響應式佈局，你需要知道的一切

前端開發 2022-03-23 09:25

> **來自：campcc_blog**
> **鏈接：https://github.com/campcc/blog/issues/24**

2011年，Google 發布了Android 4.0，在經歷了Cupcake，Donut，Froyo 等多個甜品名稱版本的迭代後，安卓終結了Symbian（塞班）的霸主地位，迅速佔領了手機市場躍居全球第一。同年，騰訊發布了微信開始進軍移動互聯網，阿里也在2013 年宣布ALL IN 無線，隨著智能設備的普及和移動互聯網時代的到來，**響應式佈局**這個詞開始頻繁地出現在Web 設計和開發領域，作為一名優秀的前端攻城獅，**要將極致的用戶體驗和最佳的工程實踐作為探索的目標**): balabala...

所以，響應式佈局，要學。不僅要學，我們還要了解它的前世今生，前置知識，實現手段和原理，以便在實際應用時選取合適的技術方案。

閱讀完本文，你將Get 以下知識點，

- 什麼是響應式設計？
- 什麼是像素，什麼DPR？設備像素與CSS像素的區別是什麼？
- EM，REM 的計算規則是什麼？實際應用中如何選擇？
- 什麼是視口viewport，佈局視口，視覺視口，理想視口的區別？
- 百分比單位和視口單位的計算規則是什麼？
- 彈性盒與網格
- 設備斷點與CSS 媒體查詢
- 響應式佈局的一些最佳實踐

響應式設計

著名的網頁設計師Ehan Marcotte 在2010 年5 月的一篇名為《Responsive Web Design》的個人文章中，首次提到了響應式網站設計。文中講到響應式的概念源自響應式建築設計，即房間或者空間會根據其內部人群數量和流動而變化。

> 最近一門新興的學科“響應式建築(responsive architecture)”開始在探討物理空間根據流動於其中的人進行響應的方法。建築師們通過把嵌入式機器人與可拉伸材料結合的方法，嘗試藝術裝置和

> 可彎曲、伸縮和擴展的牆體結構，達到根據接近人群的情況變化的效果。運動傳感器與氣候控制系統相結合，調整圍繞人們周圍的房間的溫度以及環境照明。已經有公司製造了“智能玻璃技術”，當室內人數達到一定的閥值時，它可以自動變為不透明狀態，為人們提供更多隱私保護

Web 響應式設計的理念與之非常相似，只不過在這裡，

我們**需要適配的不是建築，而是Web 頁面**。

我們期望頁面可以根據用戶的設備環境，比如係統，分辨率，屏幕尺寸等因素，進行自發式調整，提供更適合當前環境的閱讀和操作體驗，對已有和未來即將出現的新設備有一定的適應能力。

這就是響應式設計的理念。那麼是否有對應的方法論呢？

別急，在談及實現之前，我們需要了解一些前置知識，比如像素。

像素

什麼是像素？

像素是圖像中最小的單位，一個不可再分割的點，對應到物理設備上（比如計算機屏幕），就是屏幕上的一個光點。我們常說的分辨率就是長和寬上像素點的個數，比如IPhone X 的分辨率是1125×2436，代表屏幕橫向和縱向分別有1125 和2436 個像素點，這裡的像素是設備像素（Device Pixels）。

1px ≠1像素

實際開發中，你可能發現Iphone X 的設計稿是375×812，WTF？

這裡的375×812 是CSS 像素，也叫虛擬像素，邏輯像素。為什麼我們不使用設備像素呢？

設備像素對應屏幕上的光點，如今的屏幕分辨率已經達到人眼無法區分單個像素的程度了。試想一下，要在IPhone X 寬不到7cm 的屏幕上數出1125 個像素點，想想就讓人頭疼。所以我們在實際開發中通常使用CSS 像素，你眼中的1px 可能對應多個設備像素，比如上面的IPhone X，

```
1 css px = 3 * 3 device px // IPhone X 中，1 个 CSS 像素对应 3*3 的 9 个设备像素点
```

