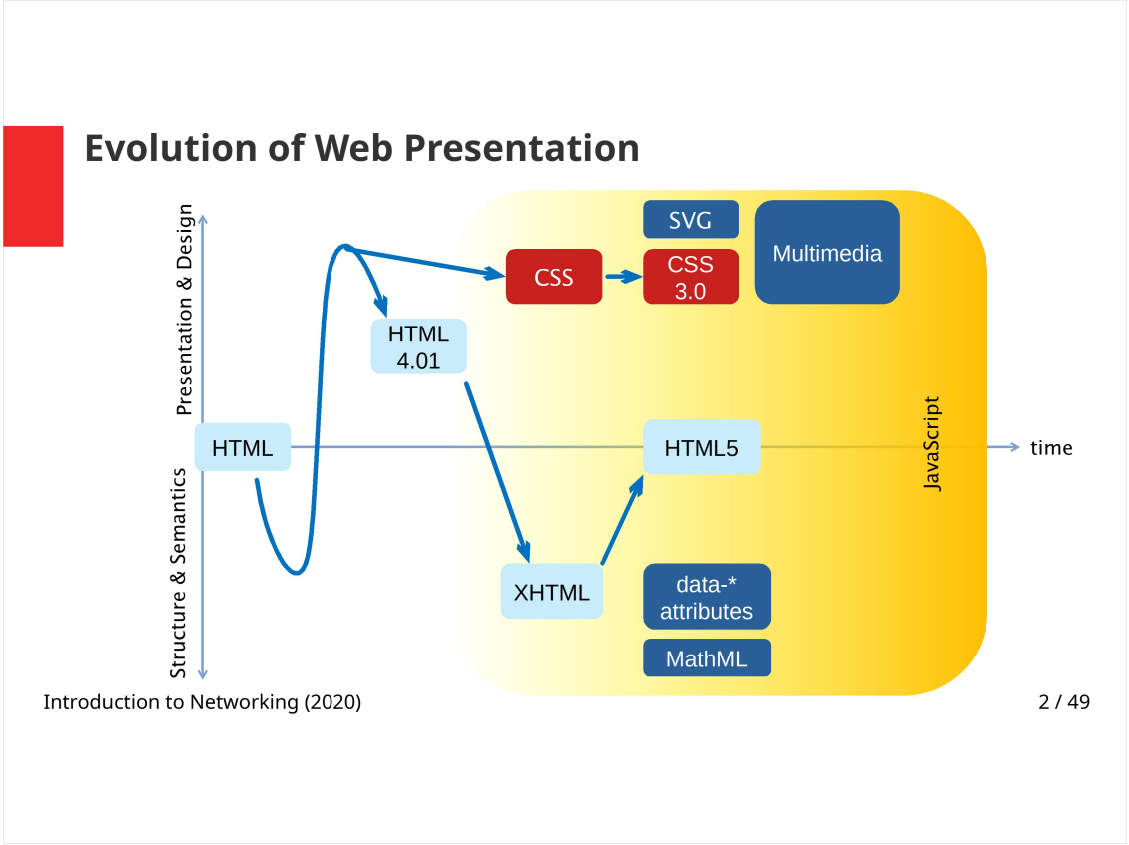
Cascading Style Sheet

Introduction to networking

Dr. Klára Pešková, Klara.Peskova@mff.cuni.cz
Department of Software and Computer Science Education



HTML was designed to define the structure of documents. The document was then rendered according to this structure.

In the course of time, more and more options were introduced to HTML to change the design of web pages – attributes to set colors, text-alignment, etc. were added.

Later on, however, the design of the web pages moved completely to CSS.

In XHMTL once again only document structure was defined.

In modern web design, lots of content is presented as multimedia; SVG support (vector graphics, animations) was added.

On the other hand, more tools were added to define the document structure – e.g. `data-*` attributes for providing data to JavaScript, or MathML – notation of mathematical expressions (However Google Chrome no longer provides support for MathML).

With the growth of web applications, a large part of web sites is built using JavaScript.

