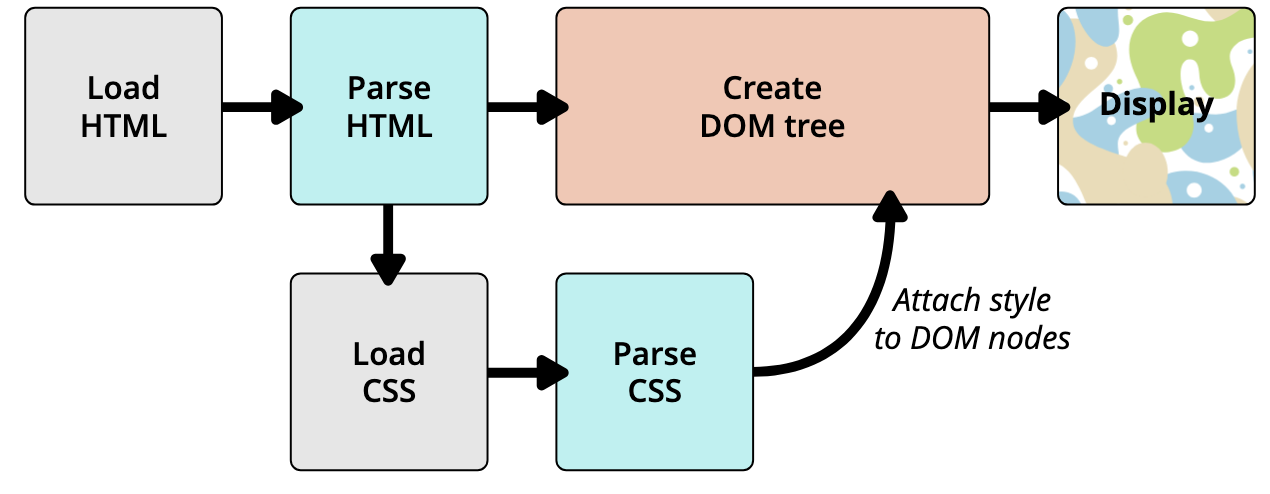
CSS

Overview

Cascading Style Sheets (CSS) are used to define where things are on a webpage and make it look pretty.

CSS process:

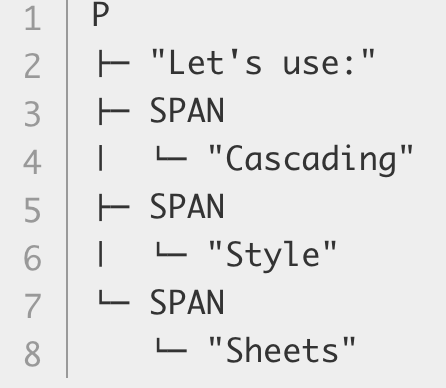
* Browser loads HTML
* Converts HTML into Document Object Model (DOM) - DOM represents the document in the computers memory
* Browser fetches resources that a linked to by the HTML document (images, videos, CSS)
* Browser parses CSS and sorts the different rules by their selector types into different ‘buckets’ (element, class, ID, etc), based on select it finds which rules should be applied to which nodes in the DOM and and attaches style as required
* Render tree is layed out after rules applied
* Visual display of page is shown on screen



DOM

Dom has a tree like structure with each element, attribute and bit of text becoming a node. Nodes are defined by their relationship to other nodes, some being parents and some children/siblings.

For example, the <p> element is a parent and its children would be a text node and three span elements, each span having their own text node children

Selectors

CSS can be applied by specifying the node which it affects using a selector. Nodes can be selected directly using element type or attributes added to an element:

* <element> - selected with element name
* type=”<type”> - selected with <element-name>[type=<type>]
* class=”<name” - selected with . before name
* id=”<anme>” - selected with # before name

The class attribute can be used to identify more than one element, where as id is unique and can only select one element.