复制代码

而上面這個比值3 就是設備像素比（Device Pixel Ratio，簡稱DPR）。

DPR 可以在瀏覽器中通過JavaScript 代碼獲取，

```
window.devicePixelRatio // IPhone X 中等于 3，IPhone 6/7/8 中等于 2，Web 网页为 1
```
复制代码

像素是一個固定單位，一般我們不會使用固定像素來做響應式佈局，但是你需要了解他。相反，響應式佈局裡經常會用到相對單位，比如EM。

EM

EM 相對於元素自身的 `font-size` ，

```
p {
  font-size: 16px;
  padding: 1em; /* 1em = 16px */
}
```
复制代码

如果元素沒有顯式地設置 `font-size` ，那麼 `1em` 等於多少呢？

這個問題其實跟咱說的 `em` 沒啥關係，這裡跟 `font-size` 的計算規則相關，回顧一下。如果元素沒有設置 `font-size` ，會繼承父元素的 `font-size` ，如果父元素也沒有，會沿著DOM 樹一直向上查找，直到根元素 `html` ，根元素的默認字體大小為16px。

理解了 `EM` ， `REM` 就很簡單了。

REM

REM = Root EM，顧名思義就是相對於根元素的EM。所以它的計算規則比較簡單，

1 `rem` 就等於根元素 `html` 的字體大小，

```
html {
  font-size: 14px;
}

p {
```

```
p {
  font-size: 1rem; /* 1rem = 14px */
}
```
复制代码

所以，如果我們改變根元素的字體大小，頁面上所有使用 `rem` 的元素都會被重繪。

EM 和REM 都是相對單位，我們在做響應式佈局的時候應該如何選擇呢？

根據兩者的特性，

- EM 更適合模塊化的頁面元素，比如Web Components
- REM 則更加方便，只需要設置 `html` 的字體大小，所以REM 的使用更加廣泛一些

實際開發中，設計圖的單位是CSS 像素，我們可以藉助一些工具將px 自動轉換為rem，

下面是一個用 `PostCSS` 插件在基於Webpack 構建的項目中自動轉換的例子，

```
var px2rem = require('postcss-px2rem');

module.exports = {
  module: {
    loaders: [
      {
        test: /\.css$/,
        loader: "style-loader!css-loader!postcss-loader"
      }
    ]
  },
  postcss: function() {
    return [px2rem({remUnit: 75})];
  }
}
```
复制代码

我們已經有響應式單位了，接下來要怎麼讓頁面支持響應式佈局呢？

第一步需要先設置頁面的 `viewport` 。

Viewport

著名的JavaScript 專家Peter-Paul Koch 曾發表過三篇有關 `viewport` 的文章，

- 《A tale of two viewports — part one》
- 《A tale of two viewports — part two》
- 《Meta viewport》

建議先看完上述文章。 `viewport` 最先由Apple 引入，用於解決移動端頁面的顯示問題，通過一個叫 `<meta>` 的DOM 標籤，允許我們可以定義視口的各種行為，比如寬度，高度，初始縮放比例等，

```
<!-- 下面的 meta 定义了 viewport 的宽度为屏幕宽度，单位是 CSS 像素，默认不缩放 -->
<meta name="viewport" content="width=device-width, initial-scale=1">
复制代码
```

Peter-Paul Koch 在文章中將移動瀏覽器的視口分為三種。

layout viewport

為了解決早期Web 頁面在手持設備上的顯示問題，Apple 在IOS Safari 中定義了一個 `viewport meta` 標籤，它可以創建一個虛擬的佈局視口（ `layout viewport` ），這個視口的分辨率接近於PC顯示器。這樣一來，由於兩者的寬度趨近，CSS只需要像在PC上那樣渲染頁面就行，原有的頁面結構也基本不會被破壞。