CSS Versions

- CSS 1 (1996)
  - Basic text properties (fonts, alignment, spacing, …)
  - Color of text and backgrounds
  - Margins, paddings, and borders
- CSS 2 (1998)
  - New types of positioning
  - Concept of media introduced
- CSS 2.1 (2004–2011)
  - Fixes serious problems of CSS 2

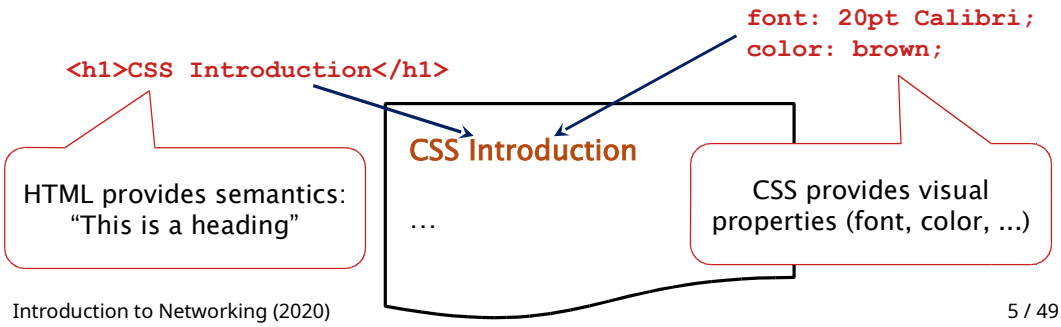
CSS Versions

- CSS 3 (1999–present)
  - Improves existing properties – more elaborate backgrounds, custom borders, …
  - Introduces additional visual effects – round corners, shadows, …
  - Allows using custom fonts
  - Adds transitions and animations

CSS and HTML Content

- CSS describes how HTML elements are displayed – on screen, mobile phone, when printed on paper...

+ one CSS can be used for several pages



A design of a webpage is created by setting CSS styles for HTML elements. CSS properties define how the HTML elements are rendered.

One CSS stylesheet – a file with CSS declarations, can be used for more pages. This way a unified style of web site can be easier to maintain.

CSS syntax

- Format of CSS rules:

```
selector {
    property: value;
    property2: value2
}
```

- Example:

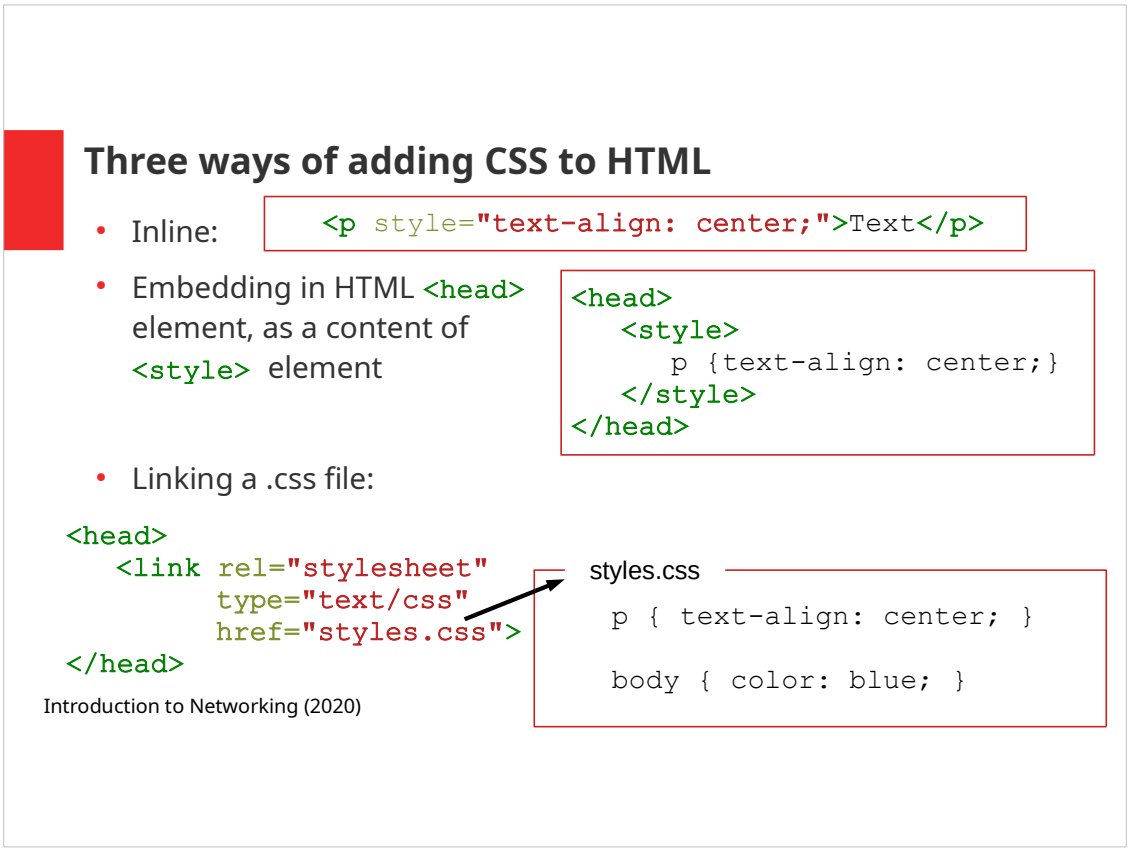
```
p {
    color: red;
    text-align: center
}
```


.css file consists of CSS rules. Each rule has a selector and a block of declarations.

