

C'est une exception où ne peut pas limiter la spécificité des sélecteurs.

Question n°6

Est-ce la bonne façon de déclarer les éléments d'élément ?

```
.c-search-form__label__icon { … }
```

Réponse n°6

```
.c-search-form__label__icon { … }
```

Non. Il n'y a pas d'élément d'élément. On préfère :

```
.c-search-form__label-icon { … }
.c-search-form__icon { … }
```

FAQ

On retient :

- On préfixe ses classes
- Les éléments d'éléments n'existent pas
- On respecte les règles

Les conseils pratiques

Adoptez BEM !

Google, Twitter, BBC, theguardian, Financial Time, DuckDuckGo, Dropbox, JetBrains, SoundCloud, WordPress, BuzzFeed, Overblog, 6play, LeFigaro, ...

Adoptez BEM !

BEM est une méthodologie tout terrain :

- Toutes les tailles de projet
- Toutes les tailles d'équipes
- Toutes les stacks techniques
- ...

Adoptez BEM !

Surtout BEM respecte la philosophie des CSS

- BEM est facile à apprendre

Adoptez BEM !


WE JUST INTRODUCED BEM
Spongebob Squarepants vector trace by kssael. ©Nickelodeon
EVERYTHING IS GOING TO BE ALRIGHT.
memegenerator.net

La blague de papa !

Surtout BEM respecte la philosophie des CSS

- BEM est difficile à maîtriser

L'expérience

Qui suis-je ?

Thomas Zilliox, expert CSS freelance.

Découvre le CSS depuis 12 ans maintenant.

J'arrive enfin à produire du code maintenable !

Qui suis-je ?

Et avec 16 développeurs front-end actifs ?

Toujours maintenable ?

Attention aux modifieurs

Les modifieurs sont un hack !

Attention aux modifieurs

BEM permet à un élément de se déplacer sans changer de nom.

```
<div class="block block--big-title">
    <div class="block__title">
        <div class="block__icon"></div>
    </div>
</div>
```

Attention aux modifieurs

Alors on évite les opérateurs entre les classes :

```
.block--big-title > .block__title { … } /* non */
.block--big-title .block__title { … }   /* oui */
```

Attention aux modifieurs

Pour limiter la spécificité des sélecteurs,

on préfère les modifieurs d'éléments aux modifieurs de blocks :

```
<div class="block">
    <div class="block__title block__title--big"></div>
</div>
```

Attention aux modifieurs

Pour limiter la dépendance entre plusieurs règles CSS,

on préfère encore plus créer un nouveau sélecteur :

```
<div class="block">
    <div class="block__big-title"></div>
</div>
```

Repeat Yourself

Tout le principe de BEM est basé sur l'isolation des styles.

Attention aux partage de code :

- `@extends` en Sass
- `@mixins` en Sass

Repeat Yourself

C'est aussi valable pour l'implémentation des blocks :

```
<div class="block">
    <div class="block__big-title"></div>
</div>
```

Repeat Yourself

```
.c-alert--info,
.c-alert--warning,
.c-alert--error {
    padding: 5px 10px;
}
```

Ne pas styler avec le contexte

On ne peut pas styler un composant avec son contexte.

```
.c-sidebar .c-article { ... } /* non */
```

Ne pas styler avec le contexte

Il faut concevoir différement, pas juste changer la façon décrire.

```
.c-article--sidebar { ... } /* non */
```

Sinon vous aller juste limiter la réutilisation de votre code.

Décrivez plutôt quel changement visuel vous souhaitez.

Ne pas styler avec le contexte

Utiliser un Bemlinter peut aider, il vérifie :

- 1 seul block par fichier
- un seul fichier par block

Le découpage

> One of the hardest parts of BEM is deciding when to start and stop scope, and when (or not) to use it
>
> — *Harry Roberts*

Le découpage

Il faut d'abord éviter le sur-découpage.

Se laisser guider par l'usage :

- Si deux blocks sont toujours utilisés ensemble,

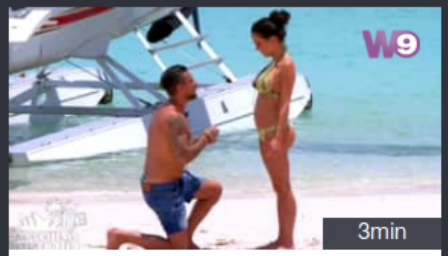
c'est sûrement le même block

- Si un élément peut être utilisé ailleurs,

c'est sûrement un block différent

Le découpage

Bande-annonce



Les premières images des Mars...

LES MARSEILLAIS AUSTRALIA

Les Marseillais s'envolent à l'autre bout du monde et posent leurs valises en Australie. Ensemble, ils relèvent le défi de s'imposer en Australie et de découvrir un pays aussi magnifique que surprenant ! Certains sont venus tester leur couple alors que d'autres vont se préparer à un changement de vie radical ! Pour Julien, Jessica, Kevin, Carla, Paga, Manon, Thibault et le reste de la famille, l'heure des grandes décisions a sonné !

PROCHAINE DIFFUSION

MERCREDI 16/05 À 18H50

SUR W9

Vous aimerez aussi...

