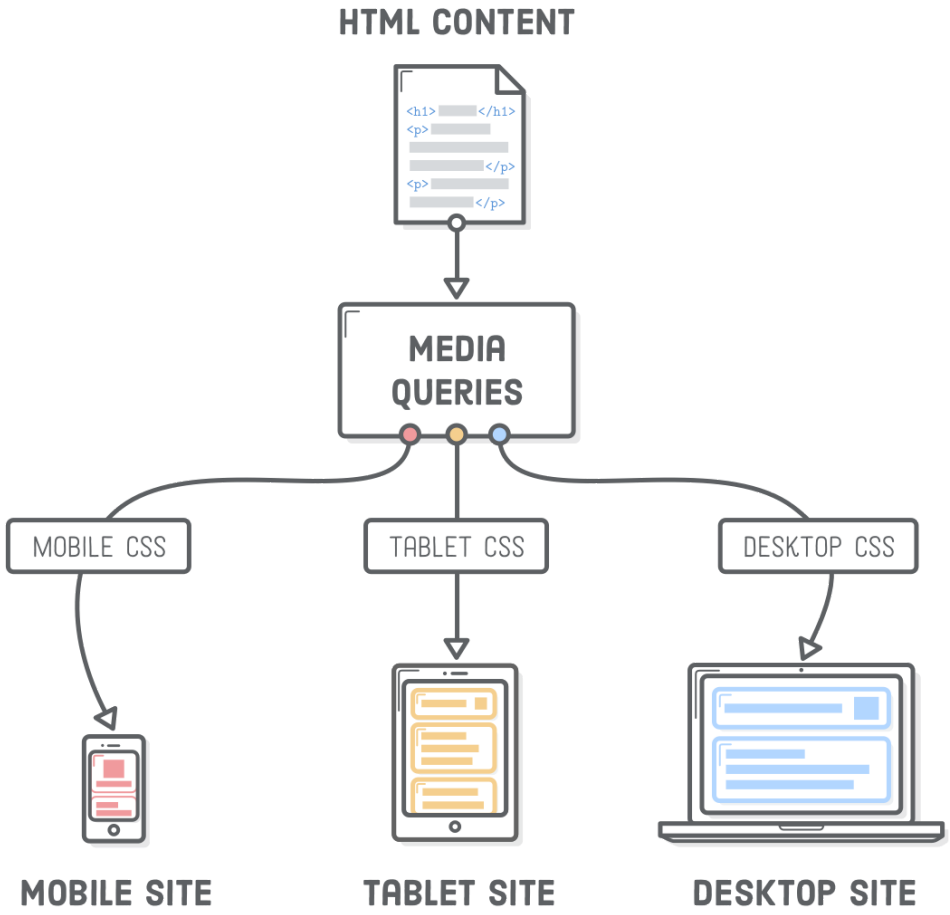
responsive design

Le «design réactif» ou «responsive design» en bon français se réfère à l'idée que votre site Web devrait s'afficher aussi bien dans tous les systèmes, qu'il s'agisse de moniteurs grand écran ou de téléphones portables. C'est une approche de la conception et du développement web qui élimine la distinction entre la version mobile de votre site Web et son homologue de bureau.

Le design réactif s'effectue grâce aux «media query» de CSS. Pensez ces requêtes sur les médias comme un moyen d'appliquer de façon conditionnelle des règles CSS. Ils indiquent au navigateur qu'il doit ignorer ou appliquer certaines règles en fonction du périphérique de l'utilisateur.



Les media queries nous permettent de présenter le même contenu HTML en autant de mises en page CSS distinctes. Ainsi, au lieu de maintenir un site web pour les smartphones et un site totalement indépendant pour les ordinateurs portables / ordinateurs de bureau, nous pouvons utiliser le même balisage HTML (et serveur Web) pour chacun d'eux. Cela signifie que chaque fois que nous ajoutons un nouvel article ou modifions une faute de frappe dans notre HTML, ces modifications seront automatiquement répercutées dans les deux configurations mobiles et écran large. C'est la raison pour laquelle nous séparons le contenu de la présentation .

Dans ce chapitre, nous allons apprendre comment les media queries sont vraiment juste un petit ajout qui va envelopper notre bon vieux CSS hien clair que nous avons travaillé avec jusqu'à ce point. Comme nous allons bientôt découvrir, il est en fait assez facile à mettre en œuvre une mise en page responsive (pour les images responsives , c'est une toute autre histoire).

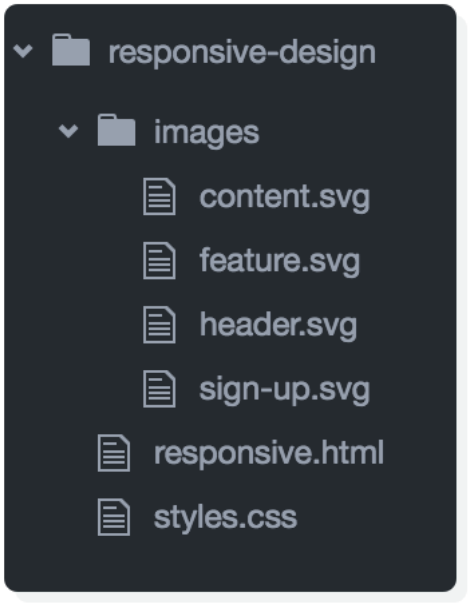
installer

Créer un nouveau projet appelé responsive-design et un nouveau fichier appelé responsive.html.

```
<!DOCTYPE html>
<html lang='en'>
  <head>
    <meta charset='UTF-8'/>
    <title>Responsive Design</title>
    <link rel='stylesheet' href='styles.css'/>
  </head>
  <body>
    <!-- There's nothing here! -->
  </body>
</html>
```

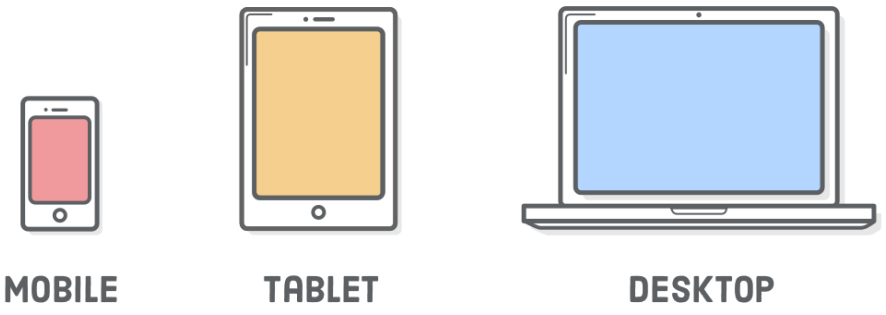
Vous aurez également besoin de télécharger des images pour plus tard dans le chapitre. Décompressez tout dans le même dossier que `responsive.html`, en gardant le dossier parent `images`.

Votre projet devrait ressembler à ceci:



css media queries

Nous allons commencer petit en mettant à jour simplement la couleur de fond sur l'élément `<body>` en fonction de la largeur de l' appareil. Ceci est un bon moyen de vérifier que nos media queries fonctionnent réellement avant d'entrer dans des mises en formes plus compliquées.

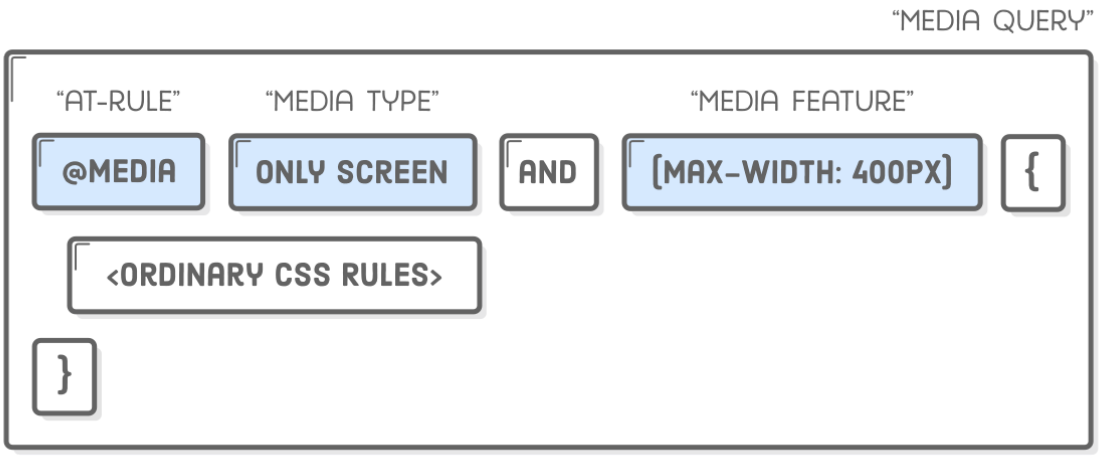


Faisons la distinction entre nos mise en page petite, moyenne et large en créant un nouveau fichier styles.css et en ajoutant ce qui suit:

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

/* Mobile */
@media only screen and (max-width: 400px) {
  body {
    background-color: #F09A9D; /* Red */
  }
}

/* Tablette */
@media only screen and (min-width: 401px) and (max-width: 960px) {
  body {
    background-color: #F5CF8E; /* Yellow */
  }
}

