codecademy

Learn CSS: Selectors and Visual Rules

Class and ID Selectors

CSS classes can be reusable and applied to many elements. Class selectors are denoted with a period `.` followed by the class name. CSS ID selectors should be unique and used to style only a single element. ID selectors are denoted with a hash sign `#` followed by the id name.

```
/* Selects all elements with
class="column" */
.column {
}

/* Selects element with id="first-item" */
#first-item {
}
```

Groups of CSS Selectors

Match multiple selectors to the same CSS rule, using a comma-separated list. In this example, the text for both `h1` and `h2` is set to red.

```
h1, h2 {
  color: red;
}
```

Selector Chaining

CSS *selectors* define the set of elements to which a CSS rule set applies. For instance, to select all `<p>` elements, the `p` selector can be used to create style rules.

Chaining Selectors

CSS selectors can be chained so that rule sets apply only to elements that match all criteria. For instance, to select `<h3>` elements that also have the `section-heading` class, the selector `h3.section-heading` can be used.

```
/* Select h3 elements with the section-heading class */
h3.section-heading {
  color: blue;
}

/* Select elements with the section-heading and button class */
.section-heading.button {
  cursor: pointer;
}
```

CSS Type Selectors

CSS *type selectors* are used to match all elements of a given type or tag name. Unlike for HTML syntax, we do not include the angle brackets when using type selectors for tag names. When using type selectors, elements are matched regardless of their nesting level in the HTML.

```
/* Selects all <p> tags */
p {
}
```

CSS class selectors

The CSS class selector matches elements based on the contents of their `class` attribute. For selecting elements having `calendar-cell` as the value of the `class` attribute, a `.` needs to be prepended.

```
.calendar-cell {
  color: #fff;
}
```

HTML attributes with multiple values

Some HTML attributes can have multiple attribute values. Multiple attribute values are separated by a space between each attribute.

```
<div class="value1 value2 value3"></div>
```

Selector Specificity

Specificity is a ranking system that is used when there are multiple conflicting property values that point to the same element. When determining which rule to apply, the selector with the highest specificity wins out. The most specific selector type is the ID selector, followed by class selectors, followed by type selectors. In this example, only `color: blue` will be implemented as it has an ID selector whereas `color: red` has a type selector.

```
h1#header {
  color: blue;
} /* implemented */

h1 {
  color: red;
} /* Not implemented */
```

CSS ID selectors

The CSS ID selector matches elements based on the contents of their `id` attribute. The values of `id` attribute should be unique in the entire DOM. For selecting the element having `job-title` as the value of the `id` attribute, a `#` needs to be prepended.

```
#job-title {
  font-weight: bold;
}
```

CSS descendant selector

The CSS *descendant* selector combinator is used to match elements that are descended from another matched selector. They are denoted by a single space between each selector and the descended selector. All matching elements are selected regardless of the nesting level in the HTML.

```
div p { }

section ol li { }
```

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

```
opacity: 0.5;
```

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;
}
```

`<link>` Link Element

The `<link>` element is used to link HTML documents to external resources like CSS files. It commonly uses:

- `href` attribute to specify the URL to the external resource
- `rel` attribute to specify the relationship of the linked document to the current document
- `type` attribute to define the type of content being linked

```
<!-- How to link an external stylesheet
with href, rel, and type attributes -->

<link
href="./path/to/stylesheet/style.css"
rel="stylesheet" type="text/css">
```

Purpose of CSS

CSS, or Cascading Style Sheets, is a language that is used in combination with HTML that customizes how HTML elements will appear. CSS can define styles and change the layout and design of a sheet.

Write CSS in Separate Files

CSS code can be written in its own files to keep it separate from the HTML code. The extension for CSS files is **.css**. These can be linked to an HTML file using a `<link>` tag in the `<head>` section.

```
<head>
  <link href="style.css" type="text/css"
rel="stylesheet">
</head>
```

Write CSS in HTML File

CSS code can be written in an HTML file by enclosing the code in `<style>` tags. Code surrounded by `<style>` tags will be interpreted as CSS syntax.

```
<head>
  <style>
    h1 {
      color: blue;
    }
  </style>
</head>
```

Inline Styles

CSS styles can be directly added to HTML elements by using the `style` attribute in the element's opening tag. Each style declaration is ended with a semicolon. Styles added in this manner are known as *inline styles*.

```
<h2 style="text-align: center;">Centered
text</h2>

<p style="color: blue; font-size:
18px;">Blue, 18-point text</p>
```

Separating HTML code from CSS code

It is common practice to separate content code in HTML files from styling code in CSS files. This can help make the code easier to maintain, by keeping the syntax for each file separate, and any changes to the content or styling can be made in their respective files.

Print Share

0

codecademy

Learn CSS: The Box Model

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 `box-sizing` 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;
}
```

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;
}
```

codecademy

Learn CSS: 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 {
  position: 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 {
  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;
}
```