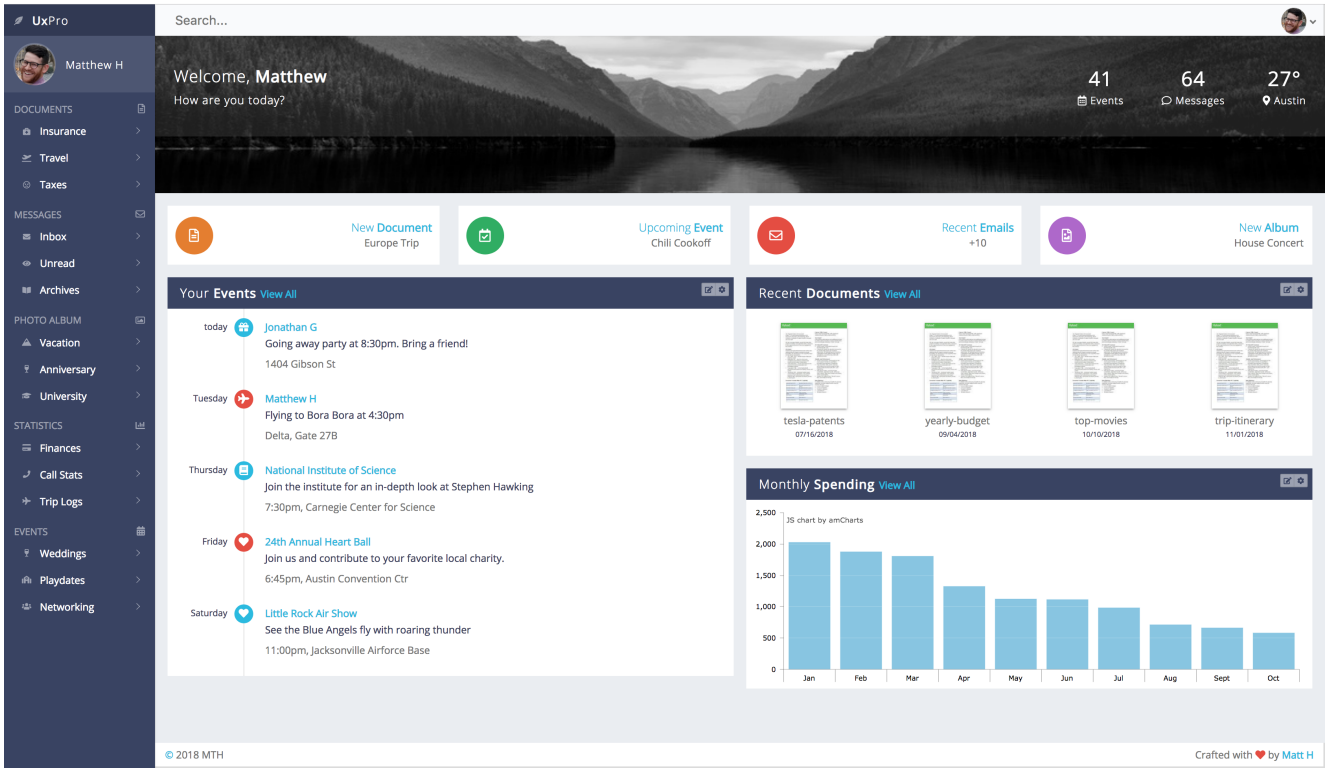


Avez-vous tous été mystifiés par ces magnifiques démos de tableaux de bord en ligne qui vous font vous demander, "Est-ce que je pourrai un jour construire ce... ?" Eh bien, je suis ici pour vous dire que vous pouvez le construire !

Comment s'y prendre ?

Nous allons d'abord introduire quelques bases de la grille CSS. Nous les utiliserons ensuite pour construire notre tableau de bord de base. Ensuite, nous explorerons la configuration et la structure de nos blocs de contenu internes, y compris certains flexbox. Enfin, nous parlerons brièvement de la réactivité mobile responsable, tout en intégrant une fonctionnalité de glissement adaptée aux mobiles dans notre navigation latérale. Qu'allez-vous faire ? Un tableau de bord dynamique qui fera en sorte que les gens vous apprécieront beaucoup.

