CSS

Web Engineering
Dirk König
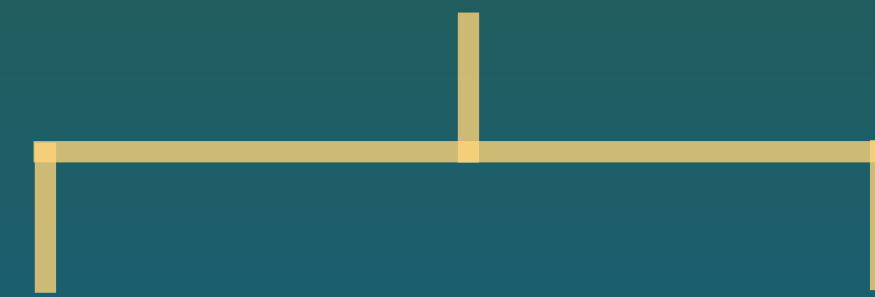## Christian Ribeaud

The Two Faces

CSS Rule

selector

value

h1 { color: blue; }

property

CSS Selectors

h1   <h1></h1>   element

#top   <.. id="top">   id

.cool   <.. class="cool">   class

*   all

can be a list like
"cool drink"

Special Selectors

[title]	<.. title="..">	attribute
[title="x"]	<.. title="x">	attribute/value
		^= $= *= ~=
p:hover	<p ...	pseudo-class

Special Selectors

h1 +   next sibling

h1 ~   all siblings

P.S. there is much more that we don't cover
:not, :nth-child, :nth-child-of-type pseudo-class-functions
::first-letter pseudo-elements

Selector Combinations

h1, h2   union

main p   descendant

ul > li   direct child

Selector Combinations

div.cool   element with class

div[cool]   element with
                        attribute

What selects: _____

What selects: div > span

CSS Inline

```
HTML
..
<p style="color:blue;">
..
</p>
..
```

Style Element

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

Link Element

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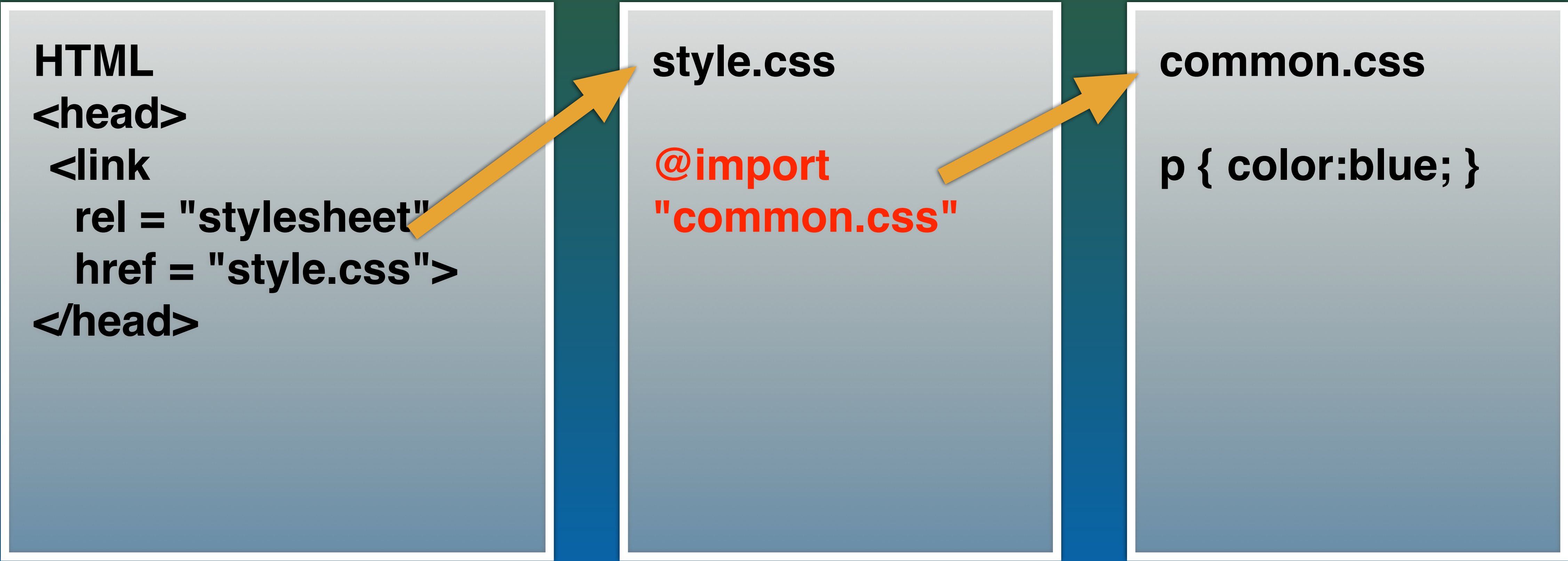
```
style.css

p { color:blue; }
```

CSS Import

HTML
<head>
 <link
   rel = "stylesheet"
   href = "style.css">
</head>

style.css

@import
"common.css"

common.css

p { color:blue; }

Precedence

1. Inline css (html style attribute) overrides css rules in style tag and css file

2. A more specific selector takes precedence over a less specific one

Precedence

3. rules that appear later in the code override earlier rules if both have the same specificity.

4. A css rule with !important always takes precedence.

Precedence

Detailed info:

https://stackoverflow.com/questions/25105736/what-is-the-order-of-precedence-for-css

How to avoid complexity in precedence?

