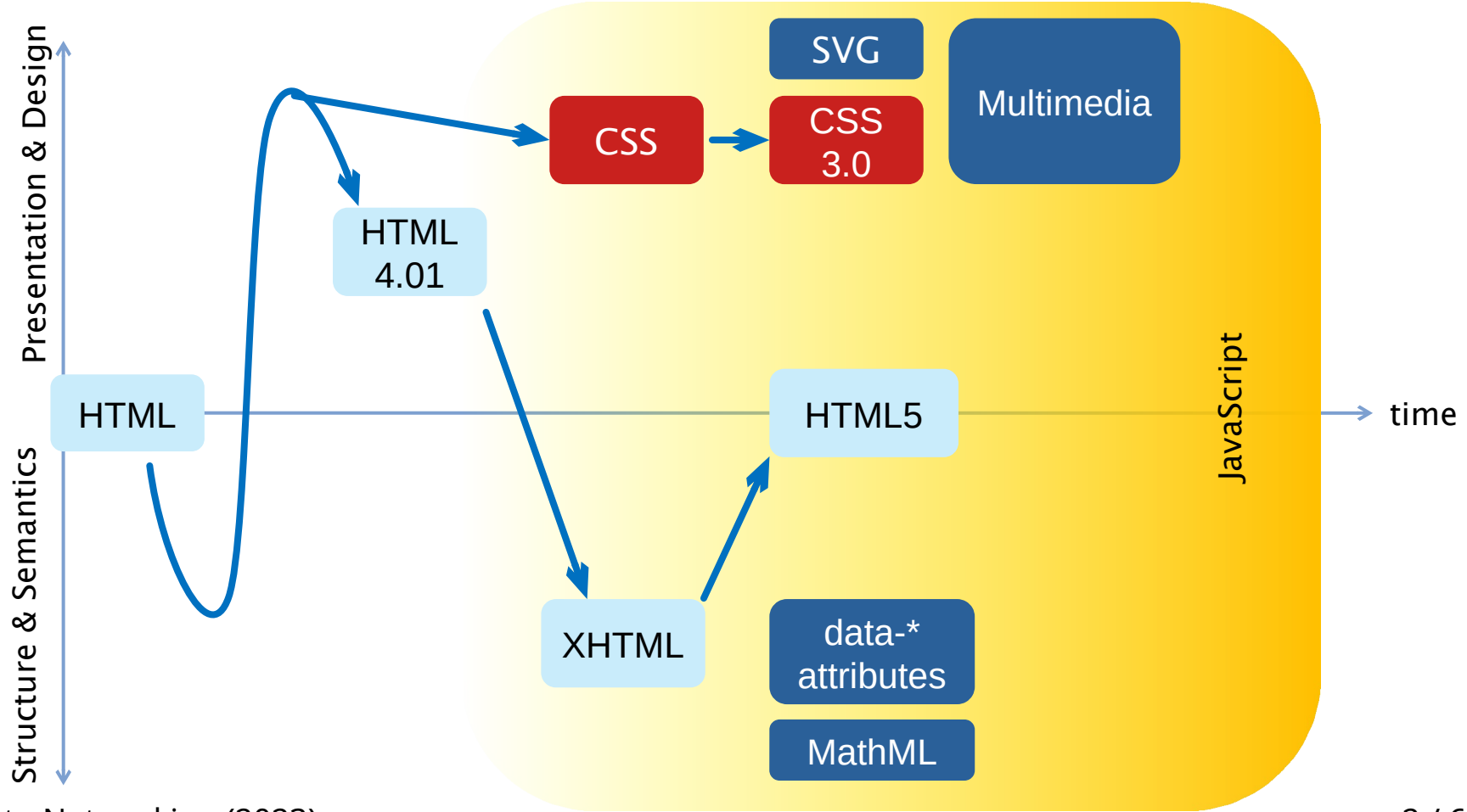
Cascading Style Sheet

Introduction to networking

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

Evolution of Web Presentation



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

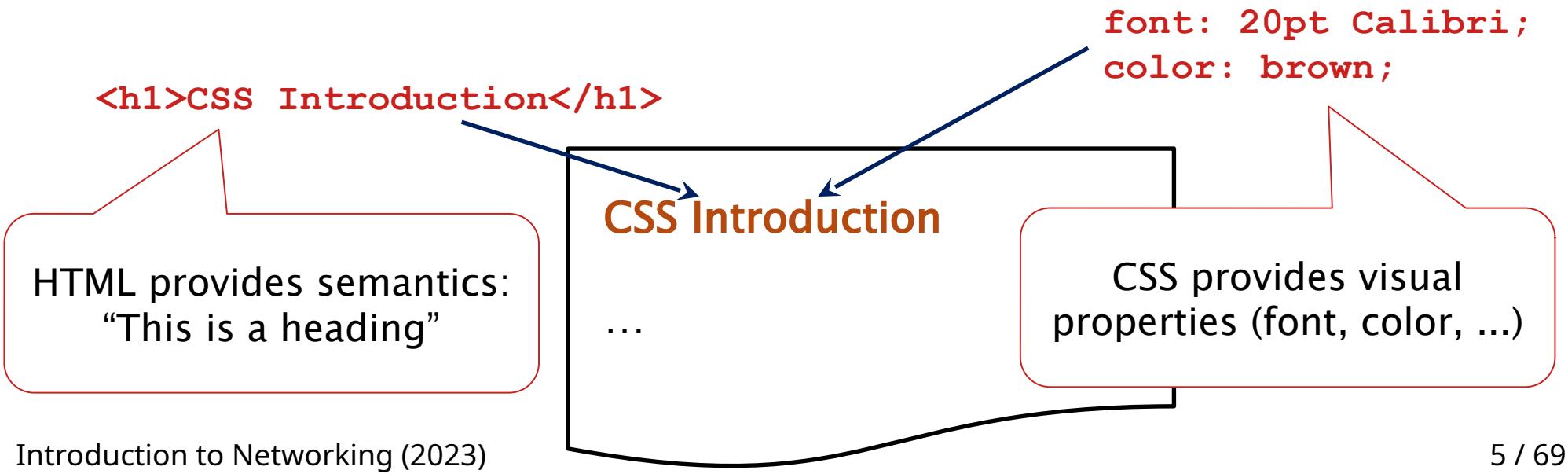
CSS Versions

- 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



CSS syntax

- Format of CSS rules:

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

- Example:

