CSS3

REM UNITS

For many years, there have been debates about **which unit of measure should be used** to determine the size of text on web pages.

Common options include:

Pixels (px)
Ems (em)
Percents (%)
And more recently... Rems (rem)

We'll take a **quick look at some of the older options** before talking about Rems.

Font sizing with pixel units

Pixels are one of the **“absolute” length units**.

http://www.w3.org/TR/2013/CR-css3-values-20130730/#lengths

```
h1 { font-size: 32px; }
```

While this method is easy to implement and consistent across devices, it does have **some potential issues**.

Internet Explore 5 through to 8 do not allow users to easily increase or decrease the size of the text. This can present **usability and accessibility issues**.

Recent versions of Internet Explorer include zooming which resolves these issues.

Font sizing with em units

The em unit is one of the **“Font-relative”** length units.

http://www.w3.org/TR/2013/CR-css3-values-20130730/#lengths

Elements specified using em units are **sized relative to the font-size of their parent** element.

For example, if the <body> element has a font-size of 16px and an <h1> element has a font-size of 2em, the actual size of the <h1> will be:

16px x 2 = 32px

```
h1 { font-size: 2em; }
```

Using em units also has some **potential issues**.

If you specify .75em on an <li> element, this could cause **“compounding”** if there are nested <li> elements.

```
li { font-size: .75em; }
```

- item 1
- item 2
    - sub-item 1
    - sub-item 2
    - sub-item 3
    - sub-item 4
- item 3
- item 4

Nested list items are smaller in size

In this example, the parent <li> is reduced in size and then the nested <li> is reduced further.

16px x .75 = 12px (<li>)
12px x .75 = 9px (nested <li>)

Font sizing with percentage units

Percentage values are not a length value. However, **they can be used to define font-size**. They work in a very similar way to em units.

```
h1 { font-size: 200%; }
```

Like em units, percentage units can also cause compounding.

A solution?

So, how can this **compounding issue** be resolved?

Well, first of all, if you are writing font-sizes for things like <p> elements and <li> elements then **you are doing it wrong**!

