Şcoala informală de IT®

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language.

What Is CSS?

- Most web pages are made from **HTML**, or hypertext markup language. This is the standard way to decorate plain web text with fonts, colors, graphic doodads, and hyperlinks (clickable text that magically transports the user somewhere else). But websites can get really big. When that happens, **HTML** is a very hard way to do a very easy thing. **CSS (cascading style sheets)** can make decorating web sites easy again!

CSS can be easily separated from HTML

- Think of **CSS** as a kind of computer dress code. **CSS** mainly does just one thing: it describes how web pages should look. Even better, **CSS** can be easily separated from **HTML**, so that the dress code is easy to find, easy to modify, and can rapidly change the entire look of your web site. Like a dress code at school, you can change your **CSS** and the look of your students will change with it. Style sheets allow you to rapidly alter entire websites as you please, just like a fashion craze allows people to change with the times yet remain the same people.

CSS cascades

- A really neat thing about **CSS**, is that it cascades. Each style you define adds to the overall theme, yet you can make the most recent style override earlier styles. For example, with **CSS** we can start by saying we want all of our text 12px (12 units) high. Later we can say we want it to be red, too. Still later, we can tell it we want one phrase to be in bold or italics, or blue rather than red.

Types of CSS

CSS comes in three types:

- In a separate file (**external**)
- At the top of a web page document (**internal**)
- Right next to the text it decorates (**inline**)

  - **External** style sheets are separate files full of **CSS** instructions (with the file extension .css). When any web page includes an external stylesheet, its look and feel will be controlled by this **CSS** file (unless you decide to override a style using one of these next two types). This is how you change a whole website at once. And that's perfect if you want to keep up with the latest fashion in web pages without rewriting every page!

Types of CSS

- **Internal** styles are placed at the top of each web page document, before any of the content is listed. This is the next best thing to external, because they're easy to find, yet allow you to 'override' an external style sheet -- for that special page that wants to be a nonconformist!
- **Inline** styles are placed right where you need them, next to the text or graphic you wish to decorate. You can insert inline styles anywhere in the middle of your HTML code, giving you real freedom to specify each web page element. On the other hand, this can make maintaining web pages a real chore!

Inline CSS

- An inline CSS is used to apply a unique style to a single HTML element.
- An inline CSS uses the style attribute of an HTML element.
- This example sets the text color of the <h1> element to blue:

```
<h1 style="color:blue;">This is a Blue Heading</h1>
```

Internal CSS

- An internal CSS is used to define a style for a single HTML page.
- An internal CSS is defined in the <head> section of an HTML page, within a <style> element:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
    <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    </body>
</html>
```

External CSS

- An external style sheet is used to define the style for many HTML pages.
- **With an external style sheet, you can change the look of an entire web site, by changing one file!**
- To use an external style sheet, add a link to it in the <head> section of the HTML page:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet"
href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Școala informală de IT®

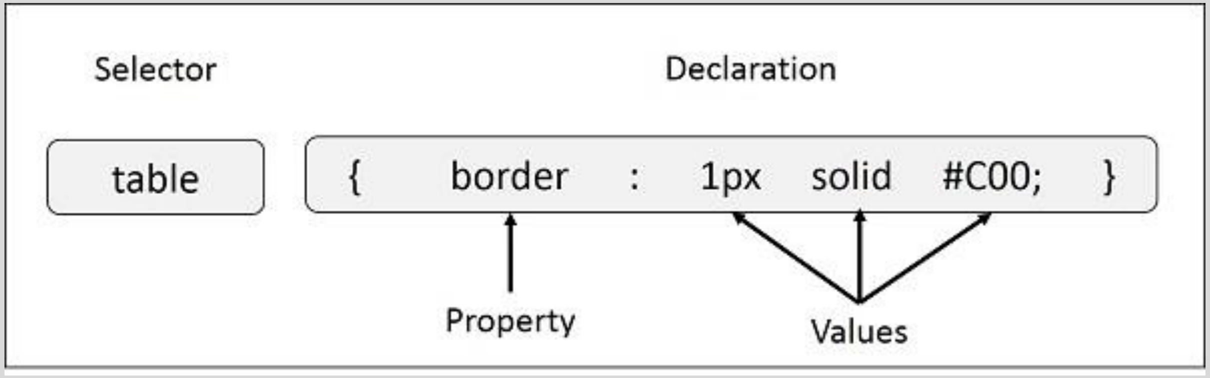
`<link rel="stylesheet" href="styles.css">`

- An external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.
- Here is how the "**styles.css**" looks:

```
body {
            background-color: powderblue;
}
h1 {
            color: blue;
}
p {
            color: red;
}
```

Școala informală de IT®

CSS - Syntax



A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document.

A style rule is made of three parts

- **Selector** − A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.
- **Property** - A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.
- **Value** - Values are assigned to properties. For example, color property can have value either red or #F1F1F1 etc.