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

Three ways of adding CSS to HTML

- Inline:

```
<p style="text-align: center;">Text</p>
```

- Embedding in HTML `<head>` element, as a content of `<style>` element

```
<head>
   <style>
      p {text-align: center;}
   </style>
</head>
```

- Linking a .css file:

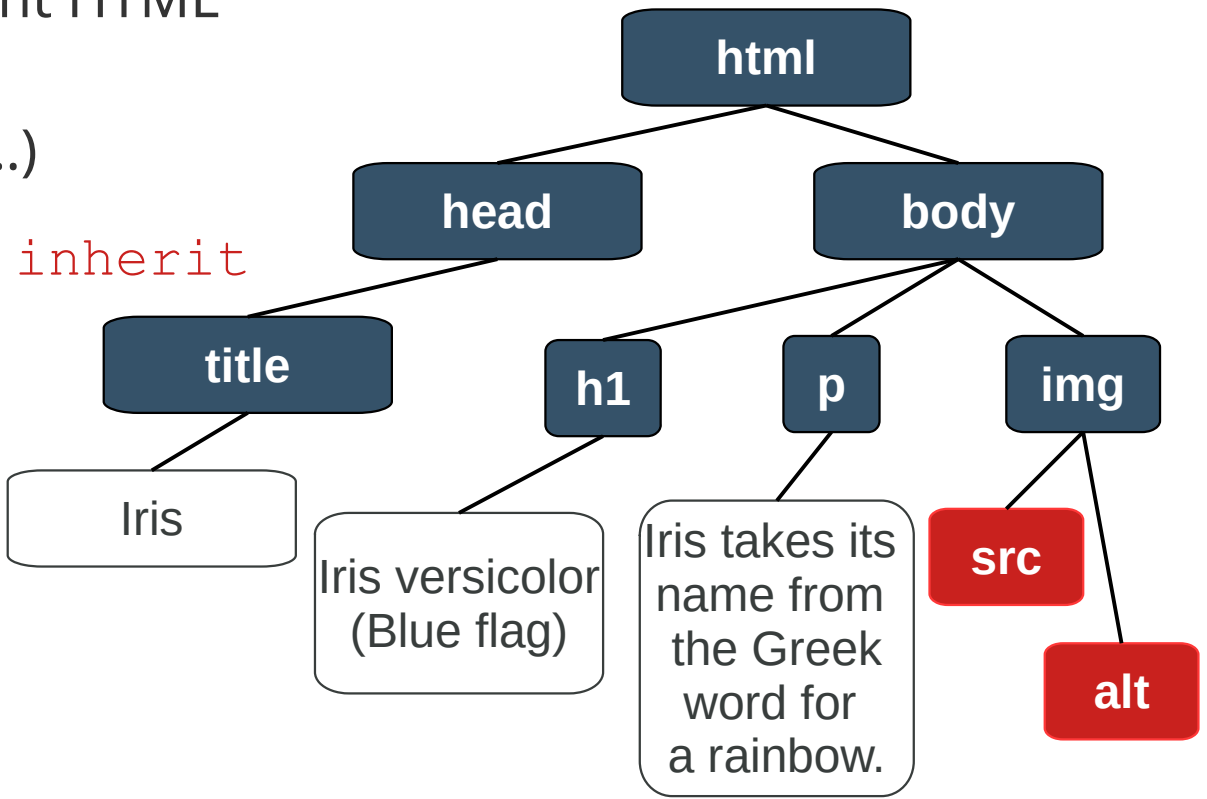
```
<head>
   <link rel="stylesheet"
         type="text/css"
         href="styles.css">
</head>
```

styles.css

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

body { color: blue; }
```

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



CSS selectors

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 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`

Combining selectors

```
div.info { … }

p.info { … }

li+li { … }

li+li+li { … }

section.small p { … }

section.small > p {…}
```

```
<section>
  <div class="info">...</div>
  <p class="info">...</p>
  <ul>
    <li>first item</li>
    <li>second item</li>
    <li>third item</li>
    <li>fourth item</li>
  </ul>
</section>
<section class="small">
  <section>
    <p>Paragraph of smaller text</p>
  </section>
  <p>Another one of smaller text</p>
</section>
```

Combining selectors

```
div.info { … }

p.info { … }

li+li { … }

li+li+li { … }

section.small p { … }

section.small > p {…}
```

```
<section>
  <div class="info">...</div>
  <p class="info">...</p>
  <ul>
    <li>first item</li>
    <li>second item</li>
    <li>third item</li>
    <li>fourth item</li>
  </ul>
</section>
<section class="small">
  <section>
    <p>Paragraph of smaller text</p>
  </section>
  <p>Another one of smaller text</p>
</section>
```

Combining selectors

```
div.info { … }

p.info { … }

li+li { … }

li+li+li { … }

section.small p { … }

section.small > p {…}
```

```
<section>
  <div class="info">...</div>
  <p class="info">...</p>
  <ul>
    <li>first item</li>
    <li>second item</li>
    <li>third item</li>
    <li>fourth item</li>
  </ul>
</section>
<section class="small">
  <section>
    <p>Paragraph of smaller text</p>
  </section>
  <p>Another one of smaller text</p>
</section>
```

Combining selectors

`E F`	elements `F`, which have ancestor `E`
`E>F`	elements `F`, which have parent `E`
`E+F`	elements `F`, which are immediately preceded by `E`
`E~F`	elements `F`, which are preceded by `E`

```
div.info { … }

p.info { … }

li+li { … }

li+li+li { … }

section.small p { … }

section.small > p {…}
```

```
<section>
<div class="info">...</div>
<p class="info">...</p>
  <ul>
    <li>first item</li>
    <li>second item</li>
    <li>third item</li>
    <li>fourth item</li>
  </ul>
</section>
<section class="small">
  <section>
    <p>Paragraph of smaller text</p>
  </section>
  <p>Another one of smaller text</p>
</section>
```

Combining selectors

`E F`	elements `F`, which have ancestor `E`
`E>F`	elements `F`, which have parent `E`
`E+F`	elements `F`, which are immediately preceded by `E`
`E~F`	elements `F`, which are preceded by `E`

