**Extra notes - CSS - Box Model**


- Dr Nick Hayward


A brief introduction to the CSS Box Model and Resets.

**Contents**



**Intro** A **Cascading Style Sheet**, or CSS, allows us to define stylistic characteristics for our HTML. In effect, it helps us define how our HTML is displayed and rendered. The colours used, font sizes, borders, padding, margins, links, and so on.

**CSS Box Model** This brings us now to a consideration of the CSS **box model**. In effect, a document's attempt to represent each element as a rectangular box. These boxes, and their properties, are determined by the rendering engine of each browser. As a page is rendered, each browser will calculate the size, properties, and position of these required boxes. Properties can include, for example,

- colour, background features, borders, width, height…

So, to help us, CSS describes each of these rectangular boxes using the standard **box model**. This model has been designed to effectively describe an element's required space and content. As you might imagine, each box has a series of edges,

- **margin** edge
- **border** edge
- **padding** edge
- **content** edge

**content** A box's **content area** naturally describes the actual content of the element itself. Properties can often include `color` , `background` , `img` , and this will apply inside the **content** edge. The dimensions will include **content width** and **content-height**.

Content size can be set using the following properties, assuming that the `box-sizing` property remains default,

- `width` , `min-width` , `max-width` , `height` , `min-height` , `max-height`

**padding** A box's **padding area** includes the extent of the padding to the surrounding border. If we set background, colour &c. properties for a content area, this content will extend into the padding. This is why we often consider the padding as extending the content. The padding itself is located in the box's **padding edge**, and its dimensions are the width and height of the **padding-box**.

We can also control the space between the padding and the content edge using the following properties,

- `padding-top` , `padding-right` , `padding-bottom` , `padding-left`
- `padding` (sizes calculated clock-wise - top, right, bottom, left)

An example is as follows,

- CSS box model - padding
  - JSFiddle - CSS Box Model

**border** The **border area** of a box extends this **padding area** to the area containing the borders. In effect, it becomes the area inside the **border edge**, and we can define its dimensions as the width and height of the **border-box**. Of course, this calculated area depends upon the width of the `border` we set in the CSS. We can set the size of our border using the following properties in CSS,

- `border-width`
- `border`

An example is as follows,

- CSS box model - border
  - JSFiddle - CSS Box Model

**margin** Our box's **margin area** can extend this border area with an empty area, which can simply be used to create a defined separation of one element from its neighbours. The dimensions of this area are defined as the width and height of the **margin-box**. We can control the size of our margin area using the following properties,

- `margin-top` , `margin-right` , `margin-bottom` , `margin-left`
- `margin` (sizes calculated clock-wise - top, right, bottom, left)

An example is as follows,

- CSS box model - margin
  - JSFiddle - CSS Box Model

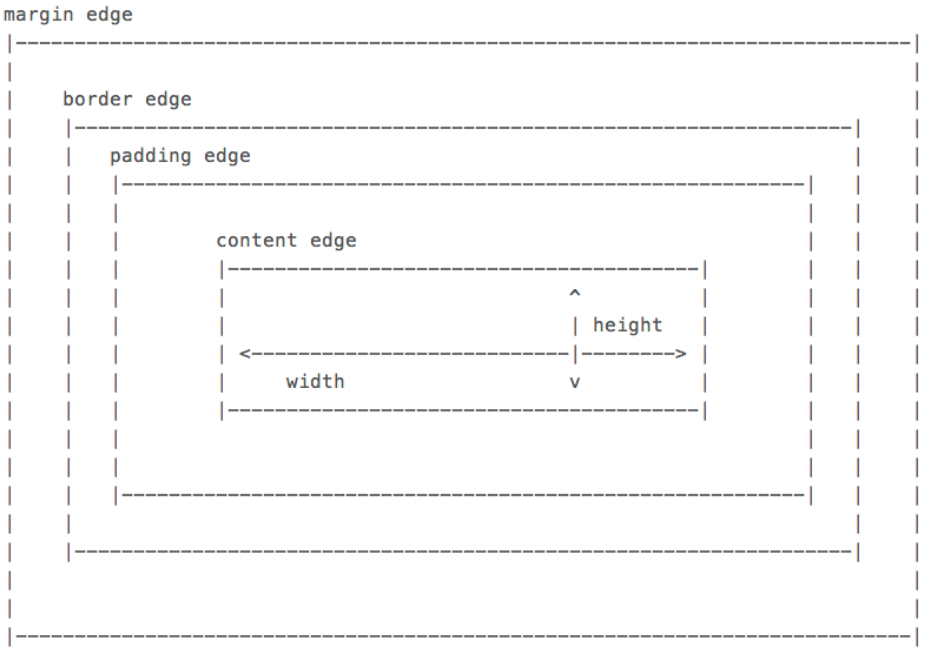


Figure 1: CSS Box Model

**CSS reset options** To help us reduce browser defaults, we can use a CSS reset. Whilst often considered a rather controversial option, at least it used to be, a reset allows us to start from scratch, and customise aspects of the rendering of our HTML documents in browsers.

They were often considered controversial for the following primary reasons,

- **accessibility** - we need to ensure that we cover options for accessibility, such as users navigating solely via a keyboard
- **performance** - another criticism is leveled at rendering performance. Resets are often reliant upon use of the universal selector, the wildcard `*` character, which can often lead to inefficient performance.
- **redundancy** - they can create a lot of redundancy within our styles, instead of simply relying upon some browser defaults

Therefore, use resets with care. One of the notable examples of resets is by Eric Meyer who originally discussed the logic behind this reset in a May 2007 blog post. He provides a stylesheet for resetting default styles, which can then be linked in our own documents. It can be a useful starting point for implementing such resets. Meyer also notes that resets often become part of frameworks and other libraries to help with overall consistency. At least you know you're starting from scratch, a blank canvas so to speak.

### References


- CSS Resets - Eric Meyer
- MDN - CSS
  - CSS box model
- W3 CSS
- W3 Schools - CSS