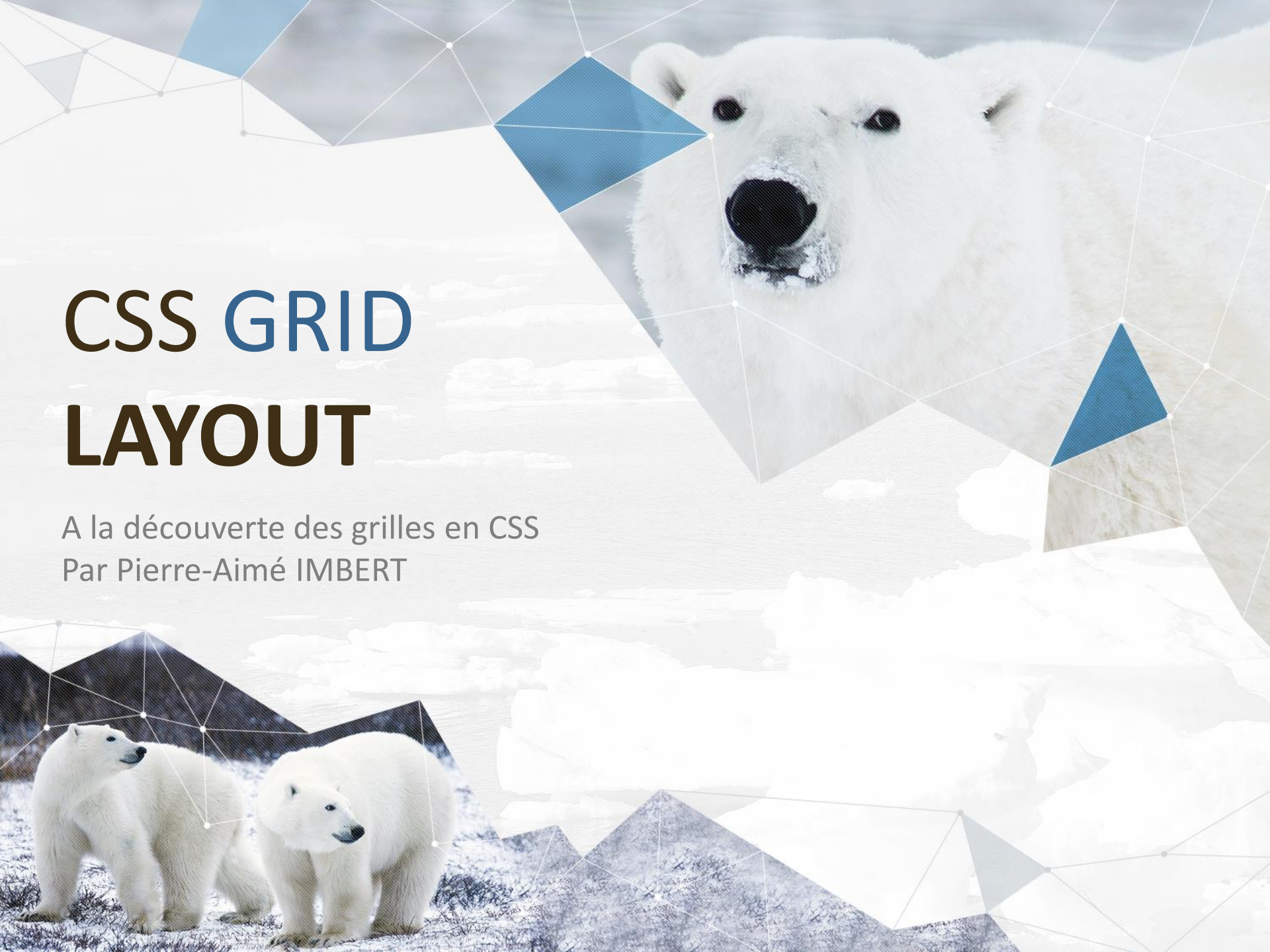


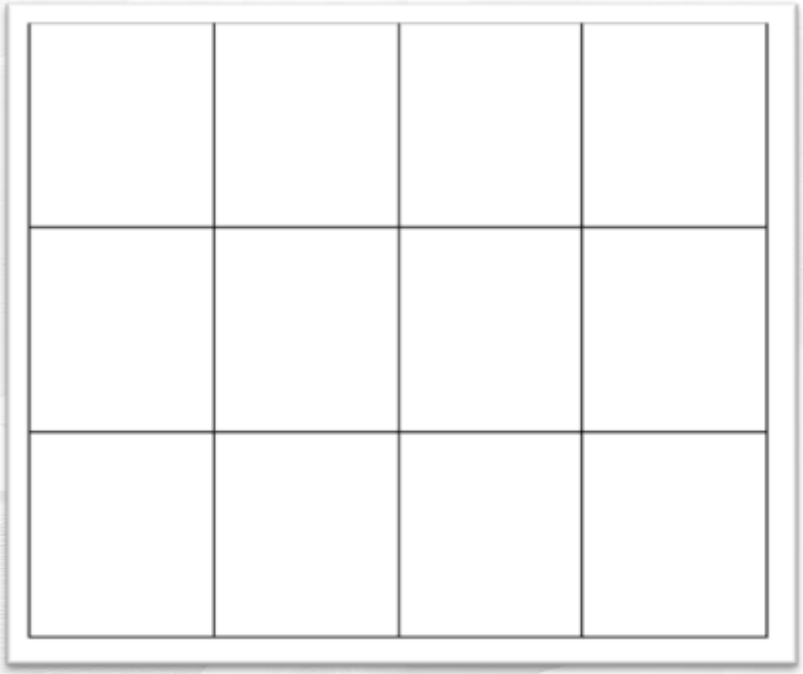
CSS GRID LAYOUT

A la découverte des grilles en CSS

Par Pierre-Aimé IMBERT

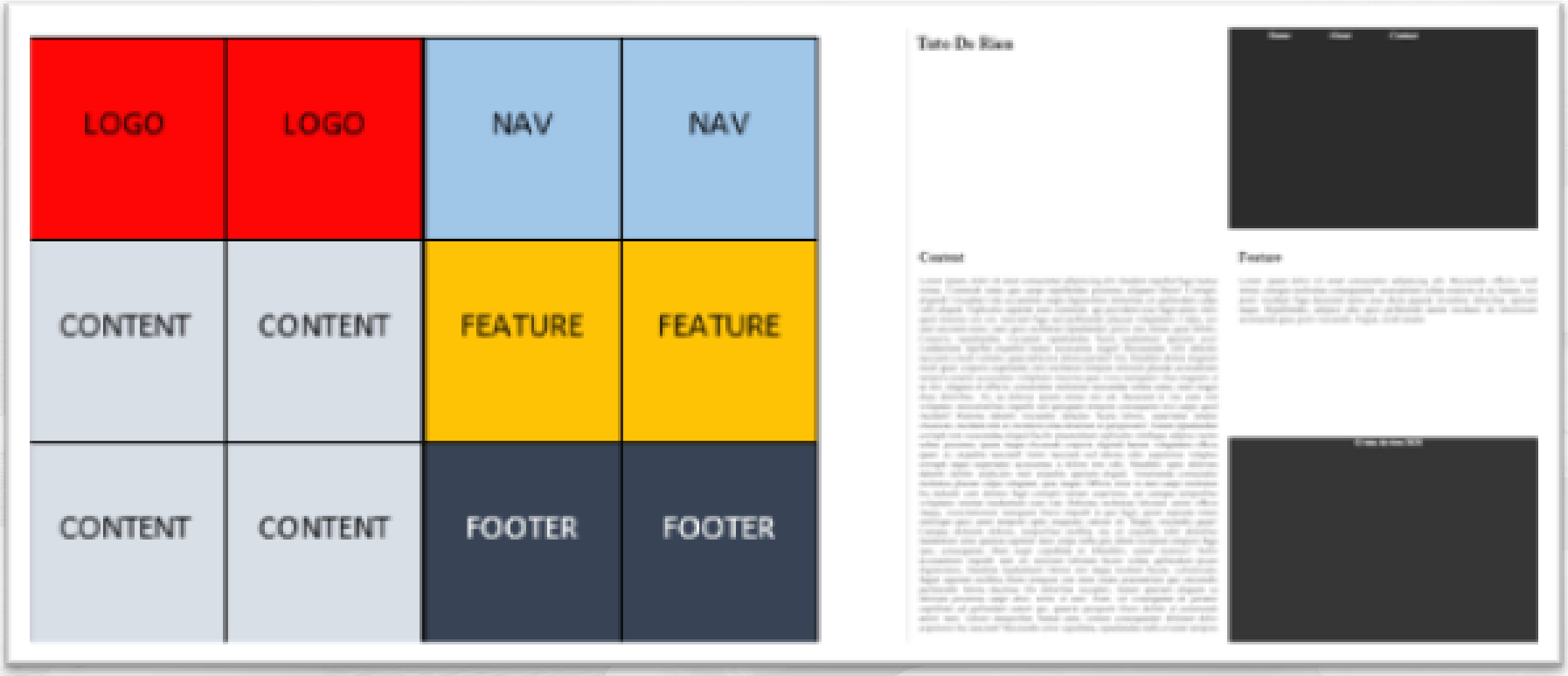
CSS GRID LAYOUT : INTRODUCTION

Nous allons voir ensemble **comment créer une mise en page** de notre site ou un **layout grâce au CSS Grid**. Imaginez désormais votre site sous la **forme d'une grille avec des lignes et des colonnes.**



CSS GRID LAYOUT : INTRODUCTION

Dans chaque emplacement de notre grille, nous allons placer **un élément** et ainsi concevoir notre layout. Nous allons voir ensemble comment cela fonctionne !



CSS GRID LAYOUT : LA PREPARATION

Nous allons comment par créer un page html avec les éléments suivant dans le body qui vont représenter notre site.

```
<body>
    <div class="container">

        <div class="logo">Logo</div>
        <div class="navigation">Navigation</div>
        <div class="content">Content</div>
        <div class="feature">feature</div>
        <div class="footer">Footer</div>
    </div>
</body>
```

Ici, rien de vraiment spécial : un bloc qui va contenir un logo, une navigation, du contenu, une sidebar (barre latérale) et un footer (bas de page).