```
div.info { … }

p.info { … }

li+li { … }

li+li+li { … }

section.small p { … }

section.small > p {…}
```

```
<section>
<div class="info">...</div>
<p class="info">...</p>
  <ul>
    <li>first item</li>
    <li>second item</li>
    <li>third item</li>
    <li>fourth item</li>
  </ul>
</section>
<section class="small">
  <section>
    <p>Paragraph of smaller text</p>
  </section>
  <p>Another one of smaller text</p>
</section>
```

Combining selectors

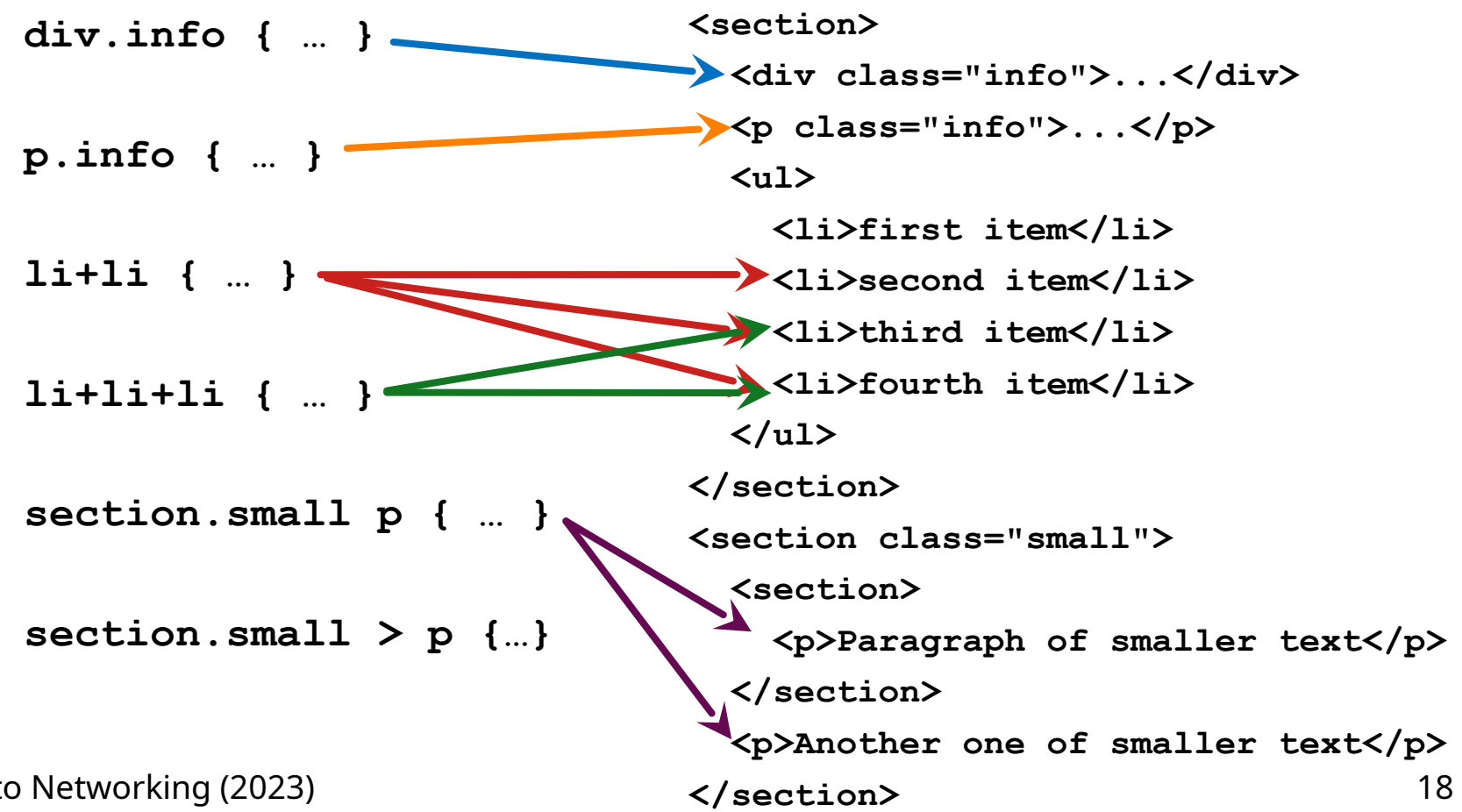
`E F`	elements `F`, which have ancestor `E`
`E>F`	elements `F`, which have parent `E`
`E+F`	elements `F`, which are immediately preceded by `E`
`E~F`	elements `F`, which are preceded by `E`