/* Bureau*/
@media only screen and (min-width: 961px) {
  body {
    background-color: #B2D6FF; /* Blue */
  }
}
```

Lorsque vous redimensionnez votre navigateur, vous devriez voir trois couleurs de fond différentes: bleu quand elle est supérieure à l'échelle 960px, jaune quand il est entre 401px et 960px, et rouge quand il est inférieur 400px.

Les media queries commencent toujours par la règle @media suivie par une sorte de déclaration conditionnelle, puis quelques accolades. A l' intérieur des accolades, vous mettez un tas de règles CSS ordinaires. Le navigateur ne prête attention à ces règles que si la condition est remplie.



Le «media type» only screen signifie que les styles contenus ne doivent être appliquées qu'aux appareils avec des écrans (opposés aux documents imprimés, comme lorsque vous appuyez sur **Cmd + P** dans un navigateur). Le min-width et le max-width sont appelées media feature, ils précisent les dimensions de l'appareil que vous ciblez.

Les media queries ci-dessus sont de loin les plus courantes que vous allez rencontrer, mais il y a beaucoup d' autres conditions que vous pouvez vérifier pour, y compris si l'appareil est en mode portrait ou paysage, la résolution de son écran, et si elle a une souris ou non.

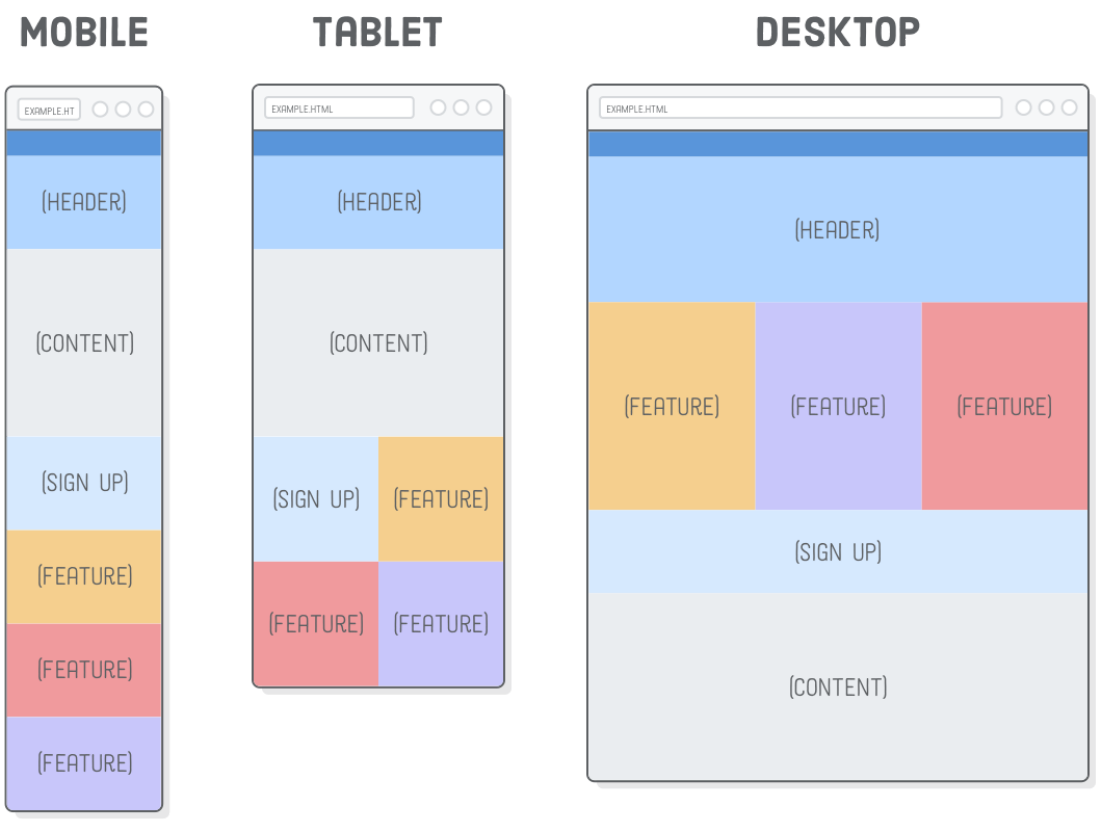
Un exemple est l'utilisation courante de la condition "orientation" qui peut prendre les valeur "portrait" ou "landscape".
Vous pouvez avoir un ensemble de propriétés CSS qui s'appliqueront uniquement lorsque la fenêtre du navigateur sera plus large que sa hauteur, que l'on appelle l'orientation "paysage".

```
/* Mobile Styles */
@media only screen and (max-width: 400px) and (orientation:portrait) {
  body {
    font-size: 12px;   }
}
```

quelques notes sur la conception

Donc avec @media nous définirons différentes configurations pour des largeurs de périphériques spécifiques, mais qu'est ce que nous sommes effectivement en train de mettre en œuvre ?

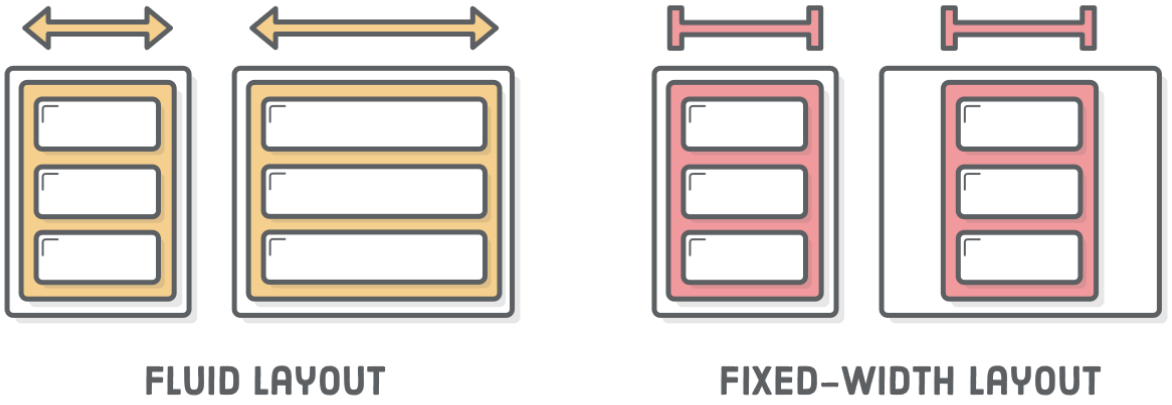
La page web exemple pour ce chapitre va ressembler à quelque chose comme ceci:



Dans le monde réel, c'est au concepteur du site Web de fournir ces types de maquettes. Votre travail en tant que développeur est de mettre en œuvre les dispositions individuelles en utilisant des media queries pour séparer les différentes règles CSS applicables à chacun.

Il y a quelques modèles bien définis pour la façon dont une disposition de bureau se redesign dans une mise en page pour des mobiles. Un grand nombre de ces décisions sont du domaine de la conception, ce qui est en dehors du cadre du cours; cependant, il y a deux concepts que vous devez comprendre en tant que développeur:

- Une mise en page "fluide" est celle qui étend et se rétrécit pour remplir la largeur de l'écran, tout comme les boites flexibles flexbox.
- La mise en page de "largeur fixe" est à l'opposé: au aura la même largeur quelles que soient les dimensions de l'écran.



Dans notre exemple de page Web, les versions mobiles et tablettes sont fluides, et la version de bureau est de largeur fixe.

Choisir les points de rupture -breakpoints-

La plupart de ces modèles de conception responsive ont un comportement similaire, l'utilisation des dispositions de fluides pour les appareils mobiles / tablettes et mises en page à largeur fixe pour des écrans plus larges. Il y a une raison pour cela.

La mise en page fluide nous permet de cibler une *gamme* de largeurs d'écran au lieu d'appareils mobiles spécifiques. Ceci est très important pour les concepteurs Web. Quand ils ont décidé de créer une mise en page

mobile, ils ne cherchent pas à faire quelque chose qui semble bon sur un iPhone 6s, Galaxy S7, ou mini - iPad- ils font la conception d'une mise en page fluide qui se comporte bien *partout* entre 300 pixels et 500 pixels (ou autre).

En d'autres termes, les valeurs exactes de pixels pour les paramètres min-width et max-width dans media query (appelés les «points de rupture ou breakpoint» pour un site responsive) n'ont en fait pas d' importance. Notre site ne se soucie pas de l'appareil spécifique avec lequel l'utilisateur parcours le site. Tout ce qu'il a besoin de savoir est qu'il doit afficher une mise en page pour un écran de 400 pixels de large (ou autre).

Développement mobile-first

Il est toujours une bonne idée de commencer avec la mise en page mobile et de la faire évoluer jusqu'à la version de bureau. Les mises en page de bureau sont généralement plus complexes que leurs homologues mobiles, et cette approche «mobile first» maximise la quantité de CSS que vous pouvez réutiliser dans vos mises en page.

Premièrement, nous devons remplir responsive.html avec l'élément <body> et avec des sections vides. Chaque section a une image en elle afin que nous puissions leur dire à part un peu plus facile.

```
<div class='page'>
  <div class='section menu'></div>
  <div class='section header'>
    <img src='images/header.svg'/>
  </div>
  <div class='section content'>
    <img src='images/content.svg'/>
  </div>
  <div class='section sign-up'>
    <img src='images/sign-up.svg'/>
  </div>
  <div class='section feature-1'>
    <img src='images/feature.svg'/>
  </div>
```

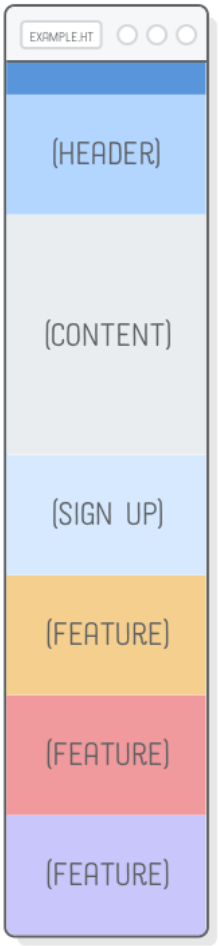
```
  <div class='section feature-2'>
    <img src='images/feature.svg'/>
  </div>
  <div class='section feature-3'>
    <img src='images/feature.svg'/>
  </div>
</div>
```

Et voici nos styles de base, qui devraient être appliquées à *toutes* les mises en page (mobile, tablette, et de bureau). Assurez-vous d'ajouter ces au dessus des règles @media que nous avons créé plus tôt et en dessous de la règle sur le sélecteur universel qui réinitialise nos marges et padding:

```
.page {
  display: flex;
  flex-wrap: wrap;
}

.section {
  width: 100%;
  height: 300px;
  display: flex;
  justify-content: center;
  align-items: center;
}

