DIFERENCIAS ENTRE UNIDADES EM Y REM.

Semejanzas:

- Ambos son unidades de longitud relativas.
- Ambos se basan tamaño de letra "m" de la tipografía declarada en **font-family**.
- Los dos priman la accesibilidad y los tamaños de tipografía y cajas dependerían (en teoría) de las "preferencias/necesidades" del visitante.

Diferencias:

- **Em's**: la base de cálculo del valor computado es el tamaño de fuente/letra del elemento padre o superior.
- **Rem's**: la base es el valor de **font-size** del elemento raíz. En principio debería ser el tamaño de la fuente que el usuario haya marcado en la opción que a tal efecto tiene el navegador, pero en muchos casos, los diseñadores reescriben dicho valor en el selector **<html>**.

Estas son las diferencias más notables entre ambos pero aún hay más;

EM

A la expansión del uso del valor em ayudó la técnica de fijar el valor de **font-size** en el **<html>** o **<body>**. Allá por 2.004 *Richard Rutter* publicó el artículo *"How to size text using ems"*. Basándose en que la mayoría de navegadores tiene el tamaño de la tipografía establecida en 16px echó cuentas y vio que el 62.5% de esos 16px son 10px. Así que para tranquilidad y control del "em" sólo es necesario declarar lo que se convirtió en regla de fe universal:

```
body { font-size: 62.5% }
```

Porque basado en lo anterior, cualquier elemento se podía controlar al píxel declarando ***ems***:

```
header { font-size: 1em } /*Aprox a 10px*/
section { font-size: 1.6em } /*Aprox a 16px*/
footer { font-size: 0.8em } /*Aprox a 8px*/
```

Tradicionalmente el mayor problema de usar el valor ***em*** surge de su propia naturaleza y definición: **la herencia**. Al calcularse su valor sobre el computado de su caja padre, suelen presentarse problemas. El mismo elemento puede tener distinto peso visual (tamaño, márgenes, paddings…) en función de la caja que lo acoja.

A los inconvenientes del uso de em ya tradicionales se han unido otros relacionados con las *media queries* en Responsive Web Design y los dispositivos actuales:

1. El valor de 1em en una regla como **@media (min-width: 32em)** no se ve afectado por el **font-size** del elemento **body**.
2. En las resoluciones actuales de multitud de dispositivos un tamaño de 10px para el texto **font-size:62.5%** es demasiado pequeño e ilegible. El uso de esa declaración automáticamente significa tener que declarar el **font-size** de todos y cada uno de los elementos textuales. Aparte del trabajo de hacerlo inicialmente, complica el mantenimiento del CSS.
3. La tendencia actual, motivada por la disparidad de dispositivos y necesidades, es que cada elemento tenga una presencia basada en el entorno donde se encuentra y escale según él.

REM

La diferencia del valor ***rem*** respecto al ***em*** es, como ya mencionaba, que el valor de cálculo **siempre** es la misma. Con independencia de dónde se encuentre el elemento y de su profundidad en el DOM. Su padre y ancestros no lo modifican.

Lo anterior favorece la relación del peso visual de los distintos elementos entre sí. Así puedes hacer algo como:

```
h1 { font-size: 2.5rem; }
h2 { font-size: 2rem; }
h3 { font-size: 1.5rem; }
p { font-size: 1rem; }
```

Tienes la certeza de que la relación de tamaños y jerarquía entre ellos siempre será la misma, con independencia de que el visitante tenga un tamaño de tipografía establecido en 50px o en 15px.

Pero el valor ***rem*** también tiene sus particularidades. Especialmente en su aplicación a las propiedades que permiten "aligerar" la vista en torno a los elementos. Los espacios vacíos: márgenes, rellenos… e incluso alturas de línea. En ellos hay veces que es más efectivo, práctico y funcional declararlas en función del tamaño de la caja, no del elemento.

Además si tienes que tener presente IE8 o anteriores, la mala noticia es que NO entenderán el valor rem. Aunque siempre podrás utilizarlo con una pizca más de código: declarando primero el valor en ***ems*** y a continuación en ***rems***.