Selector selects HTML element or elements to which CSS rules are applied.

Block of declarations contains a list of declarations of `property:value`, properties are separated by semicolons.

CSS syntax is different than HTML syntax, however both languages can be written in the same file (see next slide).



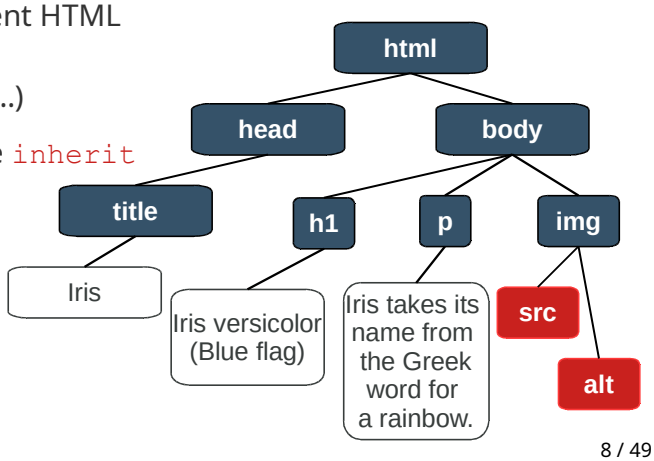
- Inline styles
  - Can be used for any HTML element
  - Styles are set as a value of `style` attribute
  - Used in special cases (typically by scripts)

- Embedding CSS in HTML `<style>` element
  - Styles are defined in HTML `<head>`
  - Used for web sites that only have one page (the page is downloaded faster)

- In external .css file
  - It is possible to define more .css files for different use cases
  - One .css file can be used multiple times for more HTML pages

Inheritance

- Some properties, e.g. font, inherit their values from parent HTML elements (`<body>` → `<h1>`, `<p>`...)
- These properties have `inherit` set as a default value



Inheritance depends on the hierarchical tree structure of HTML document. E.g. font set for `<body>` element is used for `<body>'s` children and the children of these children.

CSS selectors

- `p` selects all paragraphs (styles for HTML element type)
- `#mouse` selects an element with `id="mouse"`

  ```
  <h1 id="mouse">House mouse</h1>
  ```

- `.info` selects all elements with `class="info"`
  - One element may have multiple classes assigned

    ```
    <p class="info red">Watch out!</p>
    ```

- `*` universal selector (selects all elements)

More selector types can be used to define CSS styles for chosen HTML elements:

- Styles can be defined for all HTML elements of a selected type, e.g. to all paragraphs
- or to a specific element, using its unique `id` attribute (selector in CSS starts with # character)
- It is also possible to create your own CSS class, that can be applied to multiple HTML elements (of different types, e.g. `p` and `h2`), by setting their `class` attribute to the class name; class selector in CSS starts with a dot character.
- Styles defined for * selector are applied to all HTML elements in the document.

More on CSS selectors

- Aggregating rules
  - `s1, s2 {…css…}` one declaration block can be used for multiple selectors
- Combining selectors
  - `p.info` selects all paragraphs with class `info`
  - `h1#main` selects `<h1 id="main">`
  - Using relative position in the tree structure of HTML document
    - `E F` selects elements `F`, which have ancestor `E`
    - `E > F` selects elements `F`, which have parent `E`
    - `E + F` selects elements `F`, which are immediately preceded by `E`
    - `E ~ F` selects elements `F`, which are preceded by `E`

Aggregating rules

A block of declarations can be applied to multiple selectors – selectors are separated by commas.

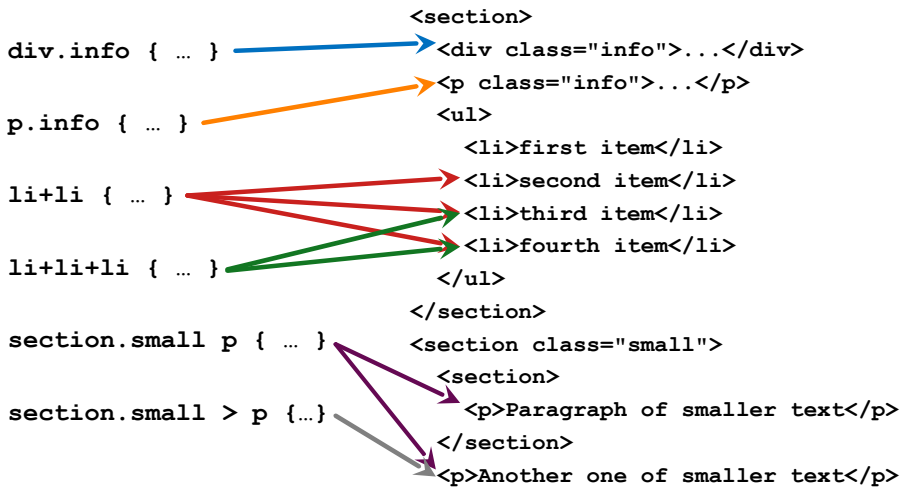
Combining selectors

A selector can be a complex combination of simple selectors (element type, class, id).

