Introduction to HTML and CSS

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University
https://softuni.bg

Table of Contents

Have a Question?

Software University

sli.do

#html-css

Introduction to HTML

What is HTML?

- The **HTML** language describes **Web content** (Web pages)
  - Text with formatting, images, lists, hyperlinks, tables, forms, etc.
  - Uses **tags** to define **elements** in the Web page

```
<p>
  <b>Document</b> content goes here…
</p>
```

Opening tag

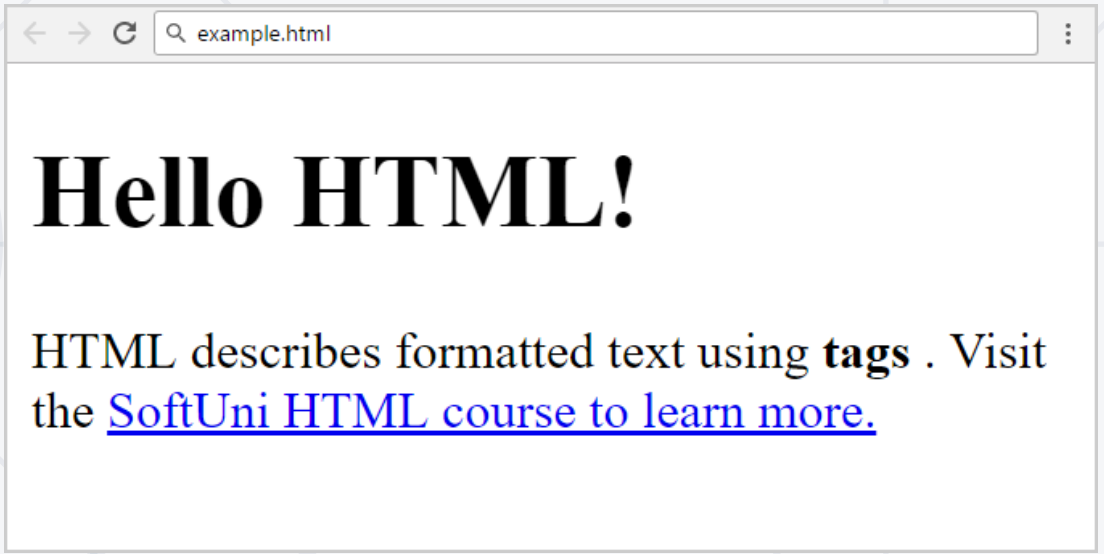
Element

Closing tag

Document content goes here…

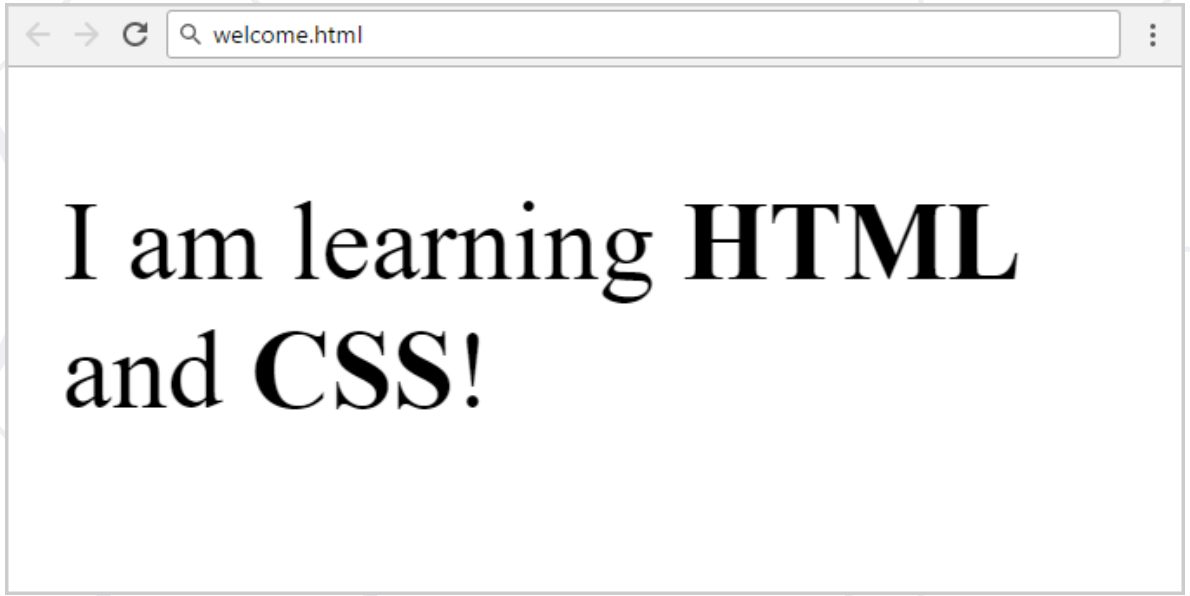
HTML Page – Example

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HTML Example</title>
  </head>
  <body>
    <h1>Hello HTML!</h1>
    <p>HTML describes formatted text using <strong>tags
      </strong>. Visit the
        <a href="https://softuni.bg/trainings/
      courses">SoftUni HTML course to learn more.</a></p>
  </body>
</html>
```



Problem: Welcome to HTML

- Create your first HTML page
  - File name: **welcome.html**
  - Title: **Welcome**
  - Paragraph of text:
    I am learning **HTML** and **CSS**!



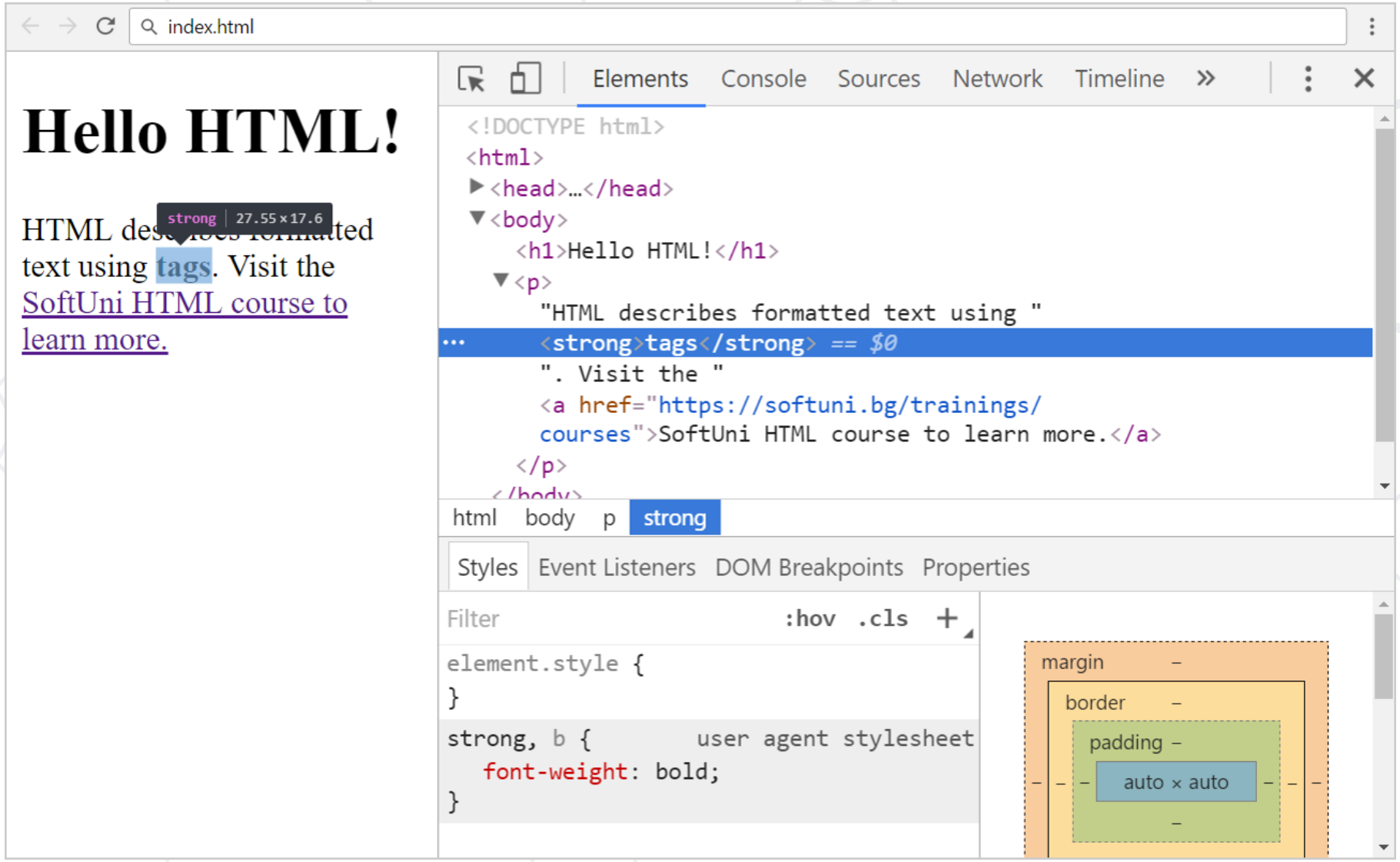
- Hints:
  - Modify the code from the previous slide, use **** tag
  - Submit the page in the judge: **welcome.html** in a **ZIP file**

HTML – Developer Environments

- **Visual Studio Code, Brackets, NetBeans**
  - Good free tools for HTML5, cross-platform
- **WebStorm**
  - Powerful IDE for HTML, CSS and JavaScript, paid product
- **Visual Studio**
  - Many languages and technologies, Windows & Mac
- **Sublime Text, Vim, Notepad++**
  - For hackers

Developer Tools: [F12] in the Browser

Software University



Zen Coding (Emmet) for Fast HTML Coding

```
ul>li.red*6
```

```
<ul>
  <li class="red"></li>
  <li class="red"></li>
  <li class="red"></li>
  <li class="red"></li>
  <li class="red"></li>
  <li class="red"></li>
</ul>
```

```
div#page>div.logo+ul#menu>li*3>a
```

```
<div id="page">
  <div class="logo"></div>
  <ul id="menu">
    <li><a href=""></a></li>
    <li><a href=""></a></li>
    <li><a href=""></a></li>
  </ul>
</div>
```

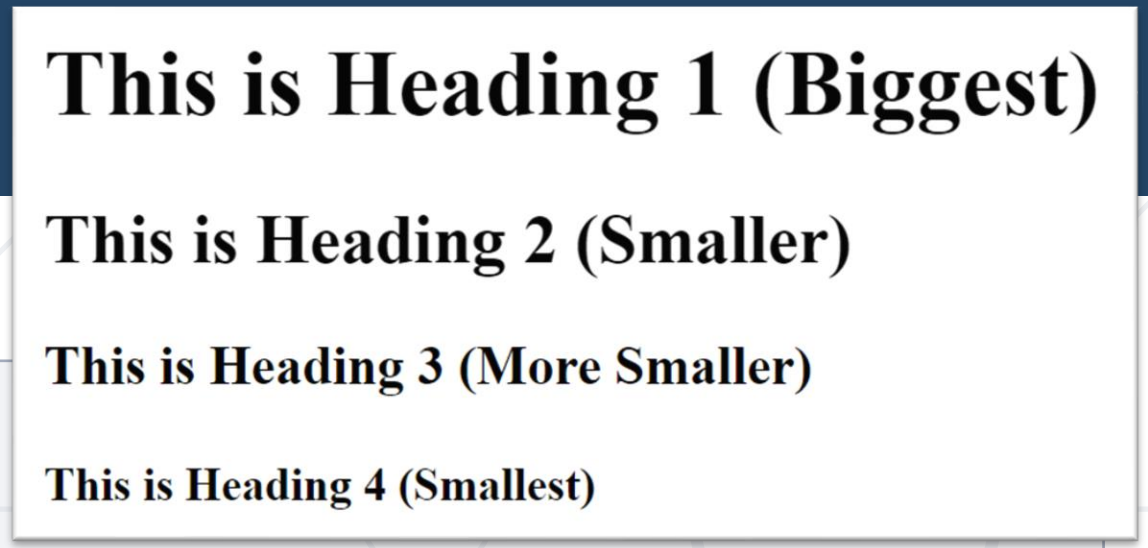
HTML Common Elements

Used in 90% of All Internet Sites

Headings and Paragraphs

- Headings: <h1> to <h6>

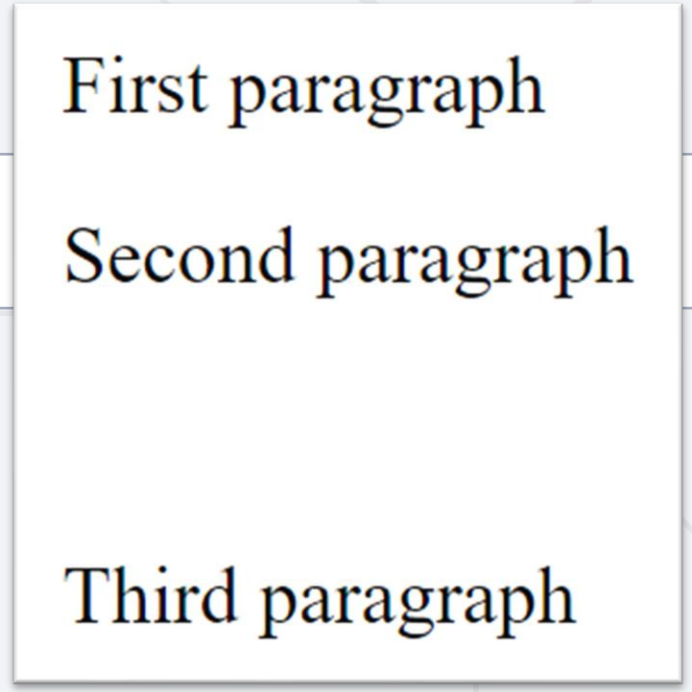
```
<h1>This is Heading 1 (Biggest)</h1>
<h2>This is Heading 2 (Smaller)</h2>
<h3>This is Heading 3 (More Smaller)</h3>
<h4>This is Heading 4 (Smallest)</h4>
```



- Paragraphs:

```
<p>First paragraph</p>
<p>Second paragraph</p>
<br /> <!-- empty line -->
<p>Third paragraph</p>
```

Comment



Hyperlinks

- External hyperlink

Specify the URL

```
<a href="https://softuni.bg">SoftUni</a>
```

SoftUni

- Local hyperlink

```
<h1 id="exercises">Exercises</h1>
…
See the <a href="#exercises" target="_blank">exercises</a>
```

Exercises

See the exercises

- Relative hyperlink

```
<a href="../2.%20HTML5-Overview.pptx">presentation</a>
```

presentation

Images

- Images are **external files**, inserted through the `<img>` tag

```
<img
  src="images/SoftUni-logo.png"
  width="400"
  height="313" />
```

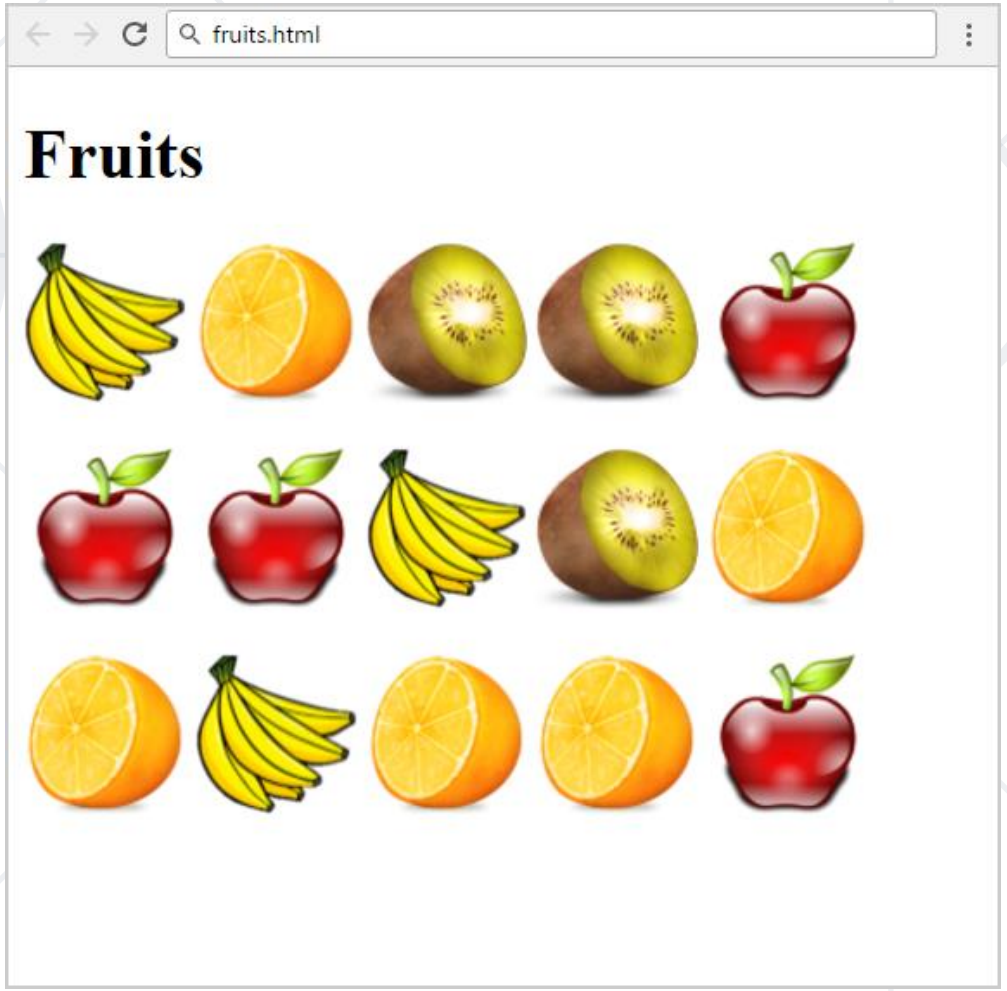
- Embedded image (**data URI**)