Note : Nous allons construire une version simplifiée du tableau de bord ci-dessus, qui utilise les mêmes concepts que la version complète. La version complète rendrait ce morceau ridiculement long...
Tout d'abord, notre grille de base

Je vais vous décomposer la grille CSS, courte et douce. Nous avons besoin d'un conteneur de grille principal, et ensuite nous avons besoin d'un div (ou élément sémantique) pour chaque élément à l'intérieur du conteneur de grille :

```
// Simple dashboard grid layout
<div class="grid-container">
  <header class="header"></header>
  <aside class="sidenav"></aside>
  <main class="main"></main>
  <footer class="footer"></footer>
</div>
```

Une structure assez simple, non ? Notre mise en page aidera à produire cette toile immédiatement belle ci-dessous. Ne vous inquiétez pas, nous allons ajouter plus de contenu.

Ensuite, établissons la loi...je veux dire CSS

Votre conteneur principal doit être défini comme affichage : grille ; pour que toute fonctionnalité de grille fonctionne réellement. Nous lui donnons aussi une hauteur de 100% pour lui dire que nous voulons que notre tableau de bord remplisse toute la page. Et, pour chaque conteneur enfant, nous allons lui attribuer un nom afin de pouvoir dire à la grille ce qu'elle doit en faire. Ensuite, nous utiliserons les noms pour créer la structure de la page dans une déclaration de type tableur en utilisant des zones de modèle de grille :

```
/* Simple dashboard grid CSS */

/* Assign grid instructions to our parent grid container */
.grid-container {
  display: grid;
  grid-template-columns: 240px 1fr;
  grid-template-rows: 50px 1fr 50px;
  grid-template-areas:
    "sidenav header"
    "sidenav main"
    "sidenav footer";
  height: 100vh;
}

/* Give every child element its grid name */
.header {
  grid-area: header;
  background-color: #648ca6;
}

.sidenav {
  grid-area: sidenav;
  background-color: #394263;
}

.main {
  grid-area: main;
  background-color: #8fd4d9;
}

.footer {
  grid-area: footer;
  background-color: #648ca6;
}
```

Cette unité, fr, représente une fraction de l'espace disponible dans le conteneur de la grille. Le code suivant crée trois colonnes égales qui se redimensionnent en fonction de l'espace disponible

Explication des domaines du modèle de grille

Nous avons attribué un nom à chacun de nos contenants pour enfants, puis nous les avons placés dans un format de type tableur par le biais de zones de grille. C'est très simple.

Nous avons un total de deux colonnes de gauche à droite. La première colonne est de 250px de large (le nav latéral), et la seconde est de 1fr, ou fraction. Cela signifie qu'elle occupera l'espace restant du conteneur après que la première colonne ait été dessinée.

Ensuite, nous déclarons trois lignes au total. Les lignes vont de haut en bas. Ainsi, en commençant par le haut, nous avons un élément <header> qui est de 50px de haut. Ensuite, nous déclarons la zone de contenu <main>, qui a une hauteur de 1fr. Cela signifie qu'elle s'étirera verticalement pour remplir l'espace restant de la fenêtre après que les hauteurs explicitement déclarées aient été dessinées. Enfin, nous déclarons notre <pied>, également à une hauteur de 50px.

Ajout de l'en-tête et du pied de page

Le <header> et le<footer> seront des conteneurs flex, avec un espacement et un alignement flex :

```
// Create the styles and structure for our header and footer elements; grid-area declared
previously
<style>
  .header, .footer {
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 0 16px;
    background-color: #648ca6;
  }
</style>

<header class="header">
  <div class="header__search">Search...</div>
  <div class="header__avatar">Your face</div>
</header>

<footer class="footer">
  <div class="footer__copyright">&amp;copy; 2018 MTH</div>
  <div class="footer__signature">Made with love by pure genius</div>
</footer>
```

Nous utilisons justify-content: space-between pour étaler le premier et le dernier élément de façon à ce qu'ils s'étendent aux deux extrémités de leur contenant. C'est tellement facile comparé aux instructions de flottaison de la vieille école. Et, grâce à align-items: center; nous avons nos éléments parfaitement alignés sans avoir à compter sur le rembourrage, etc. pour le centrage.