Definitions:

- Ancestor – HTML element, that is higher in the tree structure
- Parent – one level higher in the tree structure
- Child – one level lower in the tree structure
- Immediately preceded – at the same level in the tree structure, right after in the source code
- Preceded – at the same level in the tree structure

Combining selectors – example



- div with “info” class
- p with “info” class
- li that is preceded by li
- li that is preceded by li that is preceded by li
- p that in anywhere inside the section with “small” class
- p that has as a parent (closest ancestor) a section with “small” class

Combining selectors

- `ul li`
  - `li` anywhere inside `ul`

```
<ul>
 <li>
  <ol>
   <li> styles apply to
this list item as well
   </li>
  </ol>
 </li>
</ul>
```

- `p.info` vs. `p .info`
  - Space character has a meaning!
- `main ul, ol`
  - `main` belongs only to the first selector (`ol` stands alone)

Note that space character has a meaning when combining selectors:
The part of selector after space needs to be a child of the selector part before space for the selector combination to apply.

`p.info` vs. `p .info`

- The first example selects paragraphs with `info` class
- The second one selects all elements with `info` class inside `p`

`main ul, ol`

- This selector chooses:
  - `ul`s, that are inside HTML element `main`
  - And all `ol` elements in the whole HTML document

Pseudo-classes

- links

`a:link`	unvisited link
`a:visited`	visited link
`a:hover`	element over which a mouse cursor hovers
`a:active`	active (currently clicked on) link

- `:first-child` (`:first-of-type`) – element which is the first child of its parent / first sibling of its type
- `:last-child` (`:last-of-type`), `:only-child` (`:only-of-type`)
- `:nth-child(even)` / `:nth-of-type(3n+1)`
  - e.g. first `<i>` in each `<p>` — `p i:first-child`
  - e.g. all `<i>` in a first `<p>` — `p:first-child i`

Using pseudo-classes, a styles can be defined for elements only in a specific situation:

E.g.

- A link, if a user hovers over it by a mouse
- Visited link
- A table row, if it is an even row

Pseudo-classes – example

```
tr:nth-child(even) {background-color: #f2f2f2;}
```

First Name	Last Name	Points
Joe	Black	150
Jane	Jackson	90
Jim	Walker	10

Pseudo-elements

- Pseudo-elements select a specific part of HTML elements

```
::first-letter
::first-line
::selection
::after
::before

h1::after {content: url(smiley.gif)}
p::first-letter {color: #f00; font-size: xx-large;}
```


`::after` inserts a content right after the given element. In the example at the slide “smiley.gif” is rendered after all headings of rank 1.

`::selection` – styles are applied to the part of text that is selected by a user

Cascading

- More than one rule can apply to an element
- Complex schema of priorities (weights) is defined
- The priorities are based on (more details follow on the next slide):
  1. Style origin – the “distance” of style declaration to a HTML element (e.g. inline styles have priority to styles defined in an external file)
  2. Selector specificity
  3. Order of appearance (latter overrides former)
- CSS property may be marked as important

```
color: blue !important;
```


Note:
One selector can be used multiple times and thus more declaration blocks can be assigned to a selector. For example it is possible to define the common properties for several selectors in one block and than add specific styles for each of these selectors separately.

`!important` – use with extreme care; using `!important` breaches the structure of your styles design. (In fact, it should not be used at all).

Selector specificity

- Defines priority of selectors
- Rules:
  - Add 1000 points for inline declaration
  - Add 100 for ID
  - Add 10 for a class or a pseudo-class
  - Add 1 for element or pseudo-element selector
- In general: style attribute > ID > classes > elements
- E.g. `a:hover` has a larger priority than `a` alone
- If two selectors have the same priority, the latter overrides the former

In general, the more specific we are in defining a selector, the higher the priority.

E.g. text color for all paragraphs is blue, only one of them – with specified ID – is in red.

