Cheatsheets

Colors

Color Name Keywords

Color name keywords can be used to set color property values for elements in CSS.

```
h1 {
  color: aqua;
}

li {
  color: khaki;
}
```

CSS Color Alpha Values

Alpha values determine the transparency of colors in CSS. Alpha values can be set for both RGB and HSL colors by using `rgba()` and `hsla()` and providing a fourth value representing alpha. Alpha values can range between `0.0` (totally transparent) and `1.0` (totally opaque).
The CSS `transparent` value can also be used to create a fully transparent element.

```
.midground {
  background-color: rgba(0, 255, 0, 0.5);
}

.foreground {
  background-color: hsla(34, 100%, 50%,
0.1);
}

.transparent {
  color: transparent;
}
```

CSS Hexadecimal Colors

CSS colors can be represented in *hexadecimal* (or *hex*) notation. Hexadecimal digits can represent sixteen different values using `0` - `9` and `a` - `f`.
Hexadecimal colors are composed of 6 characters–each group of two represents a value between 0 and 255 for red, green, or blue. For example `#ff0000` is all red, no green, and no blue.
When both characters of all three colors are repeated, hex colors can be abbreviated to only three values, so `#0000ff` could also be represented as `#00f`.

```
.red {
  color: #ff0000;
}

.short-blue {
  color: #00f;
}
```

CSS HSL Colors

CSS colors can be declared with the *HSL* color system using `hsl()` syntax. This syntax contains three values: *hue* (the color value itself), *saturation* (intensity), and *lightness*. Hue values range from 0 to 360 while saturation and lightness values are represented as percentages.

```css
.light-blue {
  background-color: hsl(200, 70%, 50%);
}
```

CSS rgb() Colors

CSS colors can be declared with *RGB colors* using `rgb()` syntax.
`rgb()` should be supplied with three values representing red, green, and blue. These values range can from 0 to 255.

```css
.hot-pink {
  color: rgb(249, 2, 171);
}

.green {
  color: rgb(0, 255, 0);
}
```

Display and Positioning

CSS `z-index` property

The CSS `z-index` property specifies how far back or how far forward an element will appear on a web page when it overlaps other elements.
The `z-index` property uses integer values, which can be positive or negative values. The element with the highest `z-index` value will be at the foreground, while the element with the lowest `z-index` value will be at the back.

```
//`element1` will overlap `element2`
.element1 {
  position: absolute;
  z-index: 1;
}

.element2 {
  position: absolute;
  z-index: -1;
}
```

Fixed CSS Positioning

Positioning in CSS provides designers and developers options for positioning HTML elements on a web page. The CSS `position` can be set to `static`, `relative`, `absolute` or `fixed`. When the CSS position has a value of `fixed`, it is set/pinned to a specific spot on a page. The fixed element stays the same regardless of scrolling. The navigation bar is a great example of an element that is often set to `position:fixed;`, enabling the user to scroll through the web page and still access the navigation bar.

```
navbar {
postion : fixed;
 }
```

CSS `display` property

The CSS `display` property determines the type of render block for an element. The most common values for this property are `block`, `inline`, and `inline-block`.
Block-level elements take up the full width of their container with line breaks before and after, and can have their height and width manually adjusted.
Inline elements take up as little space as possible, flow horizontally, and cannot have their width or height manually adjusted.
Inline-block elements can appear next to each other, and can have their width and height manually adjusted.

```
.container1 {
  display: block;
}

.container2 {
  display: inline;
}

.container3 {
  display: inline-block;
}
```

CSS `position: absolute`

The value absolute for the CSS property position enables an element to ignore sibling elements and instead be positioned relative to its closest parent element that is positioned with relative or absolute . The absolute value removes an element entirely from the document flow. By using the positioning attributes top , left , bottom and right , an element can be positioned anywhere as expected.

```
.element {
  position: absolute;
}
```

CSS `position: relative`

The value relative of the CSS position property enables an element to be positioned relative to where it would have originally been on a web page. The offset properties can be used to determine the actual position of the element relative to its original position. Without the offset properties, this declaration will have no effect on its positioning, it will act as the default value static of the position property.

```
.element {
  position: relative;
}
```