Le découpage

Bande-annonce

Les premières images des Mars...

LES MARSEILLAIS AUSTRALIA

Les Marseillais s'envolent à l'autre bout du monde et posent leurs valises en Australie. Ensemble, ils relèvent le défi de s'imposer en Australie et de découvrir un pays aussi magnifique que surprenant ! Certains sont venus tester leur couple alors que d'autres vont se préparer à un changement de vie radical ! Pour Julien, Jessica, Kevin, Carla, Paga, Manon, Thibault et le reste de la famille, l'heure des grandes décisions a sonné !

PROCHAINE DIFFUSION

MERCREDI 16/05 À 18H50

SUR W9

Vous aimerez aussi...

IT'S QUIZZ TIME!

Quizz

Quelle est la taille maximale d'un block ?

- A : 50 lignes de CSS
- B : 100 lignes de CSS
- C : 200 lignes de CSS
- D : ça dépend

Quizz

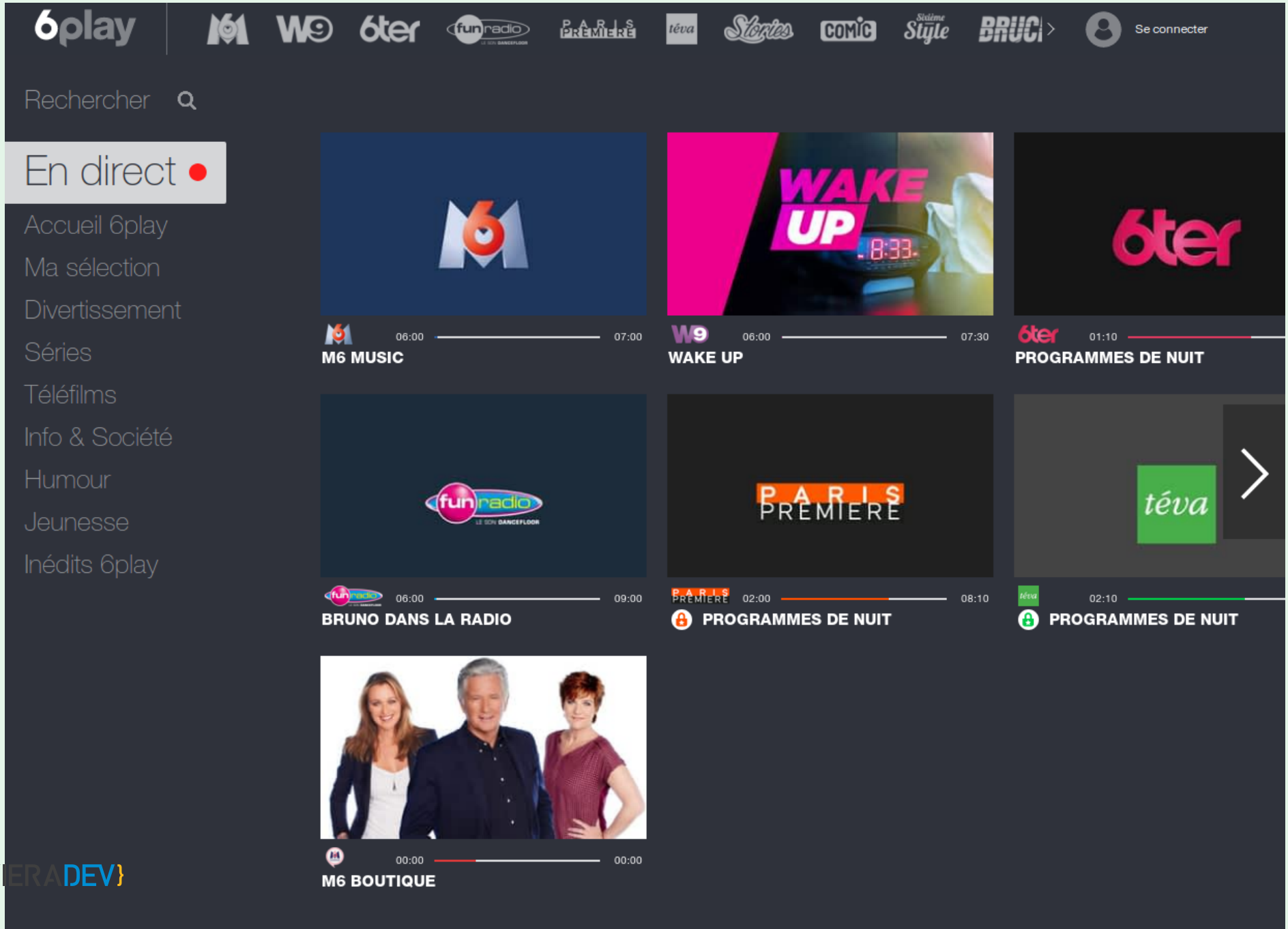
Faire confiance à son parent

> Don’t Layout Yourself: A component should style itself, but give up the task of layouting to its parent.
>
> — *Thai Pangsakulyanont*

Faire confiance à son parent

Faire confiance à son parent

Faire confiance à son parent



Les conseils pratiques

On retient :

- Les modifieurs ne sont pas la norme
- On accepte de se répéter
- On découpe en fonction de l'usage
- On ne se dimensionne pas soit-même

Les astuces Sass

Limiter les mixins

Vous ne devriez utiliser que des mixins qui ne sont pas spécifiques à votre projet :

- clearfix
- ellipse de texte
- texte caché accessible
- ...

