SITE LAYOUT.

BEST PRACTICES.

CROSS BROWSER IMPLEMENTATION.

by Anastasiia Derkach

softserve

SITE LAYOUT

What does it mean?

CSS page layout techniques allow us to take elements contained in a web page and control where they are positioned relative to their default position in normal layout flow, the other elements around them, their parent container, or the main viewport/ window.

Why it's important?

Each technique has its uses, advantages, and disadvantages, and no technique is designed to be used in isolation. By understanding what each method is designed for you will be in a good place to understand which is the best layout tool for each task.

softserve

TYPES OF SITE LAYOUT

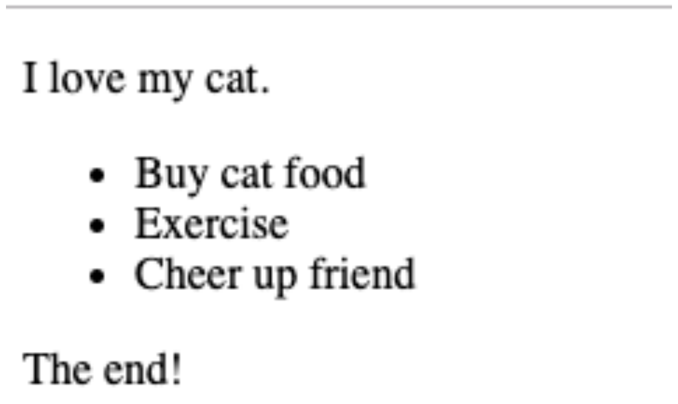
- Normal flow
- The display property
- Flexbox
- Grid
- Floats
- Positioning
- Table layout
- Multiple-column layout

softserve

NORMAL FLOW

Normal flow is how the browser lays out HTML pages by default when you do nothing to control page layout.

```
<p>I love my cat.</p>

<ul>
  <li>Buy cat food</li>
  <li>Exercise</li>
  <li>Cheer up friend</li>
</ul>

<p>The end!</p>
```



I love my cat.

- Buy cat food
- Exercise
- Cheer up friend

The end!

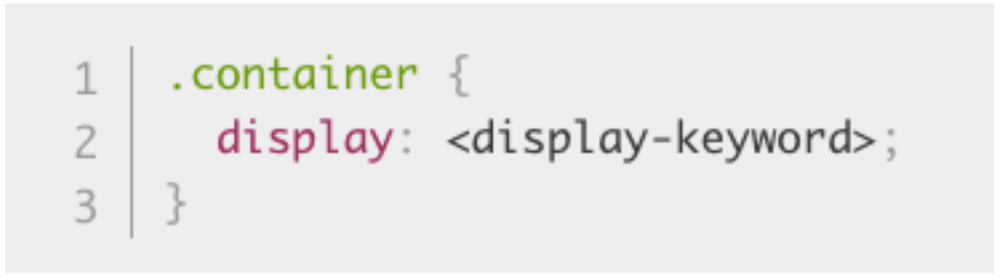
softserve

THE DISPLAY PROPERTY

This property allows us to change the default way something displays.

Everything in normal flow has a value of display, used as the default way that elements they are set on behave.

Keyword values are grouped into six value categories.



```
.container {
  display: <display-keyword>;
}
```

softserve

OUTSIDE - element’s outer display type, which is essentially its role in flow layout:

- block
- Inline

INSIDE - element’s inner display type:

- table
- flex
- grid

LIST ITEM - element generates a block box for the content and a separate list-item inline box:

- list-item

INTERNAL - this section defines those "internal" display values, which only have meaning within that particular layout mode:

- table-row
- table-cell
- table-column

BOX - these values define whether an element generates display boxes at all:

- none
- content

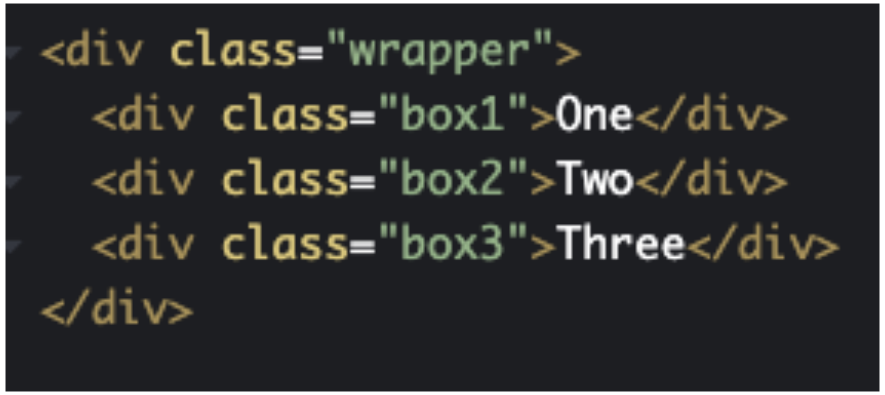
LEGACY:

- inline-block
- inline-flex
- inline-table

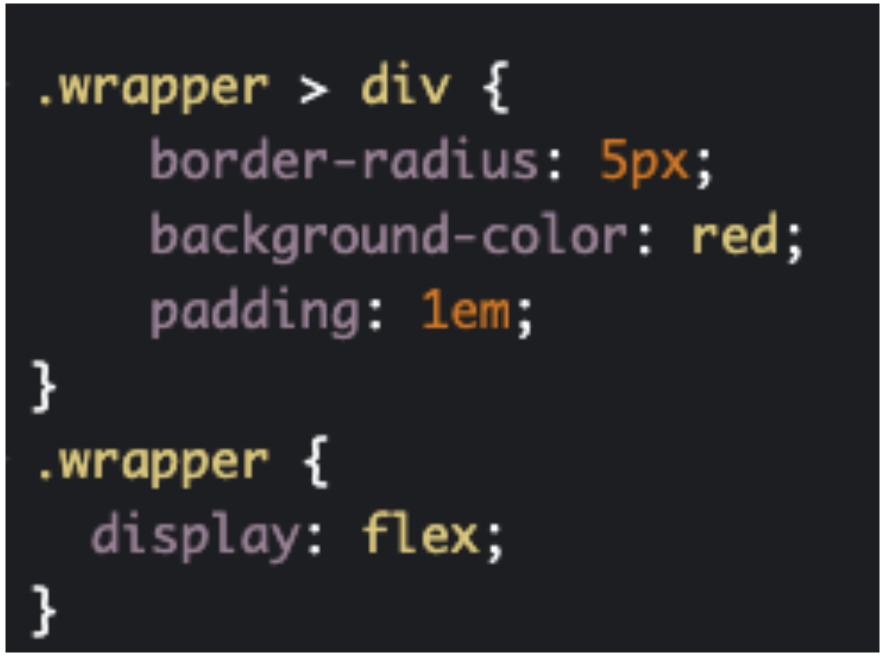
FLEXBOX

Flexbox is the short name for the Flexible Box Layout Module.

You apply display:flex to the parent element of the elements you want to lay out; all its direct children then become flex items.



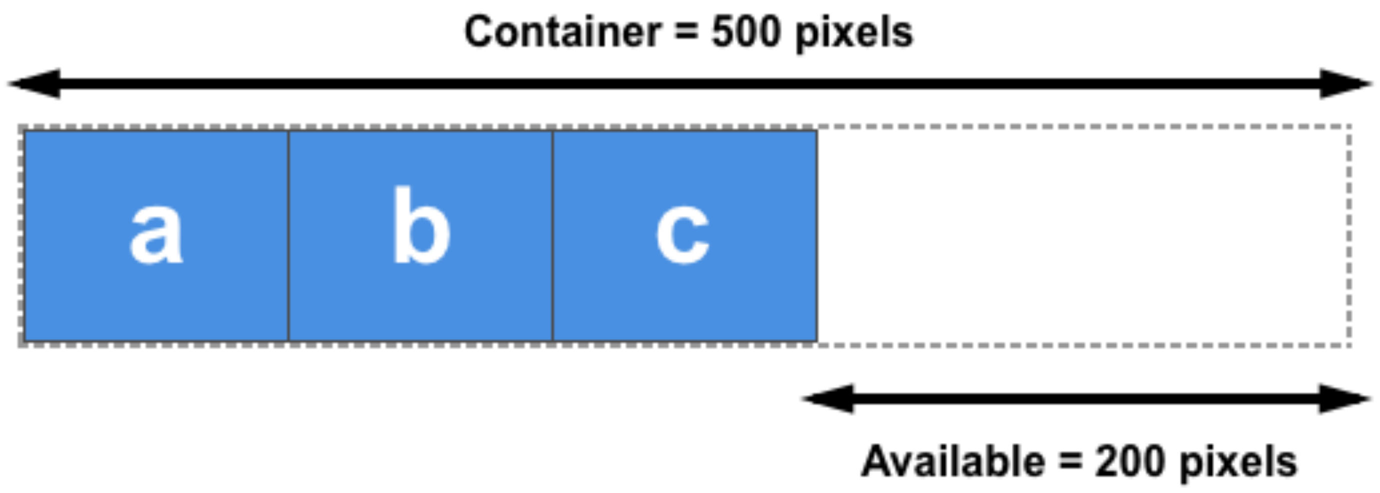
```html
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
</div>
```



```css
.wrapper > div {
    border-radius: 5px;
    background-color: red;
    padding: 1em;
}
.wrapper {
  display: flex;
}
```

softserve

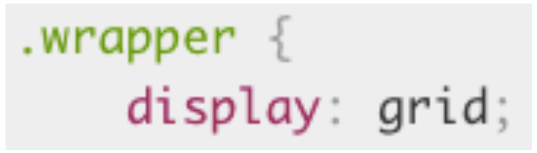
Default flexbox position



GRID

CSS Grid Layout is a two-dimensional layout system for the web.