Une brève note sur notre syntaxe de classe CSS bizarre : nous utilisons une CSS de style BEM (block-element-modifier), que je recommande pour l'évolutivité et la lisibilité. Il est recommandé d'éviter les sélecteurs de balises html brutes.
Ajout de l'élément nav latéral

Pour le contenu de la navigation latérale, nous utilisons les éléments traditionnels <ul> et <li>. Je recommande ceci plutôt que ou tout autre élément pour une bonne sémantique html et une bonne lisibilité humaine. Nous allons ajouter une fonctionnalité de glissement adaptée aux mobiles dans notre navigation latérale un peu plus bas :

Styles et structure pour notre contenu de navigation simplifié sur le côté du tableau de bord

HTML

```
<aside class="sidenav">
  <ul class="sidenav__list">
    <li class="sidenav__list-item">Item One</li>
    <li class="sidenav__list-item">Item Two</li>
    <li class="sidenav__list-item">Item Three</li>
    <li class="sidenav__list-item">Item Four</li>
    <li class="sidenav__list-item">Item Five</li>
  </ul>
</aside>
```

CSS

```
sidenav {
    display: flex; /* Will be hidden on mobile */
    flex-direction: column;
    grid-area: sidenav;
    background-color: #394263;
  }
  .sidenav__list {
    padding: 0;
    margin-top: 85px;
    list-style-type: none;
  }
  .sidenav__list-item {
    padding: 20px 20px 20px 40px;
    color: #ddd;
  }
  .sidenav__list-item:hover {
    background-color: rgba(255, 255, 255, 0.2);
    cursor: pointer;
  }
```

Notre structure de base de la navigation latérale du tableau de bord

Ajout du premier élément de la section <main> Celui-ci est simple et direct. Un autre conteneur flex :

HTML

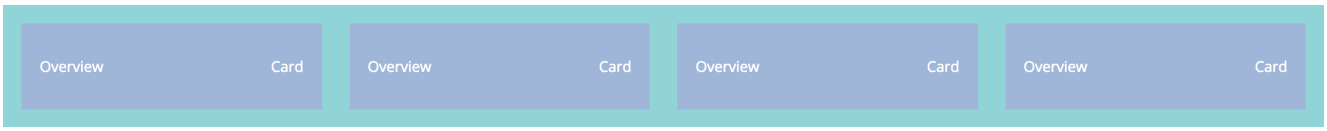
```
<div class="main-header">
  <div class="main-header__heading">Hello User</div>
  <div class="main-header__updates">Recent Items</div>
</div>
```

Le tout premier élément à l'intérieur du conteneur <main> parent

CSS

```
.main-header {
   display: flex;
   justify-content: space-between;
   margin: 20px;
   padding: 20px;
   height: 150px; /* Force our height since we don't have actual content yet */
   background-color: #e3e4e6;
   color: slategray;
  }
```

Maintenant les choses deviennent intéressantes... les cartes d'introduction des grilles réactives



C'est l'une de mes sections préférées de notre tableau de bord, parce que nous avons l'occasion d'utiliser une solution de grille super efficace et élégante. Regardez à nouveau l'animation du tableau de bord fournie précédemment. Vous avez remarqué comment ces cartes grises se comportent lorsque l'écran change ? Nos cartes gardent étonnamment une largeur de gouttière constante entre elles, elles ont un emballage constant, et quand nous ne pouvons pas remplir une rangée entière, notre carte emballée correspond exactement à la hauteur et à la largeur de la carte au-dessus, tout en restant alignée avec elle. C'est très difficile et fastidieux à réaliser sans la méthode que je vais vous montrer :

Main section de grille responsive

html

```
<div class="main-overview">
  <div class="overviewcard">
    <div class="overviewcard__icon">Overview</div>
    <div class="overviewcard__info">Card</div>
  </div>
  <div class="overviewcard">
    <div class="overviewcard__icon">Overview</div>
    <div class="overviewcard__info">Card</div>
  </div>
  <div class="overviewcard">
    <div class="overviewcard__icon">Overview</div>
    <div class="overviewcard__info">Card</div>
  </div>
  <div class="overviewcard">
    <div class="overviewcard__icon">Overview</div>
    <div class="overviewcard__info">Card</div>
  </div>
</div>
```