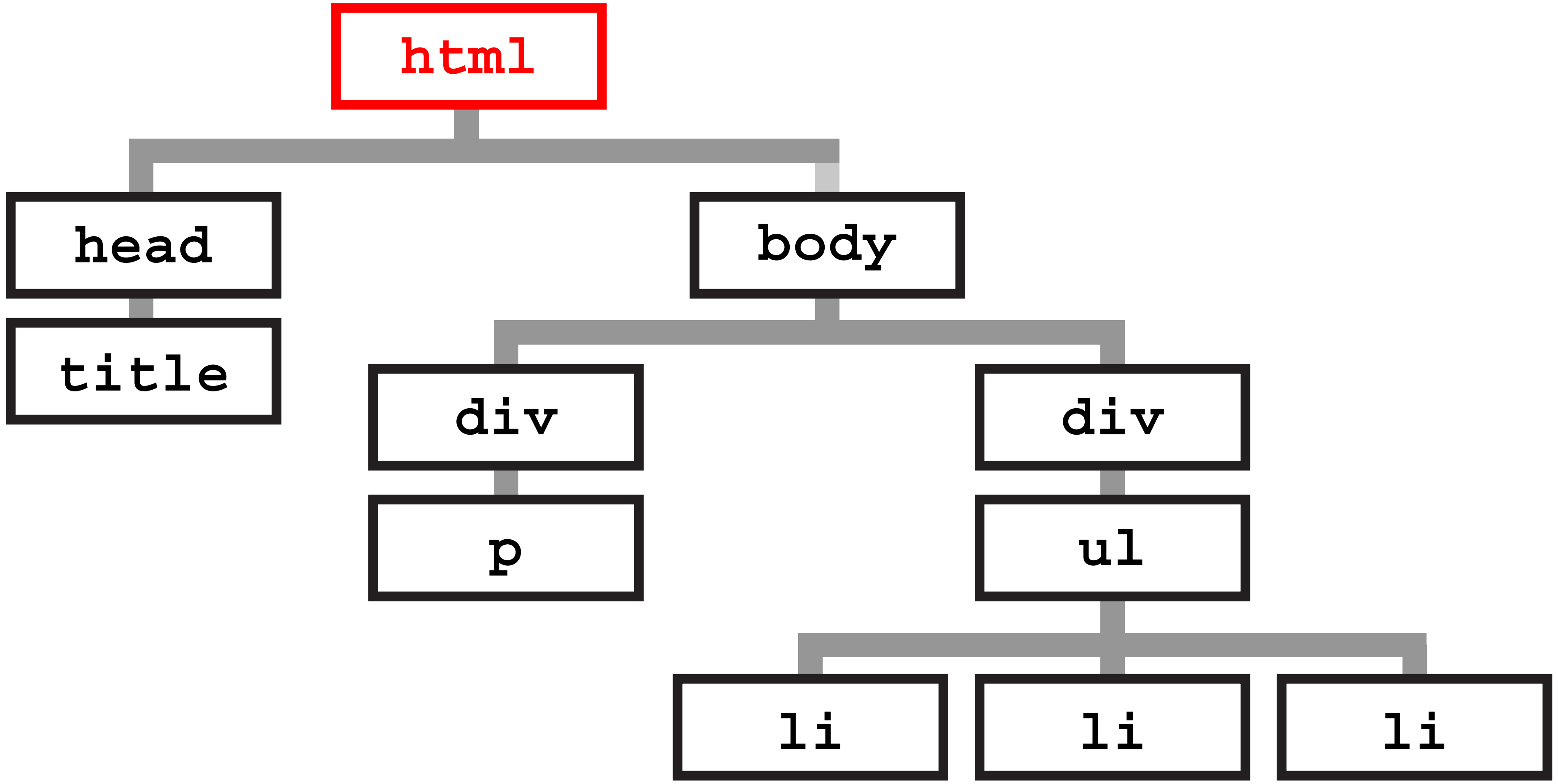
It is better to style the body element to define your base font size and then **let elements such as <p> and <li> take their sizes from the body**.

However, there may be times when you cannot control the situation. **In these cases the rem unit may be the solution**.

What are rems?

The rem unit is officially defined as “equal to the computed value of font-size on the **root element**”.

A root element is an element that has no parent element. In HTML documents, the root element is **the <html> element**.


html
head
title
body
div
p
div
ul
li
li
li

Font sizing with rem units

Rems, like the em unit, are one of the **“Font-relative” length units**.

http://www.w3.org/TR/2013/CR-css3-values-20130730/#lengths

```css
h1 { font-size: 2rem; }
```