Conflict Resolution

Source order

latest wins, unless important

p { color: blue; }
p { color: red; }

p { color: blue !important; }
p { color: red; }

Cascade Rules of Thumb
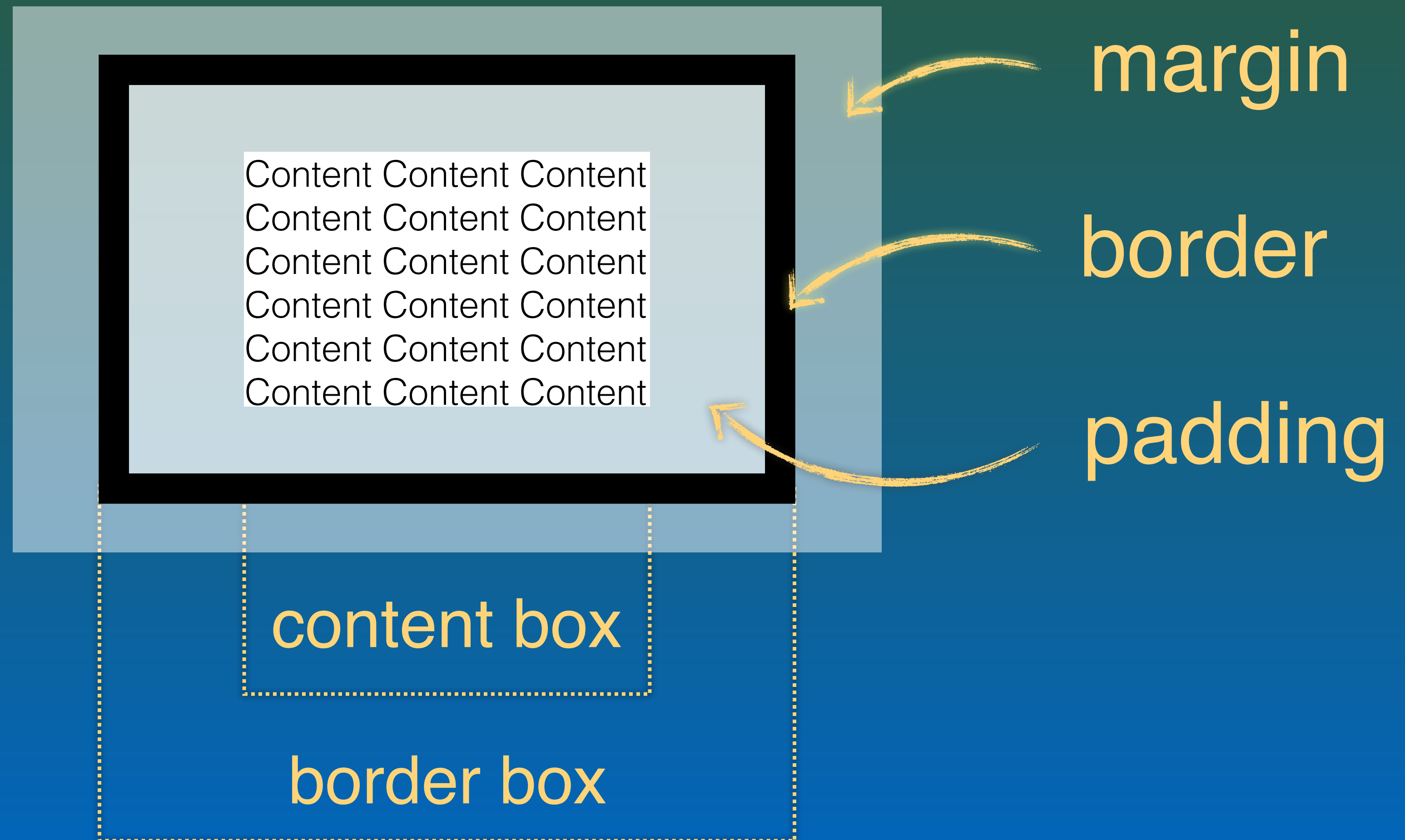
Source order   *latest wins*

Note:
p { color: blue !important; } overrides all the above!

Properties

There are too many properties and possible values to list here.

We can only cover how it works in principle.

Box Model

margin

border

padding

Content Content Content
Content Content Content
Content Content Content
Content Content Content
Content Content Content
Content Content Content

content box

border box

Margins collapse

Content Content Content
Content Content Content
Content Content Content
Content Content Content
Content Content Content
Content Content Content

Content Content Content
Content Content Content
Content Content Content
Content Content Content
Content Content Content
Content Content Content

Content Content Content
Content Content Content
Content Content Content
Content Content Content
Content Content Content

Units

px   *pixel*

cm, mm, in, pt   *absolute length (print)*

em, rem   *size of "M"*

%   *relative to parent size*

Functions

calc(100% / 12)	*formula*
attr(name)	*attribute value*
url("backg.jpg")	*external reference*
var(--left-indent)	*variable*

Further Topics: layout

display   *inline, block, none,...*

position   *static, relative,*

                     absolute, fixed

float, clear   *left, right*

Further Topics: layout

flexbox   *http://flexboxfroggy.com*

grid   *https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout*

Further Topics: color

red, blue, …   *color names*

#FF0000   *hex values*

rgba(255,0,0,0.5)   *opacity (alpha)*

linear-gradient()   *direction, color stops*

radial-gradient()   *shape, color, position*

Further Topics

Fonts, Web Fonts, Entities

Animations, Transitions

Media Query (comes later)

transform, clip, z-index, svg

flex-box, grid, table-layout