CSS

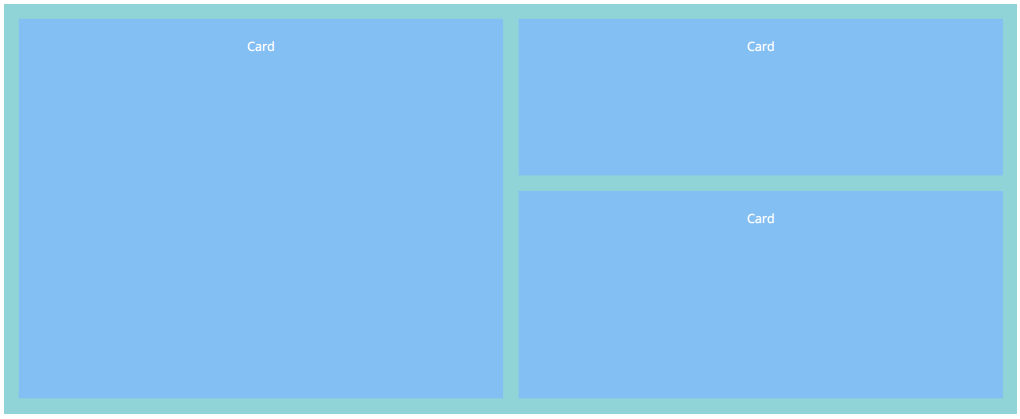
```
.main-overview {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(265px, 1fr)); /*
Where the magic happens */
    grid-auto-rows: 94px;
    grid-gap: 20px;
    margin: 20px;
  }

  .overviewcard {
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 20px;
    background-color: #d3d3;
```

C'est bien comme ça que nous avons utilisé un conteneur à grille pour ceux-ci, qui se trouve à l'intérieur de notre conteneur à grille de la page principale, n'est-ce pas ? Nous avons fait cela parce que c'est la solution la plus simple et la plus élégante à utiliser pour la réactivité nécessaire des éléments de la carte. Notre instruction repeat(auto-fit, minmax(265px, 1fr) prend en charge quelques obstacles majeurs :

1 -Si la largeur des cartes est inférieure à 265px, elles s'enrouleront sur une autre rangée.
2 -Si les cartes dépassent 265px de largeur, elles s'étireront pour occuper la largeur restante du conteneur.
3 -Lorsque les cartes s'enroulent sur une nouvelle rangée (auto-fit), elles s'alignent de gauche à droite avec les cartes au-dessus d'elles, en fonction de leur largeur ! Et vous bénéficiez également d'une réactivité intégrée sans aucune requête de média !

L'utilisation de la méthode de répétition des colonnes est également un moyen fantastique de construire de belles galeries d'images réactives, même avec des images de tailles différentes. Vous avez même accès à des algorithmes d'empaquetage dynamique avec l'instruction grid-auto-flow : dense ;. Cela permet d'éviter les espaces de lignes vides dus à des hauteurs d'images différentes. Cependant, pour que cette méthode fonctionne, il faut qu'elles soient mises à l'échelle les unes par rapport aux autres en unités fr relatives, c'est pourquoi nous ne l'utiliserons pas pour notre

Ajout des conteneurs de contenu principal



Cette section a également une tournure intéressante. Ces cartes contiendront les principaux éléments de votre tableau de bord, et elles varieront en hauteur les unes par rapport aux autres en raison de leur contenu dynamique. Sur la plupart des interfaces typiques, vous voudriez que les cartes qui se trouvent sur la même ligne aient la même hauteur et la même largeur. Et vous y parviendriez en assignant à chaque carte une valeur de flex : 1 ; de sorte qu'elles s'agrandissent pour correspondre à la carte la plus haute.

Cependant, dans notre cas, nous ne voulons pas forcer ces cartes à correspondre à la hauteur de l'autre parce que le sujet de leur contenu variera. Pour les faire circuler naturellement sur deux colonnes, nous utiliserons une propriété CSS spéciale pour cela, column-count :

HTML

```
<div class="main-cards">
  <div class="card">Card</div>
  <div class="card">Card</div>
  <div class="card">Card</div>
</div>
```

CSS

/*Construire le contenu de la section Main*/

```
  .main-cards {
    column-count: 2;
    column-gap: 20px;
    margin: 20px;
  }

  .card {
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 100%;
    background-color: #82bef6;
    margin-bottom: 20px;
    -webkit-column-break-inside: avoid;
    padding: 24px;
    box-sizing: border-box;
  }
```

/*Forcer des hauteurs variables pour simuler un contenu dynamique*/

```
.card:first-child {
    height: 485px;
  }
  .card:nth-child(2) {
    height: 200px;
  }
  .card:nth-child(3) {
    height: 265px;
  }
</style>

<div class="main-cards">
  <div class="card">Card</div>
  <div class="card">Card</div>
  <div class="card">Card</div>
</div>
```