CSS `float` property

The CSS float property determines how far left or how far right an element should float within its parent element. The value left floats an element to the left side of its container and the value right floats an element to the right side of its container. For the property float , the width of the container must be specified or the element will assume the full width of its containing element.

```
/* The content will float to the left side
of the container. */
.left {
  float: left;
}

/* The content will float to the right
side of the container. */
.right {
  float: right;
}
```

The CSS `clear` property

The CSS clear property specifies how an element should behave when it bumps into another element within the same containing element.The clear is usually used in combination with elements having the CSS float property. This determines on which sides floating elements are allowed to float.

```
/*This determines that no other elements
within the same containing element are
allowed to float on the left side of this
element.*/
.element {
```

```css
  clear: left;
}

/*This determines that no other elements
within the same containing element are
allowed to float on the right side of this
element.*/
.element {
  clear: right;
}

/*This determines that no elements within
the same containing element are allowed to
float on either side of this element.*/
.element {
  clear: both;
}

/*This determines that other elements
within the same containing element are
allowed to float on both side of this
element.*/
.element {
  clear: none;
}
```

Cheatsheets

The Box Model

CSS Margin Collapse

CSS *margin collapse* occurs when the top and bottom margins of blocks are combined into a single margin equal to the largest individual block margin.
Margin collapse only occurs with vertical margins, not for horizontal margins.

```
/* The vertical margins will collapse to
30 pixels
instead of adding to 50 pixels. */
.block-one {
  margin: 20px;
}

.block-two {
  margin: 30px;
}
```

CSS `auto` keyword

The value `auto` can be used with the property `margin` to horizontally center an element within its container. The `margin` property will take the width of the element and will split the rest of the space equally between the left and right margins.

```
div {
  margin: auto;
}
```

Dealing with `overflow`

If content is too large for its container, the CSS `overflow` property will determine how the browser handles the problem.
By default, it will be set to `visible` and the content will take up extra space. It can also be set to `hidden` , or to `scroll` , which will make the overflowing content accessible via scroll bars within the original container.

```
small-block {
  overflow: scroll;
}
```

Height and Width Maximums/Minimums

The CSS `min-width` and `min-height` properties can be used to set a minimum width and minimum height of an element's box. CSS `max-width` and `max-height` properties can be used to set maximum widths and heights for element boxes.

```
/* Any element with class "column" will be
at most 200 pixels wide, despite the width
property value of 500 pixels. */

.column {
  max-width: 200px;
  width: 500px;
}
```

The `visibility` Property

The CSS visibility property is used to render hidden objects invisible to the user, without removing them from the page. This ensures that the page structure and organization remain unchanged.

```
.invisible-elements {
  visibility: hidden;
}
```

The property `box-sizing` of CSS *box model*

The CSS *box model* is a box that wraps around an HTML element and controls the design and layout. The property box-sizing controls which aspect of the box is determined by the height and width properties. The default value of this property is content-box , which renders the actual size of the element including the content box; but not the paddings and borders. The value border-box , on the other hand, renders the actual size of an element including the content box, paddings, and borders.

```
.container {
  box-sizing: border-box;
}
```

CSS `box-sizing: border-box`

The value border-box of the box-sizing property for an element corresponds directly to the element's total rendered size, including padding and border with the height and width properties.
The default value of the border-box property is content-box . The value border-box is recommended when it is necessary to resize the padding and border but not just the content. For instance, the value border-box calculates an element's height as follows: height = content height + padding + border .

```
#box-example {
  box-sizing: border-box;
}
```

Cheatsheets

Typography

CSS `font-weight` Property

The CSS `font-weight` property declares how thick or thin should be the characters of a text. Numerical values can be used with this property to set the thickness of the text. The numeric scale range of this property is from 100 to 900 and accepts only multiples of 100. The default value is `normal` while the default numerical value is `400`. Any value less than `400` will have text appear lighter than the default while any numerical value greater than the `400` will appear bolder.
In the given example, all the `<p>` elements will appear in a bolder font.

```
/* Sets the text as bolder. */
p {
  font-weight: 700;
}
```

CSS `font-style` property

The CSS `font-style` property determines the font style in which text will appear.
It accepts `italic` as a value to set the font style to italic.

```
.text {
  font-style: italic;
}
```

CSS *@font-face* rule

The CSS *@font-face* rule allows external fonts or font files to be imported directly into stylesheets.The location of the font file must be specified in the CSS rule so that the files can be loaded from that location. This rule also allows locally hosted fonts to be added using a relative file path instead of a web URL.

```
@font-face {
  font-family: 'Glegoo';
  src: url('../fonts/Glegoo-Regular.ttf')
format('truetype');
}
```