```
div.info { … }

p.info { … }

li+li { … }

li+li+li { … }

section.small p { … }

section.small > p {…}
```

```
<section>
<div class="info">...</div>
<p class="info">...</p>
  <ul>
    <li>first item</li>
<li>second item</li>
<li>third item</li>
<li>fourth item</li>
  </ul>
</section>
<section class="small">
  <section>
    <p>Paragraph of smaller text</p>
  </section>
  <p>Another one of smaller text</p>
</section>
```

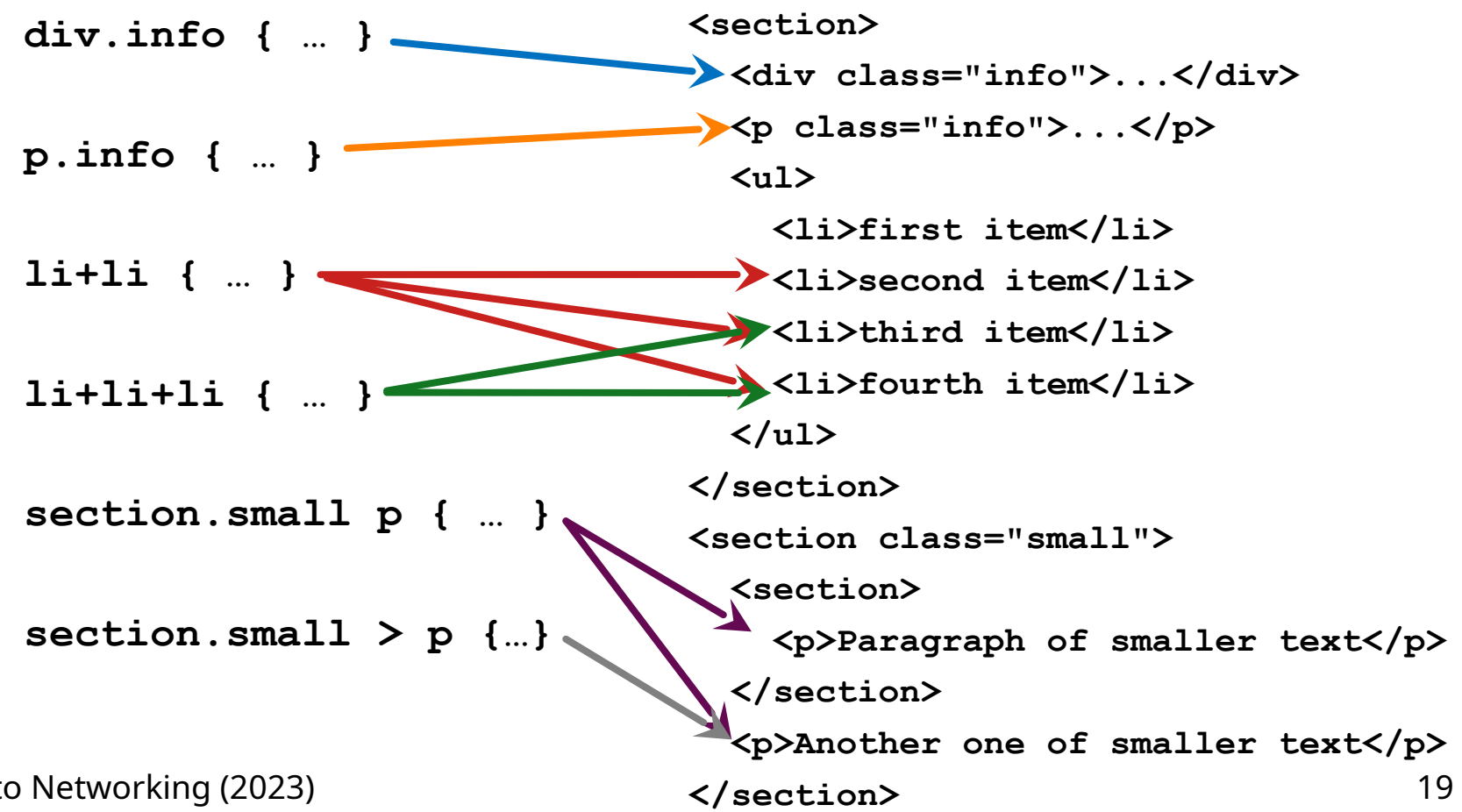
Combining selectors

`E F`	elements `F`, which have ancestor `E`
`E>F`	elements `F`, which have parent `E`
`E+F`	elements `F`, which are immediately preceded by `E`
`E~F`	elements `F`, which are preceded by `E`



Combining selectors

`E F`	elements `F`, which have ancestor `E`
`E>F`	elements `F`, which have parent `E`
`E+F`	elements `F`, which are immediately preceded by `E`
`E~F`	elements `F`, which are preceded by `E`



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)

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`

Pseudo-classes – example

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

Jméno	Příjmení	Body
Petr	Smutný	100
Ludvík	Veselý	150
Jiří	Černý	67
Klement	Nový	250

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;}
```

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;
```

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

CSS properties

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`

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)

Background

- `background-color` – fills background continuously
- Background images

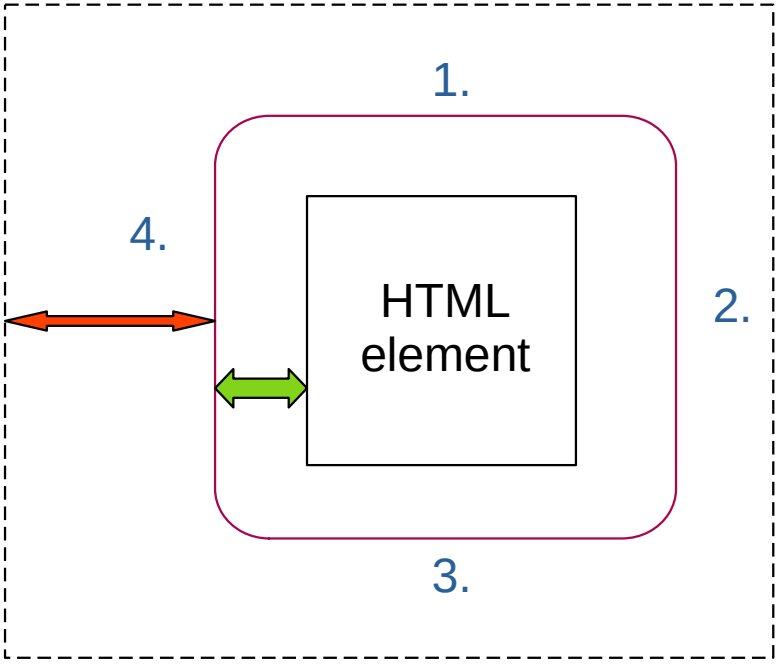
  `background-image: url("picture.gif");`
  `background-position: right top;` – position within element
  `background-repeat: repeat-x | no-repeat` – used for tile texture
  `background-attachment: fixed;` – whether background is relative to the document or window
- Shorthand declaration

  ```
  background: #ffffff url("tree.png") no-repeat right top
  ```
- Gradient background