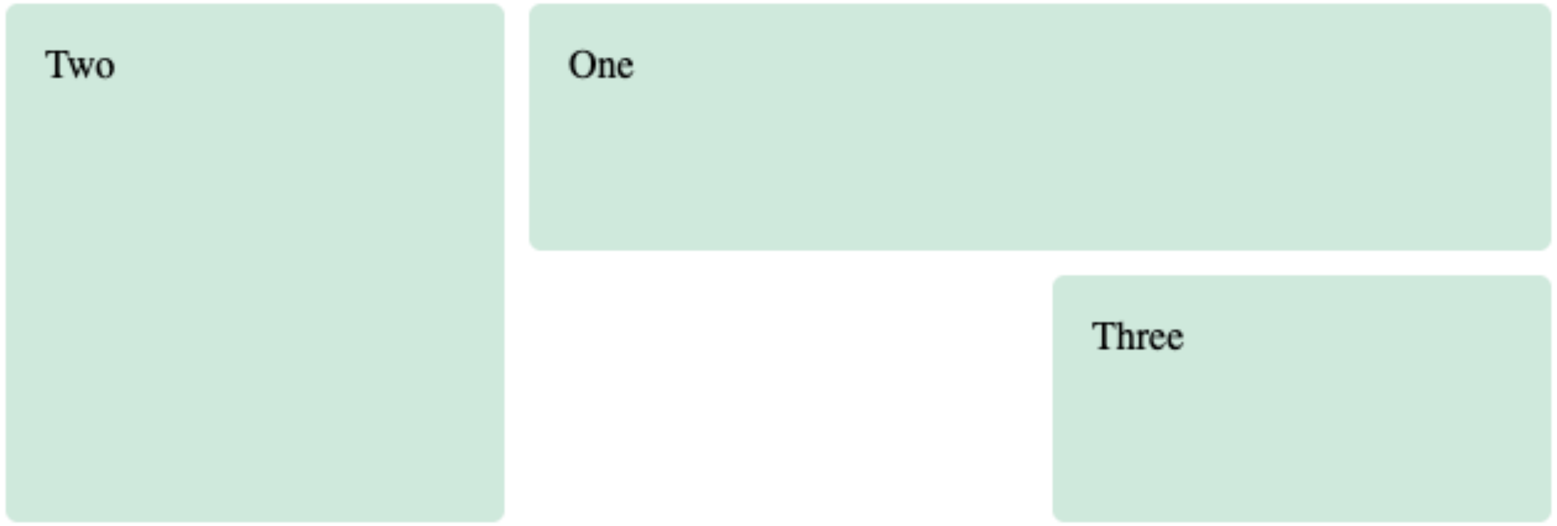
The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.



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

```html
<div class="wrapper">
    <div class="box1">One</div>
    <div class="box2">Two</div>
    <div class="box3">Three</div>
</div>
```

```css
.wrapper {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-rows: 100px 100px;
    grid-gap: 10px;
}

.box1 {
    grid-column: 2 / 4;
    grid-row: 1;
}

.box2 {
    grid-column: 1;
    grid-row: 1 / 3;
}

.box3 {
    grid-row: 2;
    grid-column: 3;
}
```



softserve

FLOATS

The float CSS property places an element on the left or right side of its container, allowing text and inline elements to wrap around it. The element is removed from the normal flow of the page.

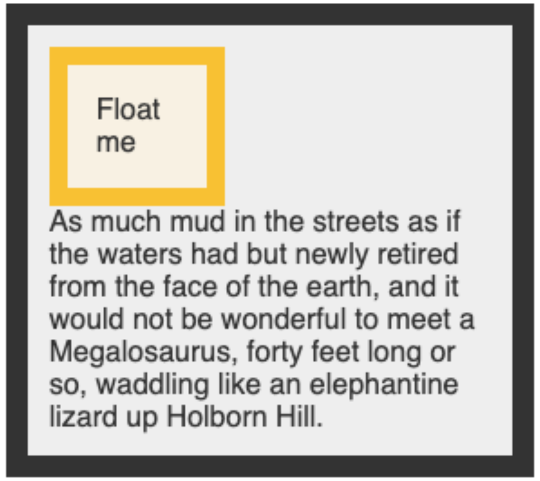
Float property:

- Left — Floats the element to the left.
- Right— Floats the element to the right.
- None— Specifies no floating at all. This is the default value.
- Inherit— Specifies that the value of the float property should be inherited from the element's parent element.