*body {*

*font-family: Arial, Helvetica, sans-serif;*

*}*

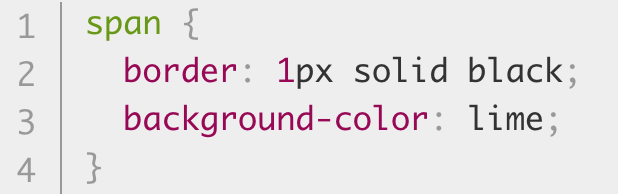
*.job-title {*

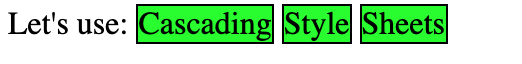
*color: #999999;*

*font-weight: bold;*

*}*

In the case below the selector picks all span elemetns and applies style to those in the document:





Selectors also have the abilty to chain attributes and values, such as a link can be selected then the attribute visited specified, to change the colour of the link once it has been visited.

*a:link, a:visited {*

*color: #fb6542;*

*}*

Incorrect CSS

If the browser encounters CSS it doesn’t understand, it will just fail silently and move on. Therefore, CSS can be used as an enhancement since any incorrect styling will be ignored.

Newer functions can be added after old ones which will then overwrite the old ones if supproted.

* class=”<name” - selected with . before name
* id=”<anme>” - selected with # before name

Normalize

Browsers have default CSS which makes developing applications for all browsers harder. It is common to use a normalise CSS sheet which resets the default CSS of a page. The normalise sheet is the first thing loaded in CSS, so everything else always has the same canvas to work on.

Fonts

Fonts can be added to the server then used in a webpage using the @font-face rule. For example:

Loading:

@font-face {

font-family: ‘<font-name>’;

src: url(<font-url>') format('truetype'); /\* Safari, Android, iOS \*/

}

Usage:

html {

font-family: '<font-name>', Fallback, sans-serif;

}

Variable Fonts

A variable font is a single font file, which behaves like mutliple styles, such as different weighting and italics. Since there is only a single font file for one family, variable fonts can dramatically increase webpage speed and it is supported on most modern browsers.

Download a variable font file, these are generally in ttf, a full system font, and therefore should be convert to WOFF2 to compatibility and compression. To convert to a WOFF2 subset of fonts, it is possible to use pyftsubset in they python library fontTools.

For webpages in english it is typical to include the following unicodes:

* U+0020-004F - standard space & numbers & letters & symbols
* U+00A0-00FF - latin supplement (copyright and fractions)

Therefore, to cut down a full fontset to these vital charaters use:

pyftsubset <font-name>.ttf --unicodes=’U+0020-004F,U+00A0-00FF’ --flavor="woff2" --output-file="<font-name>.woff2"

To use, include using the @font-face rule, set the source and backup for non-compatible browsers, and set a weight range for the font:

@font-face {

font-family: 'RobotoBody';

src: font-url('Roboto-Variable.woff2') format(woff2), font-url('Robot-Regular.woff2') format(woff2);

font-weight: 1 999;

}

Since device width don’t always match up with render widths, the option ‘width’ should always be used to defined the size of the content on the screen.

Animation

CSS animations lets an element move gradually from one style to another. To do this the @keyframes rule is used and binded to an element, which defines at what time each style is changed to.

For example keyframes can be used to create a named animation and be used to change the colour of a div background upon page load:

@keyframes example {

from {background-color: red;}

to {background-color: yellow;}

}

div {

width: 100px;

height: 100px;

background-color: red;

animation-name: example;

animation-duration: 4s;

}

The keyframe is normally used with percentages to define at what the timescale of animation:

@keyframes example {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:0px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}

Animation options:

animation-name: example; - Name of keyframe animation to run

animation-duration: 4s; - Length of time for animation

animation-delay: 2s; - Length of time after event for animation to occur

animation-direction: reverse; - Order of keyframes

* normal
* reverse - keyframes in reverse
* alternate - normal then reverse (loop)
* alternate-reverse - reverse then normal (loop)

animation-iteration-count: <n>; - Number of times to run animation

* n - specific number
* infinite - endless loop

animation-timing-function: <fnc>; - Speed curve of animation timing

* ease - slow, fast, slow
* linear - constant speed
* ease-in - slow, normal
* ease-out - normal, slow
* ease-in-out - slow, normal, slow
* cubic-bezier(n,n,n,n) - custom design of function

animation-fill-mode: <fnc>; - Allows element to retain values from last keyframe

* none - (default) element resets
* forwards - animation retains last keyframe
* backwards - animation retains first keyframe
* both - animation follows rules for both forwards and backwards
* initial - sets value to default
* inherit - inherits from parent

Production keyframes requires vendor specific implementation to allow for maximum compatibility. For example, for one keyframe defintion, the following sections of code will be required:

@-webkit-keyframes hamburger\_crown {

0% { opacity: 0; }

100% { opacity: 1; }

}

@-moz-keyframes hamburger\_crown {

0% { opacity: 0; }

100% { opacity: 1; }

}

@-o-keyframes hamburger\_crown {

0% { opacity: 0; }

100% { opacity: 1; }

}

@keyframes hamburger\_crown {

0% { opacity: 0; }

100% { opacity: 1; }

}

CSS Events

Similar to JavaScript there are various events which CSS can pick up and adjust the properties of elements on.