  ```
  background: linear-gradient(to bottom right, red, blue)
  ```

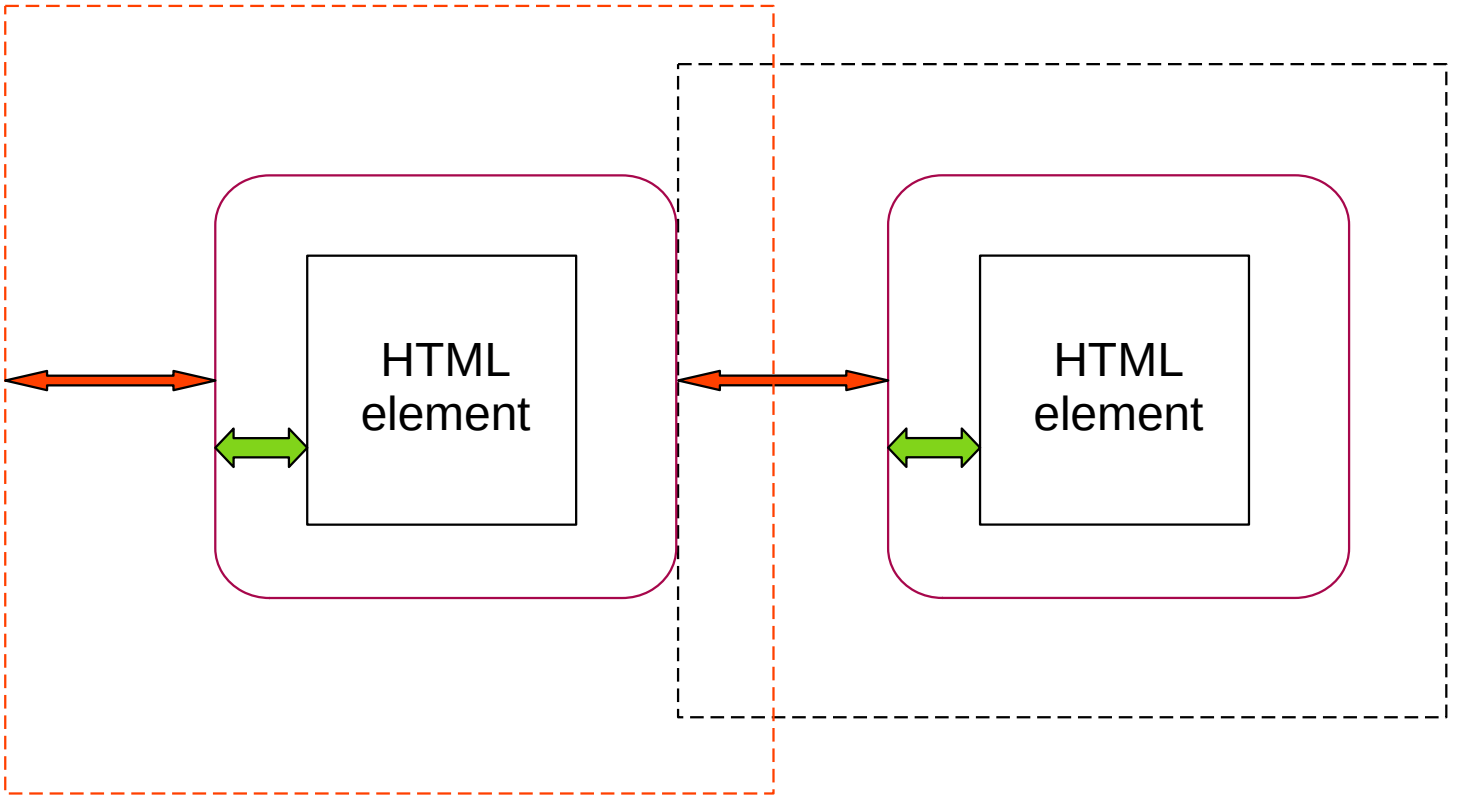
Box Model

- 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`



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

Transformations

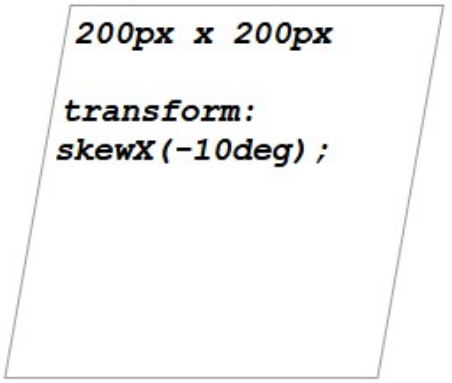
- 2D transformation:

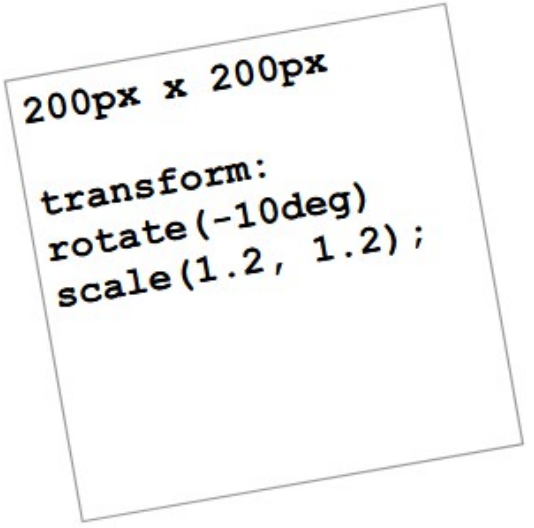
```
translate(x px, y px)
rotate(20deg)
scale(2,3) – 2x wider, 3x longer
skewX(20deg), skewY(20deg), skew(X,Y)
      – 2D skew transformation along the X- and the Y-axis
matrix(scaleX(), skewY(), skewX(), scaleY(),
       translateX(), translateY())
```

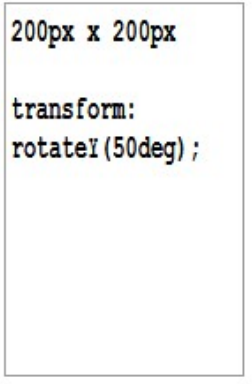
- 3D transformation:

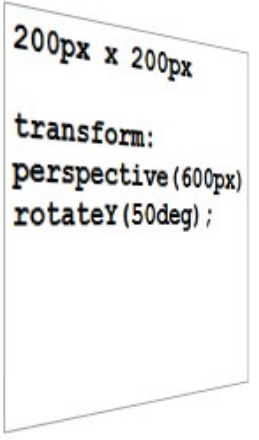
```
rotateX(90deg)
rotateY(90deg)
rotateZ(90deg)
```

Transformation examples









Transitions

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

```
transition: width 2s, height 3s
```

```
transition-timing-function:
```

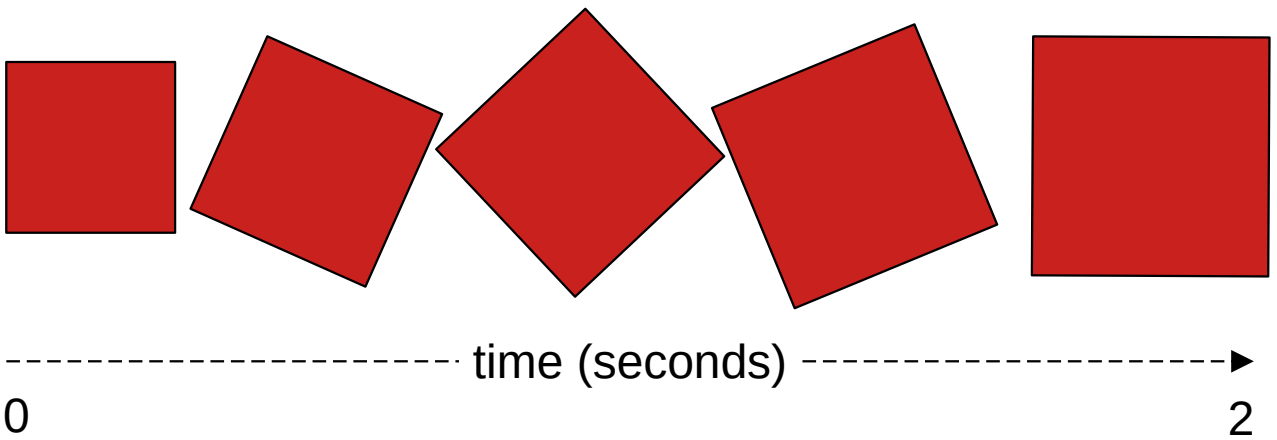
- `ease` – default
- `linear`
- `ease-in`
- `ease-out`