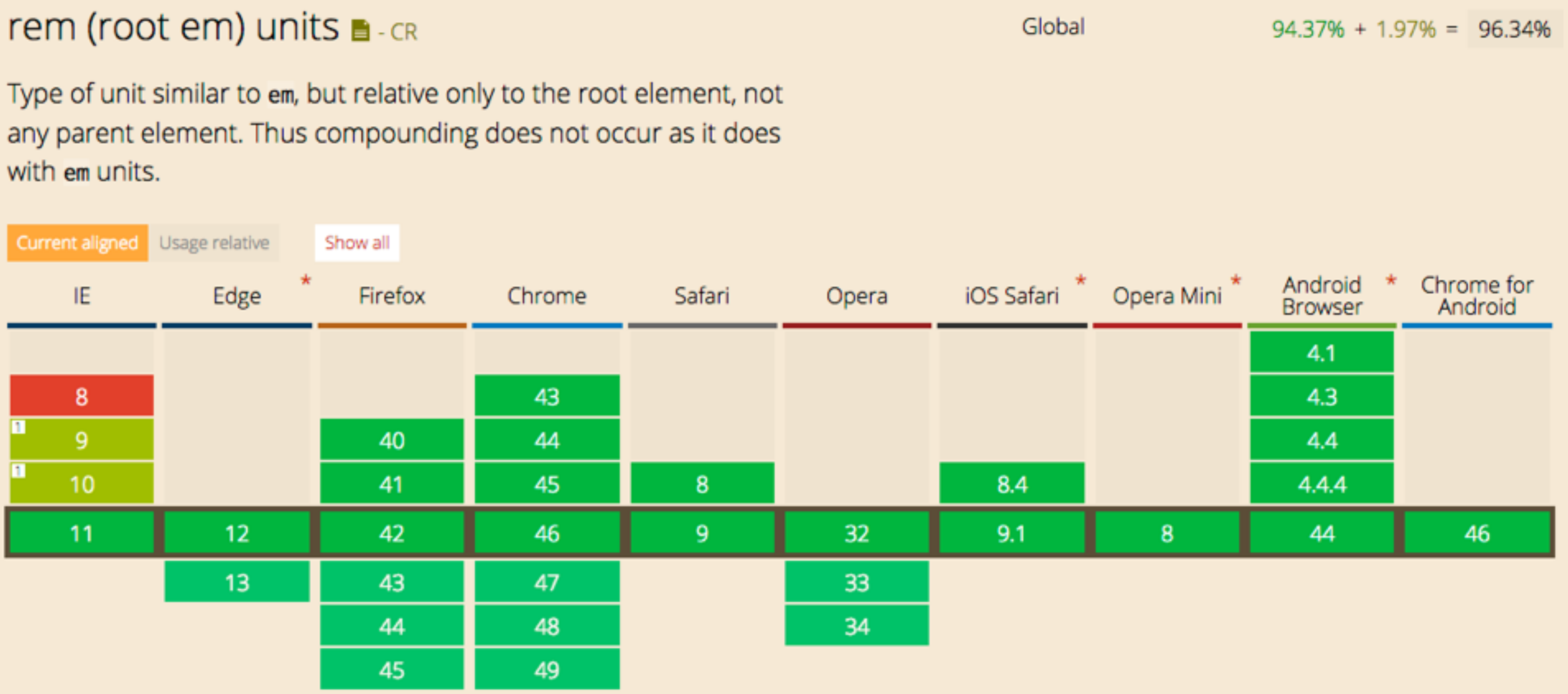
The main difference between the rem unit and the em unit is **the lack of compounding**.

Compounding doesn’t occur with rem units because the element’s font size is based on the root element - **not any parent element**.

So, in the case of the nested <li> element, the **font-sizing issue would not occur** if the <li> had been defined using rems.

```
li { font-size: .75rem; }
```

Browser support



rem (root em) units - CR

Global 94.37% + 1.97% = 96.34%

Type of unit similar to `em`, but relative only to the root element, not any parent element. Thus compounding does not occur as it does with `em` units.

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
								4.1	
8			43					4.3	
[1] 9		40	44					4.4	
[1] 10		41	45	8		8.4		4.4.4	
11	12	42	46	9	32	9.1	8	44	46
	13	43	47		33				
		44	48		34				
		45	49						

But what if you want to use rem units on sites that **need to support older browsers**?

One way to solve the problem is to **specify pixels first, and then rems**.

Only devices that support the rem unit will **apply the second declaration**.

```css
h1 { font-size: 32px; }
h1 { font-size: 2rem; }
```

Or, you could use the “REM-unit-polyfill”. This polyfill will test any browser for REM support and **patch it up if needed**.

https://github.com/chuckcarpenter/REM-unit-polyfill

Russ Weakley

Max Design

Site: maxdesign.com.au

Twitter: twitter.com/russmaxdesign

Slideshare: slideshare.net/maxdesign

Linkedin: linkedin.com/in/russweakley