```
selector { property: value }        table{ border :1px solid #C00; }
```

The Type Selectors

This is the same selector we have seen above. Again, one more example to give a color to all level 1 headings:

```
h1 {
   color: #36CFFF;
}
```

The Descendant Selectors

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to <em> element only when it lies inside <ul> tag.

```
ul em {
   color: #000000;
}
```

The Class Selectors

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black {
   color: #000000;
}
```

This rule renders the content in black for every element with class attribute set to black in our document. You can make it a bit more particular. For example:

```
h1.black {
   color: #000000;
}
```

This rule renders the content in black for only <h1> elements with class attribute set to black.

Multiple classes on one element

You can apply more than one class selectors to given element. Consider the following example:

```
<p class="center bold">
   This para will be styled by the classes center and bold.
</p>
```

Școala informală de IT®

The ID Selectors

You can define style rules based on the id attribute of the elements. All the elements having that id will be formatted according to the defined rule.

```
#black {
   color: #000000;
}
```

This rule renders the content in black for every element with id attribute set to black in our document. You can make it a bit more particular. For example

```
h1#black {
   color: #000000;
}
```

The **true power of id selectors** is when they are used as the foundation for descendant selectors, For example:

```
#black h2 {
   color: #000000;
}
```

In this example all level 2 headings will be displayed in black color when those headings will lie with in tags having id attribute set to black.

Școala informală de IT ®

The Child Selectors

You have seen the descendant selectors. There is one more type of selector, which is very similar to descendants but have different functionality. Consider the following example

```
body > p {
   color: #000000;
}
```

This rule will render all the paragraphs in black if they are direct child of <body> element. Other paragraphs put inside other elements like <div> or <td> would not have any effect of this rule.

The Attribute Selectors

You can also apply styles to HTML elements with particular attributes. The style rule below will match all the input elements having a type attribute with a value of text

```
input[type = "text"]{
   color: #000000;
}
```

The advantage to this method is that the <input type = "submit" /> element is unaffected, and the color applied only to the desired text fields.

Rules applied to attribute selectors

- **p[lang]** - Selects all paragraph elements with a lang attribute.
- **p[lang="fr"]** - Selects all paragraph elements whose lang attribute has a value of exactly "fr".
- **p[lang~="fr"]** - Selects all paragraph elements whose lang attribute contains the word "fr".
- **p[lang|="en"]** - Selects all paragraph elements whose lang attribute contains values that are exactly "en", or begin with "en-".

Multiple Style Rules

You may need to define multiple style rules for a single element. You can define these rules to combine multiple properties and corresponding values into a single block as defined in the following example

```
h1 {
    color: #36C;
    font-weight: normal;
    letter-spacing: .4em;
    margin-bottom: 1em;
    text-transform: lowercase;
}
```

Here all the property and value pairs are separated by a semi colon (;). You can keep them in a single line or multiple lines. For better readability we keep them into separate lines.

Grouping Selectors

You can apply a style to many selectors if you like. Just separate the selectors with a comma, as given in the following example

```
h1, h2, h3 {
   color: #36C;
   font-weight: normal;
   letter-spacing: .4em;
   margin-bottom: 1em;
   text-transform: lowercase;
}
```

This define style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.

Please follow this link and read more about CSS Selectors:
Basic CSS selectors

CSS - Pseudo Classes

CSS pseudo-classes are used to add special effects to some selectors. You do not need to use JavaScript or any other script to use those effects. A simple syntax of pseudo-classes is as follows

```
selector:pseudo-class {property: value}
```

CSS classes can also be used with pseudo-classes

```
selector.class:pseudo-class {property: value}
```

The most commonly used pseudo-classes are as follows

- **:link** Use this class to add special style to an unvisited link.
- **:visited** Use this class to add special style to a visited link.
- **:hover** Use this class to add special style to an element when you mouse over it.
- **:active** Use this class to add special style to an active element.
- :focus Use this class to add special style to an element while the element has focus.
- **:first-child** Use this class to add special style to an element that is the first child of some other element.
- **:lang** Use this class to specify a language to use in a specified element.