Limiter les mixins

Vos mixins ne doivent pas générer de sélecteurs !

Concaténation

On peut concaténer nos classes en Scss :

```
.block {
    &__element {
        ...
        &--modifier { ... }
    }
}
```

Concaténation

Ceci génére le code CSS suivant :

```
.block { ... }
.block__element { ... }
.block__element--modifier { ... }
```

Concaténation

Mais un modifieur de block peut limiter cette écrite :

```
.block {
    &__element { ... }
    &--modifier &__element { ... }
}
```

Local variables

Les variables peuvent être accesibles depuis un autre block :

```
$block-height: 50px;
.block {
    &__element { height: $block-height; }
}
/* Leak */
.other-block { height: $block-height; }
```

Local variables

Pour isoler vos variables :

```
.block {
    $height: 50px;
    &__element { height: $height; }
}
/* Doesn't work. No leak! */
.other-block { height: $height; }
```

Local variables

On peut alors *sauvegarder* notre block dans une variable :

```
.block {
    $block: &;
    &__element {
        ...
      #{$block}--modifier & { ... }
    }
}
```

L'anecdote de papa !

Vous savez combien il faut de lignes pour

convertir une chaîne de caracter en nombre

à l'aide de Sass ?

L'anecdote de papa !

```scss
@function to-number($value) {
  @if type-of($value) == 'number' {
    @return $value;
  } @else if type-of($value) != 'string' {
    $_: im-log('Value for `to-number` should be a number or a string.');
  }

  $first-character: str-slice($value, 1, 1);
  $result: 0;
  $digits: 0;
  $minus: ($first-character == '-');
  $numbers: ('0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9);

  // Remove +/- sign if present at first character
  @if ($first-character == '+' or $first-character == '-') {
    $value: str-slice($value, 2);
  }

  @for $i from 1 through str-length($value) {
    $character: str-slice($value, $i, $i);

    @if not (index(map-keys($numbers), $character) or $character == '.') {
      @return to-length(if($minus, -$result, $result), str-slice($value, $i))
    }

    @if $character == '.' {
      $digits: 1;
    } @else if $digits == 0 {
      $result: $result * 10 + map-get($numbers, $character);
    } @else {
      $digits: $digits * 10;
      $result: $result + map-get($numbers, $character) / $digits;
    }
  }

  @return if($minus, -$result, $result);
}
```

L'anecdote de papa !

Concaténation

Attention, à ne pas abuser de la concaténation qui rend impossible les fonctionnalités de rechercher / remplacer

Concaténation

Si vous voulez des variables locales sans concaténation :

```
.block {
    $height: 50px;
    @at-root {
      .block__element { height: $height; }
    }
}
```

Les astuces Sass

On retient :

- On limite les mixins
- On isole les variables
- On choisit si on souhaite concaténer ou non

Les limites de BEM

Le bilan

BEM a de gros avantages :

- Le concept simple à comprendre
- La mise en place est souple
- Ce n'est pas lié à un outils
- BEM est compatible avec les autres méthodologies

Le bilan

On lui reproche :

- Trop de redondance
- Trop de nommage
- Des problèmes de scalabilité avec le nombre de blocks

Avec OOCSS

Une idée est d'importer des règles d'OOCSS :

- Séparer la structure de l'habillage (CSS)
- Séparer le conteneur du contenu (HTML)

Avec OOCSS

> The component should still rely on OOCSS utilities as much as possible. Only styles that are not available through OOCSS utilities should be set in the component class.
>
> — *Rangle*

Avec OOCSS

```
<div class="c-menu">
    <div class="l-grid">
        <div class="l-grid__cell">
            <a class="c-menu__tab"></a>
        </div>
    </div>
</div>
```

Avec Atomic CSS

Une autre idée est de remplacer les modifieurs BEM par des classes atomiques :

- Les modifieurs sont difficiles à nommer
- Les modifieurs sont complexes

Avec Atomic CSS

```
<div class="c-article">
    <h1 class="c-article__title u-uppercase"></h1>
    <div class="c-article__status u-green"></div>
</div>
```

Les limites de BEM

On retient :

- BEM n'est pas une impasse
- Il faut aller voir ce qui se fait ailleurs
- Il faut trouver *sa* méthodologie

C'est fini !

Astuce de papa !

BEM sert à donner du sens à des chaînes de caractères.

On peut donc l'utiliser à plein d'occasions.

Astuce de papa !

EDF__CGV.pdf

EDF__Facture--2014.pdf

EDF__Facture--2015.pdf

EDF__Facture--2016.pdf

Astuce de papa !

Merci !

Des questions ?

@iamtzi

tzi.fr/slides/riviera2018-bem

tzi.fr/slides/riviera2018-bem.pdf