Example of computing the specificity:

```
A: h1 {color: black;}
B: #mys h1 {color: red;}
C: <div id="mouse">
     <h1 style="color: #ffffff">House mouse</h1>
   </div>
```

Specificity of A = 1 (1x element)
Specificity of B = 101 (1x ID + 1x element)
Specificity of C = 1000 (inline style)
=> rule C is used

Styling text – fonts

- Font type: `font-family: Arial CE, sans-serif`
- Font size: `font-size: 12px, 12pt, 1.2em` (relative to the size of the current element)

  `font-style: normal|italic`

  `font-weight: normal|bold`

- Shorthand declaration:

  `font: italic bold 20px Arial`

As we can never tell, what fonts are supported by user's browser, we typically provide a list of possible fonts. The fonts are specified in order of preference, separated by commas, from the most specific font to a general serif or sans-serif font.

Styling text

- Text alignment

  `text-align: left|right|center|justify`

- Text decorations

  `text-decoration: none|underline|line-through`

- Text spacing

  `text-indent, line-height, letter-spacing, word-spacing`

Colors

- With CSS, colors can be specified in different ways:

```
Red (Tomato, MediumSeaGreen)
#161616
rgb(0,0,100)
```

- Transparency: `opacity` or `rgba`

```
opacity: 0-1
rgba(255,0,0,0.5)
```

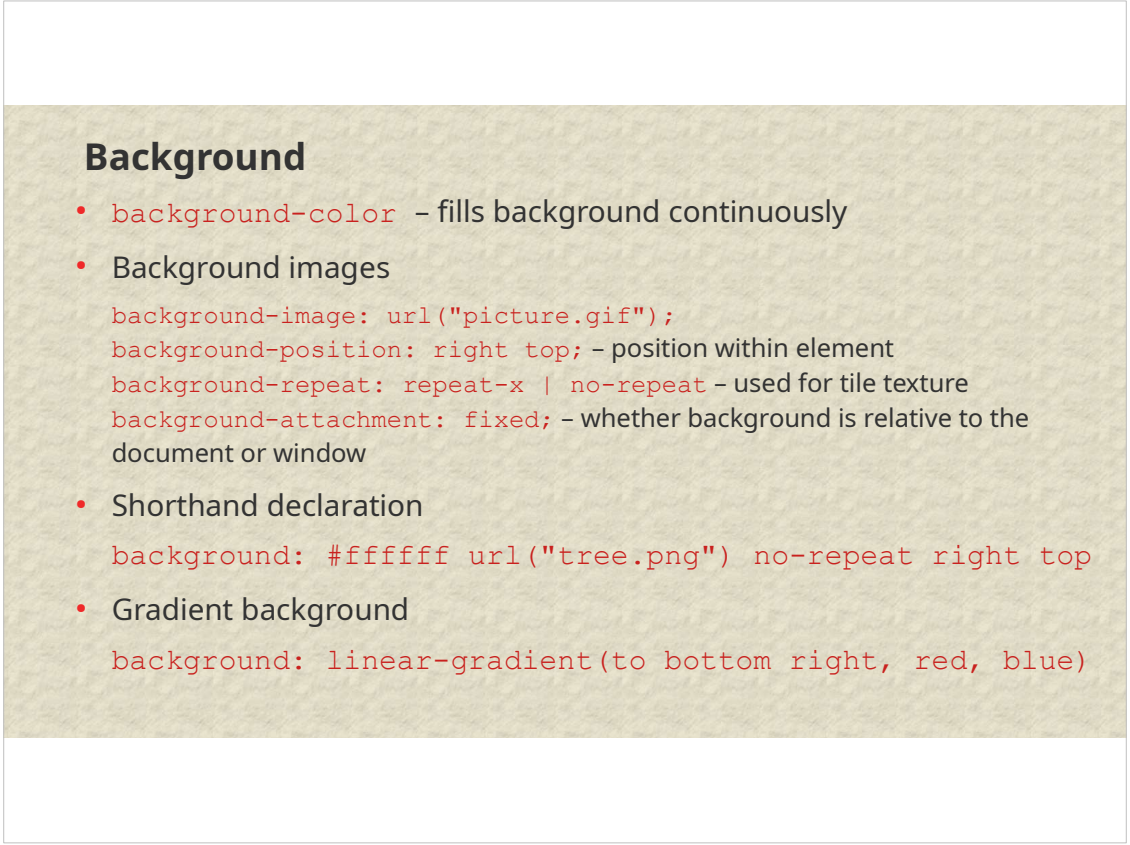
- `color` – foreground color (e.g. text color)

Colors can be specified in different ways:

- By using color names (these are “safe” colors supported by all browsers) https://www.w3schools.com/colors/colors_names.asp
- As hexadecimal values – first two digits stand for red color, next two digits are for green and the last two for blue color.
- Using RGB representation: rgb (Red, Green, Blue)

Note:

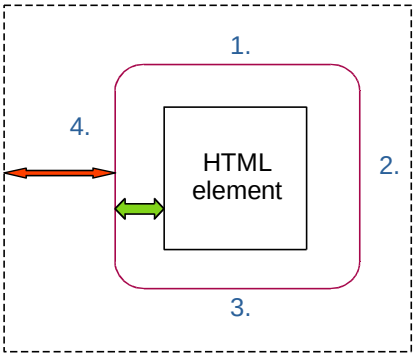
- If all color components – red, green and blue – have the same value, the result is a shade of grey.