CSS GRID LAYOUT : LA PREPARATION

```css
body{
  margin: 0;
  padding: 0;
}

.container{
  height: 100vh;
}

.logo{
  background-color: rgb(243, 11, 11);
}

.navigation{
  background-color: rgb(199, 121, 182);
}

.content{
  background-color: rgb(211, 211, 211);
}

.feature{
  background-color: rgb(236, 219, 60);
}

.footer{
  background-color: rgb(68, 61, 61);
}
```

On aura aussi un fichier CSS qui va relier à notre page html les différents éléments.
Cela nous donnera cette jolie page !

Jusque-là rien de spécial : nous avons vu comment créer un site avec le CSS et le HTML avec cette méthode où nous avons donné une taille à chaque élément. **Mais avec cela, nous sommes limités par l'emplacement des éléments car elles s'affichent dans l'ordre qu'elles se trouvent dans le fichier HTML.**

CSS GRID LAYOUT : AJOUTER LE GRID

Désormais, nous allons **ajouter un grillage** à notre site pour notre bloc avec le code suivant.

```
.container{
  height: 100vh;
  display: grid;
}
```

Et là, on peut déjà voir quelque chose de spéciale sur notre page : nos éléments prennent **toute la place sur le site et la hauteur est partagée entre chaque élément.**

CSS GRID LAYOUT : AJOUTER LE GRID

