

Styling and CSS

Spring 2024

Last time

HTML — a format for documents that represents content using nested tags

structural: `<body>`, `<h1>`, `<p>`, `<a>`, ...

formatting: ``, ``, `<code>`, ...

objects: lists, tables, images, ...

Browsers translate HTML documents into DOM trees (made up of elements)

Browsers can render however they want within the basic rendering algorithm

- block elements form blocks, inline elements are merged into blocks

Styling

Styling is the way to modify how the browser renders elements in the tree

- size, spacing, color, ...

- text properties (font, size, weight, underlining, word breaking, ...)

- layout

- ...

Styling used to be done using attributes — replaced by CSS styling properties

CSS Styling properties

Every element in a DOM tree has associated **styling properties** that start with reasonable defaults but can be overridden

CSS = Cascading Style Sheets — defines properties and how they are applied

Properties?

```
height: 40px;  
color: red;
```

More than 200 distinct properties (some properties are tag-specific)

How to style (1) — style attribute

Every element can have a style attribute that specifies CSS properties for that element:

```
<p style="font-size: 30px;">  
  This is a sample line  
</p>
```

```

```

How to style (2) — style element in <head>

Pull styling out of the HTML elements into a **style element** listing **CSS rules**

```
<style>
  p {
    color: blue;
    font-style: italic;
  }
</style>
```

"all <p> elements are blue and their text is in italics"

How to style (3) — external CSS file

Can put CSS rules in their own file — usually with extension .css

Link to it from the HTML document using a link in the <head> element

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

Can link multiple style sheets

Rules from <style> have precedence over rules in linked style sheets

Text CSS properties

font-family

font-size

font-weight

font-style

text-decoration

Text CSS properties

font-family

font-size

font-weight

font-style

text-decoration

Font family as a list of generic/specific names

```
font-family: "Open Sans",  
sans-serif;
```

Size:

```
font-size: 16px;
```

Bold or not?

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

Italics or not?

```
font-style: normal;  
font-style: italic;
```

Text CSS properties

font-family

font-size

font-weight

font-style

text-decoration

```
text-decoration: green wavy underline;
```

```
text-decoration: underline dotted red;
```

```
text-decoration: underline overline blue;
```

```
text-decoration: line-through black;
```

Size CSS properties

height

width

margin-top

margin-bottom

margin-left

margin-right

padding-top

padding-bottom

padding-left

padding-right

Size CSS properties

height

width

margin-top

margin-bottom

margin-left

margin-right

padding-top

padding-bottom

padding-left

padding-right

Sets the height and width of *block elements*

Doesn't affect *inline elements*

Different kind of units: px
in
cm
em
pt
...

Also: $n\%$ percentage of what?
height or width of parent block!

Size CSS properties

height

width

margin-top

margin-bottom

margin-left

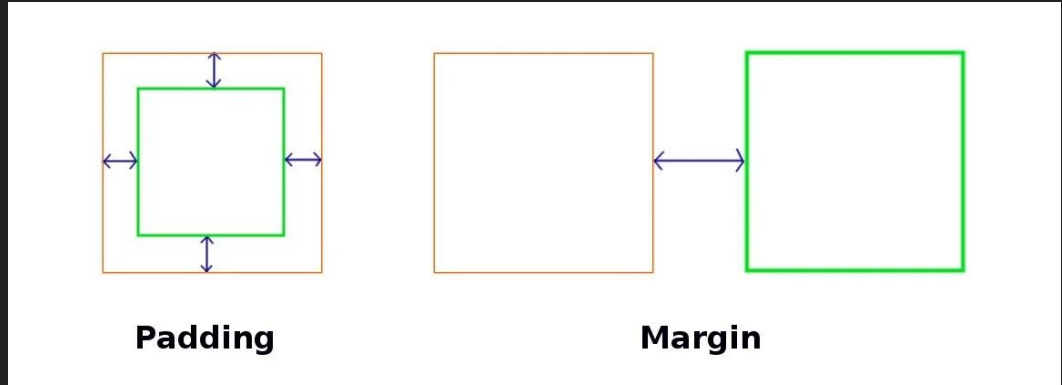
margin-right

padding-top

padding-bottom

padding-left

padding-right



Size CSS properties

height

width

margin-top

margin-bottom

margin-left

margin-right

padding-top

padding-bottom

padding-left

padding-right

Shorthand properties:

margin: a;

margin-top: a;

margin-right: a;

margin-bottom: a;

margin-left: a;

margin: a b;

margin-top: a;

margin-right: b;

margin-bottom: a;

margin-left: b;

margin: a b c;

margin-top: a;

margin-right: b;

margin-bottom: c;

margin-left: b;

margin: a b c d;

margin-top: a;

margin-right: b;

margin-bottom: c;

margin-left: d;

