Horloges

Le temps est venu. Dans cet article, nous allons relever le défi de créer et animer une horloge en utilisant des animations CSS simples et un peu de JavaScript.

Voici l'horloge que nous allons créer avec quelques lignes de HTML, de CSS et de JavaScript et un background SVG. Nous utiliserons les animations ou les transitions CSS pour tous les mouvements et nous nous appuierons sur JavaScript pour déterminer l'heure initiale et pour ajouter quelques transformations CSS basiques.

HTML

Pour commencer, nous avons besoin de HTML

```
<article class="clock">
  <div class="hours-container">
    <div class="hours"></div>
  </div>
  <div class="minutes-container">
    <div class="minutes"></div>
  </div>
  <div class="seconds-container">
    <div class="seconds"></div>
  </div>
</article>
```

À l'origine, j'avais trois éléments représentant chacun une aiguille de l'horloge, mais je les ai finalement enveloppés chacun dans un élément conteneur. La première version fonctionnait avec les animations basiques, mais nous avons besoin des éléments conteneurs pour positionner les aiguilles et les animer.

L'horloge, sans les aiguilles

Nous commencerons avec une horloge très simple, sans design compliqué.

```
.clock {
  border-radius: 50%;
  background: #fff url(/images/posts/clocks/ios_clock.svg) no-repeat center;
  background-size: 88%;
  height: 20em;
  padding-bottom: 31%;
  position: relative;
  width: 20em;
}
```

```
.clock.simple:after {
  background: #000;
  border-radius: 50%;
  content: "";
  position: absolute;
  left: 50%;
  top: 50%;
  transform: translate(-50%, -50%);
  width: 5%;
  height: 5%;
  z-index: 10;
}
```

Vous pouvez récupérer le background SVG ici. J'ai également ajouté un pseudo-élément pour avoir un cercle noir solide au centre. Les aiguilles de l'horloge seront placées sous le pseudo-élément.

Nous devrions maintenant avoir quelque chose comme ceci.

Avant d'ajouter les aiguilles, nous devons placer les conteneurs.

```
.minutes-container, .hours-container, .seconds-container {
  position: absolute;
  top: 0;
```

```
  right: 0;
  bottom: 0;
  left: 0;
}
```

Chaque conteneur est ainsi empilé au-dessus de l'horloge. Passons maintenant aux aiguilles.

Aiguille des heures

Chaque aiguille est positionnée de manière absolue (`absolute`) et placée sur la position de midi. Commençons par l'aiguille des heures.

```
.hours {
  background: #000;
  height: 20%;
  left: 48.75%;
  position: absolute;
  top: 30%;
  transform-origin: 50% 100%;
  width: 2.5%;
}
```

J'utilise les pourcentages pour la fluidité, ça donne un peu plus de boulot mais c'est plus simple ensuite pour adapter l'image aux dimensions de l'écran. Nous réglons la propriété `transform-origin` sur le bas de l'aiguille pour la faire tourner à partir du point central.

Aiguille des minutes

Elle est assez similaire, mais plus fine et plus longue.

```
.minutes {
  background: #000;
  height: 40%;
  left: 49%;
  position: absolute;
  top: 10%;
  transform-origin: 50% 100%;
  width: 2%;
}
```

Aiguille des secondes

Elle aussi est plus fine, mais elle est également placée plus bas de façon à ce qu'une partie dépasse du centre. Il nous faut donc déplacer le point de `transform-origin` à 80%, ce qui laisse 20% de l'aiguille dépasser au-delà du point central.

```
.seconds {
  background: #000;
  height: 45%;
  left: 49.5%;
  position: absolute;
  top: 14%;
  transform-origin: 50% 80%;
  width: 1%;
  z-index: 8;
}
```

Animation

Une horloge arrêtée ne donnera l'heure exacte que deux fois par jour. Ajoutons un peu d'animation pour lui donner plus d'utilité.

Certaines horloges et certaines montres ont une aiguille qui se déplace de manière saccadée, de seconde en seconde, souvent avec un petit bruit. D'autres ont une aiguille qui avance en continu. Nous allons essayer les deux, en commençant par le mouvement continu.

Nous pouvons utiliser une `keyframe` indiquant aux aiguilles de tourner sur 360 degrés (à partir d'une position 0% sous-entendue).

```
@keyframes rotate {
  100% {
    transform: rotateZ(360deg);
  }
}
```

Avec cette keyframe, nous disons que l'élément doit s'animer sur 360 degrés lorsque nous lui appliquons la propriété `animation`. Nous utiliserons une fonction de timing `linear` pour faire tourner l'aiguille en continu sans à-coups.

```
.hours-container {
  animation: rotate 43200s infinite linear;
}
.minutes-container {
  animation: rotate 3600s infinite linear;
}
.seconds-container {
  animation: rotate 60s infinite linear;
}
```

L'aiguille des heures fait le tour du cadran en 12 heures, soit 43.200 secondes. Celle des minutes le fait en une heure (3.600 secondes) et celle des secondes en une minute.