`layout viewport` 是一個固定的值，由瀏覽器廠商設定，

- IOS 和Android 基本都是980px
- 黑莓（BlackBerry）和IE10 是1024px

可以通過 `document` 獲取佈局視口的寬度和高度，

```
var layoutViewportWidth = document.documentElement.clientWidth
var layoutViewportHeight = document.documentElement.clientHeight
复制代码
```

visual viewport

視覺視口（visual viewport）可以簡單理解為**手持設備物理屏幕的可視區域**。也就是你的手機屏幕，所以不同設備的視覺視口可能不同，有了 `visual viewport` ，我們就可以實現網頁的拖拽和縮放了，為什麼？

因為有了一個承載佈局視口的容器。

試想一下，假如我們現在有一台IPhone 6（375×627），它會在寬為375px的 `visual viewport` 上，創建一個寬為980px的 `layout viewport` ，於是用戶可以在 `visual viewport` 中拖動或縮放網頁來獲得更好的瀏覽體驗。

視覺視口可以通過 `window` 獲取，

```
var visualViewportWidth = window.innerWidth
var visualViewportHeight = window.innerHeight
复制代码
```

idea viewport

我們前面一直在討論Web頁面在移動瀏覽器上的適配問題，但是如果網頁本來就是為移動端設計的，這個時候佈局視口（layout viewport）反而不太適用了，所以我們還需要另一種佈局視口，它的寬度和視覺視口相同，用戶不需要縮放和拖動網頁就能獲得良好的瀏覽體驗，這就是理想視口（idea viewport）。

我們可以通過 `meta` 設置將佈局視口轉換為理想視口，

```
<meta name="viewport" content="width=device-width">
复制代码
```

meta

視口可以通過 `<meta>` 進行設置， `viewport` 元標籤的取值有6種，

- width，正整數| device-width，視口寬度，單位是CSS像素，如果等於device-width，則為理想視口的寬度
- height，正整數| device-width，視口寬度，單位是CSS像素，如果等於device-height，則為理想視口的高度
- initial-scale，0-10，初始縮放比例，允許小數點
- minimum-scale，0-10，最小縮放比例，必須小於等於maximum-scale
- maximum-scale，0-10，最大縮放比例，必須大於等於minimum-scale
- user-scalable，yes/no，是否允許用戶縮放頁面，默認是yes

了解了視口之後，讓我們回到響應式佈局，與視口相關的幾個單位有：**vw，vh，百分比**。

vw，vh，百分比

瀏覽器對於 vw 和 vh 的支持相對較晚，在Android 4.4 以下的瀏覽器中可能沒辦法使用，下面是來自Can I use 完整的兼容性統計數據，

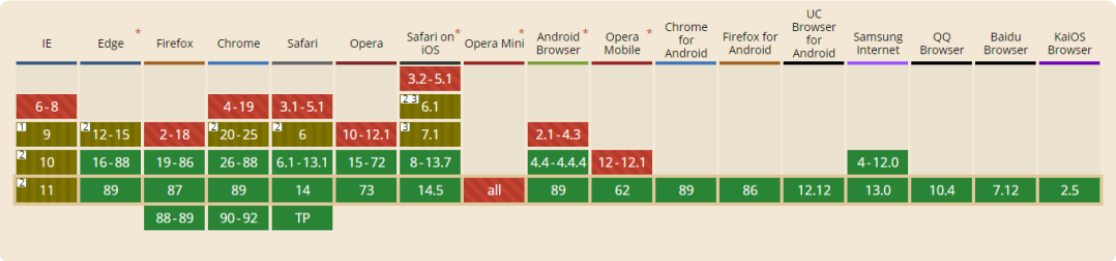

image.png

新生特性往往逃不過兼容性的大坑，但是這並不妨礙我們了解它。

響應式設計裡， vw 和 vh 常被用於佈局，因為它們是相對於視口的，

- vw，viewport width，視口寬度，所以1vw = 1% 視口寬度
- vh，viewport height，視口高度，所以1vh = 1% 視口高度