```
transition-delay: 1s
```

```
transition: width 2s linear 1s
```

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);
}
```



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

Layout and displaying elements on web page

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

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

Libějovické Svobodné Hory

Libějovické Svobodné Hory jsou malá vesnice, část obce Stožice v okrese Strakonice. Nachází se asi 2,5 km na jihozápad od Stožic, pod Svobodnou horou. Je zde evidováno 22 adres. V roce 2011 zde trvale žilo 43 obyvatel.

Libějovické Svobodné Hory leží v katastrálním území Křepice u Vodňan o výměře 3,99 km^2.

První písemná zmínka o vesnici pochází z roku 1840.

Pamětihodnosti: Zemědělský dvůr Jarov (kulturní památka ČR), Socha svatého Jiří v lese jižně od vsi, Dva křížky v jižní části vesnice

Bavorovské Svobodné Hory

`float:left`

`float:right`

`clear:both`

Float property – example 2

- The three images have `float: left` set

  → they “float” left, next to each other

- This property was 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

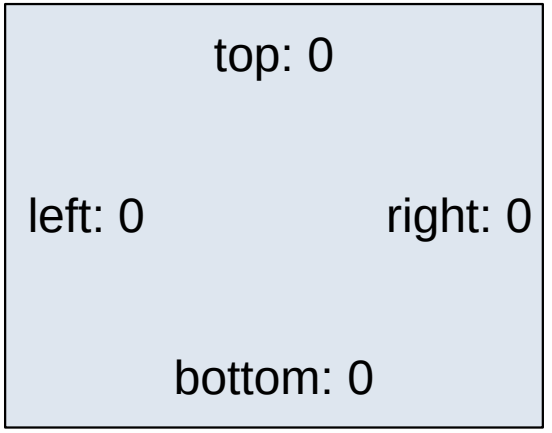
- The elements are rendered in the same order they are defined in the source code
  - Except for `float`ing elements
- This behavior can be modified by positioning

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

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

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

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`

CSS positioning properties

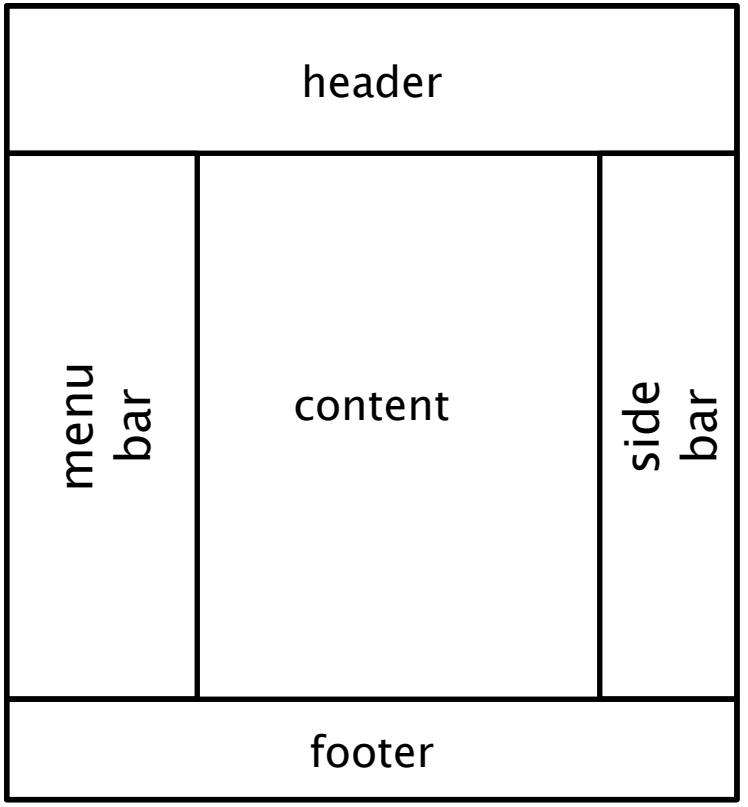
- `width`, `height` – width and height of the element

- `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

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
  - …



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!**

Flexbox, grid

- Flex
  - 1dimensional layout
  - Flexible size of the elements

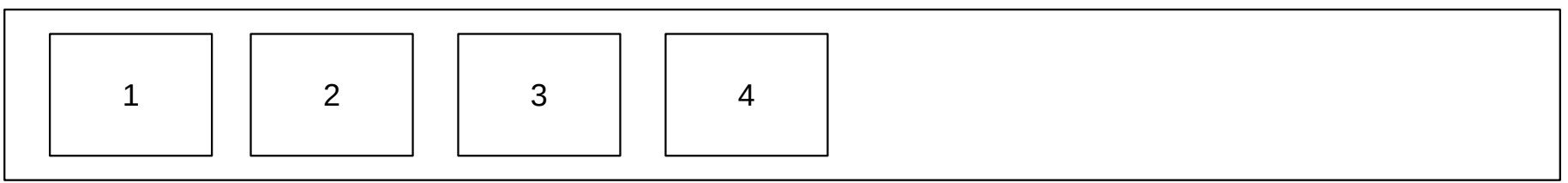
- Grid
  - 2D layout

Flexbox

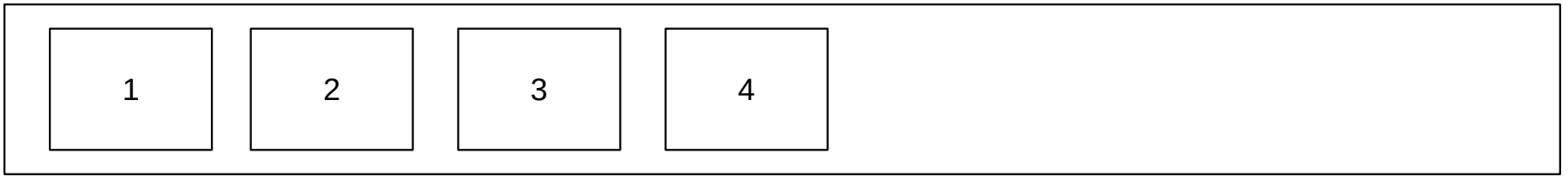
- Layout is created by one parent element and one or more children
- container
- items – elements that are to be arranged inside of the container