Clear

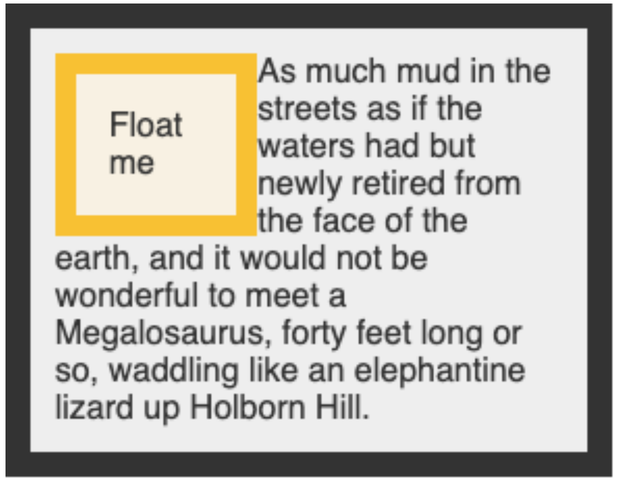
Sometimes you may want to force an item to move below any floated elements. For instance, you may want paragraphs to remain adjacent to floats, but force headings to be on their own line.

`float: none;`



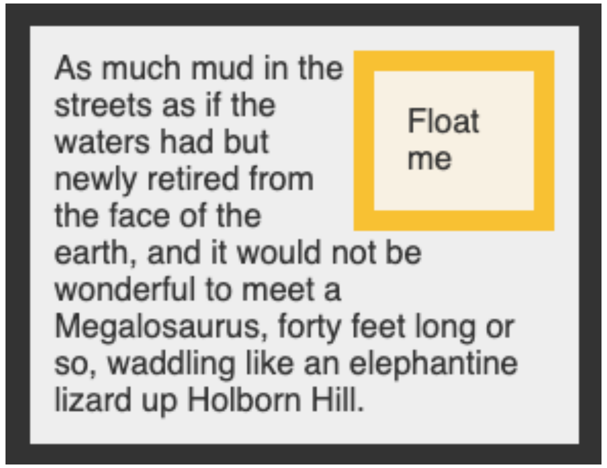
As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.

`float: left;`



As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.

`float: right;`



As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.

POSITIONING

Positioning allows you to move an element from where it would be placed when in normal flow to another location.

Important!

Positioning isn't a method for creating your main page layouts, it is more about managing and fine-tuning the position of specific items on the page.

Why to know?

Understanding positioning also helps in understanding normal flow, and what it is to move an item out of normal flow.

TYPES OF POSITION

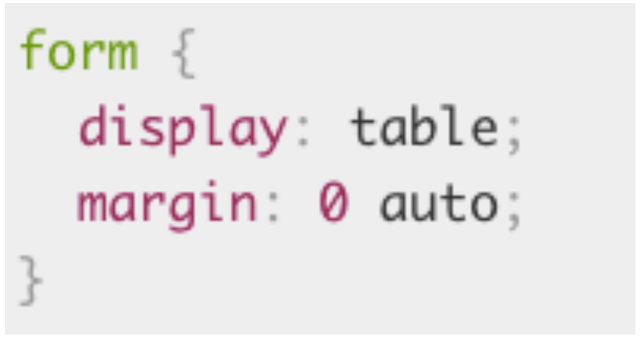
- Static
- Relative
- Absolute
- Fixed
- Sticky

TABLE LAYOUT

Before even basic CSS was supported reliably across browsers, web developers used to also use tables for entire web page layouts — putting their headers, footers, different columns, etc. in various table rows and columns.

Why it's bad?

This worked at the time, but it has many problems — table layouts are inflexible, very heavy on markup, difficult to debug, and semantically wrong.



```
form {
  display: table;
  margin: 0 auto;
}
```

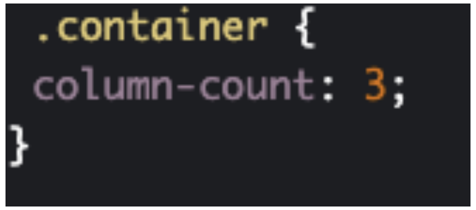
MULTIPLE-COLUMN LAYOUT

The multiple-column layout specification gives you a method of laying content out in columns, as you might see in a newspaper.

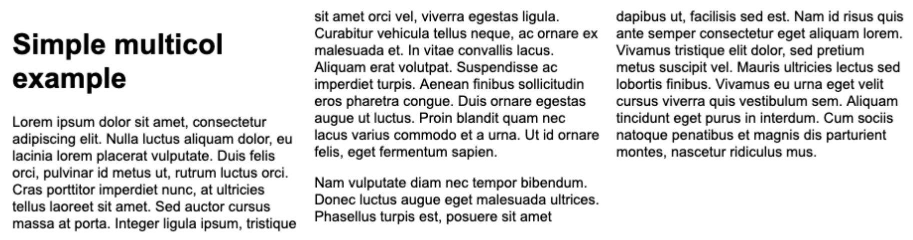
We switch on multicol by using one of two properties column-count or column-width.

Column-count: how many columns we would like to have.

Column-width: to fill the container with as many columns of at least that width.