CSS Fallback Fonts

The CSS `font-family` property can have multiple fonts declared in order of preference. In this case the fonts following the initial font are known as the *fallback fonts*.
If the initial value of the property `font-family` fails to load to the webpage, the fallback fonts will be used.

```
/* Here `Arial` is the fallback font for
<p> tags */
p {
  font-family: "Helvetica", "Arial";
}
```

The CSS `line-height` property

The CSS line-height property declares the vertical spacing between lines of text. It accepts both unitless numbers as a ratio (eg. 2) and numbers specified by unit as values (eg. 12px) but it does not accept negative numbers. A unitless number is an absolute value that will compute the line height as a ratio to the font size and a unit number can be any valid CSS unit (eg. pixels, percents, ems, rems, etc.). To set the line-height of the <p> elements to 10px , the given CSS declaration can be used.

```
p {
line-height: 10px;
}
```

CSS *Linking fonts*

Linking fonts allow user to use web fonts in the document. They can be imported in an HTML document by using the <link> tag. Once the web font URL is placed within the href attribute, the imported font can then be used in CSS declaration.

```
<head>
  <link
href="https://fonts.googleapis.com/css?
family=Droid%20Serif" rel="stylesheet">
</head>
```

Cheatsheets

Visual Rules

CSS declarations

In CSS, a *declaration* is the key-value pair of a CSS property and its value. CSS declarations are used to set style properties and construct rules to apply to individual or groups of elements. The property name and value are separated by a colon, and the entire declaration must be terminated by a semi-colon.

```
/*
CSS declaration format:
property-name: value;
*/

/* CSS declarations */
text-align: center;
color: purple;
width: 100px;
```

Font Size

The `font-size` CSS property is used to set text sizes. Font size values can be many different units or types such as pixels.

```
font-size: 30px;
```

Background Color

The `background-color` CSS property controls the background color of elements.

```
background-color: blue;
```

`!important` Rule

The CSS `!important` rule is used on declarations to override any other declarations for a property and ignore selector specificity. `!important` rules will ensure that a specific declaration always applies to the matched elements. However, generally it is good to avoid using `!important` as bad practice.

```
#column-one {
  width: 200px !important;
}

.post-title {
  color: blue !important;
}
```

Opacity

The `opacity` CSS property can be used to control the transparency of an element. The value of this property

```
opacity: 0.5;
```

ranges from 0 (transparent) to 1 (opaque).

Font Weight

The font-weight CSS property can be used to set the weight (boldness) of text. The provided value can be a keyword such as bold or normal .

```
font-weight: bold;
```

Text Align

The text-align CSS property can be used to set the text alignment of inline contents. This property can be set to these values: left , right , or center .

```
text-align: right;
```

CSS Rule Sets

A CSS rule set contains one or more selectors and one or more declarations. The selector(s), which in this example is h1 , points to an HTML element. The declaration(s), which in this example are color: blue and text-align: center style the element with a property and value. The rule set is the main building block of a CSS sheet.

```
h1 {
  color: blue;
  text-align: center;
}
```

Setting foreground text color in CSS

Using the color property, foreground text color of an element can be set in CSS. The value can be a valid color name supported in CSS like green or blue . Also, 3 digit or 6 digit color code like #22f or #2a2aff can be used to set the color.

```
p {
  color : #2a2aff ;
}

span {
  color : green ;
}
```

Resource URLs

In CSS, the url() function is used to wrap resource URLs. These can be applied to several properties such as the background-image .

```
background-image:
url("../resources/image.png");
```

Background Image

The background-image CSS property sets the background image of an element. An image URL should be provided in the syntax url("moon.jpg") as the value of the property.

```
background-image: url("nyan-cat.gif");
```

Font Family

The `font-family` CSS property is used to specify the typeface in a rule set. Fonts must be available to the browser to display correctly, either on the computer or linked as a web font. If a font value is not available, browsers will display their default font. When using a multi-word font name, it is best practice to wrap them in quotes.

```
h2 {
  font-family: Verdana;
}

#page-title {
  font-family: "Courier New";
}
```

Color Name Keywords

Color name keywords can be used to set color property values for elements in CSS.

```
h1 {
  color: aqua;
}

li {
  color: khaki;
}
```