Background of the slide above is set using a repeating image. Be careful when using images as backgrounds, the text needs to stay readable.

Box Model

- Padding
  - 10px (one value) – equal padding on all sides of the element
  - 10px 0px (two values) – top-bottom, left-right
  - 0px 5px 10px 5px (four values) – top, right, bottom, left
- Margin
- Border `border-radius:10px`



Element is surrounded by `padding`, `border` and `margin`, in this order.

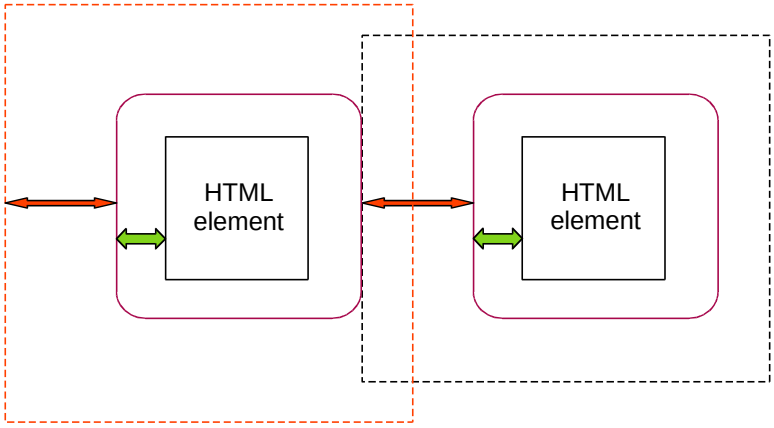
`padding`, can be set using one, two or four values. It is also possible to specify each property separately, using the following properties:

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

The same holds for `margin` and `border` properties.

Box Model

- Margins (typically) collapse – i.e., adjacent margins overlap



Border

- Shorthand declaration: `border: 2px solid blue;`
- `border-style:`
  - `dotted` – dotted border
  - `dashed`
  - `solid`
  - `double`
  - `groove` – 3D grooved border
  - `ridge` – 3D ridged border
  - `inset` – 3D inset
  - `outset` – 3D outset
  - `none` – no border
  - `hidden`
- Mixed border (top, right, bottom, left)

  ```
  border-style: dotted dashed solid double;
  ```
- Table with no border: `border-collapse: collapse;`

Shadows

- Text shadow

  `text-shadow: v-shadow h-shadow blur-radius color`

  - e.g.: `color: white; text-shadow: 2px 2px 4px #000000;`
- Box shadow

  `box-shadow: h-offset v-offset blur spread color`

  - `box-shadow: 3px 10px 10px 5px #555`

Text se stínem

More than one shadow can be applied to a HTML element, single shadow settings are separated by commas.

E.g. following shadow is composed of red and blue shadows:

Text-shadow with red and blue neon glow

```
text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
```

Transformations

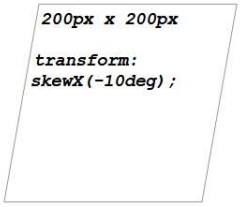
- 2D transformation:

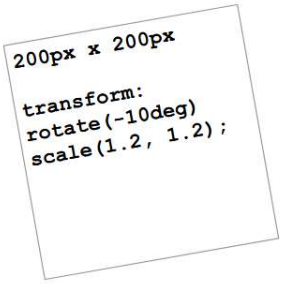
```
translate(x px, y px)
rotate(20deg)
scale(2,3) – 2x wider, 3x longer
skewX(20deg), skewY(20deg), skew(X,Y)
      – 2D skew transformation along the X- and the Y-axis
matrix(scaleX(), skewY(), skewX(), scaleY(),
       translateX(), translateY())
```

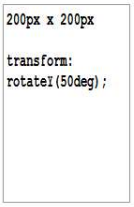
- 3D transformation:

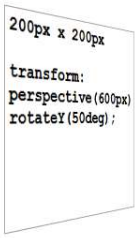
```
rotateX(90deg)
rotateY(90deg)
rotateZ(90deg)
```