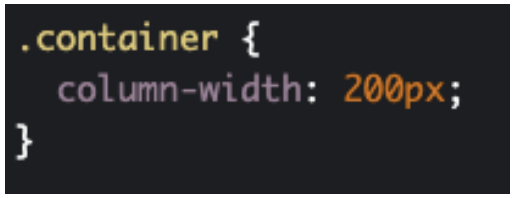
```
.container {
 column-count: 3;
}
```



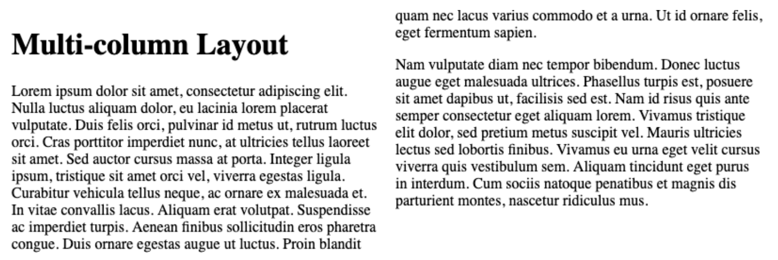
Simple multicol example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem. Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus. Vivamus eu urna eget velit cursus viverra quis vestibulum sem. Aliquam tincidunt eget purus in interdum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.



```
.container {
  column-width: 200px;
}
```



Multi-column Layout

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem. Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus. Vivamus eu urna eget velit cursus viverra quis vestibulum sem. Aliquam tincidunt eget purus in interdum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

BEST PRACTICES

- Use a CSS Reset Stylesheet
- Don’t Repeat Yourself (DRY)
- Use a CSS Framework
- Semantic
- Create it readable
- External CSS
- Avoid inline styling
- Shortland CSS

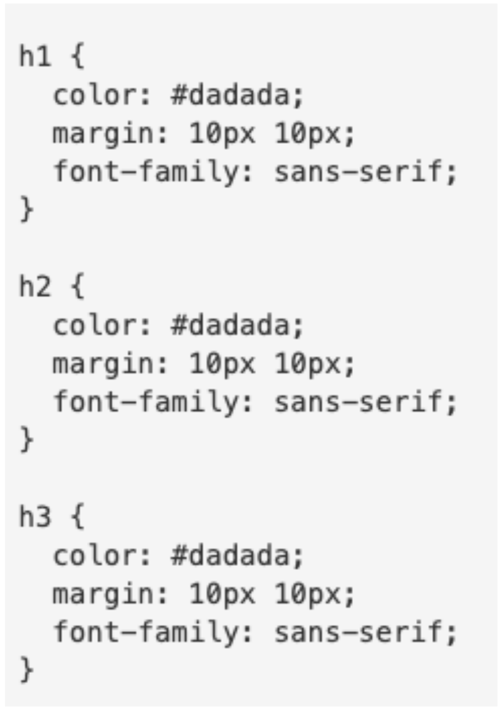
USE A CSS RESET STYLESHEET

Applying a reset stylesheet should be the first step in any website design. There are several widely-used reset stylesheets available, but the job of each one is the same: to even out the differences between different browsers' default styles and to create a clean slate on which to build your website's design.

To use a CSS reset, all you need to do is include the reset stylesheet file into your HTML page before you apply any other styles.

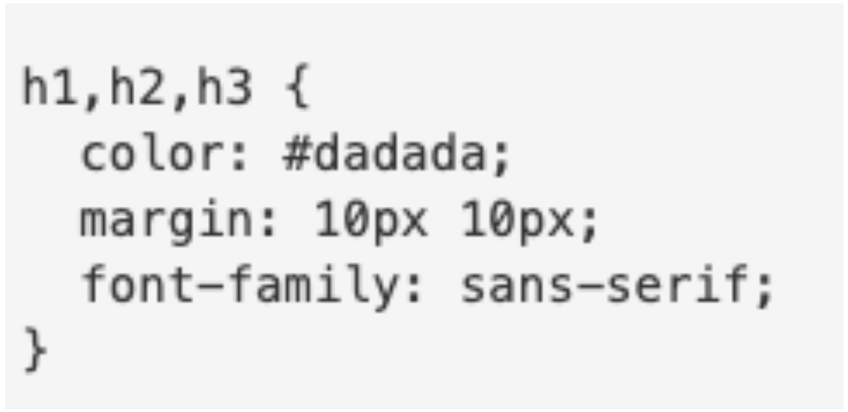
DON'T REPEAT YOURSELF(DRY)

Let's say you want to change the color, margins, and font-family for h1, h2 and h3 elements. One way you could do it is by creating three separate rules, like this:



```
h1 {
  color: #dadada;
  margin: 10px 10px;
  font-family: sans-serif;
}

h2 {
  color: #dadada;
  margin: 10px 10px;
  font-family: sans-serif;
}

h3 {
  color: #dadada;
  margin: 10px 10px;
  font-family: sans-serif;
}
```

However, this is far from efficient. A much more DRY way of achieving the same thing is to use the comma operator to apply the same styles to all three elements at once, like this:



```
h1,h2,h3 {
  color: #dadada;
  margin: 10px 10px;
  font-family: sans-serif;
}
```

USE A CSS FRAMEWORK

A CSS framework, such as Bootstrap, collects together frequently applied styles into a single stylesheet and implements all of the above best practices for you so that you can spend more time thinking about how to improve your website, rather than on making your CSS more manageable.