* :hover - cursor hovers over element
* :focus - reacts to user input
* :active - when element is clicked
* :target - when different element is clicked

Box Sizing

The default box sizing is content box, where border items are not included in sizing of the element, this can lead to confusion when using padding etc. The ‘box-sizing: border-box’ property allows a element to include its padding and border in the total width and height. For example:

With border box: display width => 100px, inner-width => 50px

.box {

box-sizing: border-box;

width: 100px;

padding: 20px 20px;

border: 5px solid red;

}

Without borderbox: display width => 150px, inner-width => 100px

.box {

width: 100px;

padding: 20px 20px;

border: 5px solid red;

}

Therefore if setting an element to 100% width with padding or border, it is important to make it a border box to stop elements from off the page and giving a bad user experience.

Flex

Flexbox - Items

Flex is a type of CSS container which allows its contents to adaptively fit a screen by shrinking, growing, and self aligning. Assigning a containers display to flex makes its children flex items, these items don’t create a new line and will only take up as much width as their content requires by default.

Flex items have three main properties with value 0, 1, auto:

* flex-shrink (default: 1) - When set to 1, this forces flex items to shrink to the size of the flex container, equally proportional to each flex item. This number can be increase for specific items to define how much of the overflow should be proportioned to it. For example, overflow of 200px with three items of flex shrink 1, 2, 1; will have total of 4 and the 200 will be split proportionally, giving the middle element a shrink of 50px ((200/4)\*2)
* flex-grow (default: 0) - Flex grow of 0 means the items will only take up as much space as they require, a value of 1 will make the item will fill as much space as it can in the flex container. Again these values can be increased and divided proportionally between the flex items in the container.
* flex-basis (default: auto) - Equivalent of width, auto means it streches to the size of its content and the ‘width’ property of the element is read. Setting this value to anything other than auto will overwrite any other width property.

Min and max width can be used alongside the above to create an item which ranges between the min and max, shrinking and growing to fit the flex container as required.

Align Items

The ‘align-items’ property of the flex container is used to set the cross axis (vertical) alignment of flex items. Options are:

* strech (default) - strech to fill vertical container
* flex-start - items placed at start of container (top generally, dependant of flex-direction)
* flex-end - items placed at end of container (bottom generally, dependant on flex-direction)
* center - items centered on cross axis
* baseline - items aligned so their baselines (text line) are aligned with the cross axis

Responsive Design

Responsive design is where the is rendered to the user suitably for each size of screen it can be viewed upon, generally splitting app design into mobile and desktop versions.

Responsive websites should be designed for mobile first, since its generally easier to fit content arranged for a smaller device into a bigger viewport compared to vice versa.

@media

To create responsive webapp, there must be different CSS set for different sizes of device screen. The @media rule is used to include a block of CSS only if certain conditions are true.

Media breakpoints (pixel limits) should be based on the size of the content (images etc) included in the site.

Notes:

* Logos and icons should all be in SVG vectors so the scale correctly on any screen size.

The width property is often used to find what size screen the content is being viewed upon. There are several options to choose from to query what type of device (mobile or desktop) is viewing the content and customise it appropiratly:

* width - describes the width of the rendering surface of the output device, ie viewport width of document window
* device-width - describes the width of the output device, meaning the entire screen rather than just the area of the document window.

Mordern mobile devices with high resolution displays scale many real pixels to CSS pixels, therefore images should be provided in high resolution and scaled down using CSS to get the maximum quality. To get the correct size of CSS pixel and websites to adhere to media queries and not scale the width the of viewport, the following meta tag should be added to the head of the page html:

<meta name="viewport" content="width=device-width,initial-scale=1">

Grids

Grids are used to make consistent positioning of responsive applications quick and easy.