```
<img src=" R0lGODlhEAAOAL…"/>
```

  - Example: https://codepen.io/snakov/pen/poNxXry

Problem: Fruits

- You are given 4 image files:
  - `apple.png`, `banana.png`, `kiwi.png`, `organge.png`
- Create a Web page like the screenshot on the right
  - Hints: use 3 paragraphs, each holding 5 images



Ordered Lists: <Ol> Tag

- Create an **O**rdered **L**ist
- Use `<ol></ol>`
  - Each holding `<li></li>`

```
<ol type="1">
  <li>One</li>
  <li>Two</li>
  <li>Three</li>
</ol>
```

1. One
2. Two
3. Three

- Attribute values for `type` are `1`, `A`, `a`, `I`, or `i`

1. One
2. Two
3. Three

A. One
B. Two
C. Three

a. One
b. Two
c. Three

I. One
II. Two
III. Three

i. One
ii. Two
iii. Three

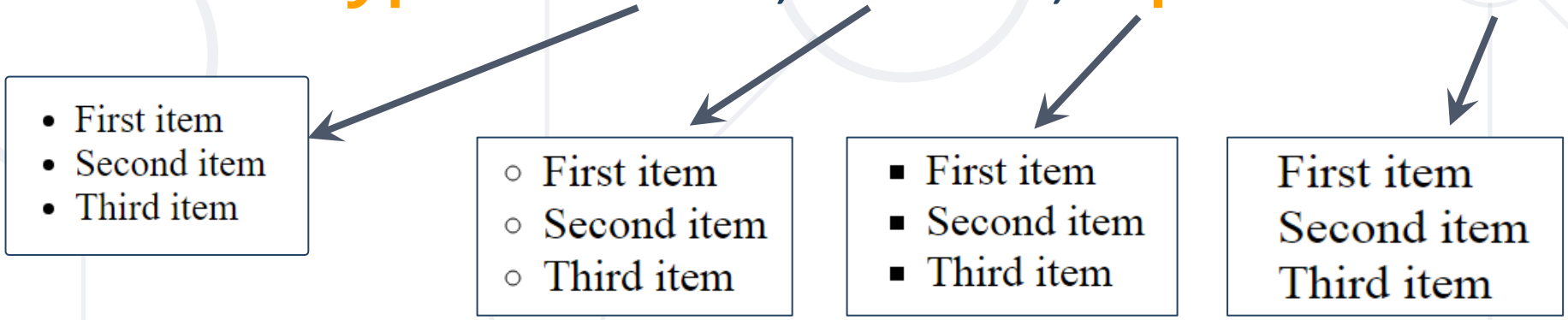
Unordered Lists: <Ul> Tag

- Create an Unordered List using `<ul></ul>`:

```
<ul type="disc">
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ul>
```

- First item
- Second item
- Third item

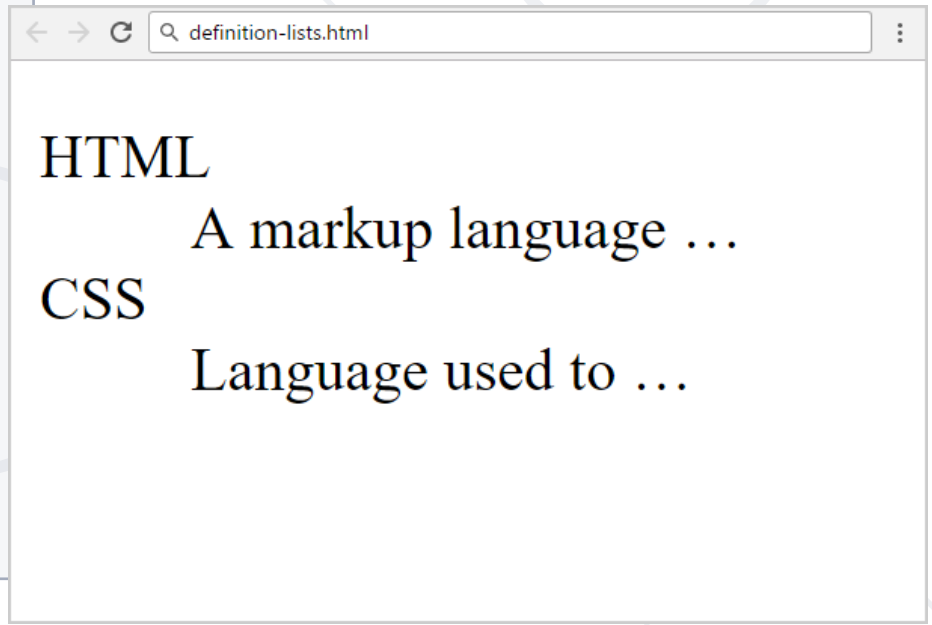
- Attribute values for `type` are: `disc`, `circle`, `square` and `none`



Definition Lists: <Dl> Tag

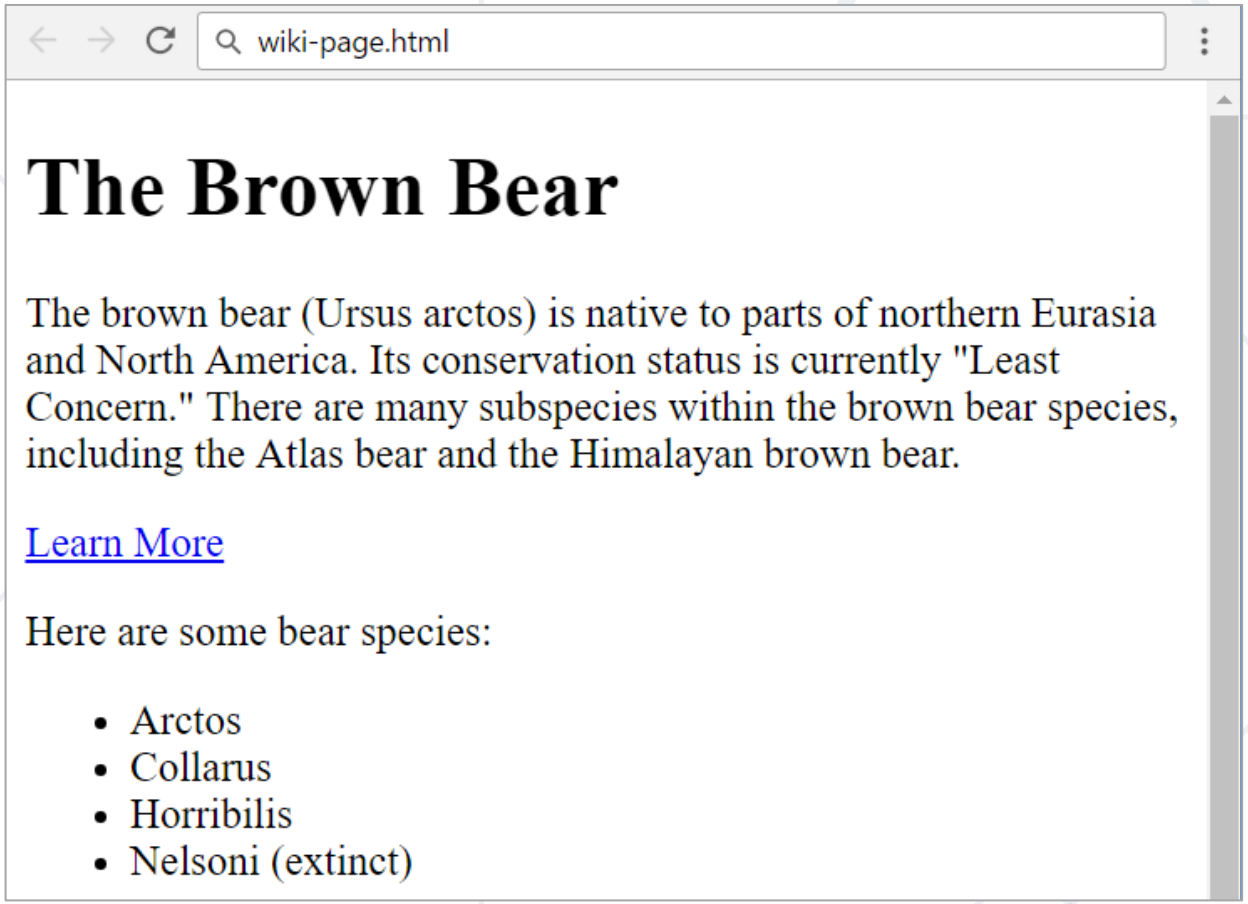
- Create definition lists using **<dl>**
  - Holds **terms** (**<dt>**) with their **definitions** (**<dd>**)

```
<dl>
  <dt>HTML</dt>
  <dd>A markup language …</dd>
  <dt>CSS</dt>
  <dd>Language used to …</dd>
</dl>
```

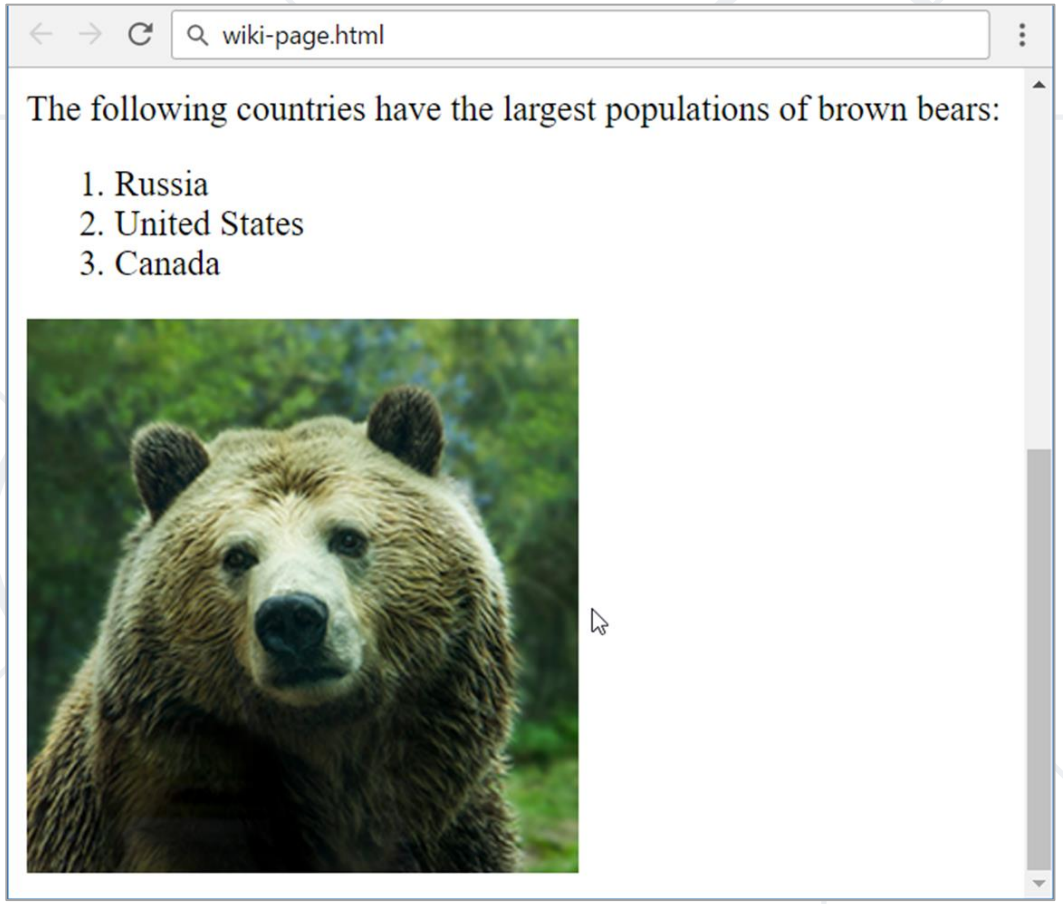


Problem: Wiki Page

- Create the following HTML page:



(page continues here …)



Hints: Wiki Page

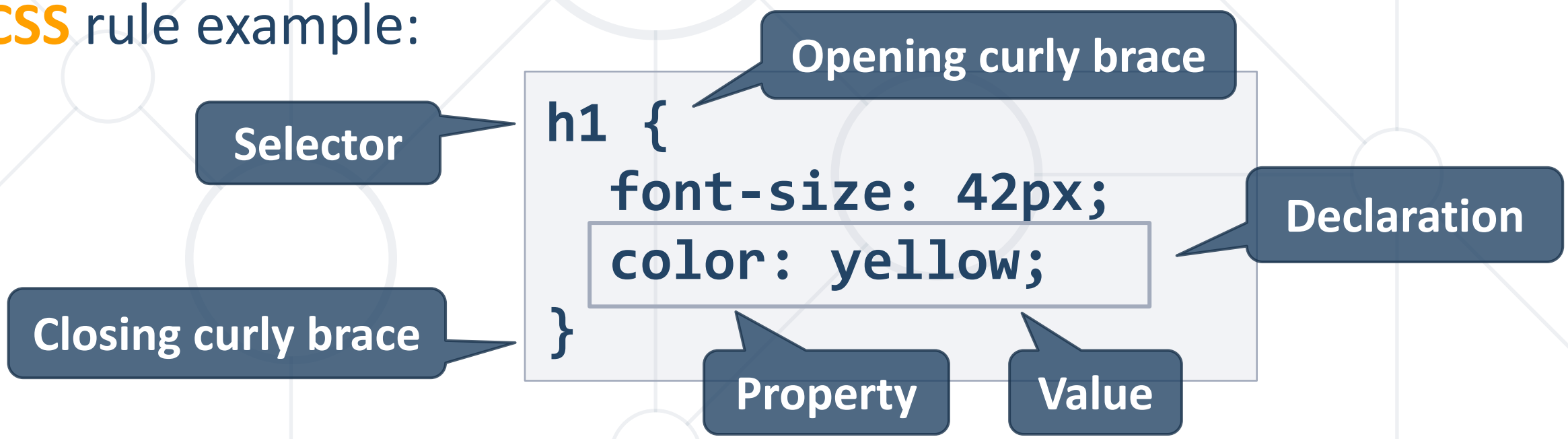
- File name: `wiki-page.html`
- Title: `The Brown Bear`
  - Use `<h1>`
- Hyperlink: **https://en.wikipedia.org/wiki/Brown_bear**
- List: use **ordered list** and **unordered list**
- Text: use **paragraph**
- Image: use the file `bear.jpg`

What is CSS?

Cascading Style Sheets

What is CSS?

- **CSS** defines **styling** of the HTML elements
  - Specifies fonts, colors, margins, sizes, positioning, floating, …
  - CSS **rules** format: `selector { prop1:val1; prop2:val2; … }`
- **CSS** rule example:



Combining HTML and CSS Files (External Style)

Software University

using-css.html

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css"
href="styles.css">
  </head>
  <body id="modern"><p>
    This is a <span class="special">special
drink</span> for <span class="special">special
drinkers</span>.</p>
  </body>
</html>
```

styles.css

```
.special {
  font-style: italic;
  font-weight: bold;
  color: blue;
}

#modern {
  background: #EEE;
}

p {
  font-size: 24pt;
}
```

CSS Selectors

- Select elements by **name**

```
<h1>Page Title</h1>
```

```
h1 { color: blue; }
```

- Select by **class** name

```
<p class="odd">Text</p>
```

```
.odd { font-size: 10px; }
```

- Select by element **id**

```
<span id="login">Go</span>
```

```
#login { width: 150px; }
```

- Select **element** with certain **class**

```
<a class="login">Login</a>
```

```
a.login { width: 80px; }
```

Combined CSS Selectors

```
<section id="news">
  <h1>Hot News</h1>
  <article>
    <h1>New Release!</h1>
    <p>Today we released
      <b class="red">ver. 7</b>
      of our unique software …
    </p>
  </article>
  <p>Published: <span
class="date">1/1/2021</span></p>
</section>
```

```
#news > h1
```

```
#news article > h1
```

```
#news article p
```

```
#news p > b.red
```