It can be thought of as a set of reliable precision tools that you can use to make crafting your website easier and to benefit from the knowledge and experience of Bootstrap's creators and users.

Learning and using a framework saves time and leads to better results in most cases.

SEMANTIC

In HTML, for example, the ⟨h1⟩ element is a semantic element, which gives the text it wraps around the role (or meaning) of a top level heading on your page.

```
<h1>This is a top level heading</h1>
```

By default, most browser's will style an ⟨h1⟩ with a large font size to make it look like a heading.
On the other hand, you could make any element look like a top level heading.

```
<span style="font-size: 32px; margin: 21px 0;">Is this a top level heading?</span>
```

This will render it to look like a top level heading, but it has no semantic value, so it will not get any extra benefits as described above. It is therefore a good idea to use the right HTML element for the right job.

CREATE IT READABLE

The readability of your CSS is necessary, although the general public overlooks its importance. Nice readability of your CSS makes it a lot easier to keep up with the future, as you will be able to notice components faster.

Usually while writing CSS, most developers adopt one of the following approaches.

```
.example{background:red;padding:2em;border:1px solid black;}
```

```
.example {
  background: red;
  padding: 2em;
  border: 1px solid black;
}
```

softserve

EXTERNAL CSS

The use of external CSS is beneficial. You can pull all the CSS in an external file and include this files in your HTML.

Benefits of external CSS:

- All the CSS are stored within single Stylesheet.
- All the style changes are done in a single CSS for each page (Easier to maintain the website).
- Single CSS will be cached and page load faster.

AVOID INLINE STYLING

Using inline styling show the HTML code messy. Updating each HTML events style through global CSS is hard.

Bad code:

```
<div style="float:left">
    <img src="images/logo.gif" alt="" />
</div>
```

Good code:

```
<div class="left">
    <img src="images/logo.gif" alt="" />
</div>
```

SHORTLAND CSS

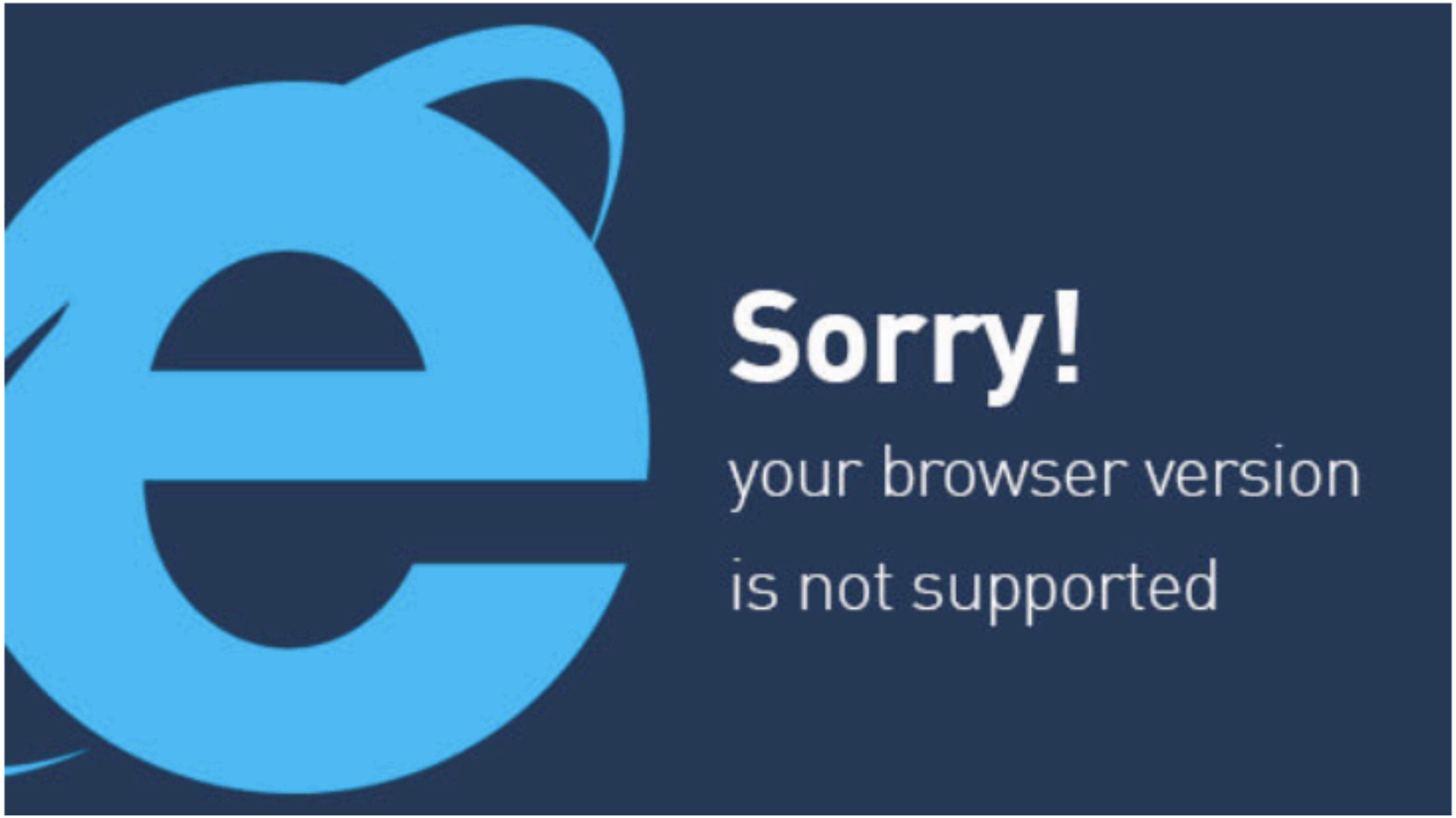
One feature of CSS is the ability to use shorthand properties and values. Most properties and values have acceptable shorthand alternatives.

```
img {
    margin-top: 5px;
    margin-right: 10px;
    margin-bottom: 5px;
    margin-left: 10px;
}

button {
    padding: 0 0 0 20px;
}
```

```
img {
    margin: 5px 10px;
}

button {
    padding-left: 20px;
}
```