L'utilisation du comptage par colonne permet de s'assurer que notre contenu dans la section des cartes principales est divisé en deux colonnes. Nous appliquons également un écart entre les cartes avec column-gap. C'est très similaire à nos cartes de vue d'ensemble, où nous avons utilisé le quadrillage
(grid-gap).
La raison pour laquelle nous n'avons pas utilisé display:grid ; pour cette section, c'est que nos hauteurs pour chaque carte sont dynamiques. Nous voulons qu'elles s'écoulent naturellement dans deux colonnes, tout en observant leurs hauteurs variables. Cette méthode nous évite également d'avoir à utiliser une grille flottante traditionnelle, qui nous ferait calculer des pourcentages de largeur, des gouttières et des règles de marge spéciales pour le premier et le dernier élément enfant.
Nous avons également utilisé la méthode column-break-inside : avoid ; pour s'assurer que chaque carte n'obtienne pas son contenu split.column-count va en fait briser le contenu de chaque élément pour faire en sorte que les deux lignes aient la même hauteur, ce que nous ne voulons pas.
Une brève note sur la réactivité mobile responsable
Regardez à nouveau la vidéo de réponse en haut de la pièce. Voyez comment la section des cartes principales se transforme en une colonne lorsque vous vous approchez des écrans de la taille d'une tablette ? Et voyez comment la navigation latérale disparaît sur les écrans de taille mobile ? Pour cette réactivité, nous devrions vraiment écrire notre CSS dans un format mobile d'abord.
Cela signifie que notre CSS initial devrait être adapté aux mobiles. Ensuite, à mesure que la taille de notre écran augmente, nous observons des styles d'écran plus grands en utilisant des requêtes médias graduées en largeur minimale. C'est une meilleure pratique que de remplacer les styles de bureau par des requêtes média de largeur maximale, car cette approche peut entraîner certains maux de tête :

/*Rendre notre tableau de bord réactif dans un format mobile */

CSS

```
.grid-container {
  display: grid;
  grid-template-columns: 1fr; /* La navigation latérale est cachée sur le mobile*/
  grid-template-rows: 50px 1fr 50px;
  grid-template-areas:
    'header'
    'main'
    'footer';
  height: 100vh;
}

.sidenav {
  display: none;
  grid-area: sidenav;
  background-color: #394263;
}

.main-cards {
  column-count: 1;
  column-gap: 20px;
  margin: 20px;
}

/* Styles non mobiles,, 750px breakpoint */

@media only screen and (min-width: 46.875em) {
  /* Show the sidenav */
  .grid-container {
    grid-template-columns: 240px 1fr; /* Afficher la navigation latérale pour les écrans non mobiles */
    grid-template-areas:
      "sidenav header"
      "sidenav main"
      "sidenav footer";
  }
  .sidenav {
    display: flex;
    flex-direction: column;
  }
}
/* Medium-sized screen breakpoint (tablet, 1050px) */
@media only screen and (min-width: 65.625em) {
  /* Break out main cards into two columns */
  .main-cards {
    column-count: 2;
  }
}
```

Rendre notre navigation latérale coulissante sur les appareils mobiles

Notre travail n'est pas vraiment terminé tant que nous n'aurons pas rendu la navigation latérale utilisable sur les appareils mobiles. Nous devons le faire :

- **Ajouter notre icône de menu et l'icône de fermeture.**
- **Ajouter quelques transitions réactives pour l'action de glissement.**
- **Ecrire du JavasSript pour que nos clics activent le nav latéral.**

Nous utiliserons la bibliothèque Font Awesome pour nos icônes, et apporterons jQuery pour une manipulation facile du DOM (Voir le codepen pour référence).

Icône du menu principal du tableau de bord et icône de fermeture de la navigation latérale

```
<div class="grid-container">
  <div class="menu-icon">
    <i class="fas fa-bars"></i>
  </div>
</div>

<aside class="sidenav">
  <div class="sidenav__close-icon">
    <i class="fas fa-times"></i>
  </div>
</aside>
```