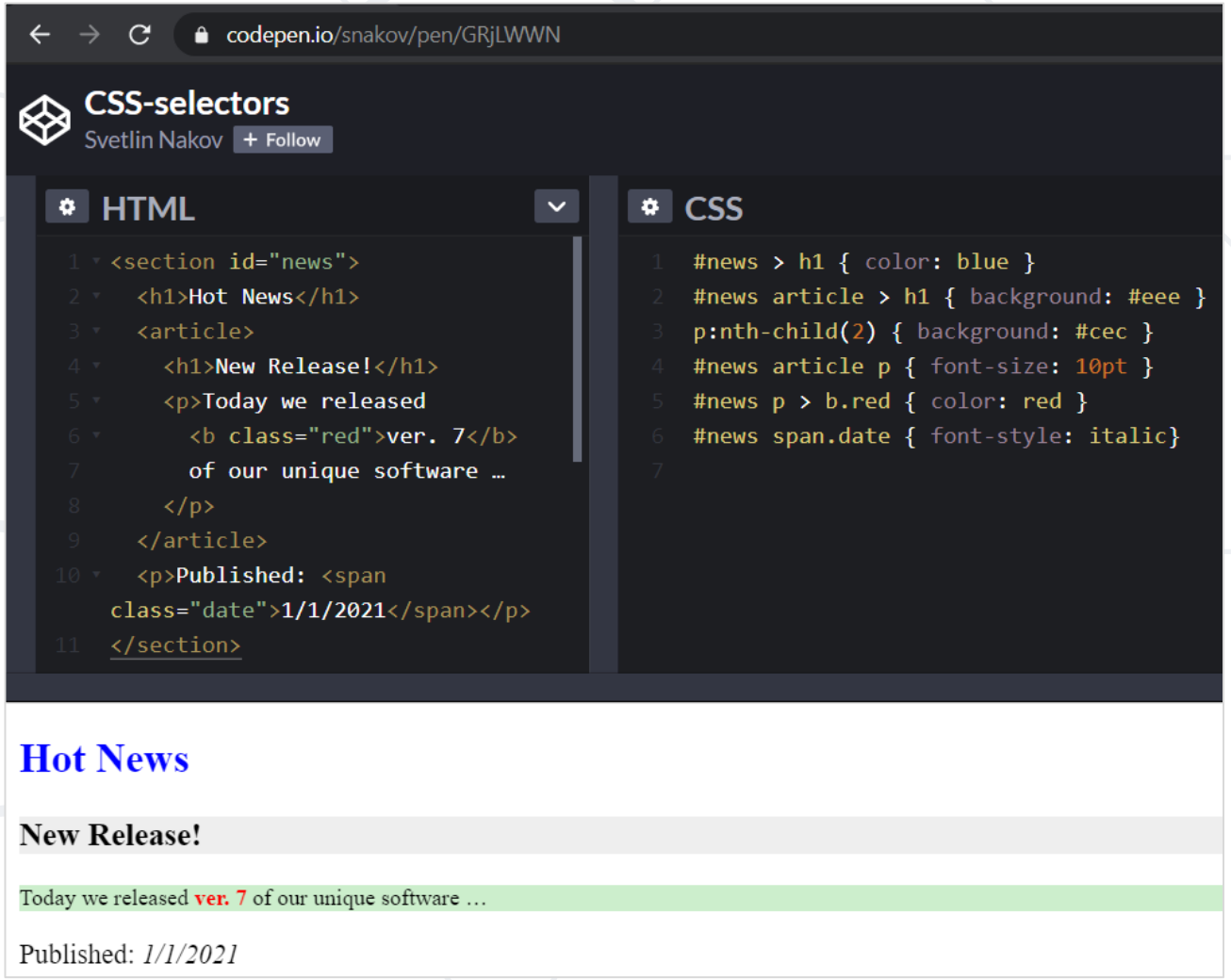
```
#news span.date
```

```
article>p, section>p
```

CSS Selectors: Live Demo

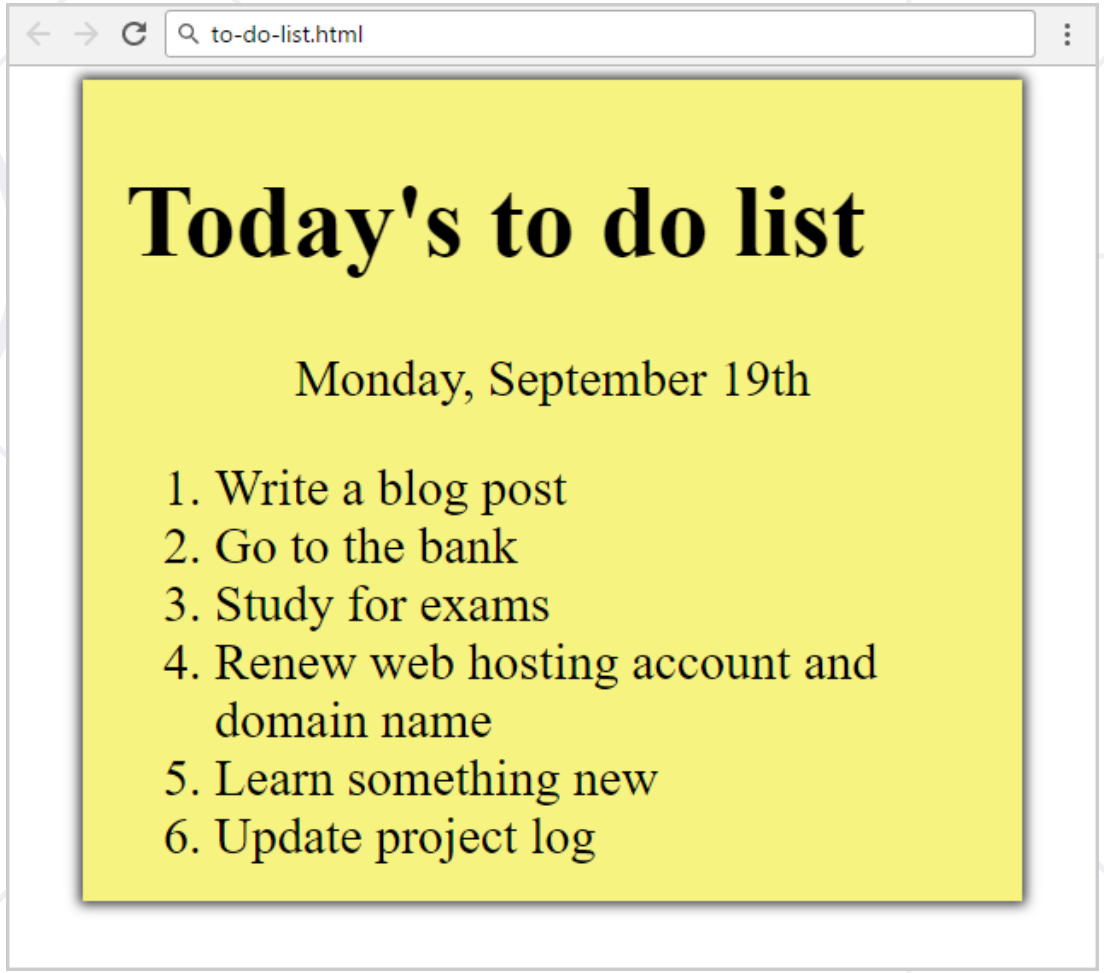
Software University

- https://codepen.io/snakov/pen/GRjLWWN



Problem: to Do List

- Create the following page (HTML + CSS files)
- Hints:
  - Container: use `<main>`
    - Background color: `#f7f381`
  - Heading: use `<h1>`
  - Date: use `<p>` + center it
  - List: use `<ol>`

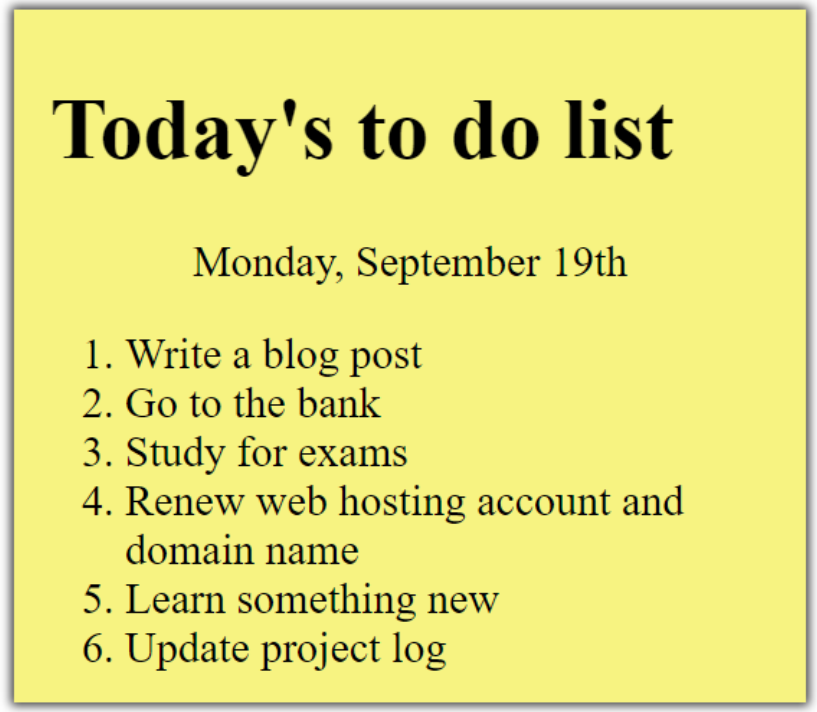


Solution: to Do List (HTML)

Software University

to-do-list.html

```
<!DOCTYPE html>
<html>
  <head><!-- TODO: Link the CSS here --></head>
<body>
  <main class="my-list">
    <h1>Today's to do list</h1>
    <p>TODO: Put day-info here</p>
    <ol>TODO: Put list-item here</ol>
  </main>
</body>
</html>
```



Solution: to Do List (CSS)

Software University

to-do-list.css

```
.my-list {
  margin: 0 auto;
  padding: 8px 24px;
  width: 500px;
  font-size: 30px;
  border: 1px solid #f7f381;
  background: #f7f381;
  box-shadow: 0 0 10px 2px #333333;
}
```

to-do-list.css

```
.my-list ol {
  margin: 12px;
}

.my-list p {
  text-align: center;
}
```

Inline CSS Style

- The `style` attribute defines **inline CSS**

Attribute "style"

```
<h1 style="color:blue">This is a blue …
</h1>
```

Property

Value

```
<h2 style="color:red; font-size:2.1em">
  This is a red …
</h2>
```

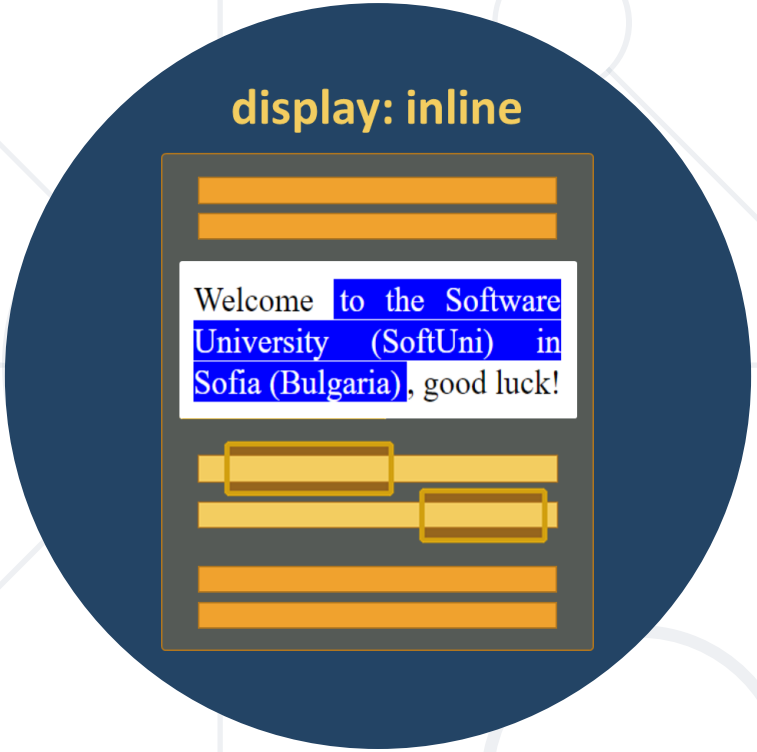
Multiple CSS declarations

Embedded CSS Styles in the HTML Page

- Put a `<style>` element in the HTML `<head>` section

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .red {color:red;}
  </style>
</head>
</html>
```

```
<body>
  <p class="red">This is red</p>
</body>
```



Inline and Block Elements

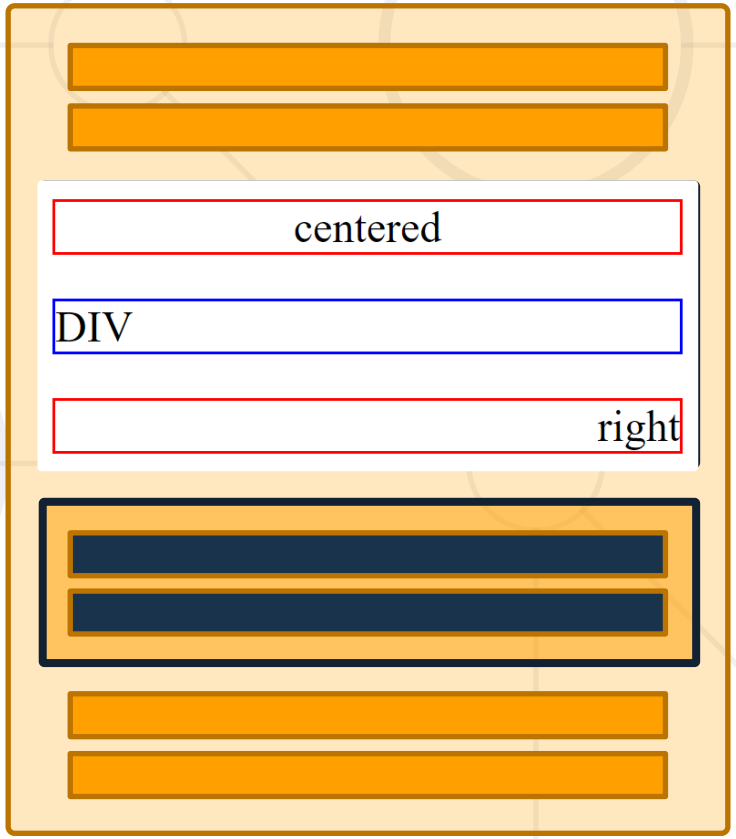
vs.

Block Elements

- `<div>` and `<p>` are **block elements** (rectangles)
  - Fill the entire container width
  - Stack vertically one after another

```
<p style="border:1px solid red;
text-align:center">centered</p>
<div style="border:1px solid
blue">DIV</div>
<p style="border:1px solid red;
text-align:right">right</p>
```

display: block

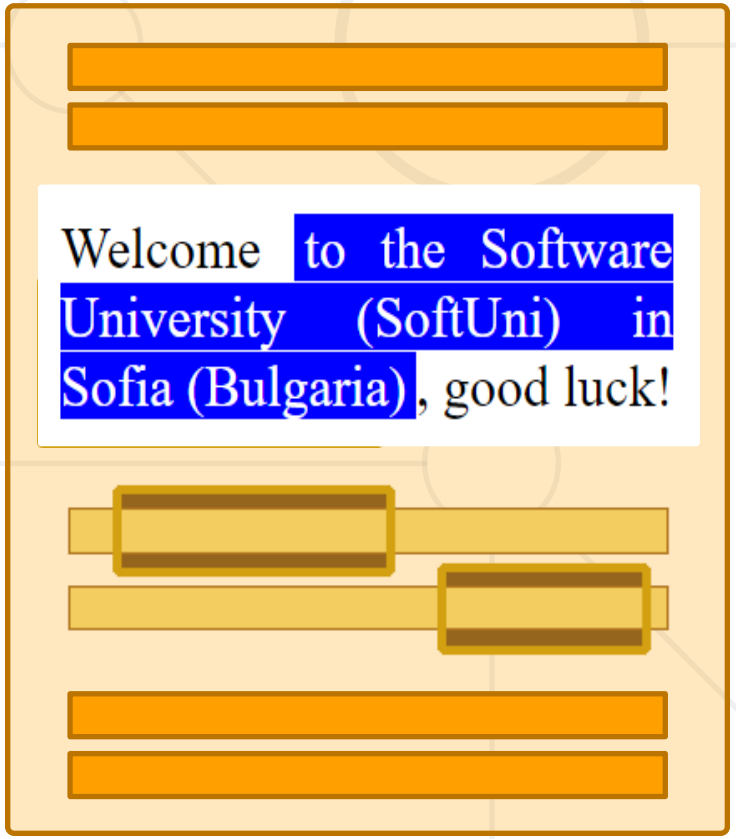


Inline Elements

- `<span>` is **inline element**
  - Its shape is not always rectangular
  - Can be split across multiple lines

```
<p style="text-align:justify"> Welcome
<span style="color:white;
background:blue; padding-right:3px;
padding-left:3px;">
to the Software University (SoftUni)
in Sofia (Bulgaria)</span>, good
luck!</p>
```

display: inline



Inline-Block Elements

- Elements can be also **inline-block**
  - Rectangles arranged one after another
  - Just like words in a sentence