Please follow this link and read more about CSS Pseudo-classes:

[CSS Pseudo-classes]

Școala informală de IT ®

CSS - Pseudo Elements

CSS pseudo-elements are used to add special effects to some selectors. You do not need to use JavaScript or any other script to use those effects. A simple syntax of pseudo-element is as follows

```
selector:pseudo-element {property: value}
```

CSS classes can also be used with pseudo-elements

```
selector.class:pseudo-element {property: value}
```

The most commonly used pseudo-elements are as follows

- **:first-line** Use this element to add special styles to the first line of the text in a selector.
- **:first-letter** Use this element to add special style to the first letter of the text in a selector.
- **:before** Use this element to insert some content before an element.
- **:after** Use this element to insert some content after an element.

Please follow this link and read more about CSS Pseudo Elements:

CSS Pseudo Elements

CSS - @ Rules

- The **@import:** rule imports another style sheet into the current style sheet.
- The **@charset** rule indicates the character set the style sheet uses.
- The **@font-face** rule is used to exhaustively describe a font face for use in a document.
- The **!important** rule indicates that a user-defined rule should take precedence over the author's style sheets.

Note: **DO NOT USE !important !!!!**

- ***Only*** *use !important on page-specific CSS that overrides foreign CSS (from external libraries, like Bootstrap or normalize.css).*
- ***Never*** *use !important when you're writing a plugin/mashup.*
- ***Never*** *use !important on site-wide CSS.*

CSS Specificity

Specificity is the means by which browsers decide which CSS property values are the most relevant to an element and, therefore, will be applied. Specificity is based on the matching rules which are composed of different sorts of CSS selectors.

How is specificity calculated?
Specificity is a weight that is applied to a given CSS declaration, determined by the number of each selector type in the matching selector. When multiple declarations have equal specificity, the last declaration found in the CSS is applied to the element. Specificity only applies when the same element is targeted by multiple declarations. As per CSS rules, directly targeted elements will always take precedence over rules which an element inherits from its ancestor. More info here: css specificity

Școala informală de IT ®

Using CSS media queries

Media queries are useful when you want to apply CSS styles depending on a device's general type (such as print vs. screen), specific characteristics (such as the width of the browser viewport, or environment (such as ambient light conditions). With the huge variety of internet-connected devices available today, media queries are a vital tool for building websites and apps that are robust enough to work on whatever hardware your users have.

For more details on using media queries please follow this link:
[CSS media queries]

