6 : REDIMENSIONNER ET POSITIONNER LES ÉLÉMENTS

Explications :

En CSS on va également pouvoir redimensionner et positionner nos éléments assez facilement et surtout très précisement. Pour ça, je vous propose de commencer par un exemple avec width(la largeur) et height(la hauteur) :

```html
<html>
    <head>
        <meta charset="UTF-8">
        <title>Accueil</title>
        <link rel="stylesheet" type="text/css" href='css/style.css'>
    </head>
    <body>
        <div class="test"></div> <-- une div vide initiée
    </body>
</html>
```

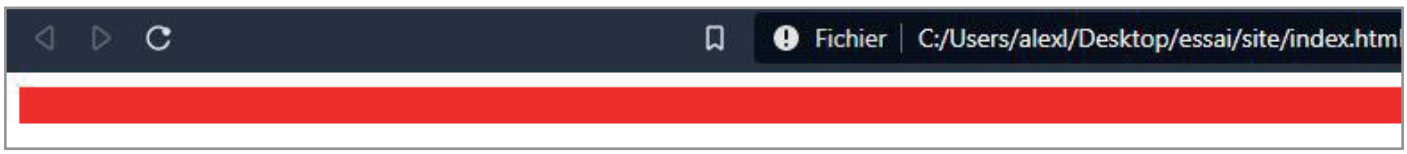
index.html

```
style.css

.test{
      background: red;
      width: 20px;
      height: 20px;
}
```

Résultat :

La div vide initiée à comme attribut une class=''test''. Dans notre fichier style.css, nous avons mis la couleur de fond sur rouge, la largeur à 20px et la hauteur à 20 px. Le résultat nous donne un carré rouge de 20x20 px car la div n'a pas de texte, elle est vide.

Si on viendrait à enlever la propriété height, la div disparaitrait, à l'inverse si on enlèverait le width, elle prendrait le plus de place possible vu que c'est une div :



Maintenant, lorsqu'on redimensionne un élément qui est un élément block par défaut (comme une div), on sait que deux block ne peuvent pas être l'un à côté de l'autre puisque un élément block essaie de prendre un maximum de largeur. Dans l'exemple on voit bien que la largeur est limité à 20 px, cet exemple va vous permettre de comprendre encore mieux le fonctionnement d'un block ou un inline-block. Nous allons faire une suite de cet exemple pour être plus explicite :

```html
<html>
    <head>
        <meta charset="UTF-8">
        <title>Accueil</title>
        <link rel="stylesheet" type="text/css" href='css/style.css'>
    </head>
    <body>
        <div class="test"></div>
        <div class="test" style="background: green"></div>
    </body>
</html>
```

index.html

Résultat avec le même style.css que dans l'exemple précédent :

Bien que la taille de la largeur(width) est définie à 20px dans style.css, les deux carré ne se mette pas côte à côte car ils restent des blocks et se mettent l'un en dessous de l'autre. Si on mettait un display:inline-block; les carrés se mettraient côte à côte avec une petite marge comme vu précédemment :

On voit bien que nos deux carrés apparaissent l'un à côté de l'autre. Cette exemple est à part car il n'a aucun sens dans une div mais c'était dans le but de ne pas confondre la largeur/hauteur d'un élément, et, son type display qui reste prédominant.
Nous allons maintenant voir la propriété **position** en instanciant beaucoup de carrés avec une classe (à noter que les noms de classe ne vont pas influer sur la couleur du carré, c'est juste pour mieux repérer le carré sur lequel on veut faire un traitement) :

```html
<html>
    <head>
        <meta charset="UTF-8">
        <title>Accueil</title>
        <link rel="stylesheet" type="text/css" href='css/style.css'>
    </head>
    <body>

        <div class="red"></div>
        <div class="green"></div>
        <div class="yellow"></div>
        <div class="black"></div>
        <div class="blue"></div>
        <div class="pink"></div>
        <div class="purple"></div>
```