```
<div style="text-align:justify;">
   <div style="display:inline-block;
   background:green">green</div>
   <div style="display:inline-block;
   background:red">red block</div>
   …
</div>
```

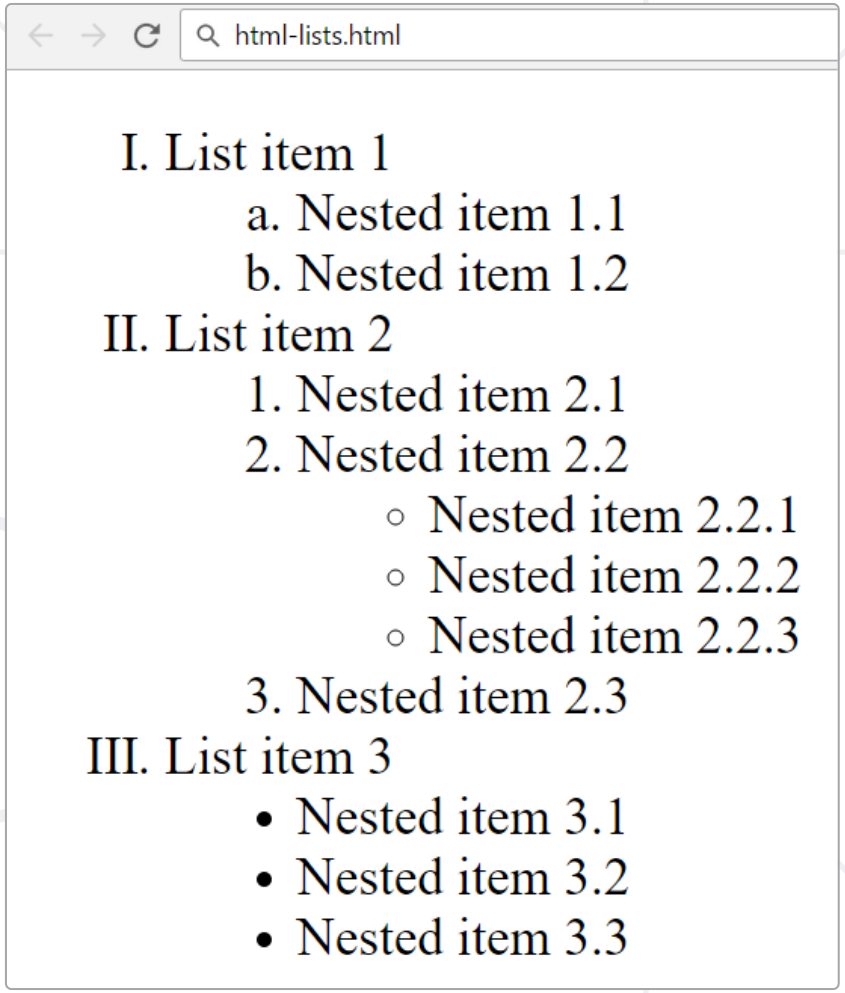
display: inline-block

More HTML Problems

Problems and Solutions

Problem: HTML Lists

- Create a HTML page, holding **nested lists**, like at the example
- Hints:
  - Use `<ol>` and `<li>` for **List Item 1**, **List item 2** and **List Item 3**
  - Use `<ol>`, `<li>`, `<ul>` and `<li>` for the nested lists



Solution: HTML Lists

Software University

html-lists.html

```
<ol type="I">
  <li>List item 1
    <ol type="a">
      <li>Nested item 1.1</li>
      <li>Nested item 1.2</li>
    </ol>
  </li>
  <!-- TODO: put List item 2 and List item 3 here
-->
</ol>
```

I. List item 1
    a. Nested item 1.1
    b. Nested item 1.2

Solution: HTML Lists (2)

Software University

html-lists.html

```
<li>List item 2
  <ol type="1">
    <li>Nested item 2.1</li>
    <li>Nested item 2.2
      <ul type="circle">
        <li>Nested item 2.2.1</li>
        <li><!-- TODO: put the next items here--></li>
      </ul>
    </li>
    <li>Nested item 2.3</li>
  </ol>
</li>
```

II. List item 2
1. Nested item 2.1
2. Nested item 2.2
   - Nested item 2.2.1
   - Nested item 2.2.2
   - Nested item 2.2.3
3. Nested item 2.3

Solution: HTML Lists (3)

Software University

html-lists.html

```
<li>
  List item 3
  <ul type="disc">
    <li>Nested item 3.1</li>
    <li><!-- TODO: put the next items here --></li>
  </ul>
</li>
```

III. List item 3
- Nested item 3.1
- Nested item 3.2
- Nested item 3.3

Summary

- What is HTML?

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HTML Example</title>
  </head>
  <body>
    <h1>Hello HTML!</h1>
    <p>HTML describes formatted text using <strong>tags</strong>. Visit the
      <a href="https://softuni.bg/trainings/courses">SoftUni HTML course to
    learn more.</a></p>
  </body>
</html>
```

- CSS styles may be: external, inline, embedded

```
.my-list p { text-align: center; }
```

Questions?

SoftUni

Software University | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids | Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

createX

Postbank
Решения за твоето утре

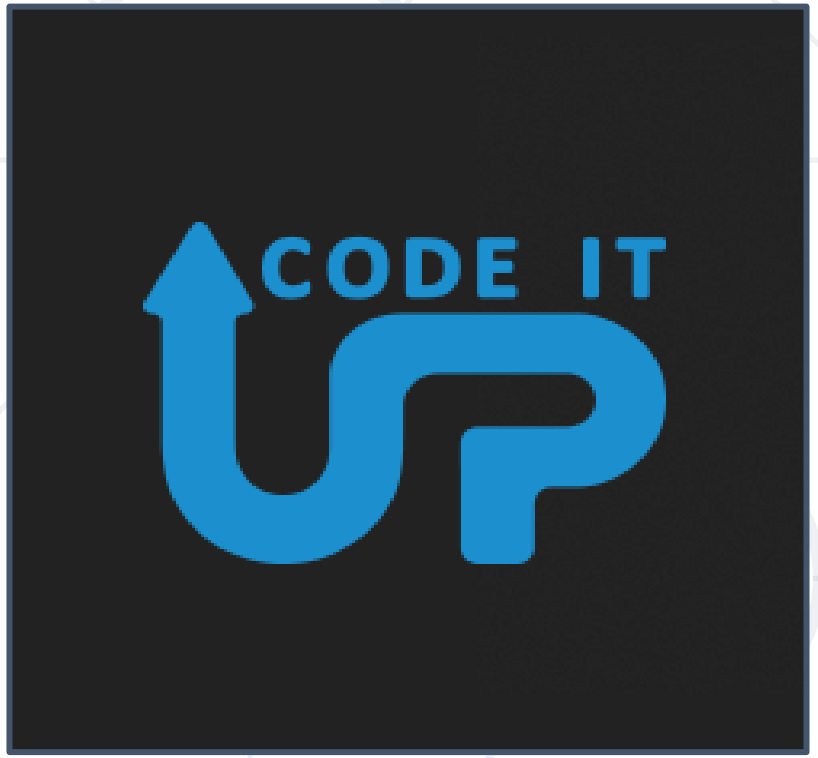
BOSCH

DXC TECHNOLOGY

SmartIT

Educational Partners

Software University



License

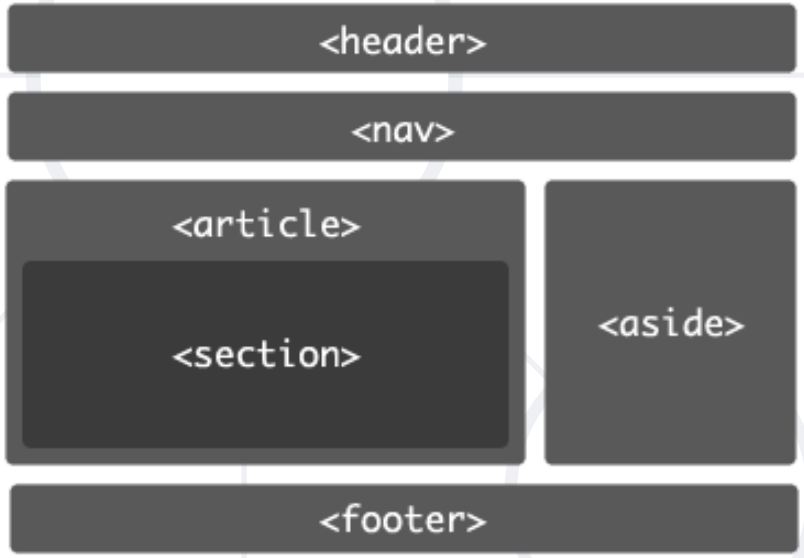
- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

HTML STRUCTURE

Semantic Tags, Document Structure



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Table of Contents

Have a Question?

Software University

sli.do

#html-css

Semantic HTML

HTML Markup to Reinforce the Semantics

What is Semantic HTML?

- Semantic element clearly describes its meaning to both the **browser** and the **developer**

```
<p>Some random text...</p>
```

Indicates that the enclosed text is a paragraph

- This is both semantic and presentational
  - People know what paragraphs are and browsers know how to display them

```
<footer>© 2021 by ABC</footer>
```

This holds a footer

The Importance of Semantic HTML

- Provides an **additional information** about that document, which **aids in communication**
- Semantic tags make it **clear** to the **browser** what the **meaning** of a page and its **content** is
  - This clarity is also **communicated** with search engines

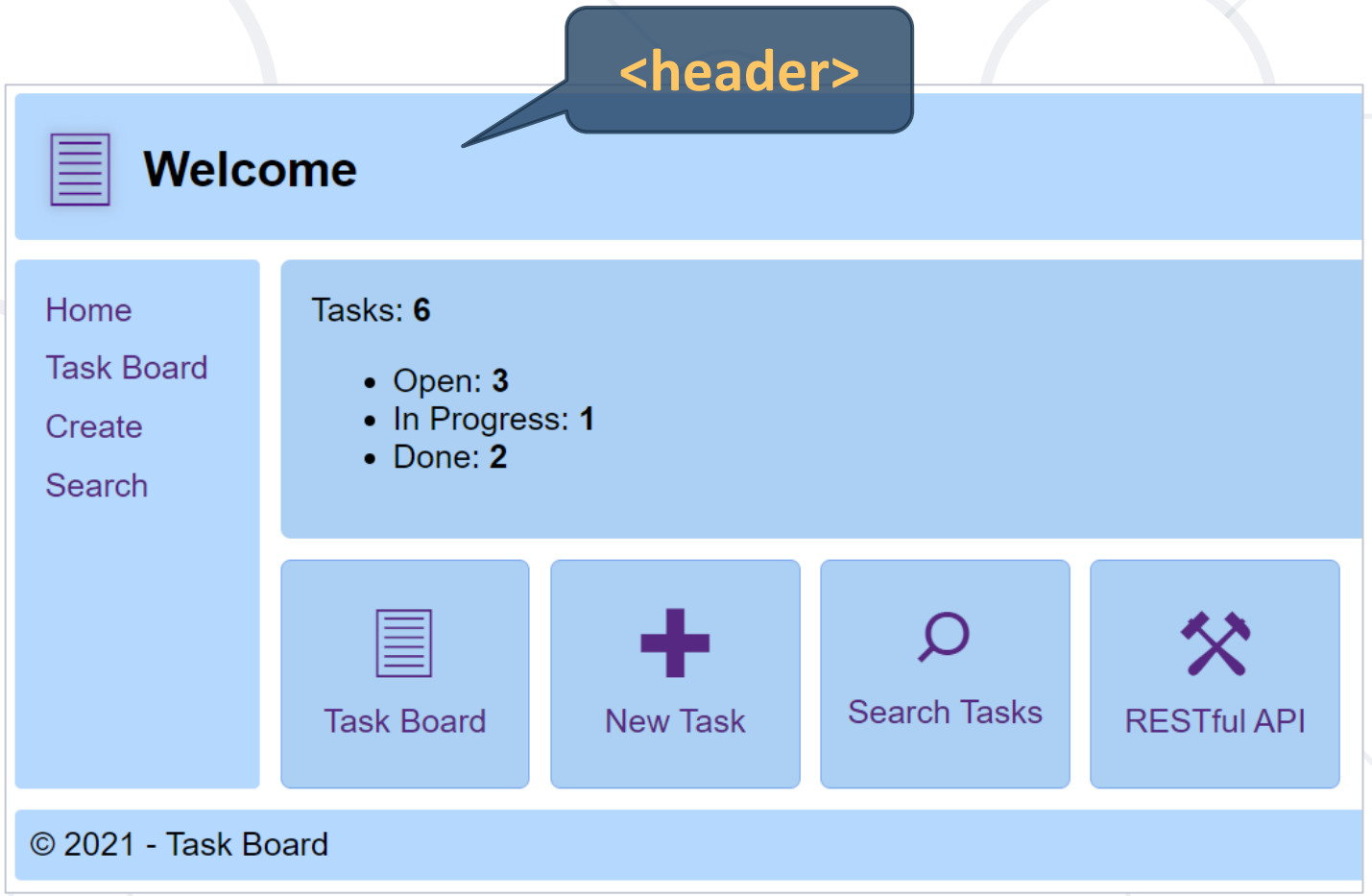
HTML Semantic Tags

<Header></header>

- Represents **introductory content**

```
<header>
    <h1>Welcome</h1>
</header>
```

- It may contain:
  - headings
  - logo
  - search form



<Nav></nav>

- Defines a set of **navigation links**

```
<nav id="leftmenu">
  <ul>
    <li><a href="/home">Home</a></li>
    <li><a href="/tasks">Task Board</a></li>
    <li><a href="/create">Create</a></li>
    <li><a href="/search">Search</a></li>
 </ul>
</nav>
```

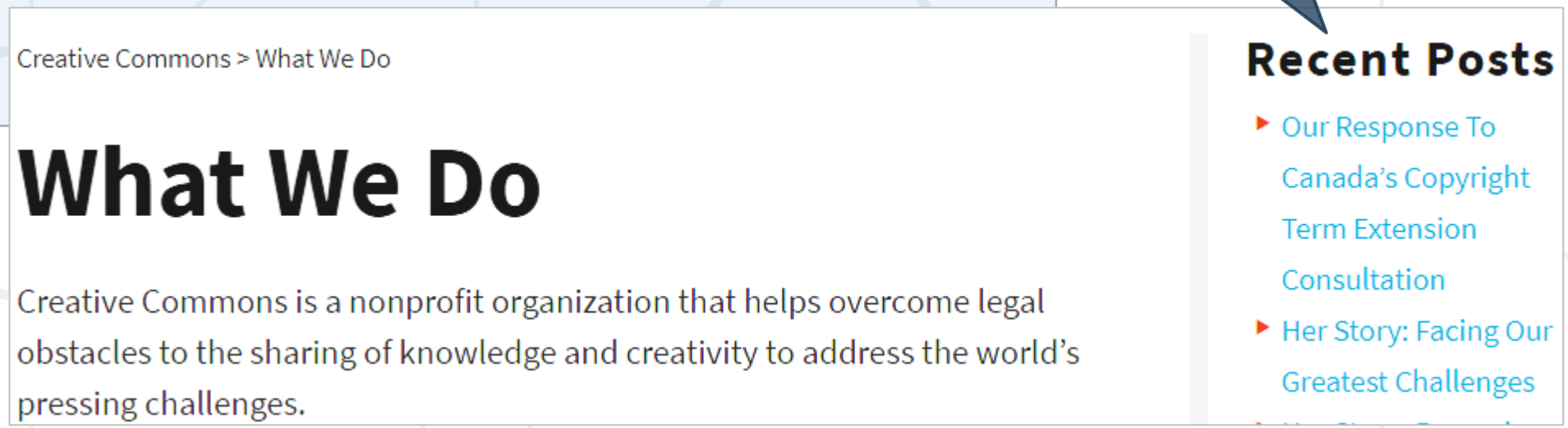
<Main></main>

- `<main>` holds the main content of a document
  - Helps crawlers
  - There must not be more than one `<main>` element in a document
  - Wrap the most important information in the body

<Aside></aside>

- Defines a sidebar (**left / right navigation**)

```
<aside>
  <h2>Recent posts</h2>
  <ul>
    <li><a href="#">Our Response</a></li>
    <li><a href="#">Her Story</a></li>
    <li><a href="#">Greatest Challenges</a></li>
  </ul>
</aside>
```



<Footer></footer>

- A document / section footer

```
<footer>
  <p>Posted by: Hege Refsnes</p>
    <p><a href="someone@exam.uk">
      2021 Task Board</a></p>
  <p>&copy;copyright</p>
</footer>
```

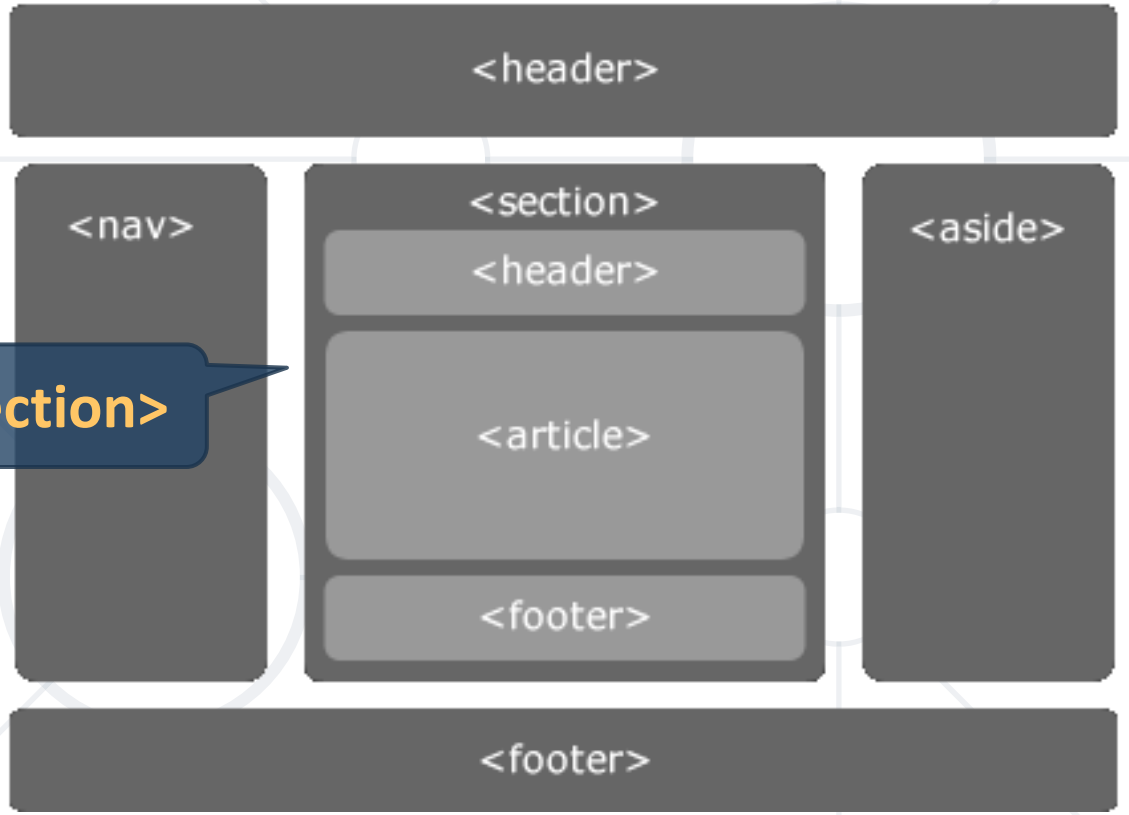
- A footer typically contains:
  - Navigation links
  - Copyright data