.menu {
  background-color: #5995DA;
  height: 80px;
}

.header {
  background-color: #B2D6FF;
}

.content {
  background-color: #EAEDF0;
  height: 600px;
}
```

```
.sign-up {
  background-color: #D6E9FE;
}

.feature-1 {
  background-color: #F5CF8E;
}

.feature-2 {
  background-color: #F09A9D;
}

.feature-3 {
  background-color: #C8C6FA;
}
```

Si vous réduisez la fenêtre du navigateur, vous verrez que cela nous donne toute notre disposition mobile. Assez facile, hein ? Pas de requêtes de médias requis. Voilà pourquoi c'est appelé «mobile first» -la version mobile ne nécessite pas de traitement spécial. Notez également que la propriété flex-wrap dans la div .page. Ainsi, il sera très facile de mettre en œuvre nos mises en page pour les tablettes et les ordinateurs de bureau.



En gardant ces styles de base en dehors des media queries, nous sommes en mesure de remplacer et d'ajouter ce que nous mettrons en œuvre pour nos dispositions spécifiques. Ceci est très pratique lorsque, par exemple, votre concepteur veut modifier le schéma de couleurs pour l'ensemble du site. Au lieu de traquer des déclarations redondantes background-color dans plusieurs règles @media, il suffit de le mettre à jour ici. Ce changement s'applique automatiquement à la mise en page mobile, tablette, et de bureau.

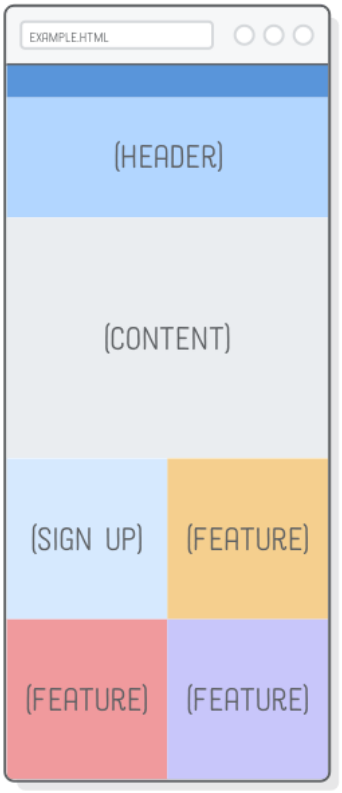
présentation pour tablette

À la disposition de la tablette. La seule différence entre les maquettes mobiles et tablettes est que les sections **Sign Up** et **Feature** forment une grille de 2 × 2 au lieu d'une seule colonne.

Flexbox rend cela vraiment facile. Il suffit de régler la largeur des éléments flexibles pour être affiché sur la moitié de l'écran et flex-wrap se chargera du reste. Bien sûr, nous voulons que ce comportement à appliquer aux écrans de tablettes de taille, donc il a besoin d'aller dans une règle @media. Remplacer l'actuel media query /* Tablettes */ avec les éléments suivants:

```
/* Tablette */
@media only screen and (min-width: 401px) and (max-width: 960px) {
  .sign-up,
  .feature-1,
  .feature-2,
  .feature-3 {
    width: 50%;
  }
}
```

Pour voir ces changements, assurez-vous que la fenêtre du navigateur se situe entre 400 pixels et 960 pixels de large, puis faites défiler vers le bas de la page. Vous devriez voir une grille colorée:



Encore une fois, il n'y a pas d'importance à ce que la largeur exacte de l'écran soit connue puisque cette mise en page de façon fluide répondra à toute les largeurs de la gamme de la media query. Notre mise en page mobile est également fluide, de sorte que nous avons maintenant un site Web qui est magnifique (même s'il est un peu vide) pour chaque appareil plus petit que 960px de large.

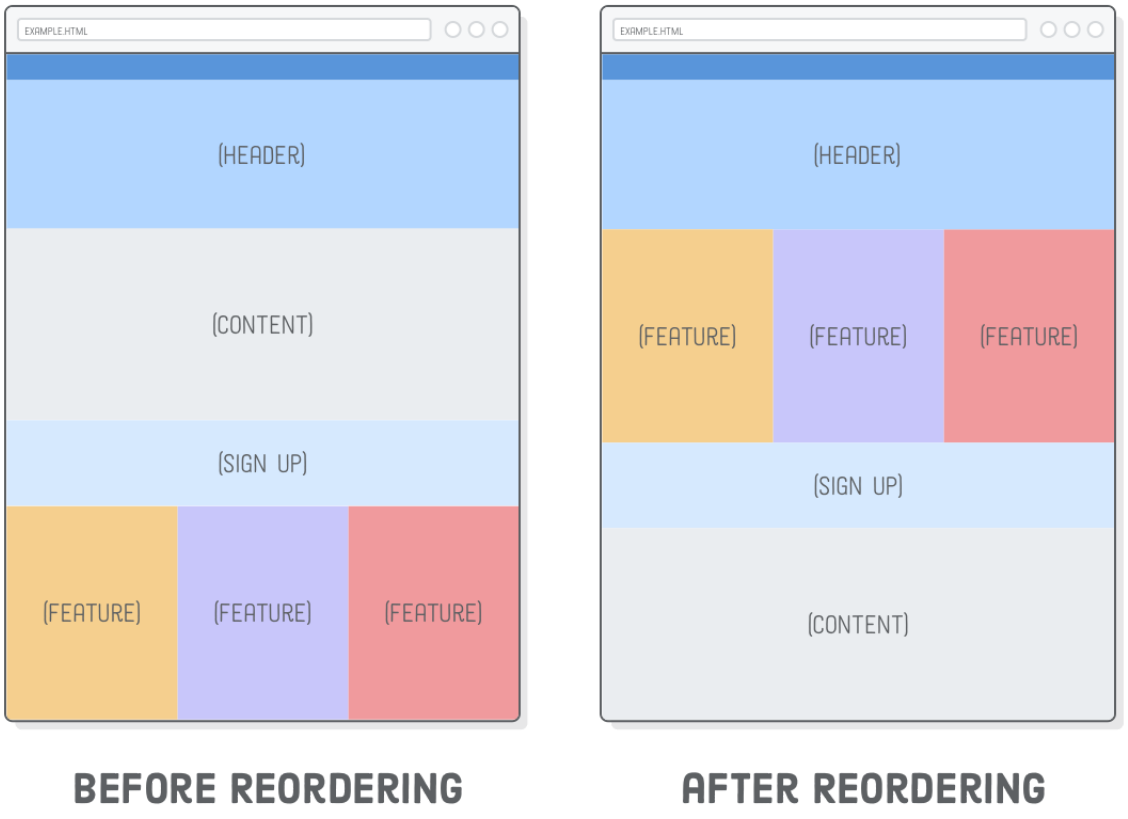
Version ordinateur de bureau

Et c'est là notre disposition de bureau entre en jeu . Nous ne voulons pas que notre page Web s'élargisse sans cesse, alors nous allons lui donner une largeur fixe et la centrer avec des marges "auto". Comme avec les styles de tablettes, cela doit aller dans une requête de médias. Remplacer l'actuelle media query /* Bureau */ avec les éléments suivants:

```
/* Bureau */
@media only screen and (min-width: 961px) {
  .page {
    width: 960px;
    margin: 0 auto;
  }
  .feature-1,  .feature-2,  .feature-3 {
    width: 33.3%;
  }
  .header {
    height: 400px;
  }
}
```

Cela nous donne les largeurs correctes pour tout, et nous avons fait un entête un peu plus grand, aussi.

On y est presque, mais notre disposition de bureau appelle quelques réordonnancement: **Sign Up** et **contenu** doivent apparaître *sous* tous les sections **Feature**.



BEFORE REORDERING **AFTER REORDERING**

C'est là que Flexbox brille vraiment. Essayer de créer cette combinaison de configurations de bureau et mobiles serait très difficile avec des float . Avec la propriété order de Flexbox, il s'agit juste d'ajouter quelques lignes de CSS. Ajouter ces règles à la requête des médias pour la configuration bureau:

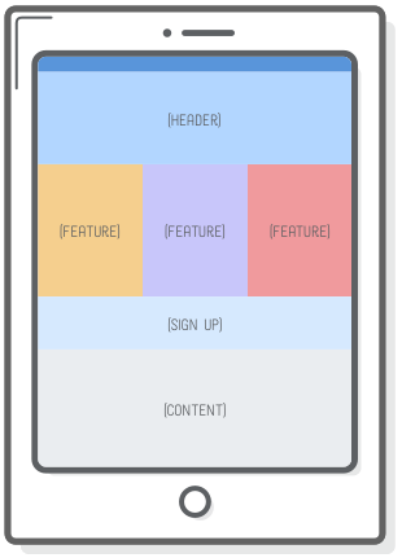
```
.sign-up {
  height: 200px;
  order: 1;
}
.content {
  order: 2;
}
```

Ta da ! Un site responsive ! Pas mal pour moins d'une centaine de lignes de CSS. Plus important encore, on n'a pas eu à modifier une seule ligne de code HTML pour réaliser nos mises en page mobile, tablettes et ordinateurs de bureau.

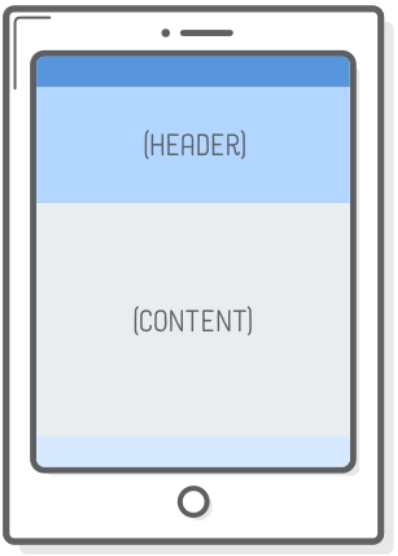
Ce fut juste un exemple, vous pouvez utiliser ces mêmes techniques pour mettre en œuvre toutes sortes d'autres conceptions. Commencez par les styles de base applicables à l'ensemble de votre site, puis les retravailler pour diverses largeurs de périphériques en appliquant sélectivement des règles CSS avec @media. Vous pouvez même ajouter une autre requête de médias, par exemple, créer une mise en page dédiée pour les moniteurs à écran ultra-grand.

désactivation du viewport zooming

Nous avons une tâche finale pour rendre une page web responsive. Avant que le responsive n'existe, les appareils mobiles ne pouvaient seulement afficher une mise en page de bureau. Pour faire face à cela, ils faisaient zoom arrière pour adapter la mise en page entière de bureau dans la largeur de l'écran, permettant à l'utilisateur d'interagir avec elle en faisant un zoom si nécessaire.



ZOOM ENABLED

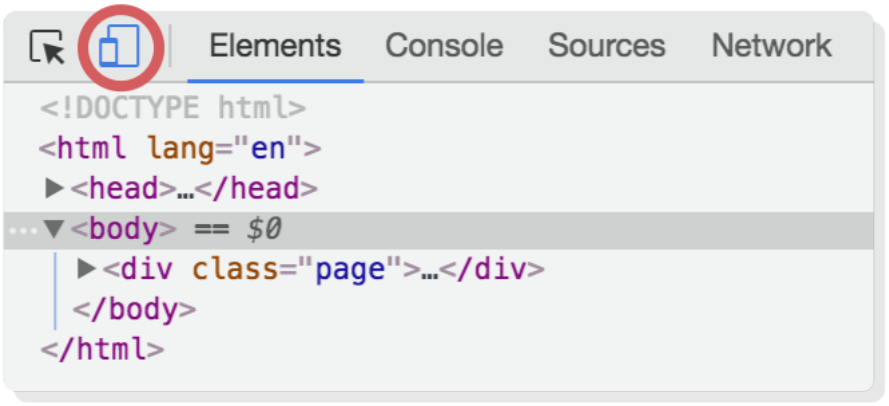


ZOOM DISABLED

Ce comportement par défaut empêche les appareils mobiles d'utiliser notre disposition mobile, ce qui est évidemment terrible. Pour le désactiver, il faut ajoutez l'élément suivant dans le <head> de notre document. Tout comme <meta charset='UTF-8'/>, il est un élément essentiel qui devrait être sur chaque page web que vous créez:

```
<meta name='viewport' content='width=device-width, initial-scale=1.0, maximum-scale=1.0' />
```

Pour le voir en action, nous aurons besoin de *simuler* un appareil mobile dans notre navigateur de bureau. Ouvrez responsive.html dans Google Chrome, puis appuyez sur **Affichage> Développeur> Outils de développement** dans la barre de menu. Ensuite, pour simuler un appareil mobile, cliquez sur la **barre d' outils Basculer périphérique** icône, mis en évidence ci - dessous.



Vous devriez voir la version avec le zoom désactivé du diagramme ci-dessus dans votre navigateur, car il fait maintenant semblant d'être un appareil mobile.

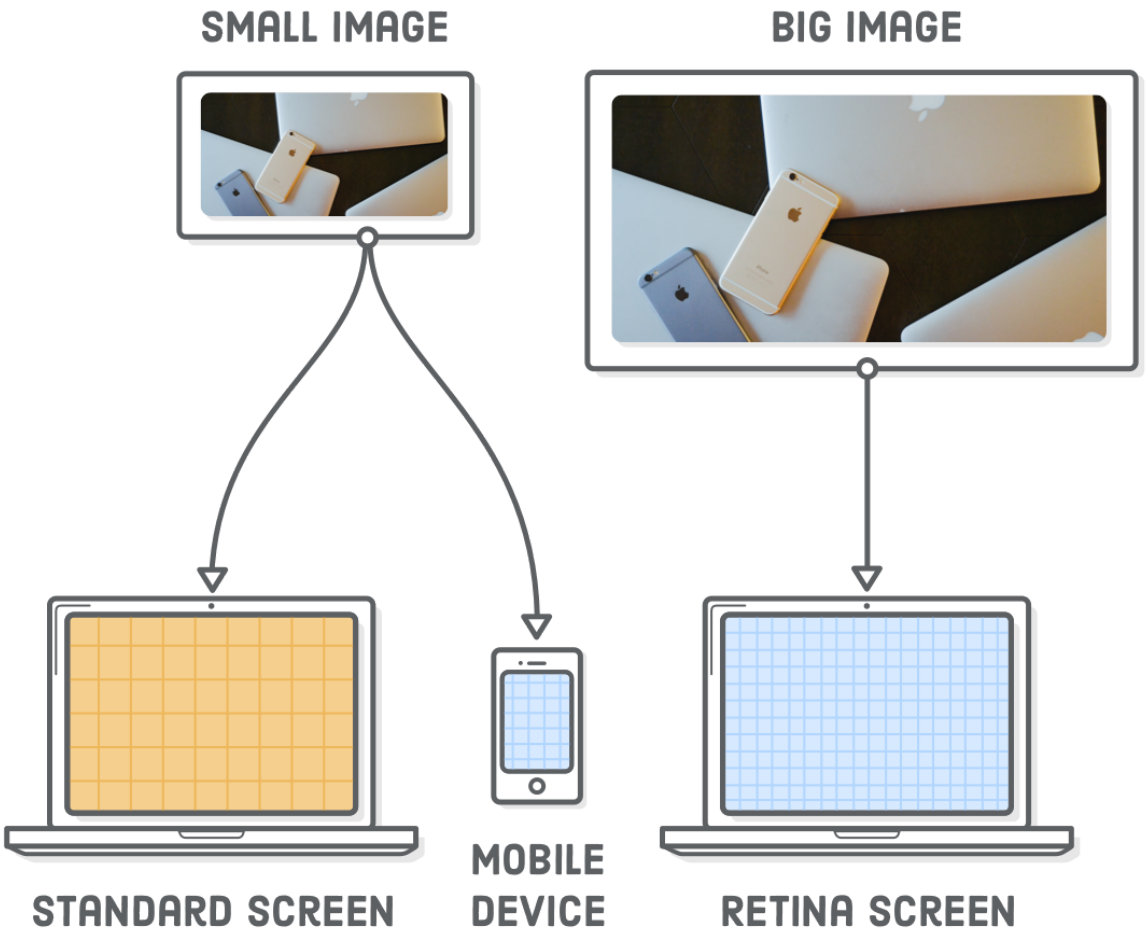
résumé

Croyez-le ou non, c'est en fait tout ce que vous devez savoir pour créer des sites responsive, à savoir trois choses:

- Le responsive de *conception* (les maquettes pour chaque mise en page)
- les règles CSS pour la mise en œuvre de chacune de ces dispositions
- les media queries pour appliquer conditionnellement ces règles CSS

Alors, c’était la partie facile du responsive design. Dans le chapitre suivant, nous allons découvrir la partie difficile: les images .L'optimisation des images pour ces appareils nécessite un peu plus de planification.

Dans Responsive Design, nous avons appris à utiliser des media queries pour créer des versions mobiles séparés, tablettes et mises en page de bureau. Maintenant, nous allons ajouter des images au mélange. Tout comme les demandes des médias nous permettent d'afficher conditionnellement différentes règles CSS, nous voulons afficher des images différentes en fonction de l'appareil de l'utilisateur.

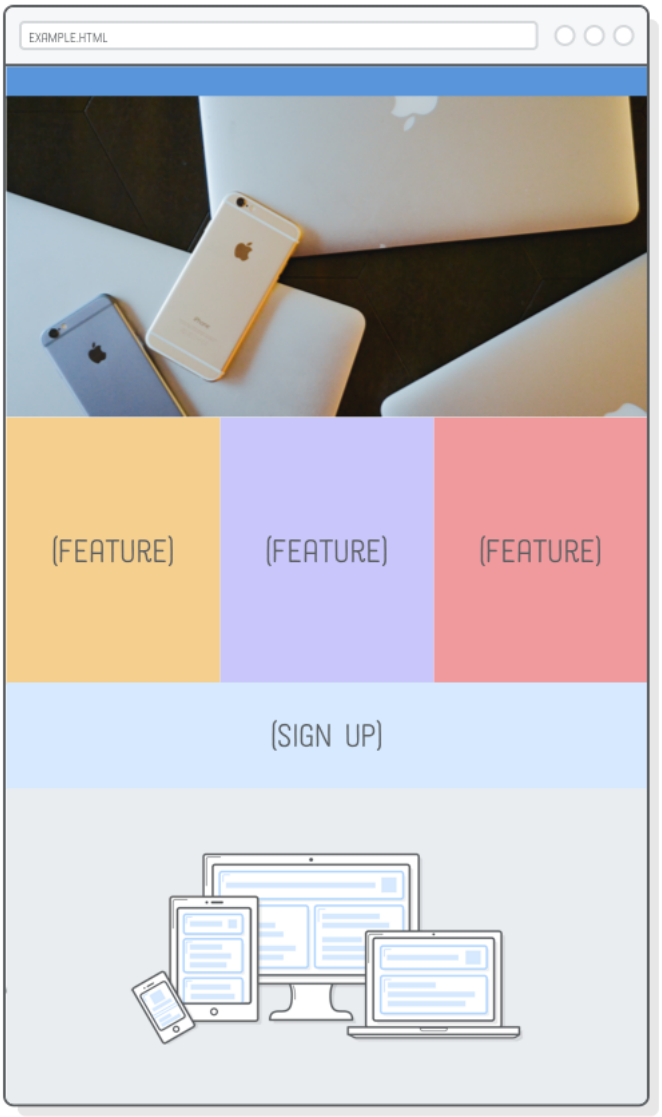


Le problème est, les images ont des dimensions prédéfinies. Nous ne pouvons pas étirer une photo qui est de 500 × 250 pixels à quoi que ce soit au-delà de 500 pixels de large, car il va se pixéliser. Les écrans Retina et les appareils mobiles compliquent les choses encore plus. Afin de rendre

nos images responsives, nous allons maintenant prendre trois choses en considération:

- Les dimensions de l'appareil
- Les dimensions de l'image
- La résolution de l'écran de l'appareil

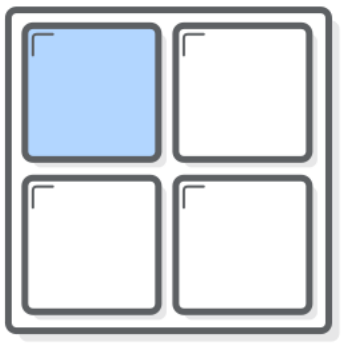
Ce sera plus difficile que les media queries, qui étaient seulement concernés par la largeur de l'appareil. Mais ne vous inquiétez pas, il existe des moyens standard pour résoudre tous ces problèmes.



Dans les deux cas, vos fichiers de projet devraient ressembler à ceci avant de passer. Remarquez comment nous avons plusieurs copies de nos images PNG et JPG (par exemple, illustration-big.png et illustration-small.png). Nous allons laisser le navigateur choisir lequel de ces devrait charger en fonction de la taille de l'appareil et la résolution de l'écran.

les écrans retina

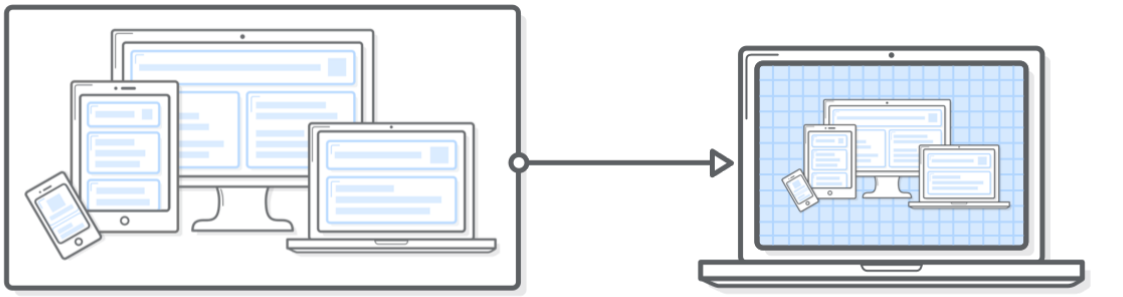
Ceci est notre première fois se soucier des écrans retina, alors parlons un peu de résolution d'écran. Les écrans retina ont deux fois plus de pixels par pouce que les écrans de résolution standard. En d'autres termes, chaque pixel retina est l'équivalent de 4 pixels standard. Cela a un impact important sur la façon dont les images sont affichées dans un navigateur Web.



STANDARD RESOLUTION

RETINA RESOLUTION

Pour rendre correctement sur un périphérique retina, une image doit être deux fois plus grand que ses dimensions d'affichage final. Par exemple, si vous souhaitez ajouter un 500 x 250 pixels de l'image à la page, le fichier image correspondant doit être 1000 × 500 pixels.



Ceci est en fait un peu une simplification, tous les écrans retina ne sont pas égaux. Par exemple, l'iPhone 6 Plus a *trois fois* plus de pixels par pouce qu'un écran standard. Ce cours se concentre sur le cas d'utilisation 2x, mais les mêmes techniques s'appliquent aux écrans retina 3x.

De plus, les écrans standard et les périphériques plus petits n'ont pas besoin de tous ces pixels supplémentaires dans des images haute résolution, et l'envoi de beaucoup de données inutiles entraîne généralement une mauvaise expérience utilisateur.

images svg responsive

La meilleure façon de résoudre tous ces problèmes est avec des images SVG . Comme ils sont à base de vecteurs, SVG évite les problèmes de résolution d'écran que nous allons voir dans la section suivante. Jetons un coup d'oeil en ajoutant une illustration à notre page responsive.html. Remplacer l'image existante dans la div .content afin qu'elle corresponde à ce qui suit:

```
<div class='section content'>
  <img class='illustration' src='images/illustration.svg' >
</div>
```

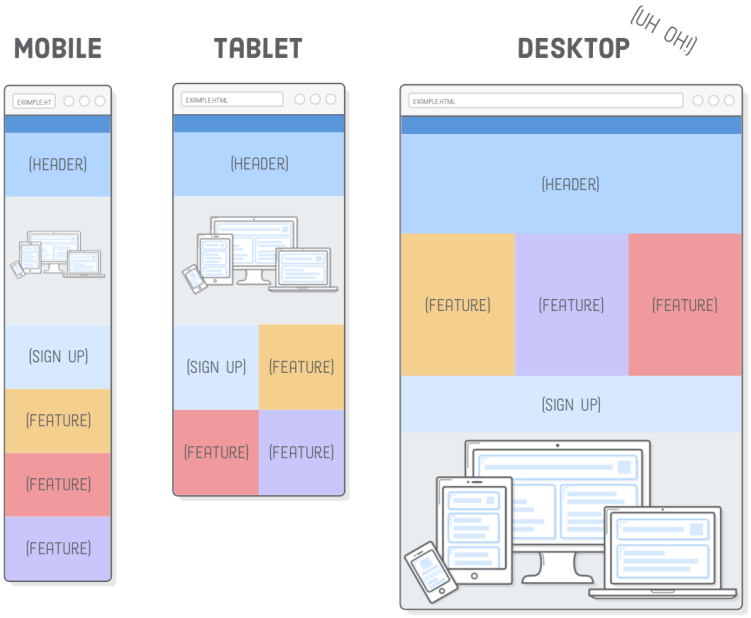
Les navigateurs mettent à l'échelle automatiquement le SVG pour les périphériques retina, de sorte que cette image SVG de 500 × 250 pixels sera bien rendue sur les appareils standard et retina.

Avec le SVG nous oublions les problèmes de résolution d'écran, mais nous devons réduire l'illustration pour tenir parfaitement dans notre tablette fluide et mises en page mobiles. Firefox le fera automatiquement, mais si vous ouvrez cette page avec Chrome et que vous réduisez la largeur votre navigateur, vous verrez que l'image reste la même taille.

Pour obtenir une image fluide dans Chrome, nous devons indiquer que l'illustration doit remplir toute la largeur de son conteneur. Dans styles.css, mettre la règle suivante avec le reste des styles de base, en dehors des des médias queries:

```
.illustration {
  width: 100%;
}
```

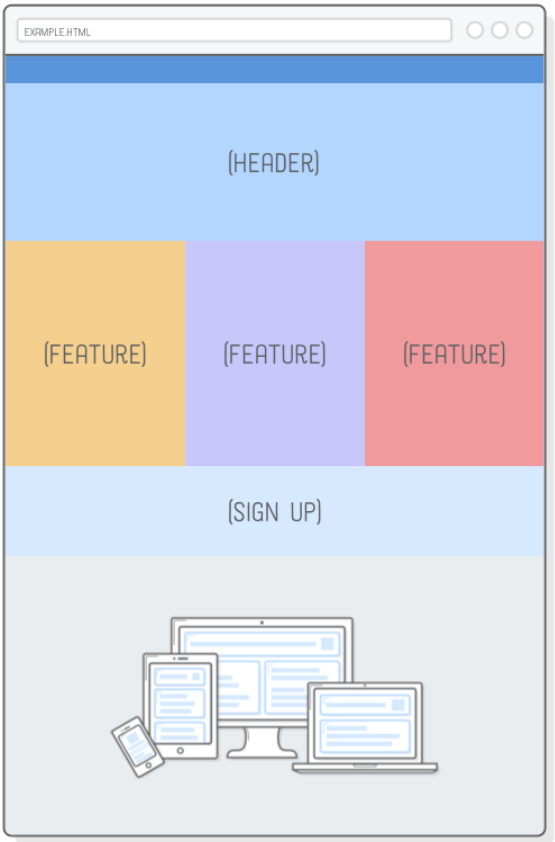
Lorsque nous précisons la largeur 100% sur une image, il supposerons que nous voulons maintenir son ratio d'aspect et de calculer automatiquement sa hauteur. Cela corrige la mise en page mobile, mais maintenant la version de bureau est *énorme* :



Ce comportement est parfait pour certains modèles (comme la photo à fond perdu que nous verrons dans la section suivante), mais pas maintenant. Nous voulons limiter la largeur de l'illustration à sa largeur intrinsèque, qui est de 500 pixels. Nous pouvons le faire avec un style en ligne :