Désormais que cela est fait, nous allons définir **combien de colonnes** nous voulons dans notre grille. Nous pouvons le faire **avec des fractions ou fr** et le paramètre **grid-template-columns**

```css
.container{
  height: 100vh;
  display: grid;
  grid-template-columns: 1fr 1fr 1fr 1fr;
}
```

```css
.container{
  height: 100vh;
  display: grid;
  grid-template-columns: repeat(4, 1fr)
}
```

CSS GRID LAYOUT : AJOUTER LE GRID

Voici le rendu avec les colonnes ajoutées :

CSS GRID LAYOUT : AJOUTER LE GRID

Ajoutons **des lignes** de la même façon avec le paramètre **grid-template-rows :**

```
.container{
  height: 100vh;
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-template-rows: repeat(3,1fr);
}
```

Ici, nous avons 3 lignes et la dernière est vide car nous n'avons pas 9 éléments.

Mais nous allons arranger cela !

CSS GRID LAYOUT : PLACER NOS ELEMENTS

Pour l'instant, un élément est placé dans un emplacement de notre **grille mais nous pouvons définir où notre élément sera placé mais aussi dans plusieurs emplacements.**

Mais avant de pouvoir faire cela, nous devons créer le modèle de la grille qui va nous permettre de dessiner notre site avec les noms d'un élément. Il y a d'autres solutions mais celle-ci est pour moi la plus facile.