<Section></section>

- Represents a standalone section
  - Typically followed by a heading

```
<section>
  <h2>Heading</h2>
  <img src="bird.jpg" />
</section>
```

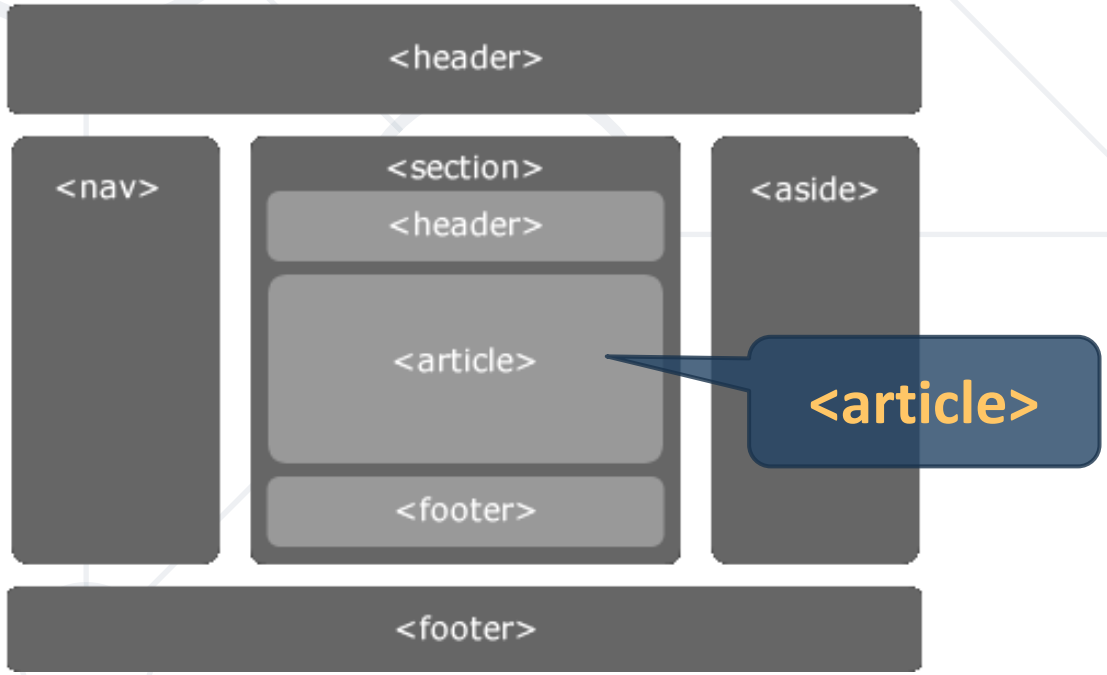
- Sections may have header, several articles, and footer



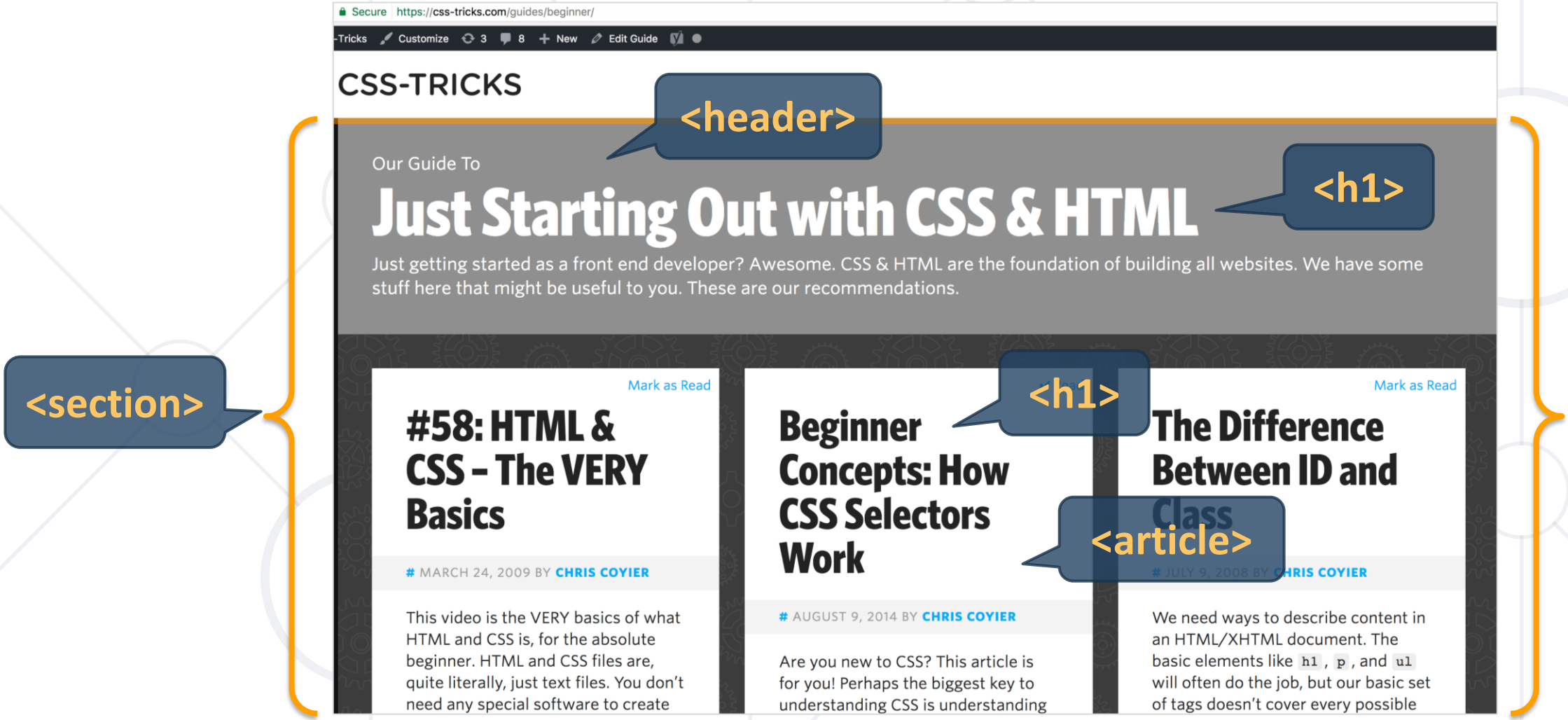
<Article></article>

- Represents a **self-contained composition** in a document, page, application, or site
- Intended to be **independently distributable** or **reusable**
- Examples:
  - Forum post
  - Magazine
  - Newspaper article
  - Blog entry

```
<article>
  <h2>Tips</h2>
  <p>Tip #1 …</p>
</article>
```



Sections and Articles – Example



<Figure></figure>

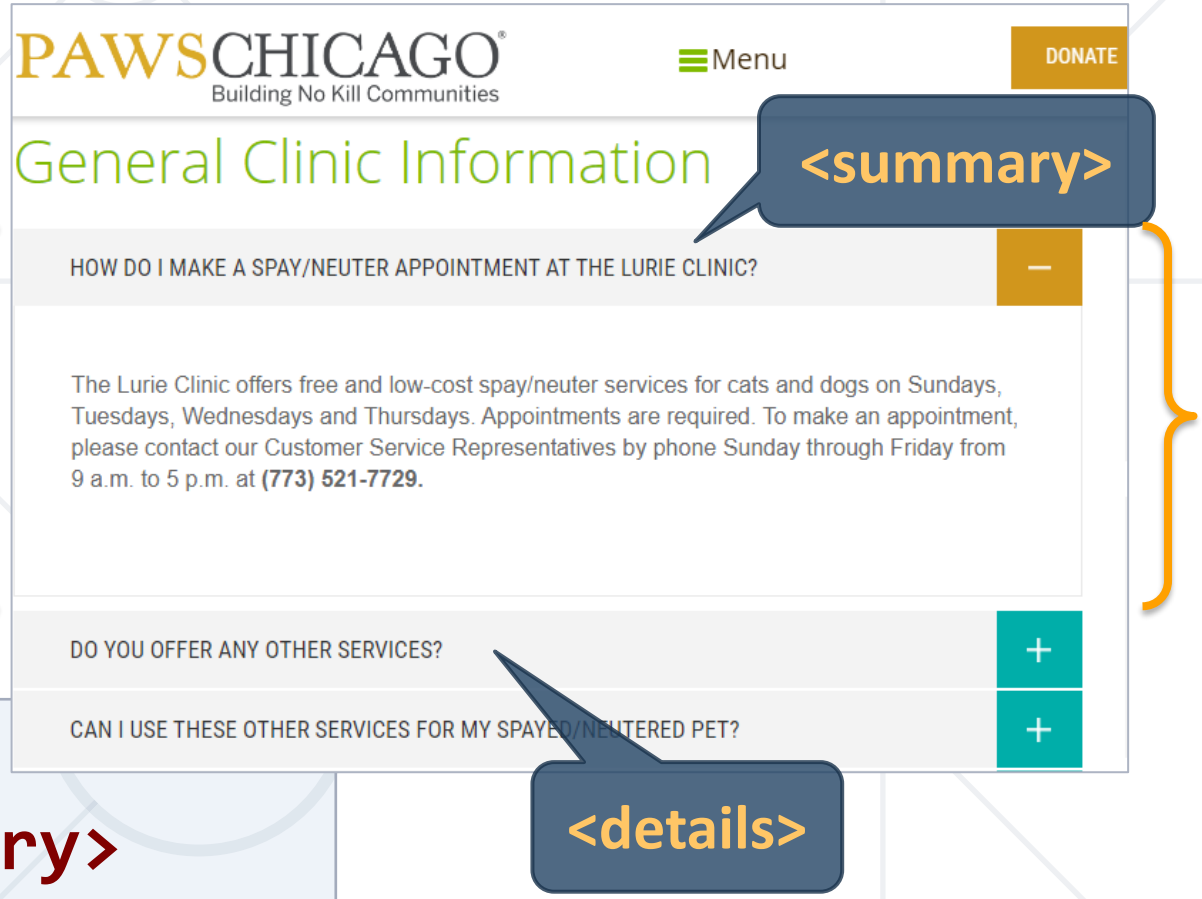
- Represents self-contained content
- Frequently with a caption "figcaption"
- Typically referenced as a single unit

```
<figure>
  <img src="pic_trulli.jpg"
    alt="Trulli">
  <figcaption>
    Fig.1 Trulli, Puglia, Italy.
  </figcaption>
</figure>
```

<figure>

<Details> + <Summary>

- **<details>** – additional details that the user can view or hide
- **<summary>** – defines a visible heading for the **<details>**




```
<details>
  <summary>Some details</summary>
  <p>More info about the details.</p>
</details>
```

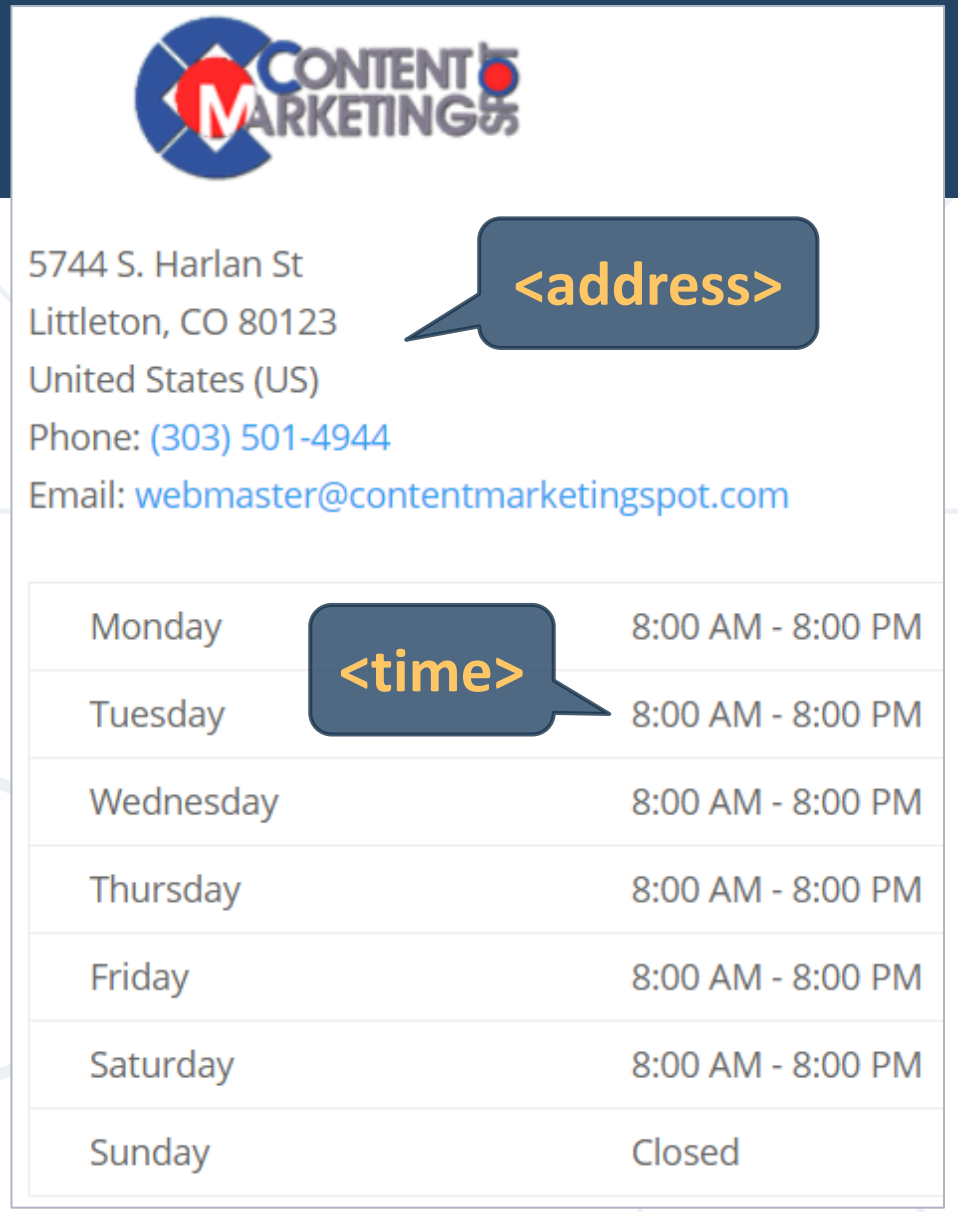
<Time> + <Address>

- **<time>** – a human-readable time
  - Search engines can produce smarter results

```
<p>We open at <time>10:00</time>
  every morning.</p>
```

- **<address>** – contact information for site author / owner

```
<address>
  <a href="mailto:tony@gmail.com">
    tony@gmail.com</a>
</address>
```



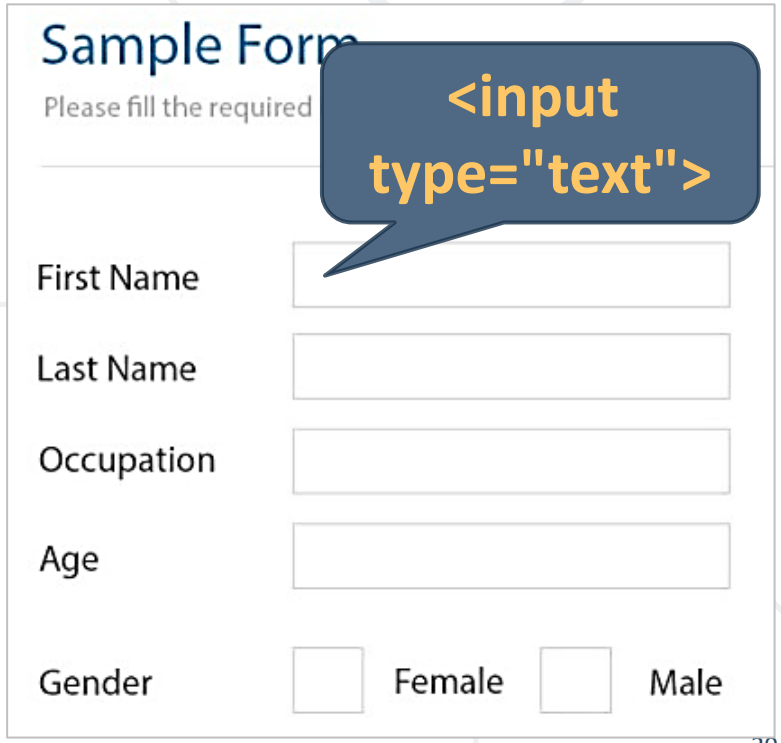
Forms

Collect User Input

Form

- The **HTML form** - a document section
  - Contains interactive controls for submitting information
  - Takes **input** from the site **visitor** and posts it

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname"
 name="fname" value="John"><br>
  <input type="submit" value="Submit">
</form>
```



Form Attributes

- **Action** - used to specify where the form data is to be sent to the server after submission of the form

```
<form action="register.php" … />
```

- **Method** - The HTTP method that the browser uses to submit the form
  - **POST** - Corresponds to **HTTP POST** (hides posted form data)
  - **GET** - Corresponds to **HTTP GET** (shows form data in the URL)