Tout cela mis ensemble, nous avons maintenant du mouvement !

Si vous avez un oeil de lynx, vous arriverez peut-être même à discerner le mouvement de l'aiguille des minutes. Il faudra une heure à l'aiguille des minutes

pour faire un tour et douze heures à l'aiguille des heures.

L'aiguille des secondes ne met que 60 secondes à faire le tour du cadran, c'est plus facile à voir.

Ajouter un mouvement saccadé

Nous pouvons donner un comportement plus naturel à notre horloge en déplaçant l'aiguille des secondes en 60 mouvements séparés. Une façon simple d'y parvenir consiste à utiliser la fonction de timing `steps` . La propriété `animation` de chaque aiguille devient :

```
.minutes-container {
  animation: rotate 3600s infinite steps(60);
}
.seconds-container {
  animation: rotate 60s infinite steps(60);
}
```

L'aiguille des secondes et celle des minutes tournent maintenant en 60 "pas" dont le navigateur calcule automatiquement l'ampleur.

À la bonne heure !

Notre horloge a fière allure, mais elle nous plairait encore plus si elle était à l'heure.

Avec un peu de JavaScript nous pouvons donner l'heure exacte à nos visiteurs.

Voici le code :

```
/*
 * Starts any clocks using the user's local time
 * From: cssanimation.rocks/clocks
 */
function initLocalClocks() {
  // Get the local time using JS
  var date = new Date;
  var seconds = date.getSeconds();
  var minutes = date.getMinutes();
  var hours = date.getHours();

  // Create an object with each hand and it's angle in degrees
  var hands = [
    {
      hand: 'hours',
      angle: (hours * 30) + (minutes / 2)
    },
    {
      hand: 'minutes',
      angle: (minutes * 6)
    },
    {
      hand: 'seconds',
      angle: (seconds * 6)
    }
  ];
  // Loop through each of these hands to set their angle
  for (var j = 0; j < hands.length; j++) {
    var elements = document.querySelectorAll('.' + hands[j].hand);
    for (var k = 0; k < elements.length; k++) {
        elements[k].style.webkitTransform = 'rotateZ('+ hands[j].angle +'deg)';
        elements[k].style.transform = 'rotateZ('+ hands[j].angle +'deg)';
        // If this is a minute hand, note the seconds position (to calculate minute position late
        if (hands[j].hand === 'minutes') {
```

```
            elements[k].parentNode.setAttribute('data-second-angle', hands[j + 1].angle);
          }
        }
      }
    }
```

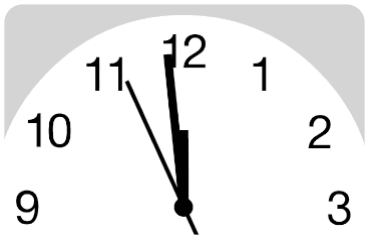
Cette fonction convertit l'heure (heures, minutes, secondes) en l'angle correspondant, pour chacune des aiguilles. Elle fait une boucle ("loop") sur chaque aiguille et applique cet angle en utilisant la propriété `style.transform` et `rotateZ` .

Ce code fonctionnera sur la plupart des navigateurs, sauf sur Safari ou d'autres qui ont besoin d'un préfixe. Nous devons donc également utiliser la propriété `style.webkitTransform` .

Et voilà, notre horloge est alignée avec l'heure du système local.

Aligner les aiguilles des minutes et des secondes

Nous devons nous assurer que l'aiguille des minutes se déplace au moment où l'aiguille des secondes atteint la position de midi.



Lorsque l'horloge est dessinée pour la première fois à l'écran, il reste moins d'une minute avant que l'aiguille des minutes ne doive se mouvoir. Pour gérer ce cas particulier, nous pouvons calculer le temps qui reste jusqu'à la prochaine minute et déplacer l'aiguille manuellement. Puisque nous utilisons JavaScript pour ce premier mouvement, nous pouvons continuer à la faire avancer de 6 degrés à chaque minute, en utilisant `setInterval` .

Avant de faire bouger l'aiguille des minutes, nous devons indiquer où nous en sommes dans la minute en cours. Vous avez sans doute remarqué ces lignes.

```
if (degrees[j].hand === 'minutes') {
  elements[k].parentNode.setAttribute('data-second-angle', degrees[j + 1].degree);
}
```

Ces lignes vérifient d'abord que l'aiguille est celle des minutes, et si c'est le cas, donnent au data attribute l'angle actuel de l'aiguille des secondes.