Ensuite, nous allons mettre à jour notre CSS pour inclure les nouvelles icônes et pour donner à notre navigation latérale quelques transitions glissantes. Encore une fois, nous utiliserons des requêtes média graduelles (media queries) :

```
/* Hamburger menu icon, stays fixed on mobile for any possible scrolling */
.menu-icon {
  position: fixed;
  display: flex;
  top: 5px;
  left: 10px;
  align-items: center;
  justify-content: center;
  border-radius: 50%;
  z-index: 1;
  cursor: pointer;
  padding: 12px;
  background-color: #DADAE3;
}

/* Make room for the menu icon on mobile */
.header__search {
  margin-left: 42px;
}

/* Mobile-first side nav styles */
.sidenav {
  grid-area: sidenav;
  display: flex;
  flex-direction: column;
  height: 100%;
  width: 240px;
  position: fixed;
  overflow-y: auto;
  box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.16), 0 0 0 1px rgba(0, 0, 0, 0.08);
  z-index: 2; /* Needs to sit above the hamburger menu icon */
  background-color: #394263;
  transform: translateX(-245px);
  transition: all .6s ease-in-out;
}

/* The active class is toggled on hamburger and close icon clicks */
.sidenav.active {
  transform: translateX(0);
}

/* Only visible on mobile screens */
.sidenav__close-icon {
  position: absolute;
  visibility: visible;
  top: 8px;
  right: 12px;
  cursor: pointer;
  font-size: 20px;
  color: #ddd;
}

/* Non-mobile styles for side nav responsiveness, 750px breakpoint */
@media only screen and (min-width: 46.875em) {
  .sidenav {
    position: relative; /* Fixed position on mobile */
    transform: translateX(0);
  }

  .sidenav__close-icon {
    visibility: hidden;
  }
```

Enfin, nous devons écrire du JavaScript pour que nos clics fonctionnent. La fonctionnalité de glissement est accomplie en basculant le nom de la classe .active, ce qui met à jour l'instruction transform : translateX() ;. N'oubliez pas d'ajouter le lien CDN jQuery avant votre balise de fin </body> :

```
<body>
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
 <script>
  const menuIconEl = $('.menu-icon');
  const sidenavEl = $('.sidenav');
  const sidenavCloseEl = $('.sidenav__close-icon');

  // Add and remove provided class names
  function toggleClassName(el, className) {
   if (el.hasClass(className)) {
    el.removeClass(className);
   } else {
    el.addClass(className);
   }
  }

  // Open the side nav on click
  menuIconEl.on('click', function() {
   toggleClassName(sidenavEl, 'active');
  });

  // Close the side nav on click
  sidenavCloseEl.on('click', function() {
   toggleClassName(sidenavEl, 'active');
  });
 </script>
</body>
```

Vous devriez maintenant avoir une navigation latérale entièrement réactive. Réduisez la taille de votre fenêtre jusqu'à ce qu'elle soit de taille mobile. Elle devrait se cacher, et l'icône du menu devrait apparaître. Cliquez dessus et la barre de navigation latérale devrait s'afficher. Cliquez sur l'icône de fermeture, et le menu latéral devrait se refermer. C'est plutôt bien, hein ? Et tu pensais que tu ne pouvais pas le faire...

En résumé...

Qu'est-ce que tu viens d'apprendre ?

- **Comment construire un tableau de bord en utilisant la grille CSS et Flexbox.**
- **Règles spéciales pour les blocs de contenu dynamique utilisant la répétition, auto-fit, min-max, column-count, grid-gap, et column-gap.**
- **La bonne façon d'écrire du CSS réactif et mobile en utilisant des requêtes média graduées de largeur minimale.**
- **Comment créer un navigateur latéral coulissant et réactif aux mobiles.**

Veuillez consulter le codepen complet du tableau de bord, qui comprend un navigateur latéral coulissant et cliquable avec des éléments de liste en accordéon, un joli menu utilisateur déroulant et une action de survol de transition.

J'espère que vous avez acquis de nouvelles connaissances grâce à ce morceau et que la disposition du tableau de bord ne vous mystifiera plus jamais. N'hésitez pas à me faire part de vos commentaires si vous avez des questions !

```
<div class="grid">
  <header
class="header">
   <i class="fas fa-bars
header__menu"></i>
   <div
class="header__search
">
    <input
class="header__input"
placeholder="Search...
" />
   </div>
   <div
class="header__avatar
">
    <div
class="dropdown">
     <ul
class="dropdown__list
">
      <li
class="dropdown__list
-item">
       <span
class="dropdown__ico
n"><i class="far fa-
user"></i></span>
       <span
class="dropdown__titl
e">my profile</span>
      </li>
      <li
class="dropdown__list
-item">
       <span
class="dropdown__ico
n"><i class="fas fa-
clipboard-list"></i></
span>
       <span
class="dropdown__titl
e">my account</span>
      </li>
      <li
class="dropdown__list
-item">
       <span
class="dropdown__ico
n"><i class="fas fa-
sign-out-alt"></i></
span>
       <span
class="dropdown__titl
e">log out</span>
      </li>
```