Transformation examples









Transitions

`transition: width 2s` – property that changes and duration of the transition

`transition: width 2s, height 3s`

`transition-timing-function:`

- `ease` – default
- `linear`
- `ease-in`
- `ease-out`

`transition-delay: 1s`

`transition: width 2s linear 1s`

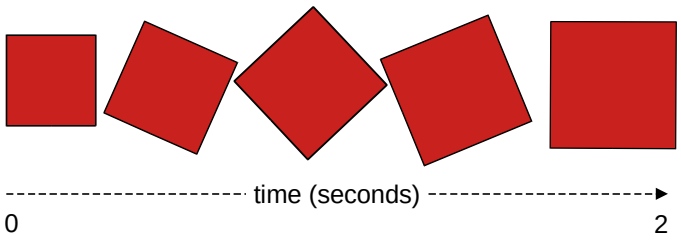
Using transition property, an animation can be created with CSS only. Different properties, like color, size or transparency can be changed.

With transitions, start and end values for the property and the transition duration are set. Also the speed curve of the transition effect can be set, e.g. the transition can start slowly and speed up to the end (`ease-in` value).

Transition – example

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s, height 2s, transform 2s;
}

div:hover {
   width: 300px;
   height: 300px;
   transform:
   rotate(90deg);
}
```



Animation runs on mouse hover over the div.

CSS properties – possible properties values

- Numerical values (size, angle, duration, ...)
  - `font-size: 12pt;`
- Color
  - `background-color: #00ff00;`
- Link to external source (e.g. an image)
  - `background-image: url("paper-texture.png")`
- Strings
  - `font-family: "Courier New";`
- Specific value enumerated in property definition
  - `border-style: solid;`

CSS properties – shorthand declaration

- Many CSS properties can be set using a shorthand declaration
- E.g. setting a border can be done separately for each property:

```
border-width: 2px;
border-style: solid;
border-color: blue;
```

- Or using a shorthand declaration:

```
border: 2px solid blue;
```

CSS properties – units

- All numbers must have a unit (except for 0)

`cm, mm, in`	Centimeters, Millimeters, Inches (1in = 2.54cm)
`px`	Pixels (1px = 1/96 in)
`pt`	Typographical points (1pt =1/72 in)
`pc`	Picas (1pc = 12pt)
`em`	Relative to the font-size of current element
`ex`	Relative to the height of 'x' in current font size
`%`	Special – relative to some existing/inherited value
`vh, vw`	Relative to 1% of width/height of the viewport
`deg`	Degrees (rotation)
`s`	Seconds

Layout and displaying elements on web page

- Basic tools:
  - `float` property
  - Content positioning
  - `display` property

These three tool will be explained on the following slides.

Floating elements

- An element “floats” (on the left or right side of the page), the rest of the contents “flows” around the floating element

  ```
  float: left|right|none
  ```

  Other elements may prevent their content to flow around floating elements, using `clear` property:

  ```
  clear: left; clear: right; clear: both;
  ```

  - `clear` property specifies that on one (or both) sides no element can be floating (the content of `clear`ed element is moved below the floating element)

Float property – example

No element can float next to the second heading. The second heading is therefore rendered bellow the floating image.

Float property – example 2

- The three images have `float: left` set

  → they “float” left, next to each other
- This can be used to create a layout of elements on a web page

Displaying elements

- Each element has a specific way of rendering – inline with the text (`inline`) or as a separate block (`block`)
- `display` property can override default behavior

  `display: block|inline`

  `display: none`

  `display: inline-block` – same as inline + width, height or borders can be set

- `visibility: hidden|visible` – space for the element is reserved on the page although the element is not visible

Content positioning

- The elements are rendered in the same order they are defined in the source code
  - Except for `float`ing elements
- This behavior can be modified by positioning

```
position: static|relative|fixed|absolute|sticky
```

- `z-index` property specifies a layer in which the element is rendered (for overlapping elements)

Single `position` values will be explained in the following slides.

Content positioning

- `static` – default value
- `relative` – element is moved relative to its computed position after the layout is created
  - Following properties can be set:
    `top`, `right`, `bottom`, `left`

- `fixed` – relative to a viewport – a device on which the page is displayed; i.e. fixed elements stay at the same place while the user is scrolling
  - Position is set using `top`, `right`, `bottom`, `left` properties

`top`, `right`, `bottom`, `left` properties set the distance from the specified edge.

Content positioning

...

- `absolute` – element is positioned inside closest positioned ancestor
- `sticky` – A sticky element toggles between `relative` and `fixed`, depending on the scroll position. It is positioned `relative` until a given offset position is met in the viewport - then it "sticks" in place (like `position:fixed`).
  - "Fixed" position is set by `top`, `right`, `bottom`, `left`

A positioned element is an element that has `position` property set to any value except for `static`.

CSS positioning properties

- `width`, `height` – width and height of the element

- `min-width`, `max-width` – minimum and maximum width
- `min-height`, `max-height` – minimum and maximum height

- `top`, `right`, `bottom`, `left` – distance from the specified edge

If the image size is set by