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
</div>
```

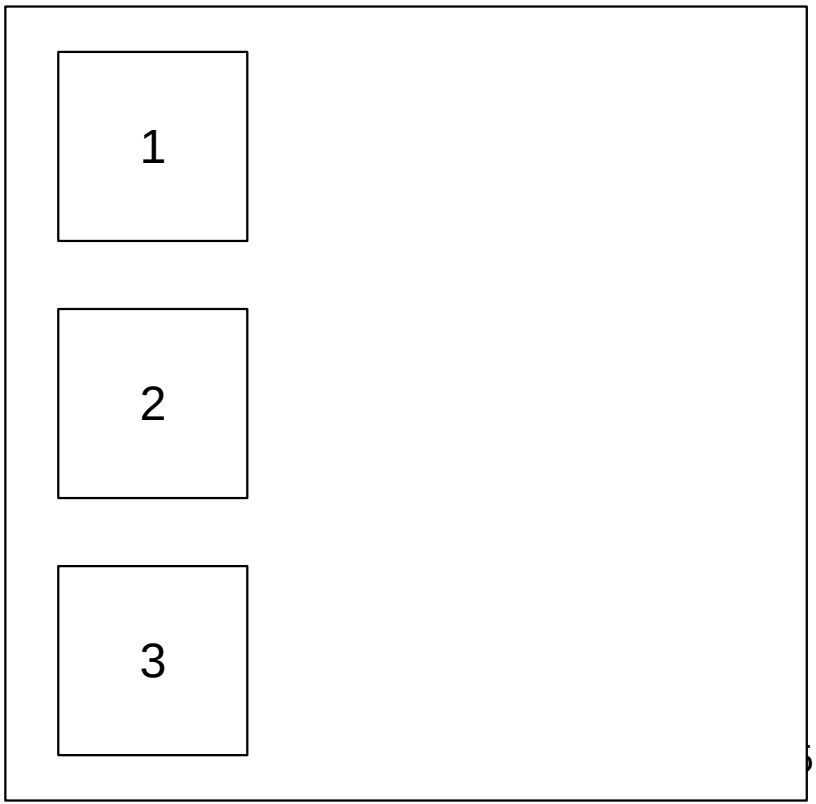
```
.flex-container {
  display: flex;
}
```



Flexbox

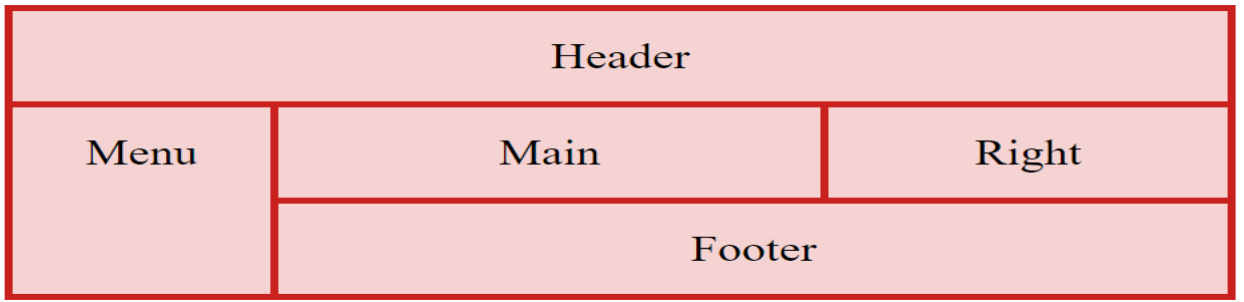


- Flexbox properties determine the way the items are put inside the container
- `flex-direction`
  - `row, column`
- `flex-wrap`
- `justify-content`



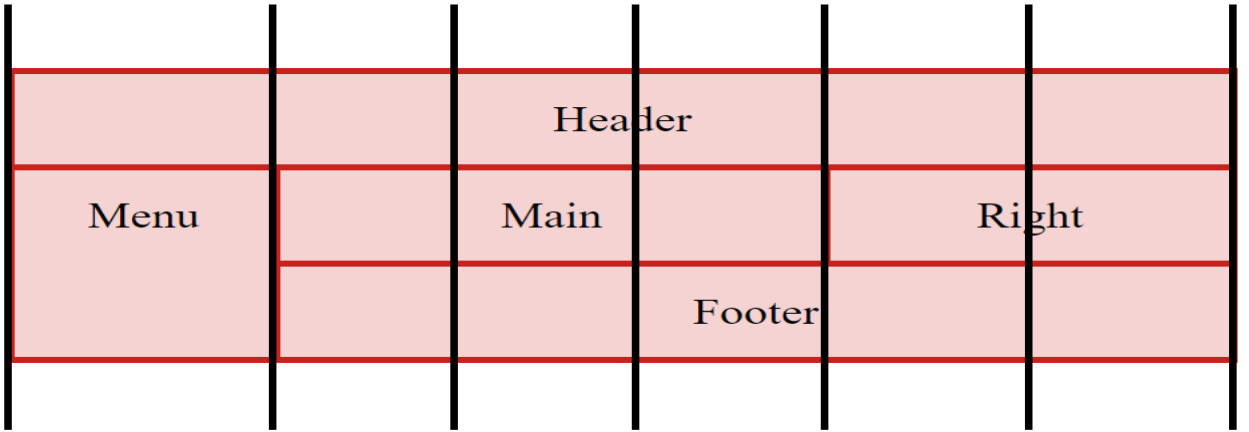
grid

- Elements are part of a grid
- A modern approach (instead of floating and positioning)



grid

- Elements are part of a grid
- A modern approach (instead of floating and positioning)
- layout is split to columns



grid

- Layout is created by one parent element (container) and one or more children (items)
  - Items are automatically part of the grid

1	2
3	4
5	6

grid

- Layout is created by one parent element (container) and one or more children (items)
  - Items are automatically part of the grid

1	2
3	4
5	6

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
</div>
```

grid

- Layout is created by one parent element (container) and one or more children (items)
  - Items are automatically part of the grid