以IPhone X 為例，vw 和CSS 像素的換算如下，

```
<!-- 假设我们设置视口为完美视口，这时视口宽度就等于设备宽度，CSS 像素为 375px -->
<meta name="viewport" content="width=device-width, initial-scale=1">

<style>
  p {
    width: 10vw; /* 10vw = 1% * 10 * 375px = 37.5px */
  }
</style>
复制代码
```

我們說百分比也可以用來設置元素的寬高，它和 vw ， vh 的區別是什麼？

這裡只需要記住一點，**百分比是相對於父元素的寬度和高度計算的。**

到這裡，相信你已經掌握了響應式佈局裡常用的所有單位。接下來，我們介紹彈性盒和柵格，它們都不是單位，而是一種新的佈局方案。

彈性盒

W3C 在2009 年提出了彈性盒，截止目前瀏覽器對 `FlexBox` 的支持已經相對完善，下面是Can I use FlexBox 完整的兼容性情況，

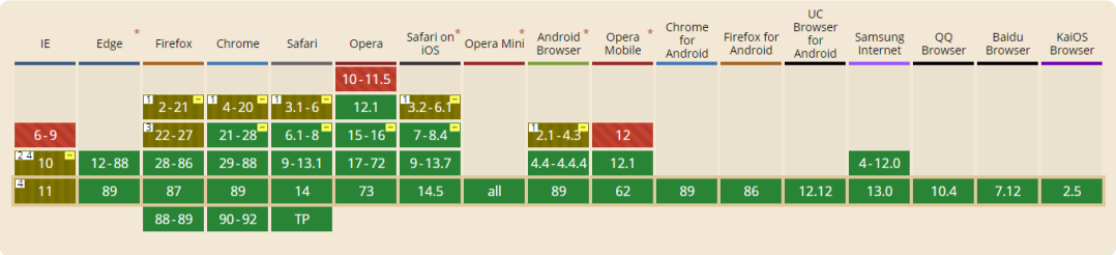

image.png

關於彈性盒模型推薦閱讀這篇文章A Complete Guide to Flexbox。

假設你已經閱讀完並了解了彈性盒模型，響應式佈局中我們需要關注 `FlexBox` 裡的兩個角色：**容器和子元素**。

container

指定 `display` 屬性為 `flex` ，就可以將一個元素設置為 `FlexBox` 容器，我們可以通過定義它的屬性，決定子元素的排列方式，屬性可選值有6 種，

- flex-direction，主軸方向，也就是子元素排列的方向
- flex-wrap，子元素能否換行展示及換行方式
- flex-flow，flex-direction 和flex-wrap 的簡寫形式
- justify-content，子元素在主軸上的對齊方式
- align-items，子元素在垂直於主軸的交叉軸上的排列方式
- align-content，子元素在多條軸線上的對齊方式

items

子元素也支持6 個屬性可選值，

- order，子元素在主軸上的排列順序
- flex-grow，子元素的放大比例，默認 0
- flex-shrink，子元素的縮小比例，默認 1
- flex-basis，分配剩餘空間時，子元素的默認大小，默認auto
- flex，flex-grow，flex-shrink，flex-basis 的簡寫
- align-self，覆蓋容器的align-items 屬性

彈性盒模型佈局非常靈活，屬性值也足夠應對大部分複雜的場景，但 `FlexBox` 基於軸線，只能解決一維場景下的佈局，作為補充，W3C 在後續提出了網格佈局（CSS Grid Layout），網格將容器再度劃

分為“行”和“列”，產生單元格，項目（子元素）可以在單元格內組合定位，所以網格可以看作二維佈局。

網格

關於網格佈局推薦閱讀這篇文章A Complete Guide to Grid。

上述文章非常詳細地介紹了網格的一些基本概念（比如容器和項目，行和列，單元格和網格線等），使用姿勢，注意事項等。作為新興的佈局方案，使用時你需要考慮兼容性是否滿足，