Color CSS properties

`color`

`background-color`

`opacity`

`border-style`

`border-width`

`border-color`

Color CSS properties

`color`

`background-color`

`opacity`

`border-style`

`border-width`

`border-color`

Colors defined by name:

`white`
`black`
`red`
`blue`
`chartreuse`
`...`

Colors defined by RGB value:

`#aabbcc`

where *aa*, *bb*, *cc* are values 00 - FF (in hex)

`#ff0000`
`#808000`

Color CSS properties

color

background-color

opacity

border-style

border-width

border-color

How opaque / transparent is the element?

```
opacity: 1;  
opacity: 0;  
opacity: 0.25;
```

Color CSS properties

color

background-color

opacity

border-style

border-width

border-color

Is there a border and what style?

```
border-style: none;  
border-style: solid;  
border-style: dashed;
```

Border's width?

```
border-width: 1px;  
border-width: 2in;
```

Border's color?

```
border-color: #ff20c0
```

Shorthand property:

```
border: width style color;
```

CSS rules

Form of rules in a style sheet:

```
selector {  
    property1 : value1;  
    property2 : value2;  
    ...  
}
```

Applies properties listed to every element matching the *selector*

Selectors

Rich language for describing sets of elements to which properties should apply

Basic selectors, e.g.,

<code>*</code>	all elements
<code><i>name</i></code>	all elements with tag <i>name</i>
<code>#<i>name</i></code>	the element with attribute <code>id="name"</code>
<code><i>.name</i></code>	all elements with attribute <code>class="name"</code>

You can also combine basic selectors, e.g.,

<code><i>selector</i>₁ <i>selector</i>₂</code>	all descendants of elements matching <i>selector</i> ₁ that match <i>selector</i> ₂
--	--

Property inheritance

```
body {  
  font-family: sans-serif;  
}
```

"<body> and all its descendants use font-family sans-serif"

```
p {  
  line-height: 1.5;  
}
```

"All <p> and all their descendants use line-height: 1.5"

Property inheritance

```
body {  
  font-family: sans-serif;  
}
```

"<body> and all its descendants use font-family"

```
p {  
  line-height: 1.5;  
}
```

"All <p> and all their descendants use line-height"

Some properties are not inherited by some elements


<h1> does not inherit the `font-size` property (why?)

Specs say which properties are inherited by which elements

Cascade

An element may have several CSS rules giving a property different values

General rules get overruled by more **specific** rules



style attribute
#*name* rule
.*name* rule
name rule

Equal specificity → most recent rule wins

There's a lot more to CSS

Calculations!

```
font-size: calc(3em + 5pt);
```

```
width: calc(100% - 20px);
```

pseudo-classes

```
p:hover {  
  border: 1px solid blue;  
}
```

media queries

Layout

Layout CSS properties

Properties that affect how an element fits in the general layout

`display`

`float`

`position`

Layout CSS properties

Properties that affect how an element fits

display

float

position

```
display: none;  
/* don't render */
```

```
display: inline;  
/* render element inline */
```

```
display: block;  
/* render element as block */
```

```
display: inline-block;  
/* render element as a block  
   but inline it */
```

```
display: flex;  
/* render content as a flexbox */
```

```
display: grid;  
/* render content as a grid */
```

Layout CSS properties

Properties that affect how an element fits

`display`

`float`

`position`

```
float: left;
```

```
float: right;
```

Don't render the element where it is but float it to the left or to the right of the next block, with content rendering around it

Layout CSS properties

Properties that affect how an element fits

display

float

position

```
position: static;  
/* render in default position */
```

```
position: relative;  
top: n;  
left: m;  
/* render in default position  
   shifted by m and m */
```

```
position: fixed;  
top: n;  
left: m;  
/* render independently of the  
   rest of the document at  
   screen position (m, n) */
```

Flexboxes and Grids

For a long time, lining up elements in columns or in rows or in grids was painful

- late 90s: perverted use of tables and frames
- it was always clunky
- does NOT work well for responsive design

Modern approach:

- flexboxes for row/column alignment
- grids for actual grid structure

Transformations

`transform:`

`scale`

`rotate`

`translate`

`skew`

`...`

Transformations are applied from right to left

Can apply to any element (but best used with blocks)