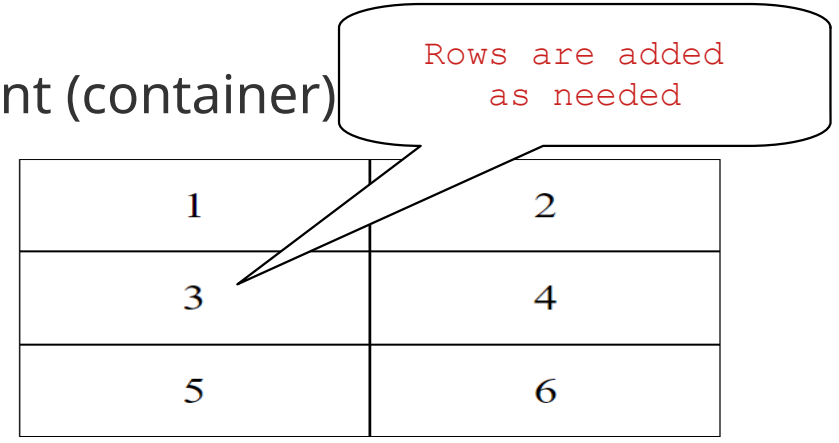
1	2
3	4
5	6

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
</div>
```

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto;
}
.grid-item {
  border: 1px solid black;
  ...
}
```

grid

- Layout is created by one parent element (container) children (items)
  - Items are automatically part of the grid

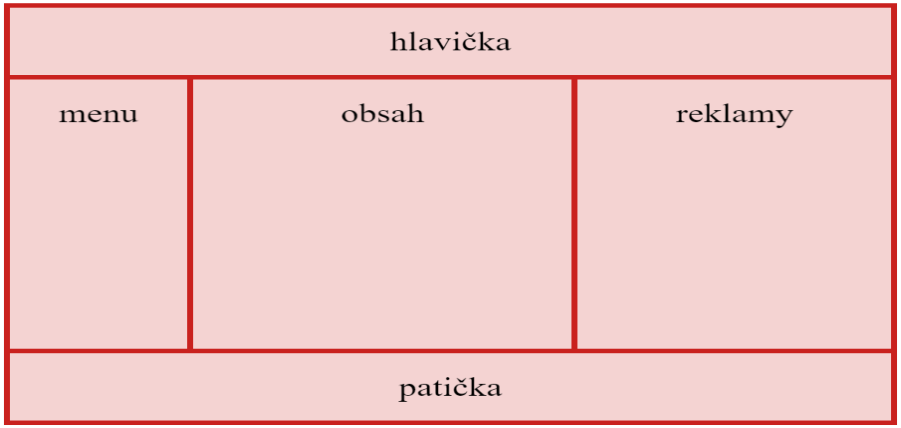



```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
</div>
```

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto;
}
.grid-item {
  border: 1px solid black;
  ...
}
```

"Holly grail" layout using `grid`

```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }

.grid-container {
  grid-template-areas:
    'header header header header header header'
    'menu main main main right right'
    'footer footer footer footer footer
footer';
}
```



@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>`

@media

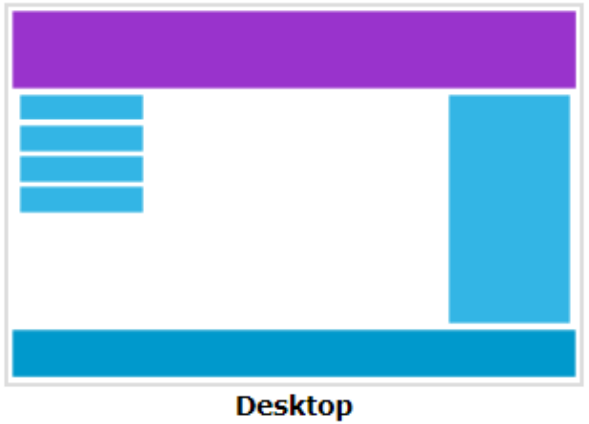
- and or comma (=or) operators can be used to join conditions

```
@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)



Viewport

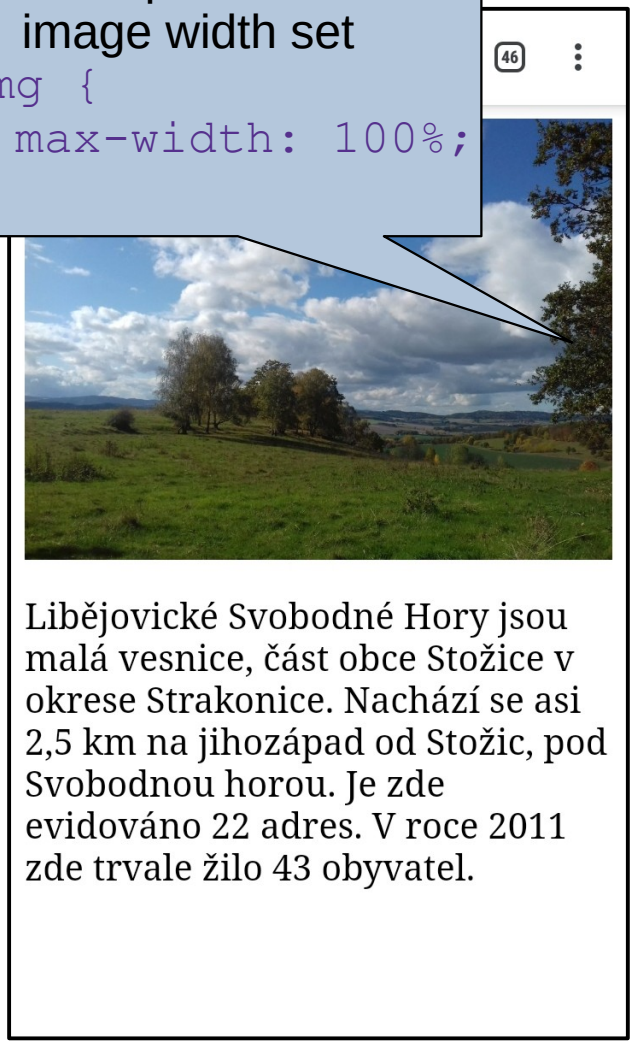
- The viewport is the user's visible area of a web page
  - E.g. phone display size, browser window size

- 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

Setting viewport – example



Questions...

?