8 essential tips for building a cross-browser compatible website

- Keep your code simple
- Use frameworks
- Define valid doctype
- CSS reset
- Validate
- Conditional comments
- Prepare for differences
- Don’t skip cross-browser testing



Sorry!
your browser version
is not supported

KEEP YOUR CODE SIMPLE

Think quality over quantity when it comes to coding.

Don't dedicate ten lines of code to a feature that only needs three.

Not only will simple code be more cross-browser friendly, it will also be more maintainable when the time comes that you do have to debug or adjust it for compatibility.

softserve

USE FRAMEWORKS

CSS frameworks like Foundation and Bootstrap will give you style code to make cross-compatibility easier for you. If you take the time to become familiar with some of the features, building a responsive web application becomes much faster and easier.

These will also help you make the application look and behave correctly in mobile browsers.

DEFINE VALID DOCTYPE

The Doctype is the first line in your code which describes the HTML that will be used in your application. Because different browsers have different standards and rules, you need to define the Doctype or the rendering engine will basically guess it for you.

Why use?

Doctype basically helps your browser to recognize in which language is your website's code written. If you don't specify that, some of the smart browsers will understand it themselves but some dumb browser will not be able to figure out what happened, and they will render some element of your website in a way that you would not like.

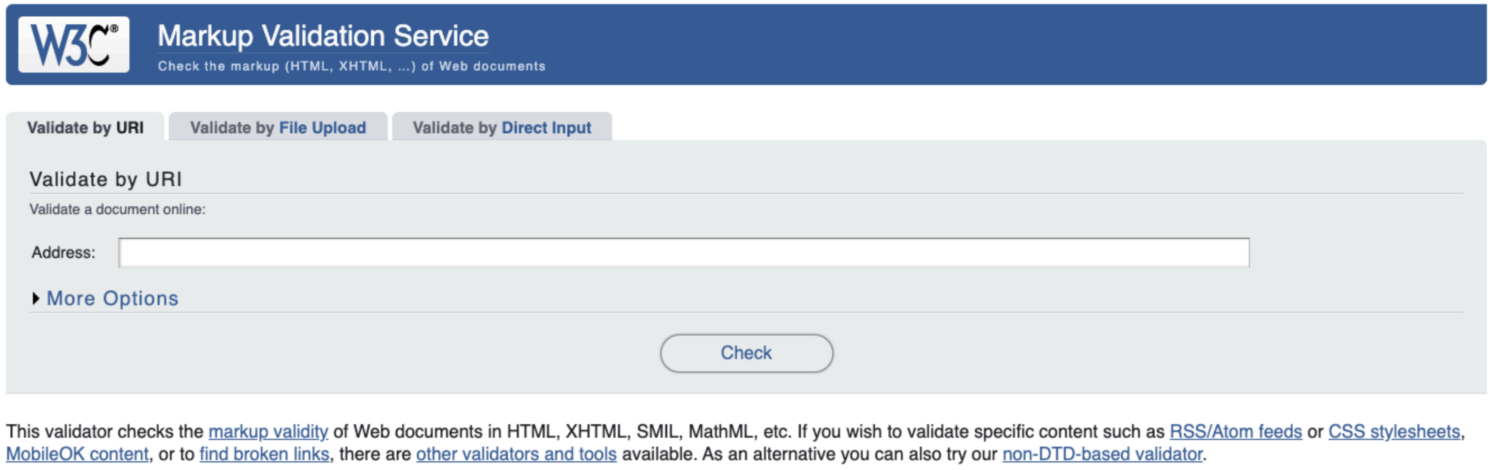
CSS RESET

Every browser has different default CSS rules that they follow.
This is why you use CSS reset stylesheet to make sure your browsers follow the same basic rules and behave consistently.
You want to add one of these as the first stylesheet in order to reset unless you use a framework which will already have one.