```
<form method="POST" … />
```

```
<form method="GET" … />
```

Form Elements – Input

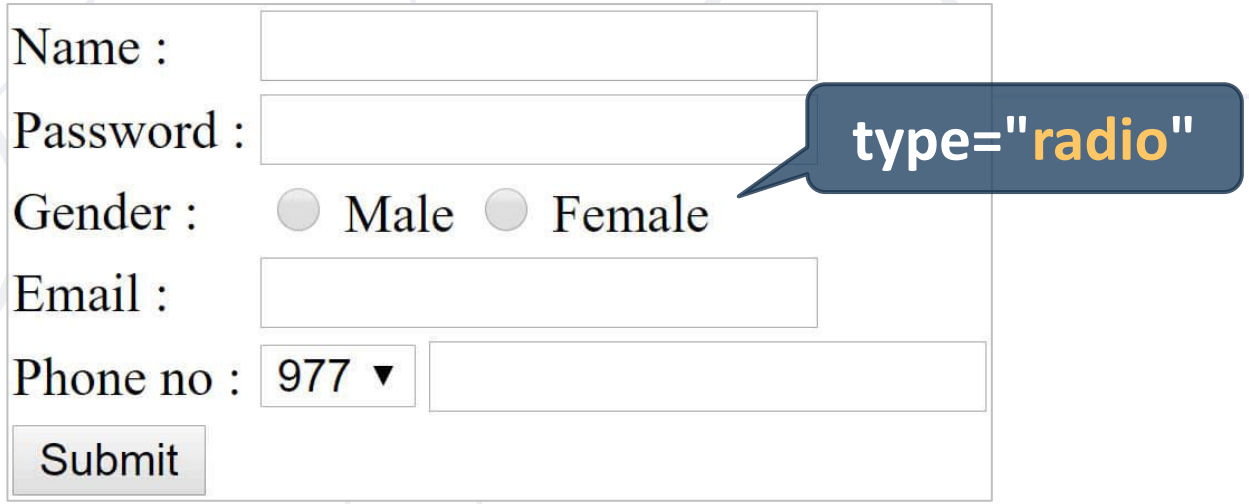
- `<input>` element is the most important form element
- It can be displayed in several ways, depending on the **type** attribute:
  - `<input type="text">`
  - `<input type="number">`
  - `<input type="password">`
  - `<input type="email">`
  - `<input type="search">`

Form Elements – Input

- `<input type="checkbox">`
- `<input type="radio">`
- `<input type="range">`
- `<input type="submit">`
- `<input type="button">`
- `<input type="file">`

Radio Buttons – Example

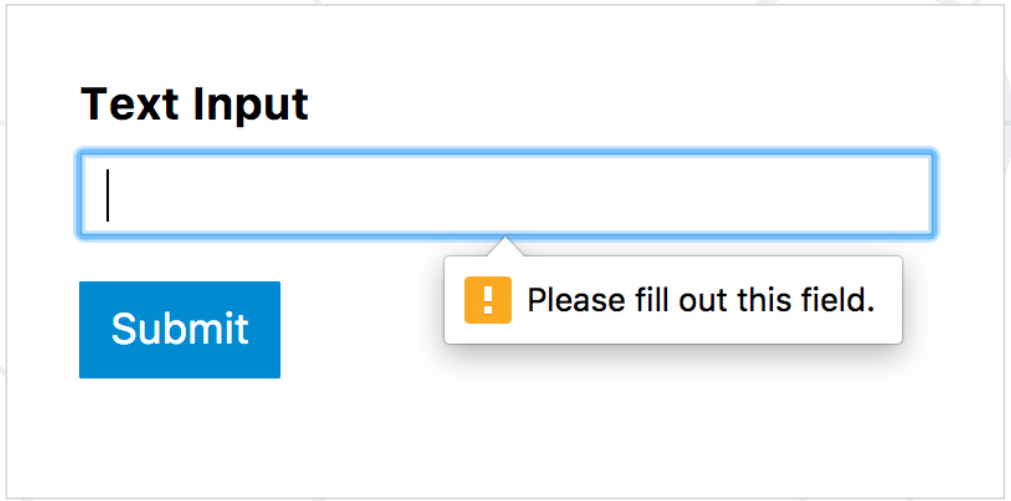
```
<form action="/register.php" method="get">
  <label for="male">Male</label>
  <input type="radio" id="male" name="male" value="Gender">
  <label for="female">Female</label>
  <input type="radio" id="female" name="female" value="Gender">
</form>
```



Input Validation

- HTML input **validation** is done automatically by the browser based on **special attributes**

```
<input type="text"
  required="true" />
```

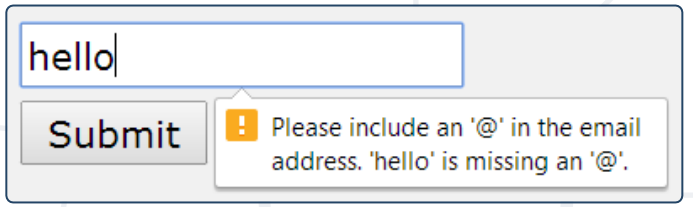


- The validation only **occurs** when attempting to **submit** the form
- Inputs which are **disabled** or **read-only** will not trigger validation

Input Fields – Examples

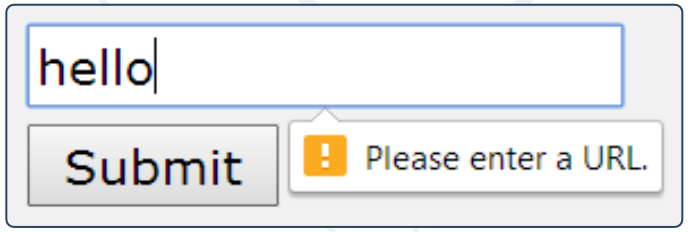
- Email – simple validation for emails

```
<input type="email" required="true" />
```



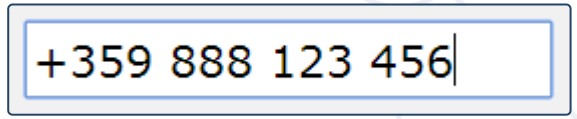
- URL – validation for URL addresses

```
<input type="url" required="true" />
```



- Telephone – validation for phone numbers

```
<input type="tel" required="true" />
```

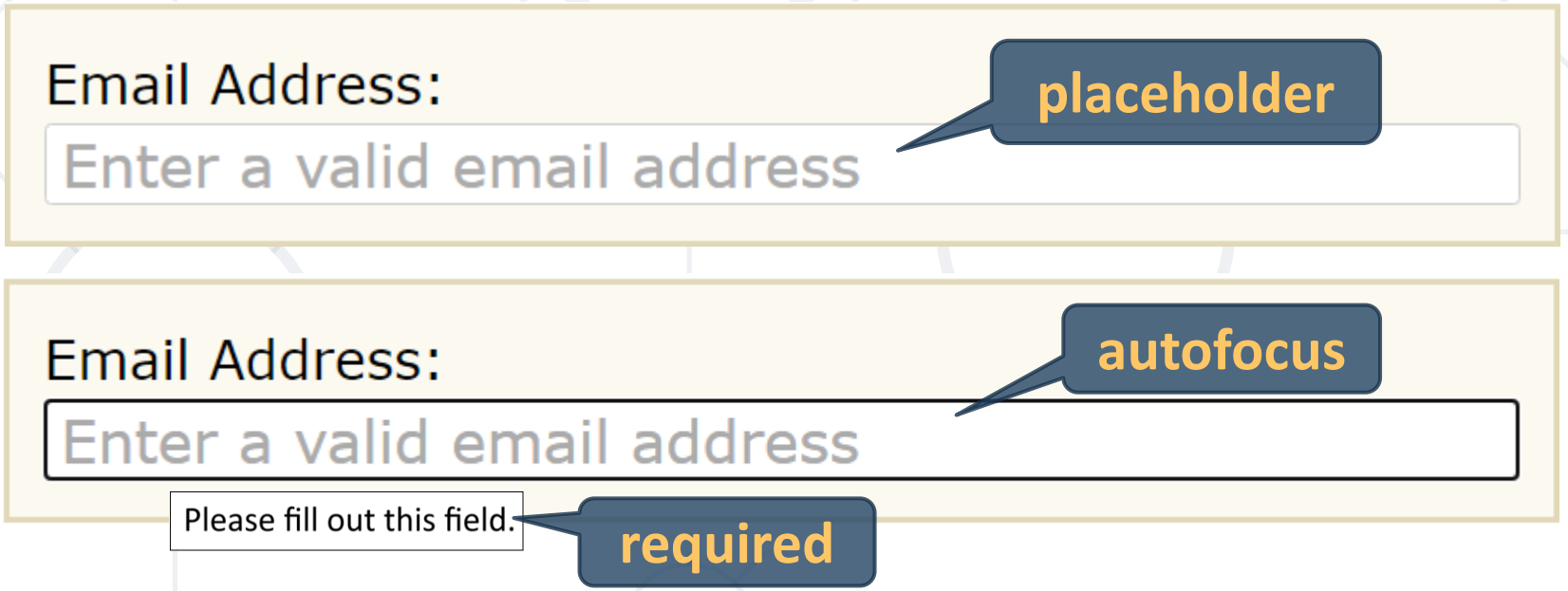


Form Elements – Input Attributes

- **value** - specifies the initial value for an input field
- **name** - specifies the name of the input element
- **placeholder** - specifies a hint that describes the expected value of the input field
- **required** - the field must be filled out before submitting the form
- **autofocus** - the input should automatically get focus when the page load
- **disabled** - specifies that the input field is disabled
- **min** and **max** - specify the minimum and maximum values

Form Elements – Examples

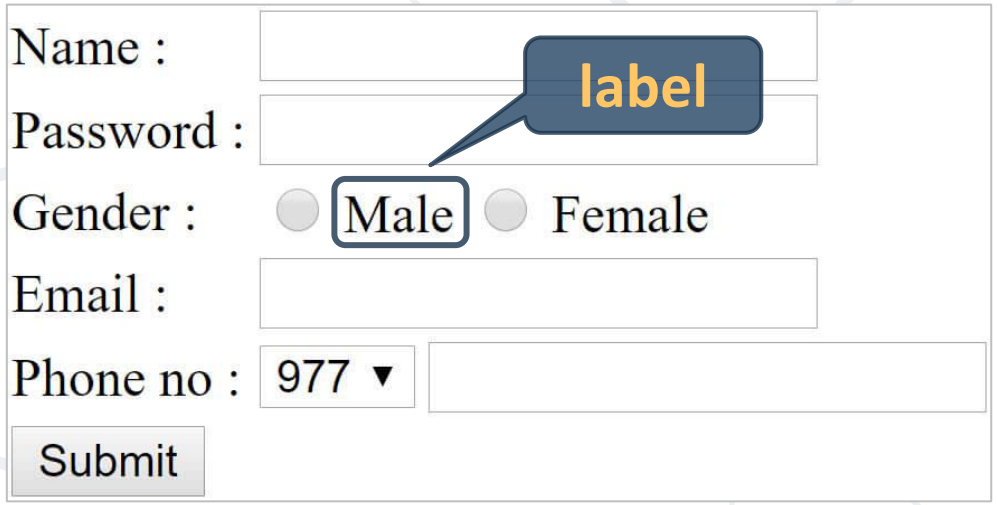
```
Email Address: <input type="email" size="48"
  name="email" required="true" autofocus
  placeholder="Enter a valid email address" >
```



Form Elements – Label

- `<label>` - defines a label for the others forms elements
- The `for` attribute should be equal to the `id` attribute of the related element to bind them together

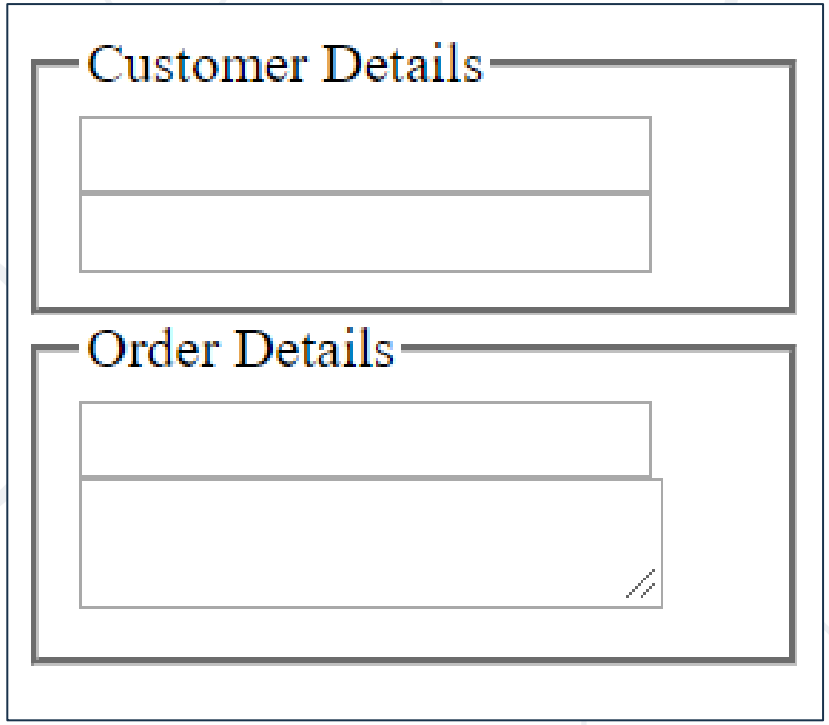
```
<form>
  <label for="male">Male</label>
  <input type="radio" name="gender"
    id="male" value="male">
</form>
```



Form Elements – Fieldset

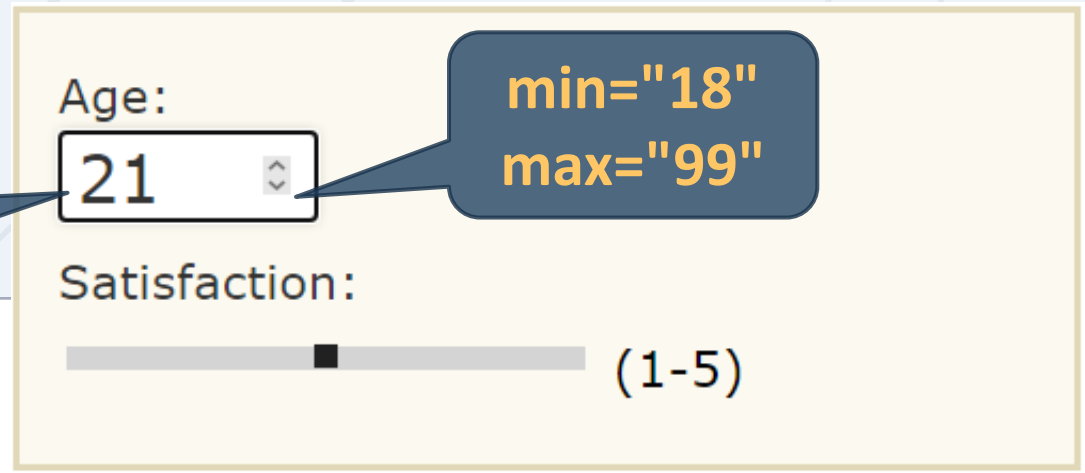
- `<fieldset>` - used to group related data in a form
- `<legend>` - defines a caption for the `<fieldset>` element

```
<fieldset>
  <legend>Customer Details</legend>
  <input type="text" name="fName" />
  <input type="text" name="lName" />
</fieldset>
<fieldset>
  <legend>Order Details</legend>
  <input type="text" name="quantity" />
  <textarea name="remarks"></textarea>
</fieldset>
```



Number and Range Fields

```
<fieldset>
  <label for="field_age">Age</label>
  <input type="number" name="age" id="field_age"
    min="18" max="99" value="21">

  <label for="field_sat">Satisfaction</label>
  <input type="range" name="satisfaction"
    min="1" max="5" value="3"
    id="field_sat"> (1-5)
</fieldset>
```



Form Elements – Select

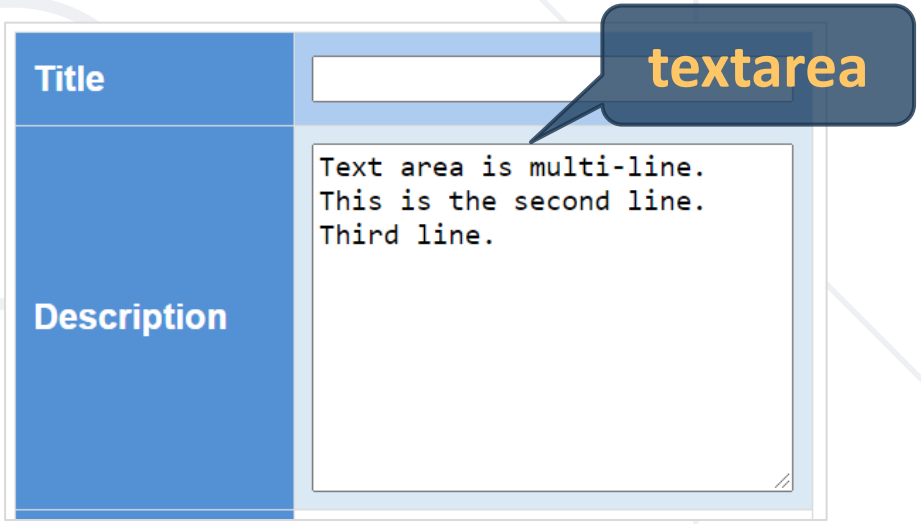
- `<select>` - defines a drop-down list
- `<option>` - defines an option that can be selected