index.html

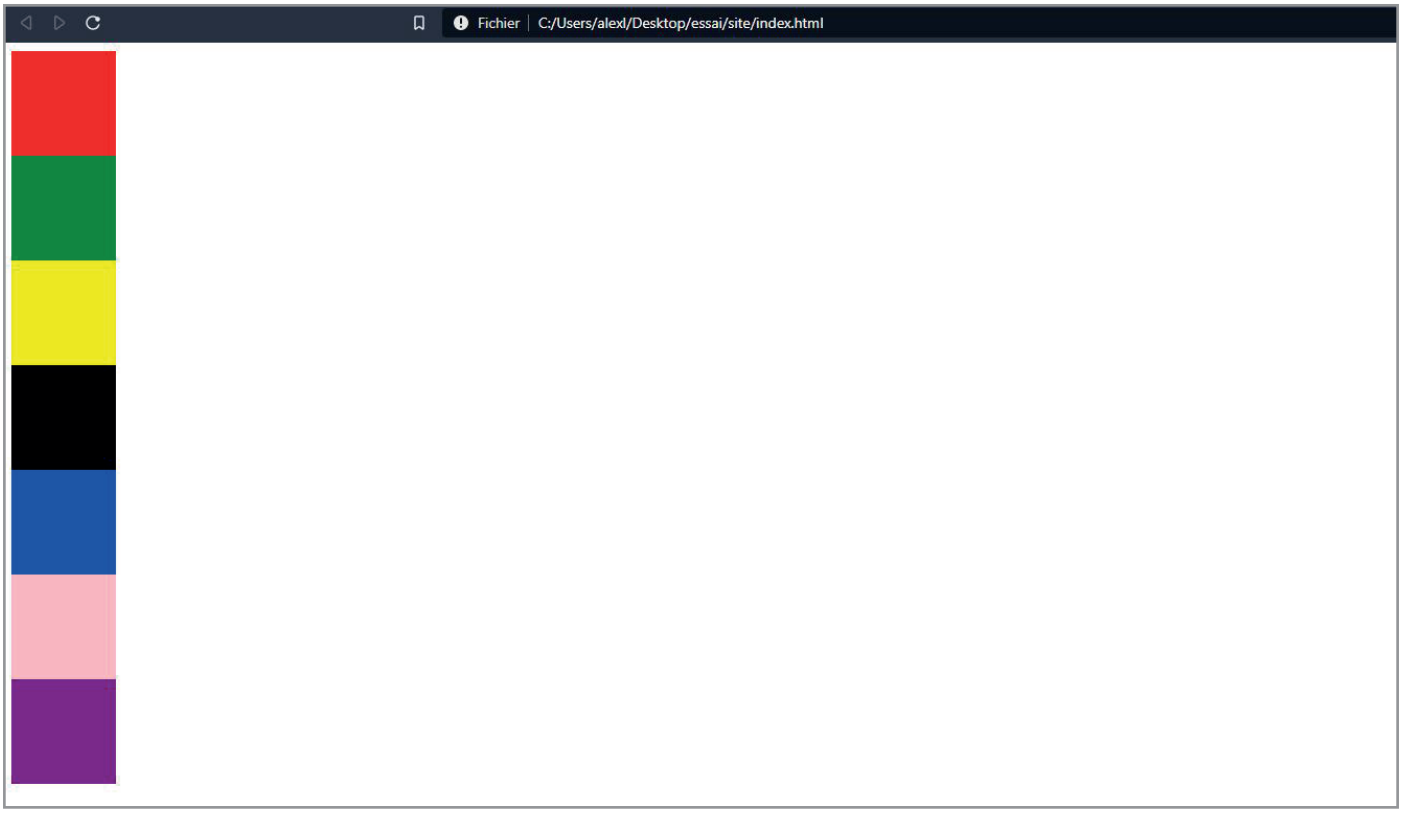
```css
div{
    width: 100px;
    height: 100px;
}

.red{
    background: red;
    position : static;
}
.green{
    background: green;
}
.yellow{
    background: yellow;
}
.black{
    background: black;
}
.blue{
    background: blue;
}
.pink{
    background: pink;
}
.purple{
    background: purple;
}
```

initiation des carrés en 100x100 px

position static est la position par défaut que nous ne sommes donc pas obligé d'écrire et qui ne va rien changer sur notre capture d'écran de résultat

style.css

Résultat :