Nous allons commencer par **nommer nos éléments dans un grid area** comme suite.

```css
.logo{
  grid-area: logo;
  background-color: rgb(243, 11, 11);
}

.navigation{
  grid-area: nav;
  background-color: rgb(199, 121, 182);
}

.content{
  grid-area: content;
  background-color: rgb(211, 211, 211);
}

.feature{
  grid-area: feature;
  background-color: rgb(236, 219, 60);
}

.footer{
  grid-area: footer;
  background-color: rgb(68, 61, 61);
}
```

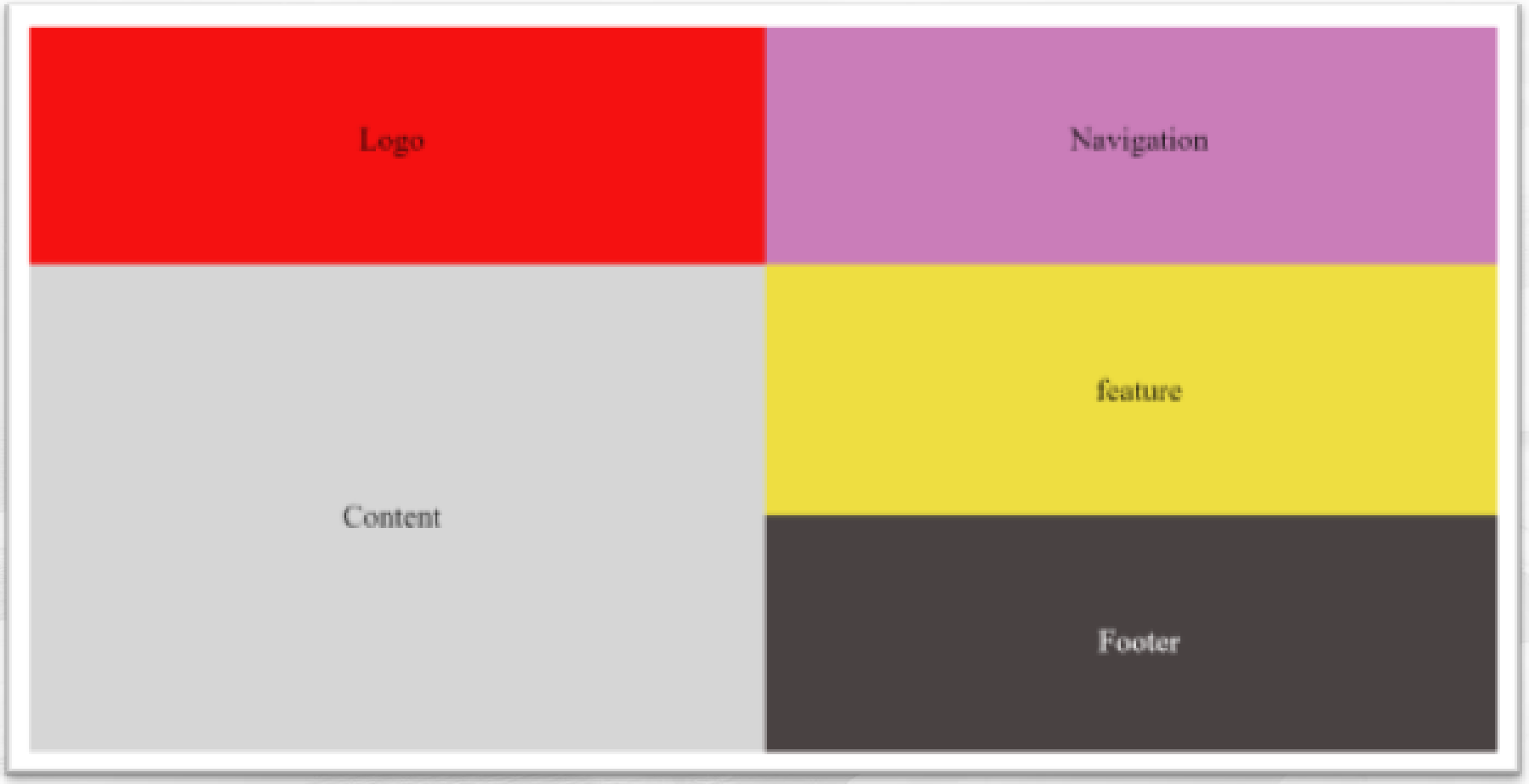
CSS GRID LAYOUT : PLACER NOS ELEMENTS

Puis, nous allons **utiliser ces noms** pour dessiner notre layout dans le bloc avec le paramètre **grid-template-areas** :

```
.container{
  height: 100vh;
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-template-rows: repeat(3,1fr);
  grid-template-areas:
  "logo logo nav nav"
  "content content feature feature"
  "content content footer footer"
  ;
}
```