```
max-width: 100%,
```

it prevents the image from scaling to larger than its original size. If left out, height property is set automatically.

Web page layout

- = visual structure of HTML elements or their blocks
- Many different approaches
  - Whether the page scrolls as whole or not
  - How each container handles content overflow
  - ...

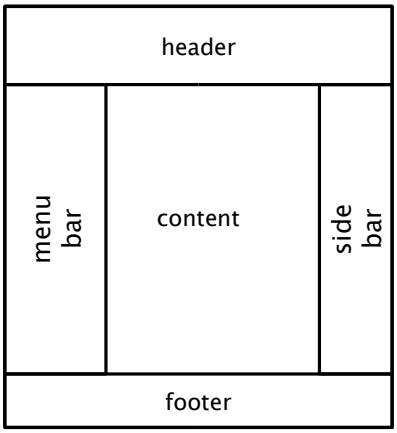


Figure on the right displays a typical web page layout with side bars, header and a footer. This is so called “holy grail” web layout. For a long time it was not easy to implement this type of layout.

Web page layout – different approaches

- Different approaches to web page layout with side bars
  - Tables – DO **NOT** USE
  - Using CSS styles
    - Using float property
    - Using content positioning
    - Using flexbox and grid – modern features of CSS – **YES!** (beyond the scope of this lecture)

- Layout using HTML tables
  - Goes against HTML philosophy of elements' semantic meaning
  - Very inflexible when making the slightest change in layout
  - Static layout, fixed width
- Floating elements
  - Easy to design
  - It is a little bit tricky to ensure correct sidebar height
- Content positioning
  - Sidebars (or other elements) can cover the content underneath
  - Sidebars can be almost anywhere in the document
- Flexbox, grid
  - Modern features of CSS (not mentioned in slides)

@media queries

- It is possible to define styles only if a certain condition is true – e.g. a type of device on which the page is displayed, or its properties (e.g. its size)
- Type of display: all, screen, print, speech

```
@media print { …applied when page is printed… }
@media screen { …applied when page is displayed
                on a screen… }
```

  - More properties:
    - Display (viewport) size, display orientation, color depth, …
- Can be used
  - In a CSS file
  - When linking a .css file in `<link>` element inside `<head>`

Using `@media` query it is possible to declare styles for different types of displaying the web page, e.g. styles for screen (display), different sizes of display or special styles for printing the page.

@media

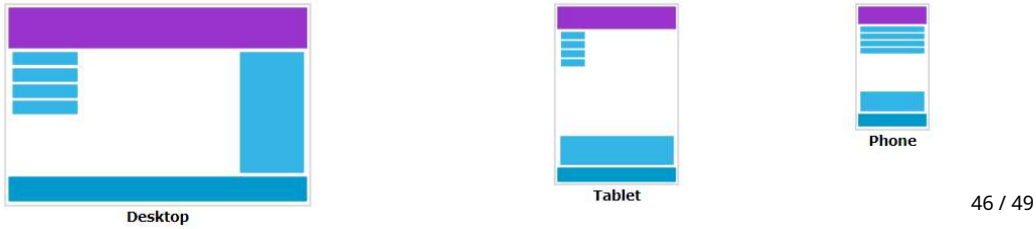
- and or comma (=or) operators can be used to join conditions

```
@media screen and (min-width: 480px) {
    CSS rules...
}
```

- Declaration block is applied when the page is displayed on screen and the window width is larger than 480px

Responsive design

- Goal: the page must look good on all devices (desktops, tablets, phones)
- How? The same content, different element sizes, different layout
- How exactly?
  - Relative sizes – in % or in `vh`, `vw` units
  - Different layouts (CSS styles) for different devices (using `@media` query)



On small devices/displays, typically all content blocks spread to 100% of display width.

Modern approach is to design for mobile first – before designing for desktop or any other device (using `@media` query). This will make the page display faster on smaller devices, which often have slower internet connection.

`vh` and `vw` units mean viewport height and viewport width.

Viewport

- The viewport is the user's visible area of a web page
  - E.g. phone display size, browser window size

- In HTML5 viewport can be set in page `<head>`

```
<meta name="viewport" content="width=device-width,
                               initial-scale=1.0">
```

- `width=device_width` sets the page width to the width of the device on which the page is displayed
- `initial-scale` sets the initial scale

Before tablets and mobile phones, web pages were designed only for computer screens. It was common for web pages to have a static design and a fixed size.

Browsers on smaller devices scaled down the entire web page to fit the screen. (This was not perfect! But a quick fix.)

Setting viewport – example

Questions...

?