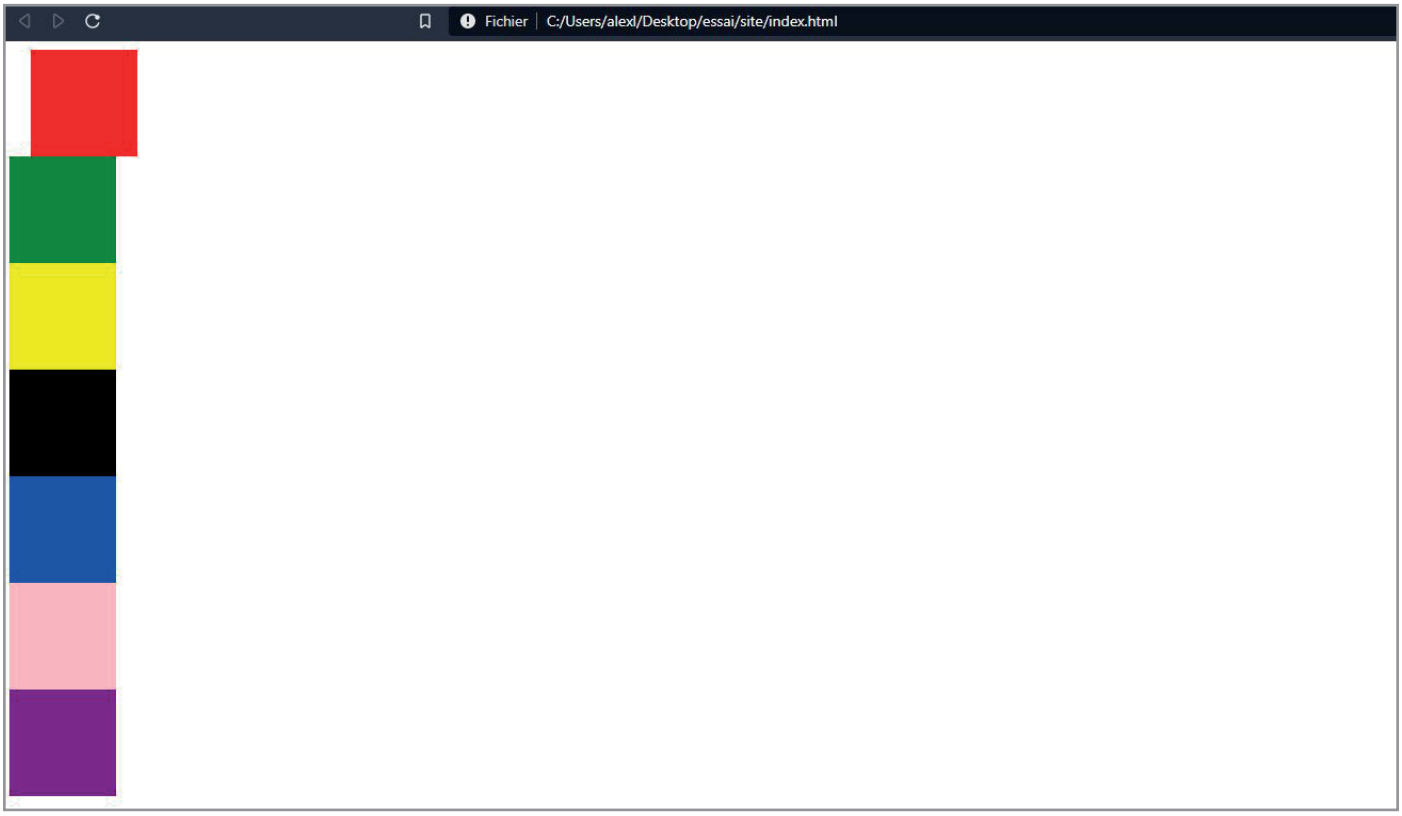


Voici le résultat en mettant la position sur static. Nous allons maintenant mettre la position en relative pour pouvoir modifier la position.

```css
.red{
    background: red;
    position : relative;
    left: 20px;
}
```

style.css

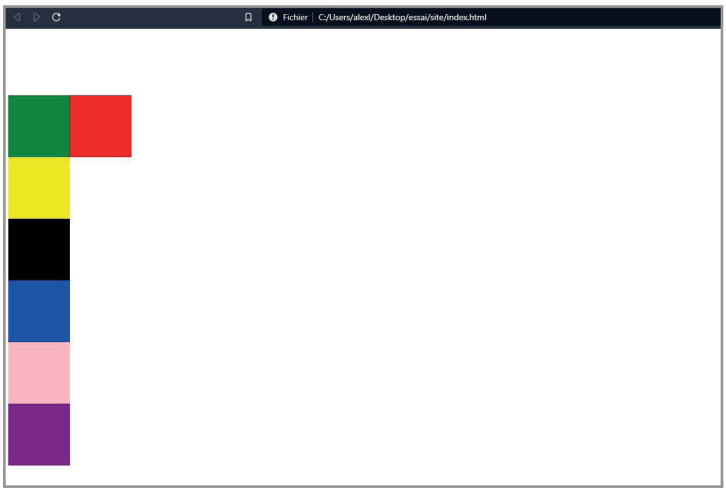
Résultat :