```
<form>
  <label for="size">Size:</label>
  <select id="size" name="size">
    <option value="39">39 EU</option>
    <option value="40">40 EU</option>
    <option value="41.5">41.5 EU</option>
    …
  </select>
</form>
```

Form Elements – Textarea

- `<textarea>` - defines a multi-line input field
- Attributes:
  - **rows** - specifies the visible **number of lines** in a text area
  - **cols** - specifies the **visible width** of a text area

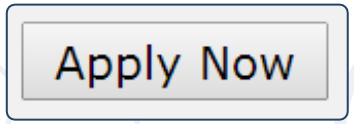
```
<textarea rows="10" cols="30">
Text area is multi-line.
This is the second line.
Third line.
</textarea>
```



Form Elements – Buttons

- **Submit** button – sends the form data to the server

```
<input type="submit" value="Apply Now" />
```



- **Reset** button – resets all form fields

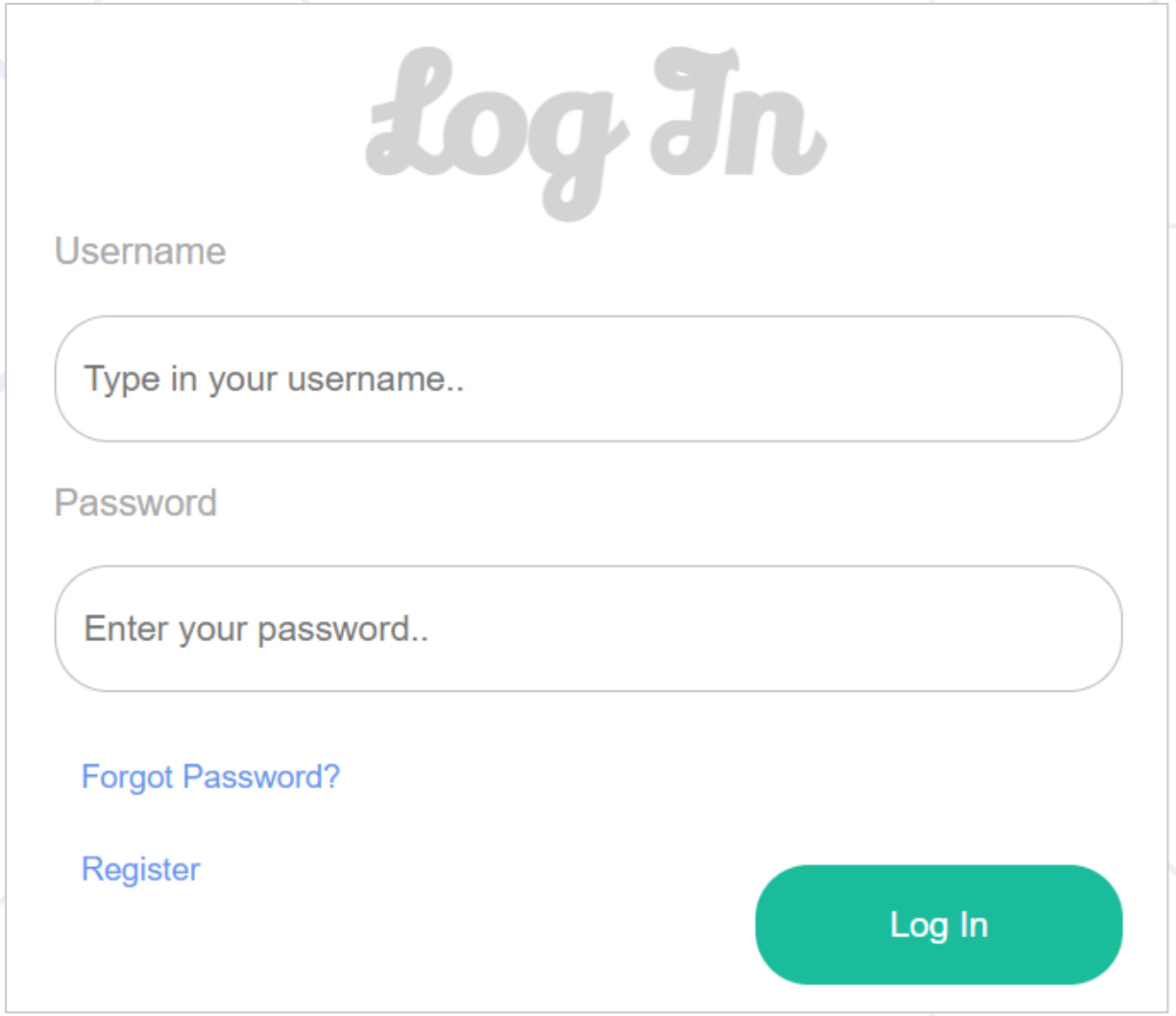
```
<input type="reset" />
```

- The `<button>` tag defines a clickable button

```
<button type="button">Click Me!</button>
```

Complete HTML Form – Example

- Example of HTML form + CSS styles
  - https://codepen.io/snakov/pen/oNYQvpB



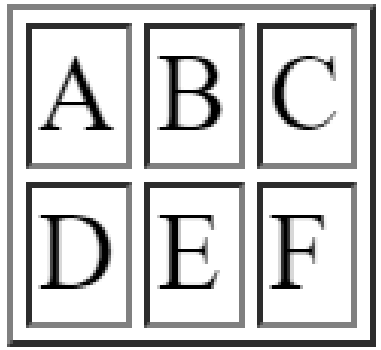
Tables

Arrange Data into Rows and Columns

Simple HTML Tables

- An HTML **table** is defined with the **<table>** tag
- Each table **row** is defined with the **<tr>** tag
- A table **cell** is defined with the **<td>** tag

```
<table border="1">
  <tr>
    <td>A</td><td>B</td><td>C</td>
  </tr>
  <tr>
    <td>D</td><td>E</td><td>F</td>
  </tr>
</table>
```

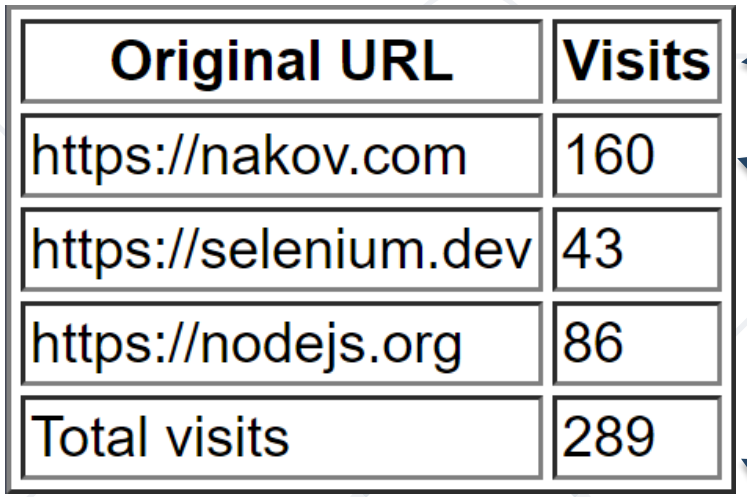


Complete HTML Tables

- There are three specific parts in every table:
  - Table **header**
  - Table **body**
  - Table **footer**
- Each table part holds rows (`<tr>`)
  - Rows hold cells (`<td>` / `<th>`)

```
<table>
    <thead>…</thead>
    <tbody>
        <tr>
            <td>Mark</td>
            <td>5,75</td>
        </tr>
    </tbody>
    <tfoot>…</tfoot>
</table>
```

Complete HTML Tables

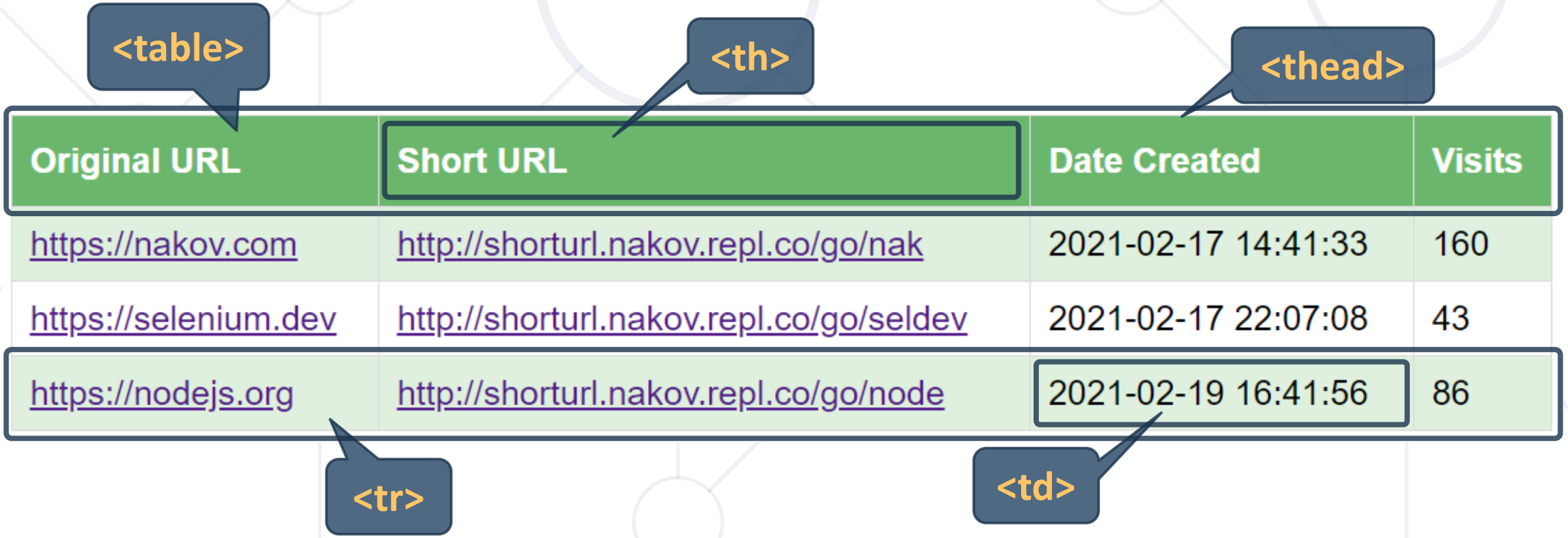


Original URL	Visits
https://nakov.com	160
https://selenium.dev	43
https://nodejs.org	86
Total visits	289

```
<table border="1">
<tbody>
  <thead>
    <tr>
      <th>Original URL</th>
      <th>Visits</th>
    </tr>
  </thead>
  <tr>
    <td>https://nakov.com</td>
    <td>160</td>
  </tr>
</tbody>
  <tfoot>
    <tr>
      <td>Total visits</td>
      <td>289</td>
    </tr>
  </tfoot>
</table>
```

Complete HTML Table – Example

- Example of HTML table + CSS styles
  - https://codepen.io/snakov/pen/XWNyreJ



Summary

- Semantic HTML
- Tags
- Forms
- Tables

Questions?

SoftUni

Software University | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids | Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

createX

Postbank
Решения за твоето утре

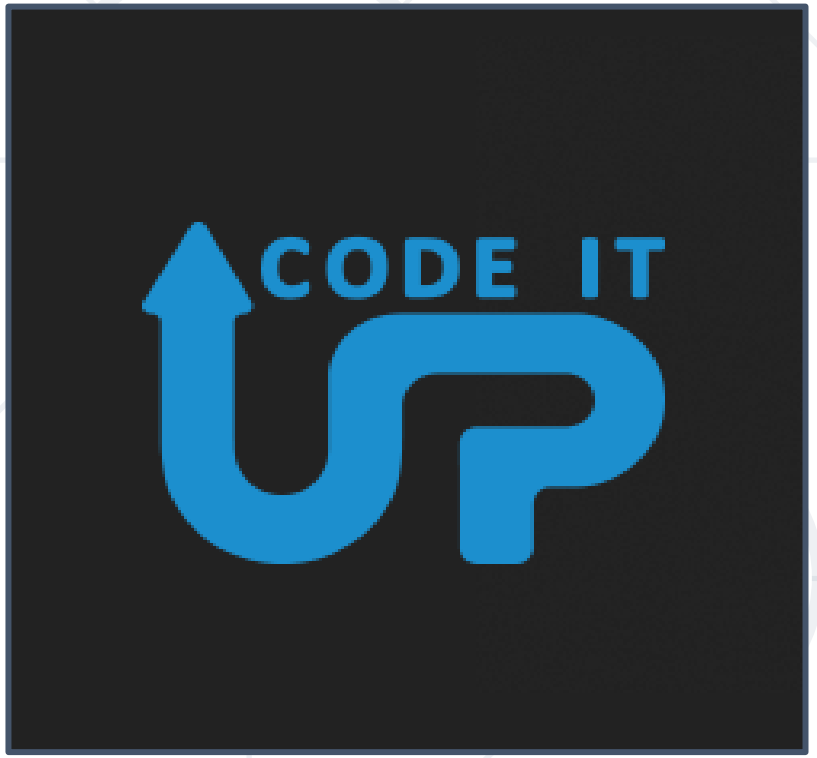
BOSCH

DXC TECHNOLOGY

SmartIT

Educational Partners

Software University



License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

CSS & TYPOGRAPHY

CSS Text Properties, Fonts, Font Properties

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

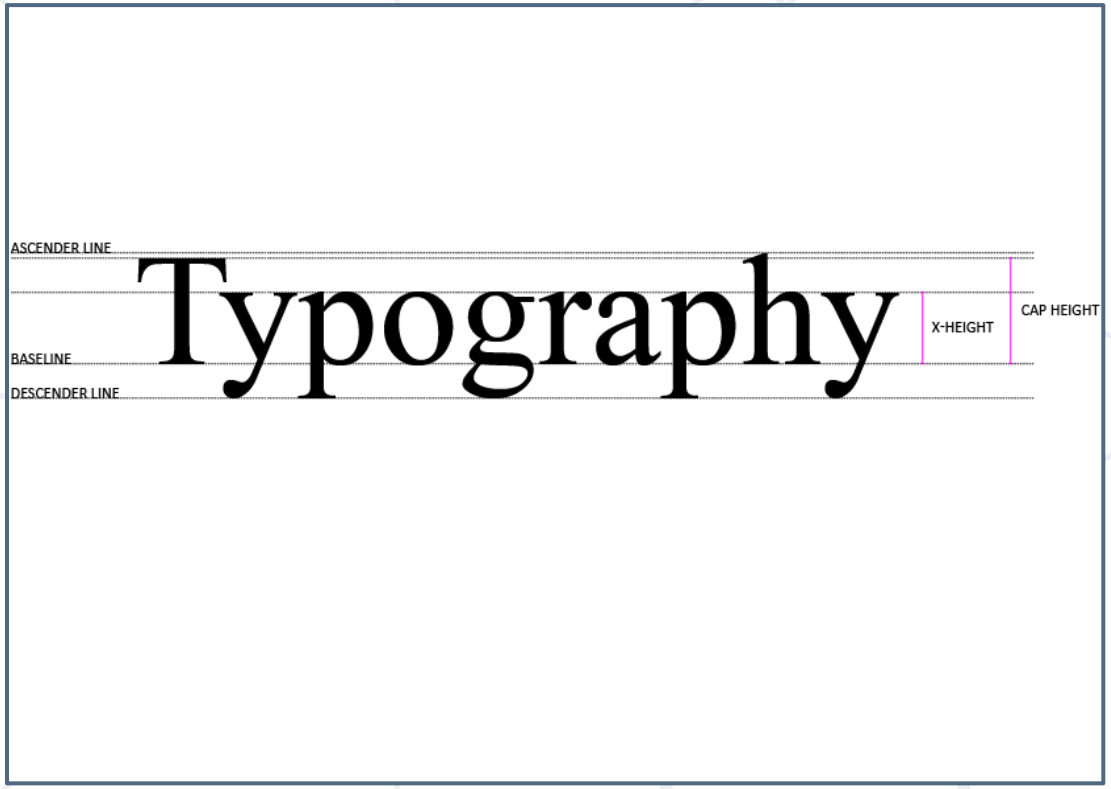
Table of Contents

Have a Question?

Software University

sli.do

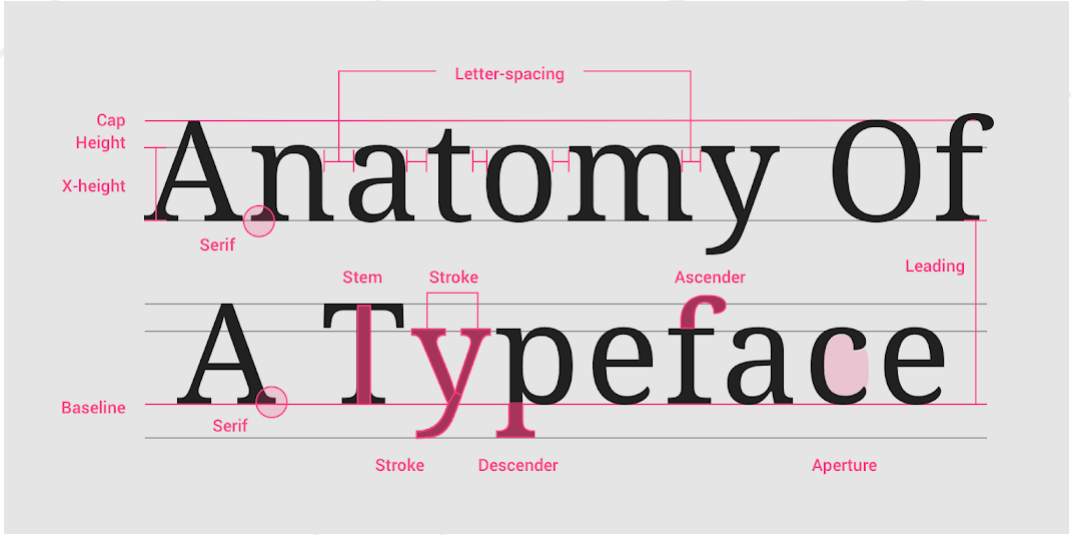
#html-css



CSS Font Properties

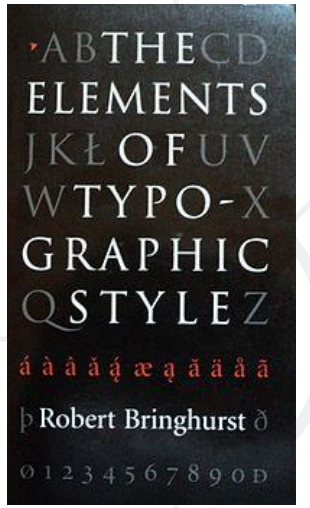
What is Typography?

- Typography is the art and technique of arranging type to make written language
  - **Legible**, **readable** and **appealing**
- Typography is the visual component of the written word

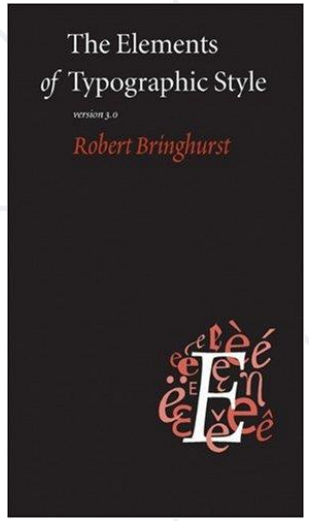


Books

- ***The Elements of Typographic Style*** - the authoritative book on typography and style



- ***The Elements of Typographic Style Applied to the Web*** - a practical guide to web typography



Principles of Readability

- **Readability** is one of the most important aspects of Web design usability
- Website readability is a measure of how easy it is for visitors to **read** and **understand** online text
- Readability depends on both a text's presentation and its context
- Poor readability scares readers away from the content

Keys to Readable Typography

- User-Friendly Headers
- White Space - scannable and consistent text
- Emphasis of Important Elements
- Good Margins to avoid walls of text
- Scannable Text
- Consistency
- Organization of Information

Web Safe Fonts

- Maximum **compatibility** between browsers/operating systems
- Generic family fonts:

Font Name	
Arial	Times New Roman
Courier New	Trebuchet MS
Georgia	Verdana

Font Files

- Font files contain one or more fonts that can be accessed by the **OS** and **applications**
- Most modern fonts are stored in the following formats:
  - OpenType (`.OTF`)
  - TrueType (`.TTF`)
  - Web Open Font Format (`.WOFF`)
  - Web Open Font Format 2 (`.WOFF2`)

Font Properties

Font Properties

- The **CSS font** properties **styles** the font of the text:
  - font-family / font-face (e. g. **sans-serif**, **Arial**)
  - font-size / line-height (e. g. **18pt**)
  - font-weight (e. g. **bold**)
  - font-style (e. g. ***italic***)
  - font-variant (e. g. **SMALLCAPS**)

Font Family Name

- **Font family name** specifies one or several system font names
  - For example, "**Arial**" and "**Helvetica**" are font families