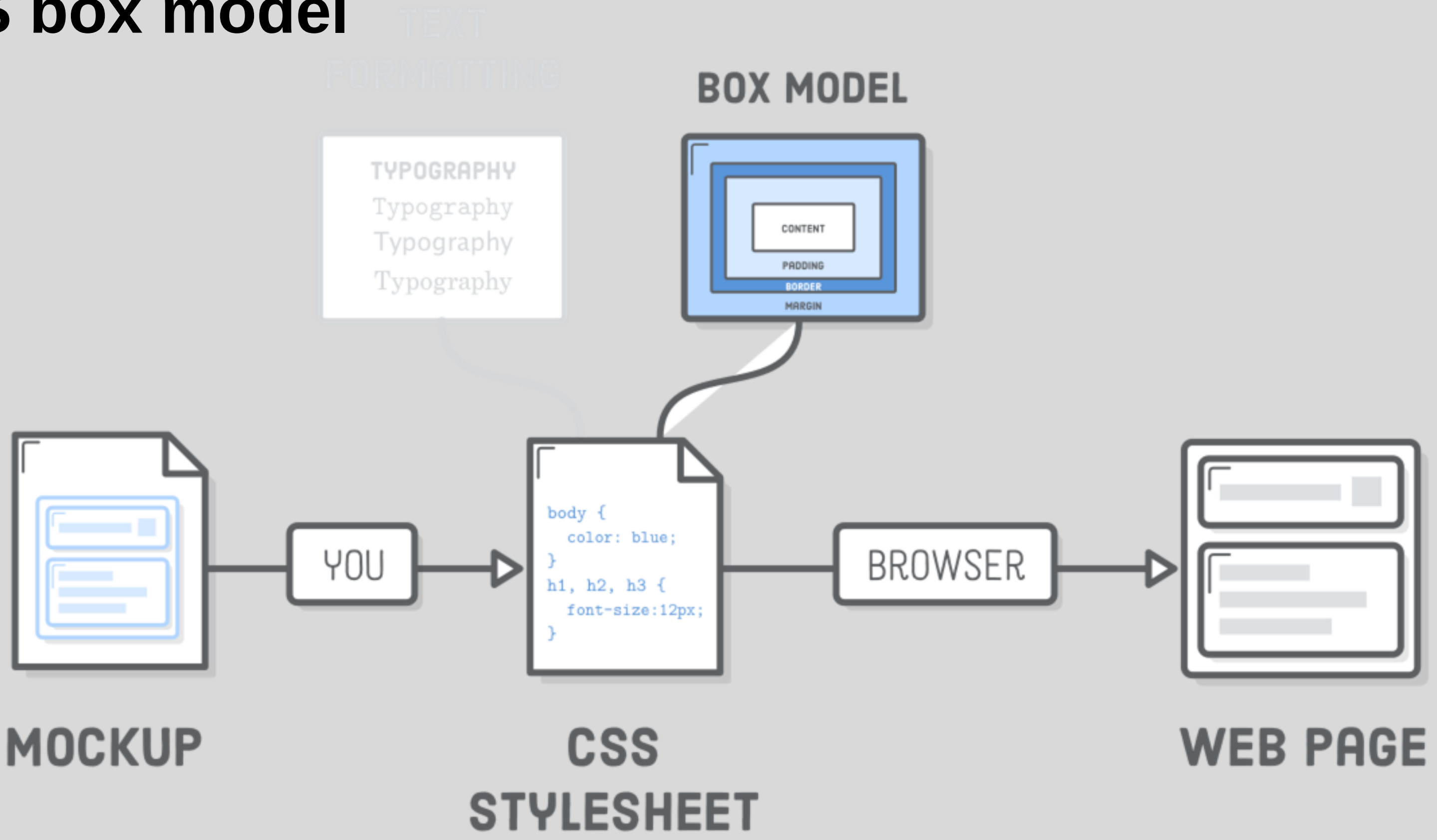
Școala informală de IT®

CSS box model

The “**CSS box model**“ is a set of rules that define how every web page on the Internet is rendered. CSS treats each element in your HTML document as a “box” with a bunch of different properties that determine where it appears on the page. So far, all of our web pages have just been a bunch of elements rendered one after another. The box model is our toolkit for customizing this default layout scheme.

A big part of your job as a web developer will be to apply rules from the CSS box model to turn a design mockup into a web page.

CSS box model



```
body {
  color: blue;
}
h1, h2, h3 {
  font-size:12px;
}
```