```
<div class='section content'>
  <img class='illustration' src='images/illustration.svg' style='max-width: 500px'/>
</div>
```

Ceci est l'une des rares fois qu'un style en ligne est acceptable, en raison du fait qu'il est décrit comme une propriété intrinsèque de l'image. Les dimensions physiques d'une image sont plus de contenu que la présentation, il est donc logique pour que cela apparaisse dans le code HTML plutôt que dans la feuille de style.



responsive et images png, gif, jpg

Bien sûr, pas toutes les images sur le web sont SVGs. Parfois, vous devez inclure une photo. PNG, GIF, JPG et images sont des « images raster », ce qui signifie qu'ils sont définis pixel par pixel au lieu de vecteurs. En conséquence, ils sont beaucoup plus sensibles à la résolution de l'écran que SVGs.

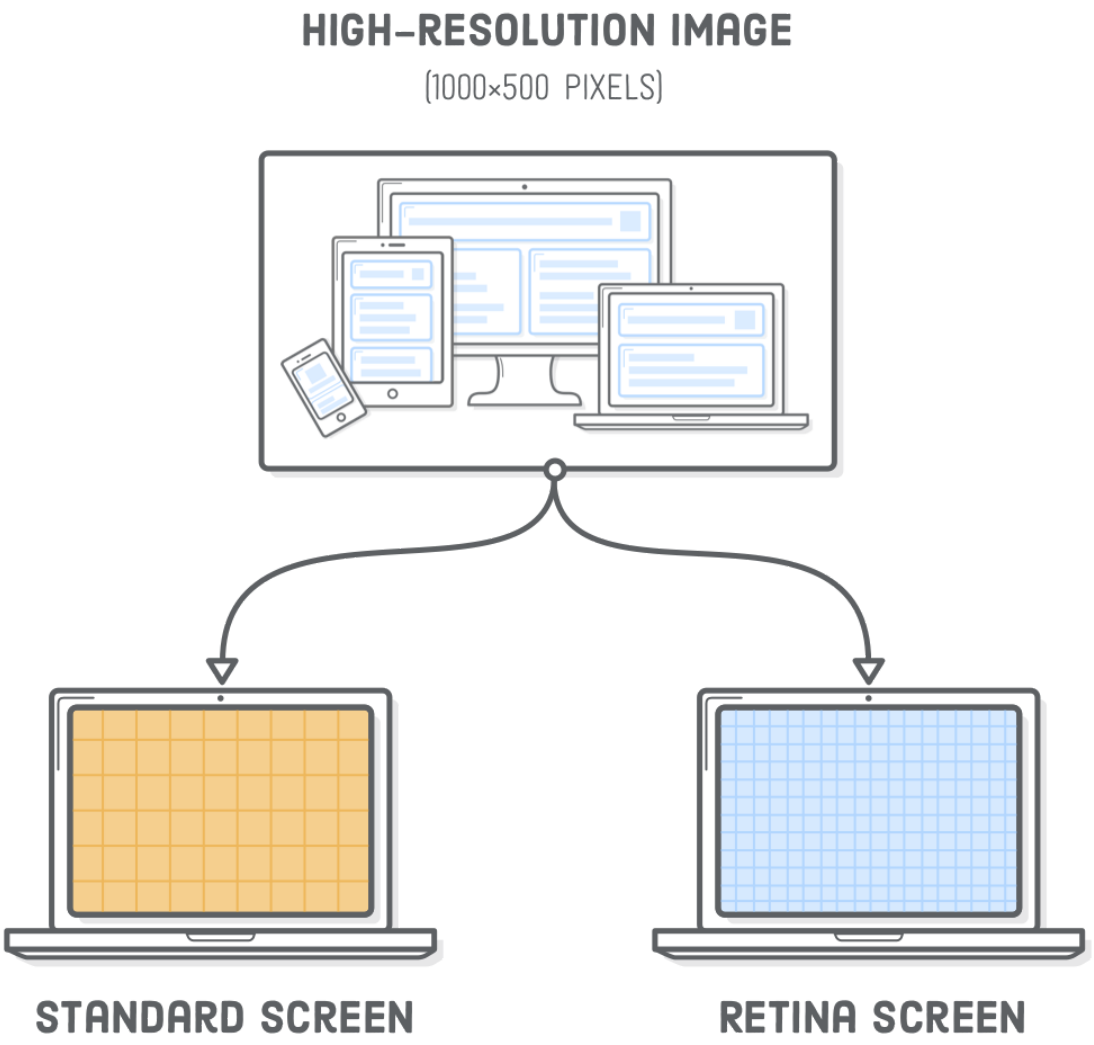
Si vous n'êtes pas inquiet sur l'optimisation, les images raster responsives ne sont pas vraiment beaucoup plus difficile à utiliser que des images SVG. Essayez de remplacer notre fichier existant `illustration.svg` avec un fichier PNG:

```
<div class='section content'>
  <div class='illustration'>
    <img src='images/illustration-big.png' style='max-width: 500px'/>
  </div>
</div>
```

Nous avons changé autour de la structure HTML un peu, l'imbrication de notre balise `<img/>` dans un autre conteneur. Sans elle, l'image serait déformée parce que flexbox tenterait de définir sa hauteur afin d'être le même que le conteneur `.content`. Cela nécessite un peu de modification à notre règle CSS `.illustration` :

```
.illustration img {
  width: 100%;
  display: block;
}
```

Notez également le suffixe -big dans le nom de fichier de l'image. Ceci est la version haute résolution de la PNG, qui a des dimensions de 1000x500. Les appareils Retina ont besoin de cette taille « 2x » pour afficher l'image de façon nette. Si nous devions utiliser la version basse résolution de cette image (500 x 250 pixels), elle sera bien sur les écrans standard, mais floue sur les appareils retina.



Considérez ceci comme la manière paresseuse de créer des images PNG, GIF ou JPG responsive, car cela suppose que tout le monde a besoin d'une image haute résolution, même ce n’est pas le cas. C'est-à-dire, une image de 1000 × 500 pixels est too much pour les dispositifs non retina.

optimisation des images responsives

Différents dispositifs ont des exigences différentes d'image. Heureusement, HTML fournit un moyen de choisir la meilleure image pour l'appareil de l'utilisateur. Nous allons jeter un oeil à trois scénarios pour l'optimisation des images sensibles:

- Un écran résolution standard qui n'a pas besoin d'une image de qualité retina.
- Un appareil mobile retina qui peut utiliser une image de qualité standard, car il a été mis à l'échelle et donc se dégrade.
- Une mise en page de bureau qui utilise une grande image et une mise en page mobile associé qui utilise un plus grand image.

La première méthode est la plus simple, et il est idéal pour les images plus petites de 600 pixels de large, car ils ne sont pas assez grand pour profiter du second scénario. La deuxième méthode est une optimisation très importante pour des images plus grandes, en particulier les photos à fond perdu. Le troisième est pour quand vous êtes feelin' de fantaisie.

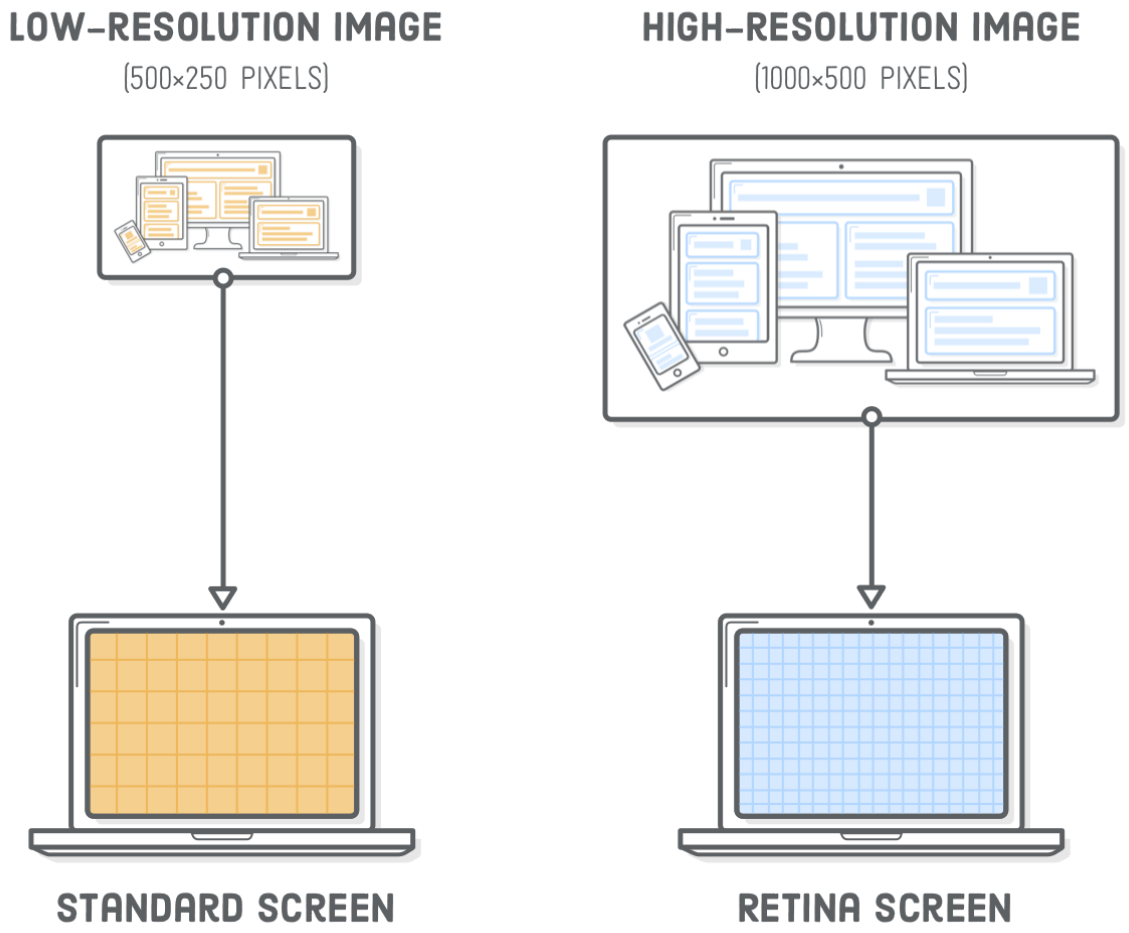
optimisation retina à l'aide de “srcset”

Des images haute résolution sont grandes. Notre fichier `illustration-big.png` prend plus de deux fois plus d'espace disque que son homologue à faible résolution. Il n'a pas de sens pour servir toutes les données supplémentaires lorsque l'utilisateur n'en a pas réellement besoin.

Ajout d'un attribut `srcset` à notre élément `<img >` nous permet de présenter notre image à haute résolution *que* pour les appareils retina, retombant à la version basse résolution pour les écrans standards. Pour cela il faut mettre à jour notre élément `.illustration` pour qu’il correspondre à ce qui suit:

```
<div class='illustration'>
  <img src='illustration-small.png'
      srcset='images/illustration-small.png 1x,  images/illustration-big.png 2x'
      style='max-width: 500px'/>
</div>
```

L'attribut srcset pointe vers une liste de fichiers d'images alternatives, ainsi que des propriétés définissant lorsque le navigateur doit utiliser chacun d'entre eux. Le 1x dit au navigateur d'afficher illustration-small.png sur les écrans de résolution standard. Les 2x moyens qui illustration-big.png est pour les écrans de retina. Les navigateurs plus anciens qui ne comprennent pas srcset se replient à l'attribut src.



En règle générale, les versions basse résolution et haute résolution d'une image serait exactement la même (sauf pour leurs dimensions), mais nous

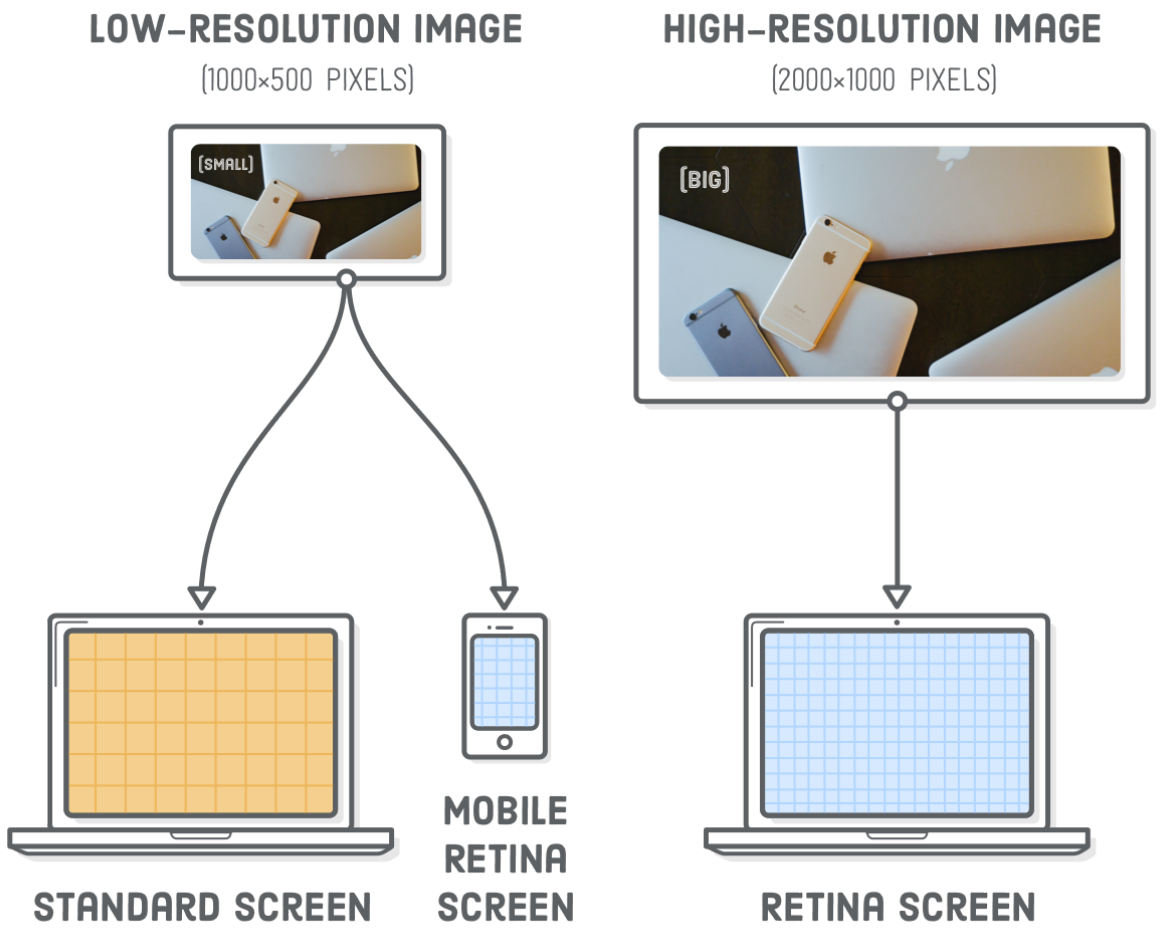
avons fait illustration-small.png jaune afin que vous puissiez facilement différencier de la version retina, qui est bleue.

Il est un peu difficile de voir en action sans site réel, donc nous avons inclus l'extrait précédent sur cette page. L'image ci-dessous devrait être bleu si vous affichez sur un périphérique retina. Dans le cas contraire, il sera jaune pour les écrans de résolution standard.

Si vous construisez ces exemples sur un ordinateur avec un écran Retina, vous pouvez aussi essayer de changer temporairement 2x pour 1x et voir à quoi une image non-retina ressemble. Elle est un peu floue (et jaune).

Optimisation de largeur d'écran en utilisant srcset

Génial ! Nous pouvons sauver quelques octets supplémentaires pour les périphériques non-retina. Malheureusement, ce qui précède la technique srcset manque d'un important cas d'utilisation pour des images plus grandes: si l'utilisateur dispose d'un smartphone retina, il va télécharger l'image en haute résolution, même si la version standard suffit.

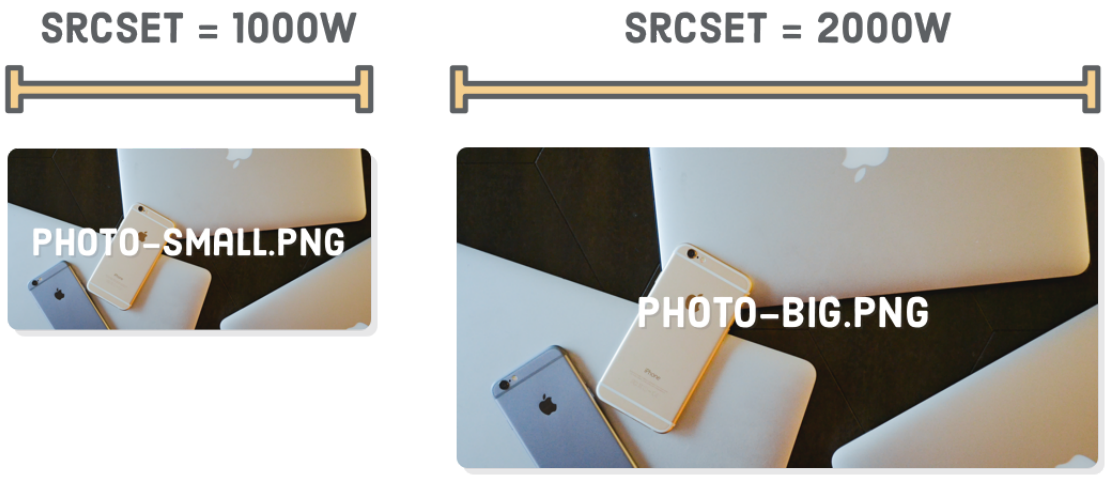


Imaginez que nous voulions afficher une grande photo dans notre élément .header. L'en-tête est de 960 pixels de large dans notre mise en page de bureau, notre photo doit être au moins 1920 pixels de large pour afficher bien sur les écrans retina. Nous fournirons également une photo à l'échelle pixel 960 pour les écrans standards. Maintenant, considérons un smartphone avec un écran retina. Les smartphones sont typiquement inférieures à 400 pixels de large en mode portrait, ce qui signifie que l'image de la qualité retina correspondante aurait seulement besoin d'être de 800 pixels de large.

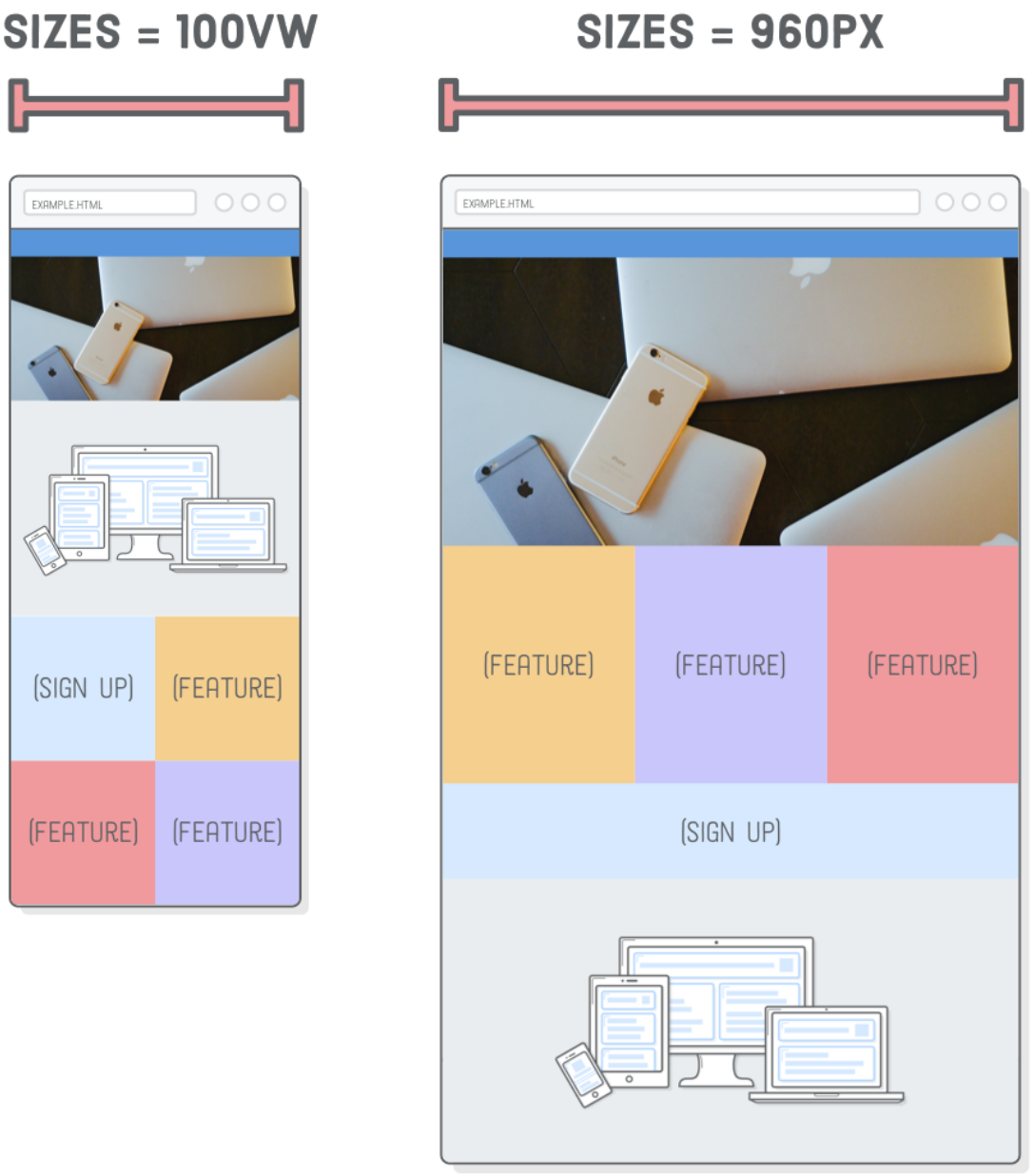
La leçon ici est que nous voulons optimiser des images plus grandes en fonction de leurs dimensions finales rendues, non seulement la résolution de l'écran de l'appareil. Allons - y et ajoutez cette grande photo à notre élément .header :