The very famous Eric Meyer's CSS reset can be used to help you out to solve this browser incompatibility issue. Or you can use of standards CSS Resets like normalize.css.

VALIDATE

Once you take care of all the coding precautions, it's time you validated your website.
You can use HTML, CSS validator for the same. This will point all your mistakes and improvements to make sure your code doesn't break.
You can use w3 validator to validate HTML of your website. Similarly, you can validate for CSS too. Try it using Jigsaw validator from W3.
Once you validate, you'll see errors and warnings which then you can solve further.

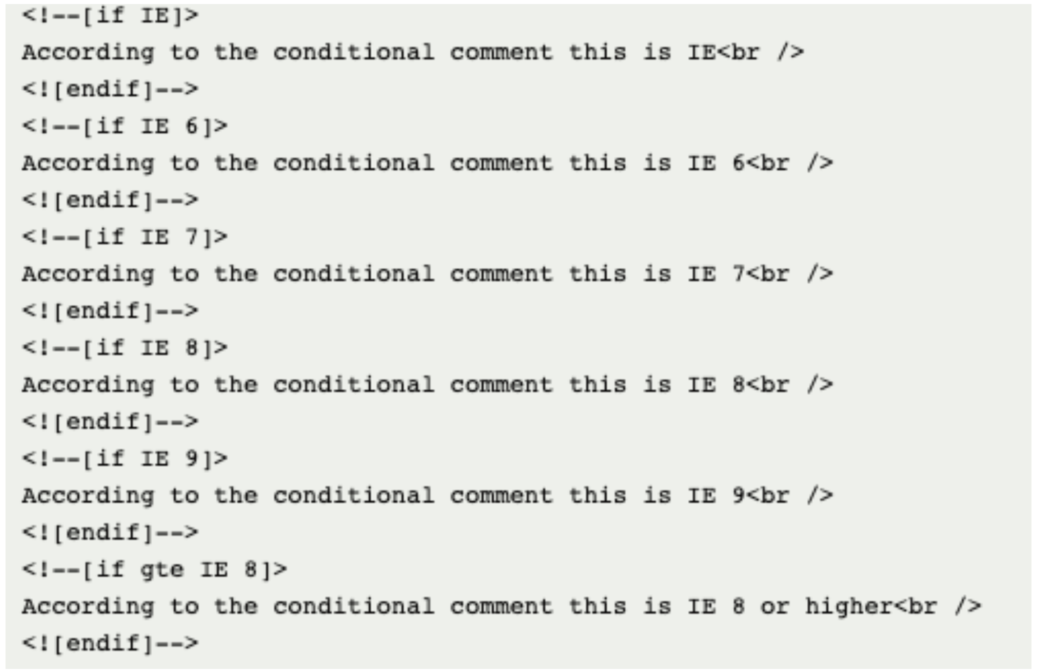

Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents

This validator checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as RSS/Atom feeds or CSS stylesheets, MobileOK content, or to find broken links, there are other validators and tools available. As an alternative you can also try our non-DTD-based validator.

softserve

CONDITIONAL COMMENTS

Conditional comments allow you to link style sheets for different browsers, which is especially helpful when it comes to design challenges that are common with Internet Explorer.



```
<!--[if IE]>
According to the conditional comment this is IE<br />
<![endif]-->
<!--[if IE 6]>
According to the conditional comment this is IE 6<br />
<![endif]-->
<!--[if IE 7]>
According to the conditional comment this is IE 7<br />
<![endif]-->
<!--[if IE 8]>
According to the conditional comment this is IE 8<br />
<![endif]-->
<!--[if IE 9]>
According to the conditional comment this is IE 9<br />
<![endif]-->
<!--[if gte IE 8]>
According to the conditional comment this is IE 8 or higher<br />
<![endif]-->
```

PREPARE FOR DIFFERENCES

It's pretty much impossible to have a design that looks the same on every browser unless it's extremely basic.

Details like forms and typography will likely vary no matter what rules you follow. Your main concern should not be making the design look identical on every browser.

Instead, you should make sure that it looks acceptable and is usable without having elements that are out of place or that prevent someone from accessing certain functions.

DON'T SKIP CROSS-BROWSER TESTING

Missing cross browser testing is like making all your efforts for cross browser compatible websites go in vain.

Without cross browser testing you won't be able to make sure that whether the chances that you've taken for making a cross browser compatible website do work or not.

Using a tool like CrossBrowserTesting or LambdaTest as your all time cross browser testing friend for that.

RESOURCES USED

- https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Introduction
- https://cssguidelin.es/
- https://crossbrowsertesting.com/blog/development/cross-platform-website-development/
- https://www.lambdatest.com/blog/how-to-make-a-cross-browser-compatible-website/
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Relationship_of_Grid_Layout
- https://www.w3.org/TR/css-multicol-1/
- https://developer.mozilla.org/en-US/docs/Web/CSS/float

softserve