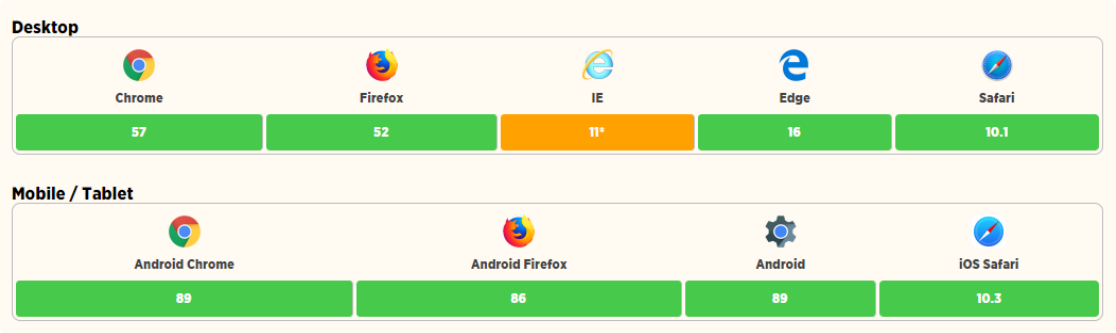

image.png

不過在標準之外，我們可能也正通過其他的一些姿勢在使用網格。如果你關注時下一些比較熱門的UI庫，比如Ant Desgin，Material UI，Element Plus 等，它們以柵格系統的方式實現了對網格部分特性的支持。

UI 庫對 `Grid` 的實現中，通常會使用到媒體查詢，這也是響應式佈局的核心技術。

媒體查詢

媒體查詢（Media Query）是CSS3 規範中的一部分，媒體查詢提供了簡單的判斷方法，允許開發者根據不同的設備特徵應用不同的樣式。響應式佈局中，常用的設備特徵有，

- min-width，數值，視口寬度大於 `min-width` 時應用樣式
- max-width，數值，視口寬度小於 `max-width` 時應用樣式
- orientation， `portrait` | `landscape` ，當前設備的方向

選擇 `min-width` 和 `max-width` 取值的過程，稱為**設備斷點選擇**，它可能取決於產品設計本身，下面是百度Web 生態團隊總結的一套比較具有代表性的設備斷點，

```
/* 很小的设备（手机等，小于 600px） */
@media only screen and (max-width: 600px) { }

/* 比较小的设备（竖屏的平板，屏幕较大的手机等，大于 600px） */
@media only screen and (min-width: 600px) { }

/* 中型大小设备（横屏的平板，大于 768px） */
@media only screen and (min-width: 768px) { }

/* 大型设备（电脑，大于 992px） */
@media only screen and (min-width: 992px) { }

/* 超大型设备（大尺寸电脑屏幕，大于 1200px） */
@media only screen and (min-width: 1200px) { }
```

复制代码

如果你需要對細分屏幕大小進行適配，ResponsiveDesign 站點上的這篇文章Media queries for common device breakpoints 可能會有所幫助。

響應式文字和圖片

相信你已經掌握了響應式佈局的所有知識，接下來我們介紹一些最佳實踐。

文字

大多數用戶閱讀都是從左到右，如果一行文字太長，閱讀下一行時容易出錯，或者用戶只會讀一行文字的前半部分，而略讀後半部分。在上世紀就有研究表明，一行45～90 個英文字符是最好的，對於漢字來說，一行文字合理的數量應該是22～45 個字符。

此外，字體大小對閱讀體驗同樣重要，基本字體一般不小於 `16px` ，行高大於 `1.2em` 。

```
p {
  font-size: 16px;
  line-height: 1.2em; /* 1.2em = 19.2px */
}
```