CSS GRID LAYOUT : PLACER NOS ELEMENTS

Cela nous donne le résultat suivant :



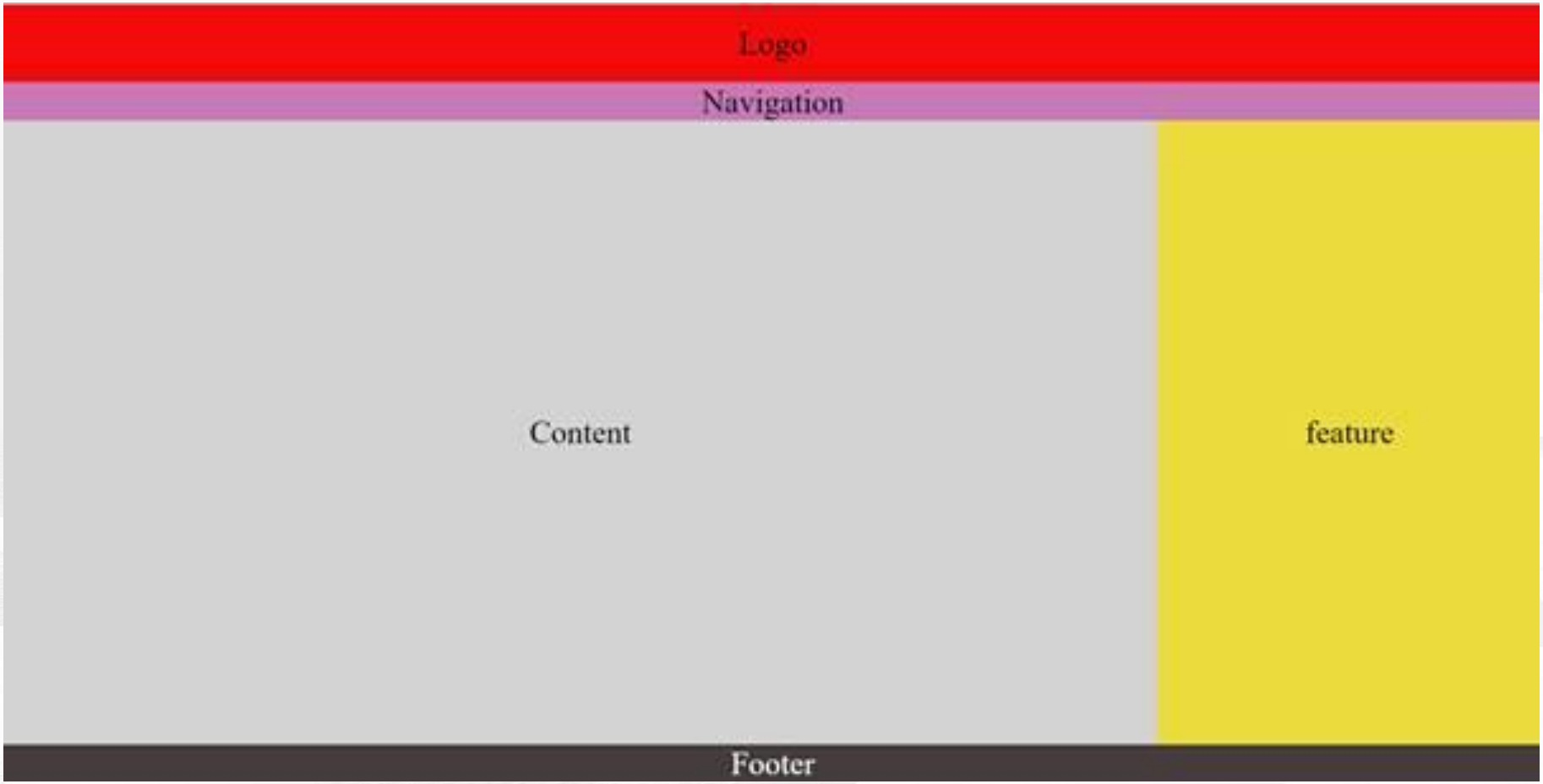
CSS GRID LAYOUT : EXEMPLE DE SITE STANDARD

Puis, nous allons **utiliser ces noms** pour dessiner notre layout dans le bloc avec le paramètre **grid-template-areas** :

```
.container{
  height: 100vh;
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-template-rows: 1fr 0.5fr 8fr 0.5fr;
  grid-template-areas:
  "logo logo logo logo"
  "nav nav nav nav"
  "feature content content content"
  "footer footer footer footer"
  ;
}
```

CSS GRID LAYOUT : EXEMPLE DE SITE STANDARD

Cela nous donne le résultat suivant :



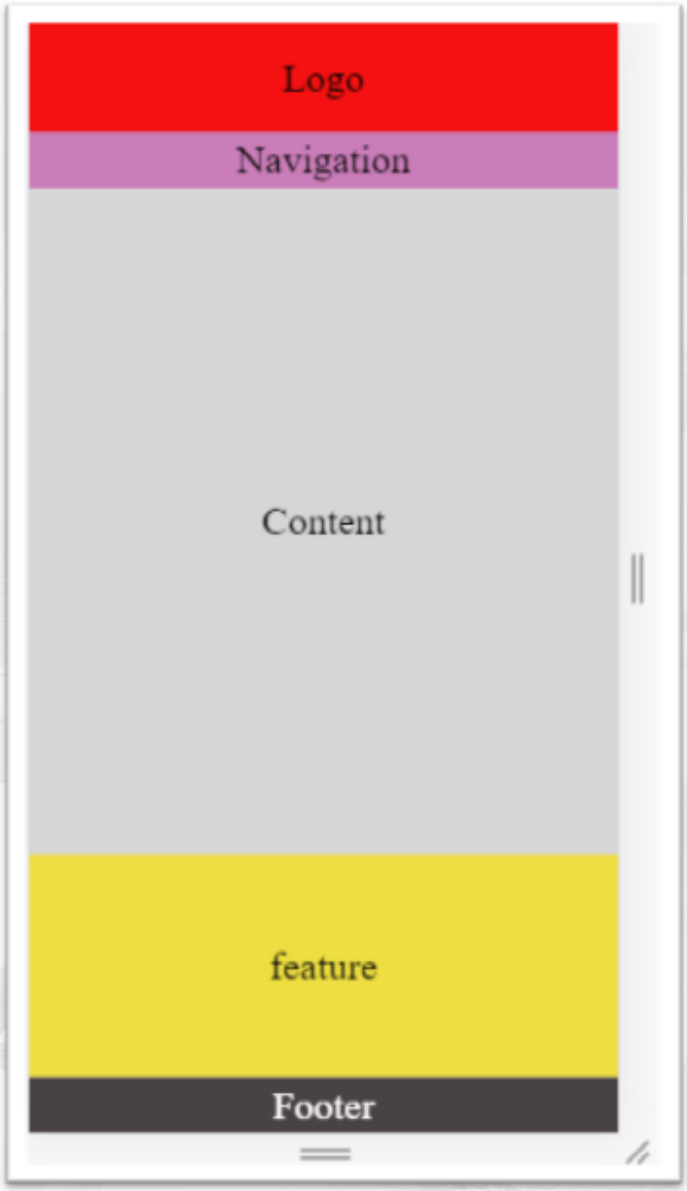
CSS GRID LAYOUT : EXEMPLE DE SITE MOBILE

Nous allons voir que nous pouvons créer une vue pour chaque type d'écran. Pour le moment notre site n'est pas mobile first car nous avons modifier notre layout mais nous allons créer une version mobile pour notre Template :

```css
@media screen and (max-width:700px){
    .container{
        grid-template-columns: 1fr;
        grid-template-rows: 1fr 0.5fr 6fr 2fr 0.5fr;
        grid-template-areas:
        "logo"
        "nav"
        "content"
        "feature"
        "footer";
    }
}
```

CSS GRID LAYOUT : EXEMPLE DE SITE MOBILE

Cela nous donne le résultat suivant :



MERCI DE VOTRE ECOUTE