```
<div class='section header'>
  <div class='photo'>
    <img src="images/photo-small.jpg"
        srcset='images/photo-big.jpg 2000w, images/photo-small.jpg 1000w'
        sizes='(min-width: 960px) 960px, 100vw' >
  </div>
</div>
```

Nous avons le même élément srcset que la dernière section, mais au lieu de descripteurs 1x et 2x, nous fournissons la largeur physique inhérente de l'image. Le 2000w indique au navigateur que le fichier photo-big.jpg est de 2000 pixels de large. De même, le moyen 1000w photo-small.jpg a une largeur de 1000 pixels. Si vous vous interrogez sur ce caractère w, c'est une unité spéciale utilisée uniquement pour ce type de scénario d'optimisation d'image.



La largeur seule ne suffit pas pour un dispositif pour déterminer quelle image il faut charger. Nous devons aussi dire ce que sera la largeur finale de l'image. C'est là l'attribut sizes entre en jeu. Il définit une série de media query ainsi que la largeur de l'image rendue lorsque cette requête de médias est en vigueur.



Ici, nous disons que lorsque l'écran est au moins 960px large, l'image sera également 960 pixels de large. Dans le cas contraire, la valeur 100vw par défaut indique au navigateur que la largeur de l'image sera de 100% de la « largeur de la fenêtre ». Vous pouvez en savoir plus sur l'unité vw sur le site MDN. Tout cela correspond au comportement de redimensionnement d'image qui est dans notre CSS.

En parlant de ça, nous devons faire quelques changements pour positionner notre nouveau correctement l'image d'en-tête. Ajouter les deux

règles suivantes à nos autres styles de base, juste au-dessus du mobile styles requête médias:

```
.header {
  height: auto;
  justify-content: inherit;
  align-items: inherit;
}

.photo img {
  width: 100%;
  display: block;
}
```

Rappelez-vous que notre photo basse résolution est de 1000 pixels de large, ce qui signifie que les appareils 2x retina peuvent l'utiliser aussi longtemps que leur écran est inférieure à 500 pixels de large. Dans Firefox, vous devriez maintenant être en mesure de redimensionner le navigateur pour voir la version retina (« Big ») lorsque la fenêtre est plus large de 500 pixels et la version non-retina (« Small ») pour des largeurs plus étroites.

Nous servons maintenant une image 115KB aux appareils mobiles au lieu de les forcer à utiliser l'image 445KB haute résolution. C'est une grosse affaire, en particulier pour les sites Web qui utilisent beaucoup de photos.

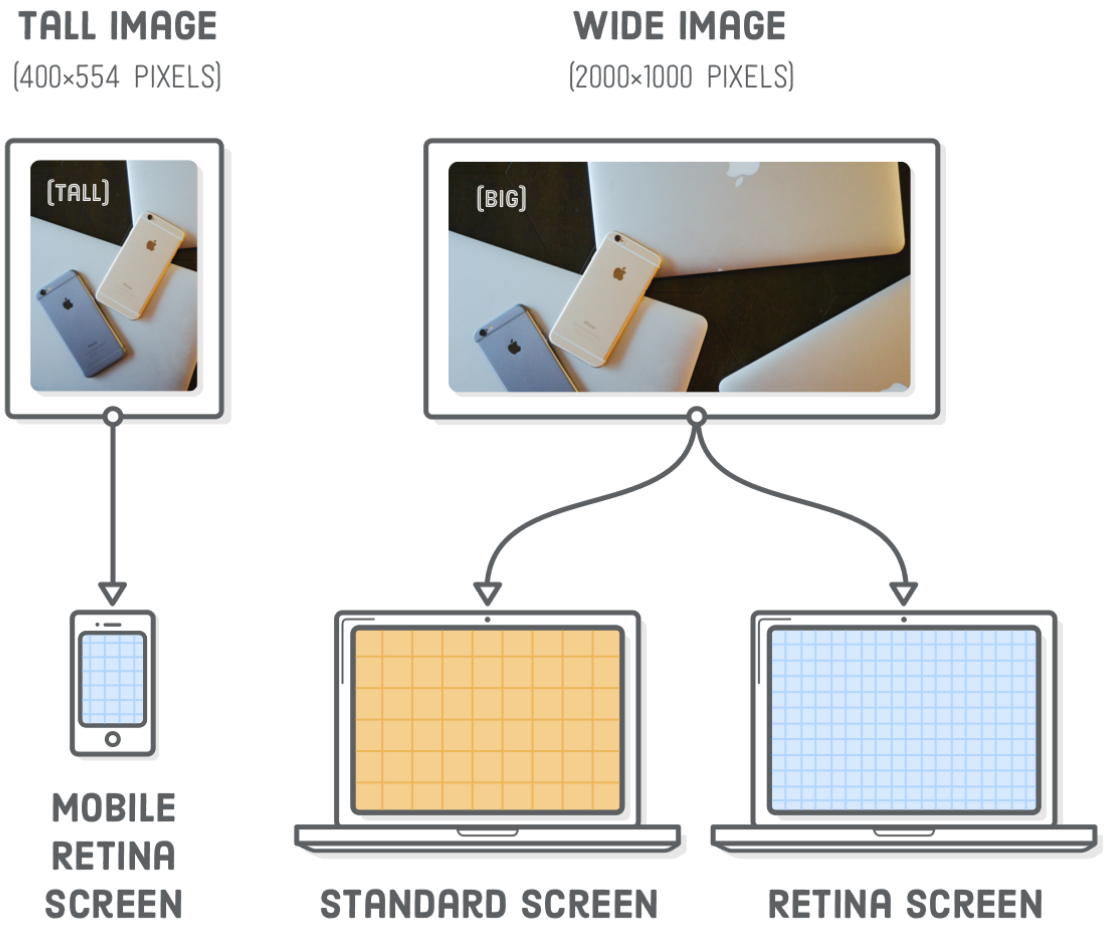
Test avec chrome

Cette technique fonctionne très bien dans Chrome, mais nous ne pouvons pas vraiment dire parce qu'il est d' être intelligent. Chrome utilise toujours la version haute résolution si elle a déjà été mis en cache localement, ce qui signifie que nous ne pouvons pas voir la version basse résolution en faisant simplement la fenêtre du navigateur étroit. Nous devons éviter le cache du navigateur local en ouvrant une nouvelle fenêtre de navigation privée, évitez le chargement photo-big.jpg en faisant la fenêtre très étroite *avant* le chargement de la page.

direction artistique et <picture>

La section précédente est parfaitement acceptable en termes d'optimisation de l'utilisation des données. Nous pourrions nous arrêter là et être très bien, mais nous allons obtenir un peu de fantaisie avec « direction artistique ».

Il vous permet d'optimiser les *mises en page* en envoyant des images complètement différentes à l'utilisateur en fonction de leur appareil. Comparez cela à la section précédente, qui a optimisé la *même* image pour l'appareils différents. Par exemple, notre photo d'en-tête est assez large. Ne serait-il pas génial si on pouvait rogner une version plus grande et présenter que les appareils mobiles au lieu de la version large de bureau?



Pour cela, nous avons besoin des éléments <picture> et <source>. Le premier est juste un container, et celui-ci charge les conditionnellement images basées sur les demandes des médias. Essayez de changer notre élément .header à ce qui suit:

```
<div class='section header'>
  <div class='photo'>
    <picture>
      <source media="(min-width: 401px)" srcset="images/photo-big.jpg">
      <source media="(max-width: 400px)" srcset="images/photo-tall.jpg">
      <img src="images/photo-small.jpg">
    </picture>
  </div>
</div>
```

Conceptuellement, cela est assez similaire à l'aide de media queries en CSS. Dans chaque élément <source>, l'attribut media définit *quand* l'image doit être chargée, et srcset définit *quel* fichier image doit être chargé. L'élément <img> est utilisé comme solution de repli pour les anciens navigateurs. Vous devriez être en mesure de voir la version haut de la photo lorsque vous réduisez la fenêtre de votre navigateur:

Ce niveau de contrôle rendra votre concepteur très heureux, mais le fait est qu'il ne laisse pas le navigateur choisir automatiquement l'image optimale. Cela signifie que nous avons perdu notre optimisation retineade la section précédente: tant que la largeur de l'écran est de 401 pixels ou plus, le navigateur va *toujours* utiliser la haute résolution, la grande image recadrée.

Bien qu'il soit possible de combiner le meilleur des deux mondes, ça se complique vite fait. Notre recommandation est de coller à la version 1x et 2x

srcset pour les images à moins de 600 pixels de large, utilisez le srcset plus la méthode sizes de la section précédente pour de plus grandes photos, et réserver <picture> pour quand vous essayez de faire quelque chose de plus jolie avec votre concepteur.

résumé

Les images responsives peuvent sembler assez compliquées, mais il n'y a vraiment que deux problèmes que nous essayons de résoudre:

- Faire des images qui s’intègrent dans des mises en page mobiles tout en respectant leur taille intrinsèque
- Évitez de faire le téléchargement d'image inutilement grosses

Nous avons accompli la première en faisant des images toujours étirée pour remplir 100% de leur contenant, tout en limitant leur taille avec un style en ligne avec la propriété max-width. Pour ce dernier, nous avons utilisé srcset pour optimiser la résolution de l'écran, srcset ainsi que sizes pour optimiser la largeur de l'appareil, et enfin l'élément <picture> qui permet une optimisation manuel sur le fichier image à afficher.