复制代码

圖片

《高性能網站建設指南》的作者Steve Souders 曾在2013 年的一篇博客中提到：

> 我的大部分性能優化工作都集中在JavaScript 和CSS 上，從早期的Move Scripts to the Bottom 和Put Stylesheets at the Top 規則。為了強調這些規則的重要性，我甚至說過，“JS 和CSS 是頁面上最重要的部分”。幾個月後，我意識到這是錯誤的。圖片才是頁面上最重要的部分。

圖片幾乎佔了網頁流量消耗的60%，雅虎軍規和Google 都將圖片優化作為網頁優化不可或缺的環節，除了圖片性能優化外，**響應式圖片**無疑帶來更好的用戶體驗。

下面是一些響應式圖片的最佳實踐，

1.確保圖片內容不會超出viewport

試想一下，如果圖片固定大小且超出理想視口的寬度，會發生什麼？

內容會溢出視口外，導致出現橫向滾動條對不對，這在移動端是非常不好的瀏覽體驗，因為用戶往往更習慣上下滾動，而不是左右滾動，所以我們需要確保圖片內容不要超出 `viewport` ，可以通過設置元素的最大寬度進行限制，

```
img {
  max-width: 100%;
}
```
复制代码

類似的，相同的規則也應該用於一些其他的嵌入式元素，比如embed，object，video 等。

2. 圖片質量支持響應式

這是一種支持優雅降級的方案，現代瀏覽器已經支持了 `srcset` 和 `sizes` 屬性，對於兼容性不好的瀏覽器，會繼續使用默認 `src` 屬性中的圖片，所以我們可以放心大膽的使用。

- srcset 支持定義幾組圖片和對應的尺寸
- sizes 支持一組媒體查詢條件

```
<!-- 响应式图片 -->
<img
  srcset="example-320w.jpg 320w,
          example-480w.jpg 480w,
          example-800w.jpg 800w"
  sizes="(max-width: 320px) 280px,
```

```
        (max-width: 480px) 440px,
        800px"
  src="example-800w.jpg" alt="An example image">
```

复制代码

如果我們書寫了上面代碼中的圖片，瀏覽器會根據下面的順序加載圖片，

1. 獲取設備視口寬度
2. 從上到下找到第一個為真的媒體查詢
3. 獲取該條件對應的圖片尺寸
4. 加載 `srcset` 中最接近這個尺寸的圖片並顯示

除了上述方式外，我們也可以使用 `HTML5` 標準中的 `picture` 標籤實現類似的效果，

```
<picture>
  <source media="(max-width: 799px)" srcset="example-480w-portrait.jpg">
  <source media="(min-width: 800px)" srcset="example-800w.jpg">
  <img src="example-800w.jpg" alt="An example img">
</picture>
```

复制代码

小結

我們從響應式佈局的設計角度出發，介紹了響應式的設計理念，前置知識（像素，DPR，視口等），相對單位（em，rem，百分比，vw，vh等），佈局方案（FlexBox，Gird）以及媒體查詢等技術，其中不乏很多前輩們的最佳實踐，作為開發者我們應該用這些經驗，以更好地優化不同尺寸大小設備的用戶體驗。

參考鏈接

- Ethan Marcotte, Responsive Web Design
- A tale of two viewports — part one
- A tale of two viewports — part two
- Meta viewport

- PWA應用實戰，2.5 響應式佈局
- 移動前端第一彈：viewport詳解

寫在最後

本文首發於我的博客，才疏學淺，難免有錯誤，文章有誤之處還望不吝指正！

如果有疑問或者發現錯誤，可以在相應的issues 進行提問或勘誤

如果喜歡或者有所啟發，歡迎star，對作者也是一種鼓勵

--- EOF ---

推薦↓↓↓

Web開發

分享Web後端開發技術，分享PHP、Ruby、Python等用於後端網站、後台系統…

公眾號

閱讀原文

喜歡此內容的人還喜歡

10個常用的JS工具庫，80%的項目都在用！

前端開發