On voit bien qu'en travaillant seulement dans la class red, le carré à bougé de 20px vers la droite.
Attention, dans le code il est écrit left car on pousse le carré de 20px en partant de la gauche.
Si on mettrait right: -20px; on aurait exactement le même résultat.
Bien sur en plus de left et right on va pouvoir gérer le haut(top) et le bas(bottom) :

```
style.css

.red{
      background: red;
      position : relative;
      left: 100px;
      top: 100px;
}
```

Résultat :

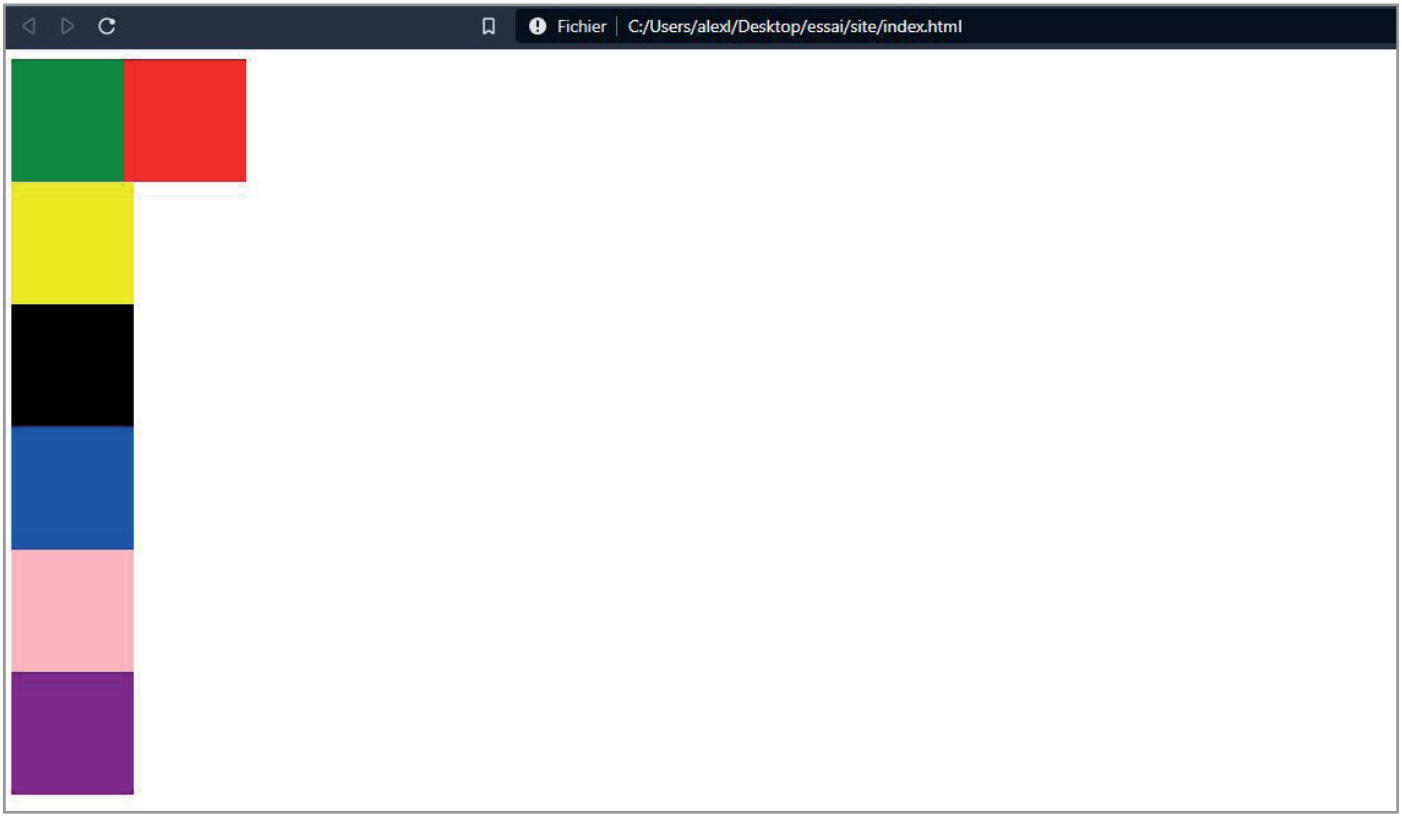


On peut voir qu'avec les attributs du dessus, le carré rouge a été poussé de 100 px vers la droite et de 100 px vers le bas et se retrouve maintenant à côté du carré vert. Par contre, l'espace crée par le carré rouge au dessus du carré vert est toujours là.
Voilà comment on peut jouer sur les positions mais il existe bien sur d'autres façons de le faire.