Școala informală de IT®

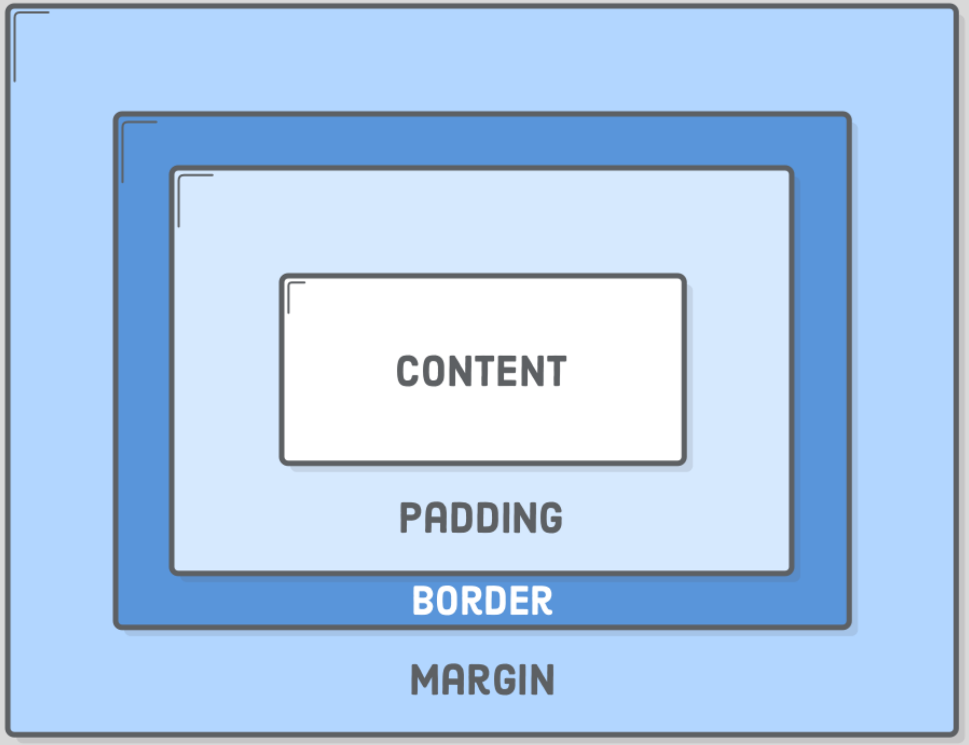
Content, padding, border, and margin

The “CSS box model” is a set of rules that determine the dimensions of every element in a web page. It gives each box (both inline and block) four properties:

- **Content** – The text, image, or other media content in the element.
- **Padding** – The space between the box’s content and its border.
- **Border** – The line between the box’s padding and margin.
- **Margin** – The space between the box and surrounding boxes.

Together, this is everything a browser needs to render an element’s box. The content is what you author in an HTML document, and it’s the only one that has any semantic value (which is why it’s in the HTML). The rest of them are purely presentational, so they’re defined by CSS rules.

Content, padding, border, and margin



[Please follow this link for a detailed explanation of the css box model]

CSS Grid Layout

CSS Grid Layout excels at dividing a page into major regions, or defining the relationship in terms of size, position, and layer, between parts of a control built from HTML primitives.

Like tables, grid layout enables an author to align elements into columns and rows. However, many more layouts are either possible or easier with CSS grid than they were with tables. For example, a grid container's child elements could position themselves so they actually overlap and layer, similar to CSS positioned elements.

CSS Grid Layout

The below example shows a three column track grid with new rows created at a minimum of 100 pixels and a maximum of auto. Items have been placed onto the grid using line-based placement.

```
<div class="wrapper">
  <div class="one">One</div>
  <div class="two">Two</div>
  <div class="three">Three</div>
  <div class="four">Four</div>
  <div class="five">Five</div>
  <div class="six">Six</div>
</div>
```

CSS Grid Layout

```
.three {
  grid-column: 1;
  grid-row: 2 / 5;
}
.four {
  grid-column: 3;
  grid-row: 3;
}
.five {
  grid-column: 2;
  grid-row: 4;
}
.six {
  grid-column: 3;
  grid-row: 4;
}
```

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 10px;
  grid-auto-rows: minmax(100px, auto);
}
.one {
  grid-column: 1 / 3;
  grid-row: 1;
}
.two {
  grid-column: 2 / 4;
  grid-row: 1 / 3;
}
```

Please follow this link for a detailed explanation of the grid layout

The Grid container

We create a grid container by declaring display: grid or display: inline-grid on an element. As soon as we do this all direct children of that element will become grid items. In this example, I have a containing div with a class of wrapper, inside are five child elements.

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

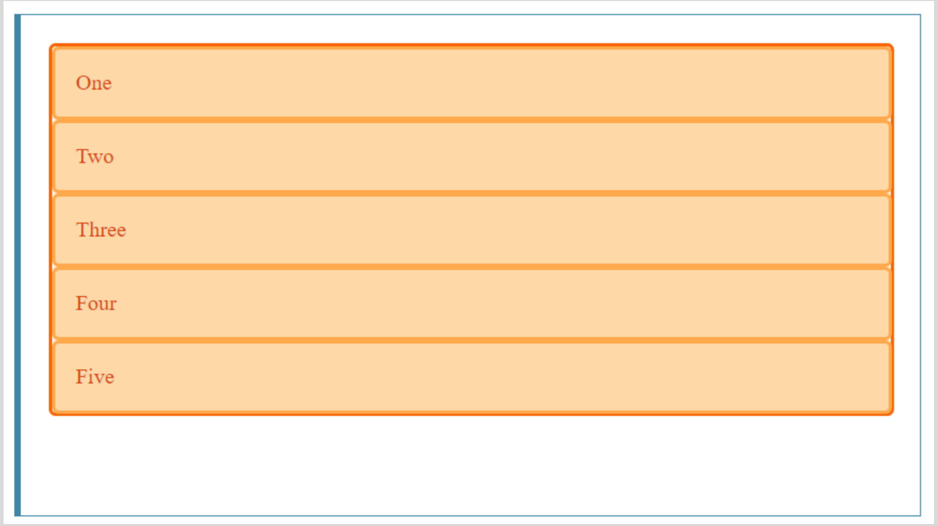
The Grid container

We make the .wrapper a grid container.