Une fois ce data attribute réglé, nous pouvons utiliser cette donnée pour déterminer à quel moment faire avancer l'aiguille des minutes.

```
/*
 * Set a timeout for the first minute hand movement (less than 1 minute), then rotate it every mi
 */
function setUpMinuteHands() {
  // Find out how far into the minute we are
```

```
  var containers = document.querySelectorAll('.minutes-container');
  var secondAngle = containers[0].getAttribute("data-second-angle");
  if (secondAngle > 0) {
    // Set a timeout until the end of the current minute, to move the hand
    var delay = (((360 - secondAngle) / 6) + 0.1) * 1000;
    setTimeout(function() {
      moveMinuteHands(containers);
    }, delay);
  }
}

/*
 * Do the first minute's rotation
 */
function moveMinuteHands(containers) {
  for (var i = 0; i < containers.length; i++) {
    containers[i].style.webkitTransform = 'rotateZ(6deg)';
    containers[i].style.transform = 'rotateZ(6deg)';
  }
  // Then continue with a 60 second interval
  setInterval(function() {
    for (var i = 0; i < containers.length; i++) {
      if (containers[i].angle === undefined) {
        containers[i].angle = 12;
      } else {
        containers[i].angle += 6;
      }
      containers[i].style.webkitTransform = 'rotateZ('+ containers[i].angle +'deg)';
      containers[i].style.transform = 'rotateZ('+ containers[i].angle +'deg)';
    }
  }, 60000);
}
```

Ajouter un rebond

Puisque nous utilisons JavaScript pour faire avancer l'aiguille des minutes, nous pouvons maintenant supprimer la propriété animation. Mais plutôt que de la supprimer, nous allons la remplacer par une transition, ce qui nous donne une bonne opportunité pour ajouter un peu plus de caractère à notre animation.

Lorsque JavaScript fixe un nouvel angle pour l'aiguille, une transition CSS sur l'élément indique au navigateur d'animer cette nouvelle position. Autrement dit, JavaScript ne va s'occuper que du changement d'angle, tandis que le navigateur s'occupera de l'animation.

Auparavant, nous devons mettre à jour le code afin d'utiliser JavaScript pour déplacer également l'aiguille des secondes. Nous allons animer l'aiguille des secondes 60 fois par minute.

```
/*
 * Move the second containers
 */
function moveSecondHands() {
  var containers = document.querySelectorAll('.seconds-container');
  setInterval(function() {
    for (var i = 0; i < containers.length; i++) {
      if (containers[i].angle === undefined) {
        containers[i].angle = 6;
      } else {
        containers[i].angle += 6;
      }
      containers[i].style.webkitTransform = 'rotateZ('+ containers[i].angle +'deg)';
      containers[i].style.transform = 'rotateZ('+ containers[i].angle +'deg)';
    }
  }, 1000);
}
```

Maintenant que les aiguilles des secondes et des minutes sont toutes deux gérées par JavaScript, mettons à jour notre CSS pour remplacer les propriétés `animation` par des `transitions` .

```
.minutes-container {
  transition: transform 0.3s cubic-bezier(.4,2.08,.55,.44);
}
```

```
.seconds-container {
  transition: transform 0.2s cubic-bezier(.4,2.08,.55,.44);
}
```

Ces transitions s'appliquent à la propriété `transform` et utilisent la fonction de timing `cubic-bezier` . C'est cette fonction qui donne aux aiguilles un léger rebond.

Style iOS 7

Je suis un grand fan de la simplicité de l'horloge qu'on trouve sur iOS 7. Apple a depuis allongé les aiguilles, mais je préfère la version courte.

Nous pouvons donner un style aux aiguilles et ajouter une image de background avec les numéros pour créer ce look.

Ce design est lui-même inspiré de l'horloge des chemins de fer suisses de Hans Hilfiker. En modifiant quelques styles et en ajoutant une nouvelle image de background, nous pouvons adapter notre horloge à ce style.

Si vous avez d'autres idées de design, faites-le moi savoir.

Utiliser Moment.js

Quand j'ai pensé à cet article, une de mes premières idées a été de recréer les panneaux d'horloges qu'on trouve dans certains hôtels, qui vous donnent l'heure selon différents fuseaux horaires.

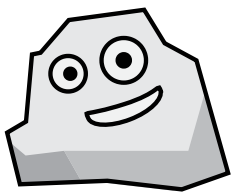
La façon la plus simple de s'ajuster à d'autres fuseaux horaires est d'utiliser la formidable bibliothèque Moment.js Timezone.

Vous pouvez voir ce code en action dans Codepen.

Compatibilité navigateurs

Les navigateurs modernes sont compatibles avec les transitions et animations CSS. IE10+, les derniers Chrome et Firefox les supportent sans préfixes et Safari requiert encore le préfixe `-webkit` .

N'oubliez pas d'utiliser les propriétés préfixées avec JavaScript également.



Make sure to follow on Twitter for all the latest updates!

Translation Info

This article translated by Pierre Choffe. (See original)

CSS Animation Weekly

A curated weekly roundup of all the latest in web animation.

Sign up now to get your FREE book, CSS Animation 101!

SUBSCRIBE

Follow on Twitter • Get in touch • CSS Courses

العربية | 中文 | Danske | Deutsch | Español | Français | 日本語 | Nederlands | Português | Polski | Ру́сский | Türkçe | Bahasa Indonesia