À defaut de repositionner les éléments par leur position relative, on va pouvoir les positionner avec leurs positions **absolu.**
La position absolu va nous permettre de modifier la position d'un élément par rapport au plus grand parent dont la position est relative. Dans notre cas, le body est le parent le plus grand qui a une position relative.

```
style.css

.red{
      background: red;
      position : absolute;
      left: 100px;
}
```

Résultat :



On voit maintenant que le carré rouge a été décalé comme tout à l'heure mais sans conserver sa place occupé (le blanc en haut sur la capture d'écran précédente) par sa position initiale. Si on change **position:absolute;** par **position:relative;** on pourra voir que sa place initiale est conservé comme sur l'exemple précédent :

Une dernière position qui existe est la **position:fixed;** que nous allons voir maintenant.

Je ne pourrais pas vous la montrer en capture d'écran mais la position:fixed; est une position qui ne va pas être impacté par le scroll et qui reste fixe. Si par exemple j'initie un élément carré orange avec comme attribut :

style.css

```css
.orange{
    background: red;
    position : fixed;
    left: 0px;
    top: 0px;
}
```

Et bien le carré orange va rester en haut à gauche de ma page tout le temps même quand je vais scroller dans ma page.

Une dernière propriété intéressante pourrait être le z-index: ; qui va permettre de donner un ordre d'importance aux éléments à afficher un peu comme dans les logiciels d'infographie avec les calques. On peut l'initier à n'importe quelle valeur tant que l'élément que nous voulons en dessous de l'autre à une valeur inférieure à l'élément qu'on veut au-dessus.
Prenons un exemple :

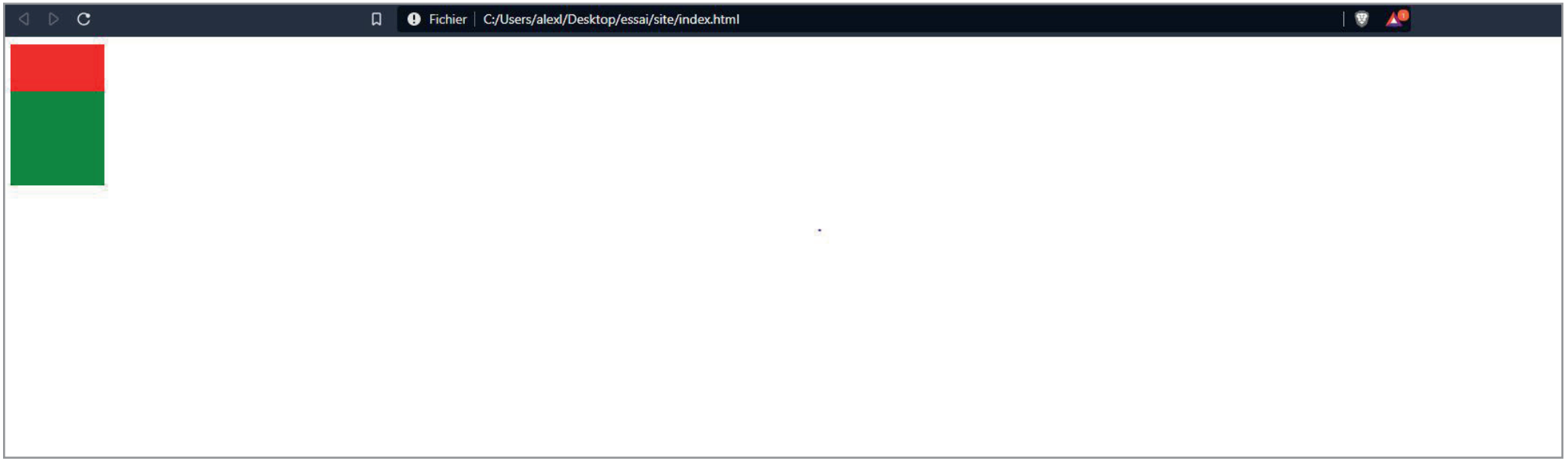
index.html

```html
<html>
    <head>
        <meta charset="UTF-8">
        <title>Accueil</title>
        <link rel="stylesheet" type="text/css" href='css/style.css'>
    </head>
    <body>

        <div class="red"></div>
        <div class="green"></div>

    </body>
</html>
```

style.css

```css
div{
    width: 100px;
    height: 100px;
}

.red{
    background: red;
    position: relative;
    z-index:0;
}
.green{
    background: green;
    position:relative;
    top: -50px;
    z-index:10;
}
```

Résultat :



On voit bien que le carré vert vient au dessus du carré rouge car le carré vert à un z-index plus élevéque le carré rouge. Si on inverse le z-index des deux :