```
.wrapper {
  display: grid;
}
```

Please follow this link for a detailed explanation of the grid container

Result:



Școala informală de IT ®

Before starting to style anything please visit CSS Zen Garden and learn as much as you can from there.

The CSS Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the time-honored techniques in new and invigorating fashion. Become one with the web.

CSS Zen Garden - Resource Guide
CSS Zen Garden - Design List

Advanced CSS

Școala informală de IT®

HTML Normal Layout Flow

- HTML Normal Layout Flow is the way the browser lays out HTML pages by default.
- HTML Elements are displayed in the order in which they appear in the source code, stacked on top of one another.
- We can override this behavior using several techniques:

1. CSS Float Property
2. CSS Position property
3. CSS Display property

CSS Float Property

The **float** property in CSS is used for positioning and layout on web pages.
Float Values:

- **none**: the element does not float. This is the initial value.
- **left**: floats the element to the left of its container.
- **right**: floats the element to the right of its container.
- **inherit**: the element inherits the float direction of its parent.

```
img  {
        float: right;
}
```

***Note**: An element that is floated is automatically **display: block;**

CSS Float Property Explained

CSS Position property

The position property can help you manipulate the location of an element, for example:

```
img  {
          position: relative;
     top: 20px;
}
```

The **position** CSS property specifies how an element is positioned in a document. The top, right, bottom, and left properties determine the final location of positioned elements.

CSS Position Property Explained

CSS Display property

The **display** CSS property specifies the type of rendering box used for an element. In HTML, default display property values are taken from behaviors described in the HTML specifications or from the browser/user default stylesheet. The default value in XML is inline, including SVG elements.

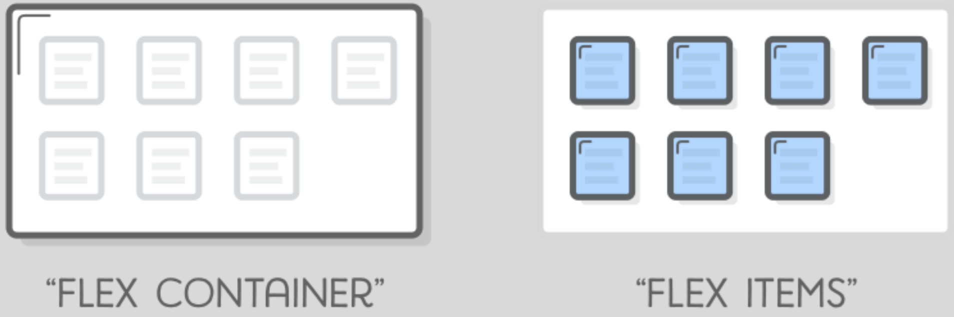
In addition to the many different display box types, the value none lets you turn off the display of an element; when you use none, all descendant elements also have their display turned off. The document is rendered as though the element doesn't exist in the document tree.

CSS Display Property Explained

Școala informală de IT®

Flexbox overview

Flexbox uses two types of boxes that we've never seen before: "flex containers" and "flex items". The job of a flex container is to group a bunch of flex items together and define how they're positioned.



Every HTML element that's a direct child of a **flex container** is an "item". **Flex items** can be manipulated individually, but for the most part, it's up to the container to determine their layout. The main purpose of flex items are to let their **container** know how many things it needs to position.

Școala informală de IT®

Flexbox overview

As with float-based layouts, defining complex web pages with flexbox is all about nesting boxes. You align a bunch of flex items inside a container, and, in turn, those items can serve as flex containers for their own items. As you work through the examples in this chapter, remember that the fundamental task of laying out a page hasn't changed: we're still just moving a bunch of nested boxes around.

Please follow this tutorial about Flexbox:
A friendly tutorial for modern CSS layouts - Flexbox

THANKS.

Școala informală de IT®