```
<p class=ar>Arial</p>
<p class=tim>Times</p>
```

Arial

Times

```
p.ar {
  font-family: Arial, Helvetica;
}
p.tim {
  font-family: "Times New Roman";
}
```

  - **White-space** in the font name must be surrounded by **quotes**
  - When the first font is **missing**, the **next** is loaded

Using External Fonts: @Font-face

- Specifies a **custom font** from external file / URL

```
@font-face {
  font-family: "Open Sans";
  src: url("/fonts/opensans.woff") format("woff");
}
```

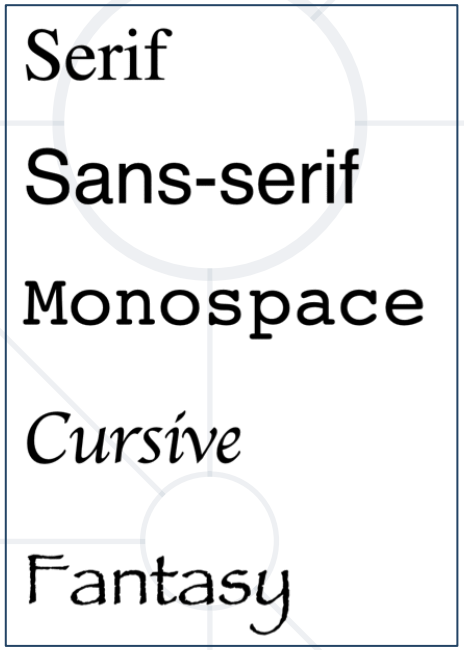
```
<p style="font-family: Open Sans">Open Sans Demo</p>
```

Open Sans Demo

- See fonts.google.com for free open Web fonts

Generic Name

- `<generic-name>` - a broad class of similar fonts used in a prioritized list of fonts
  - **serif** - all characters have stroke endings
  - **sans-serif** - no character has stroke endings
  - **monospace** - all characters have the same width
  - **cursive** - handwritten fonts
  - **fantasy** - decorative fonts



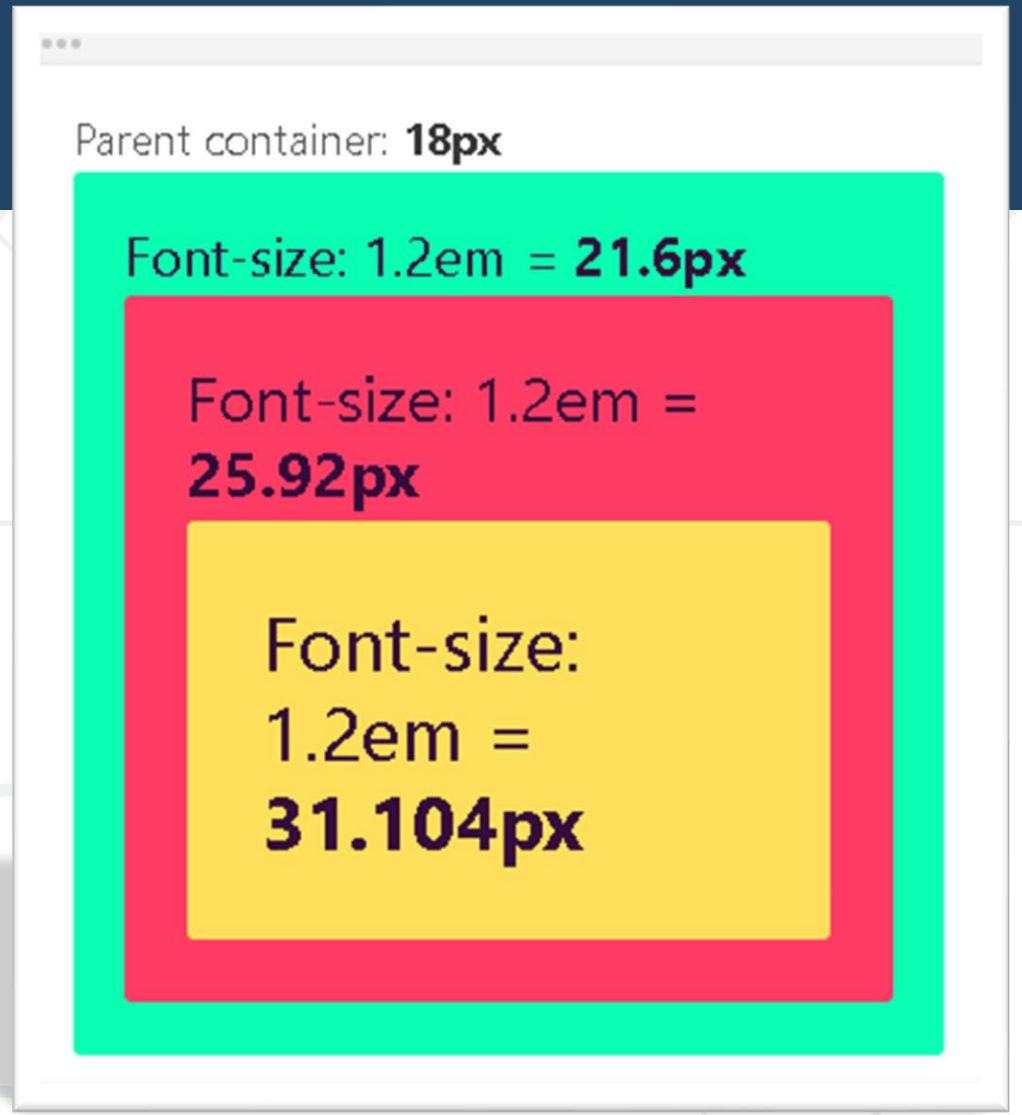
Font Size

- `font-size` – defines the text size
  - `px` / `pt` values (e. g. **18px** / **24pt**)
    - 1px == 0.75pt == 1/96 inch
  - `em` values – relative to the original size, multiplied by a **scale** factor

```
font-size: 1.2em;
```

  - `rem` values – relative to the HTML root size (the `<html>` element)

```
font-size: 1.5rem;
```



font-size: 16px

font-size: 1.5rem == 24px

Font Weight: Thin / Normal / Bold

- `font-weight` defines how weight is the font
  - Thin, normal, **bold**, or value [100 … 900]

Value	Name
100	thin
300	light
400	normal
700	bold

```
font-weight: thin;
```

```
font-weight: 300;
```

```
font-weight: 400;
```

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

Font Style: Normal / Italic

- `font-style` – defines how much the text is ***slanted***
  - `normal` – the text is not slanted

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

Normal font style

  - `italic` – the letters are slightly slanted

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

Italic font style

  - `oblique` – the letters are more slanted than italic

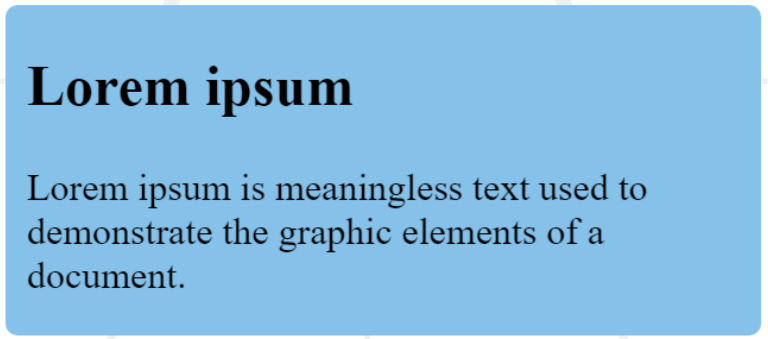
```
font-style: oblique;
```

Oblique font style

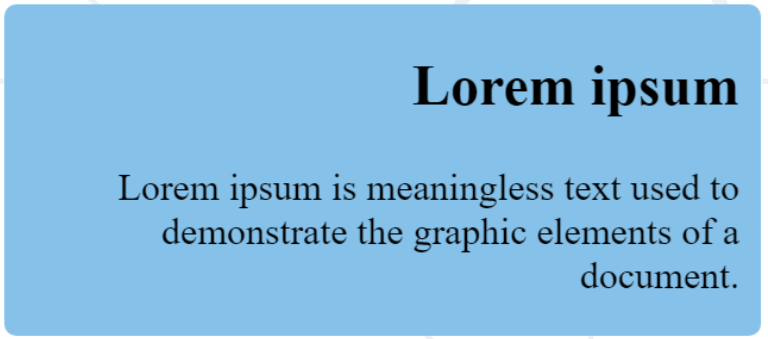
Text Align: Left / Right / Center / Justify

- `text-align`
  - Defines the **horizontal alignment**

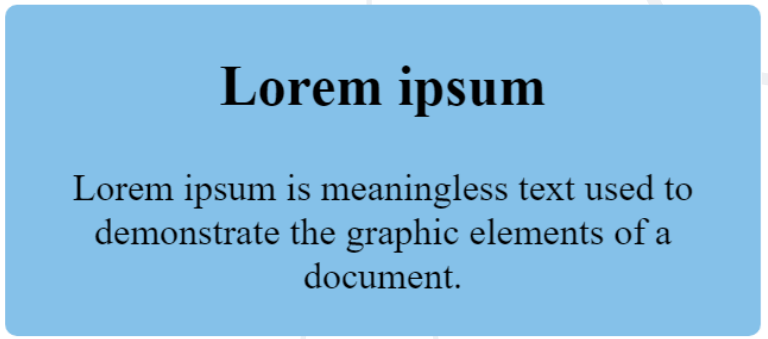
```
text-align: left;
```



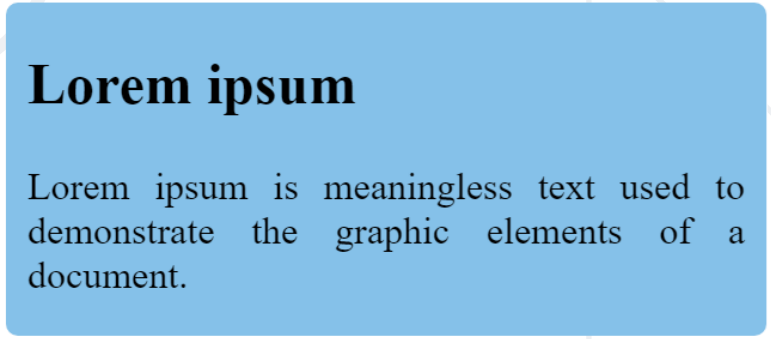

```
text-align: right;
```




```
text-align: center;
```




```
text-align: justify;
```

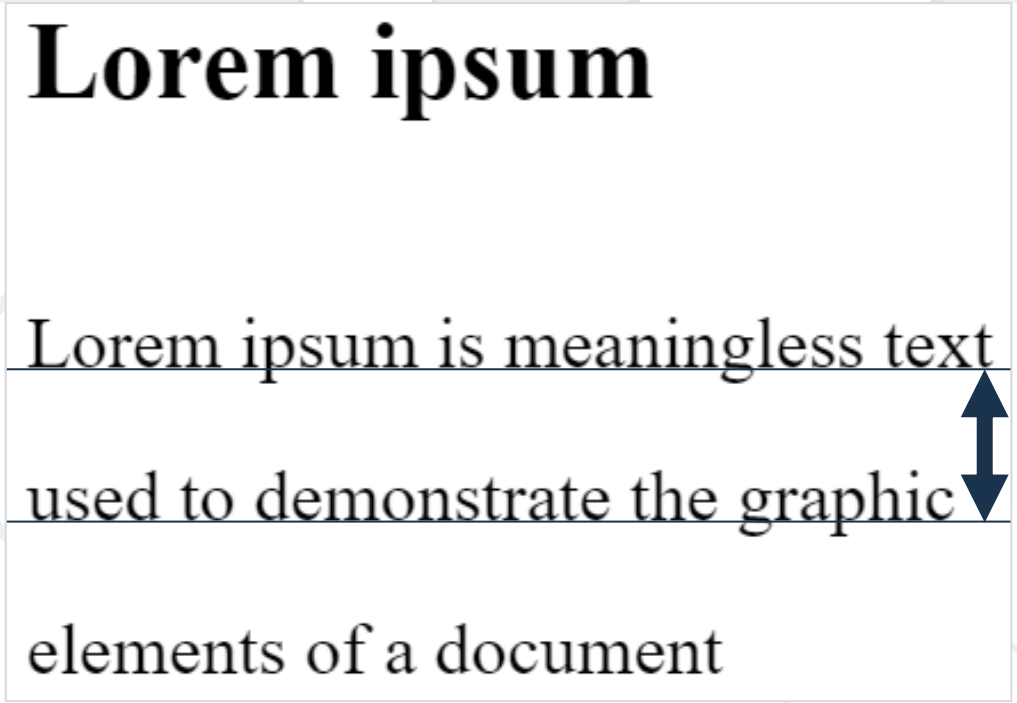


Line Height

- `line-height` – defines the **height** of a single line of text
  - Measures: **unitless** / **pt** / **px** / **em** / **rem**

```
<article>
  <h1>Lorem ipsum</h1>
    <p> Lorem ipsum is
meaningless text used to
demonstrate the graphic
elements of a document.</p>
</article>
```

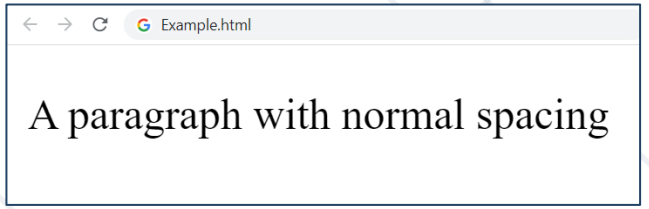
```
p { line-height: 2.1; }
```



Letter Spacing

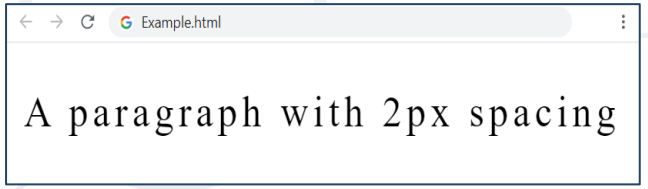
- `letter-spacing` - defines the spacing between the characters of a block of text
  - `normal` - the spacing between the characters is normal

```
letter-spacing: normal;
```



  - Using `pixels`

```
letter-spacing: 2px;
```



Text Decoration

- `text-decoration` - defines how the text content of the element is decorated: `overline`, `underline`, `line-through`
  - `none` - removes any text decoration

```
text-decoration: none;
```

  - `line-through` - draws a line across the text

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

~~text~~

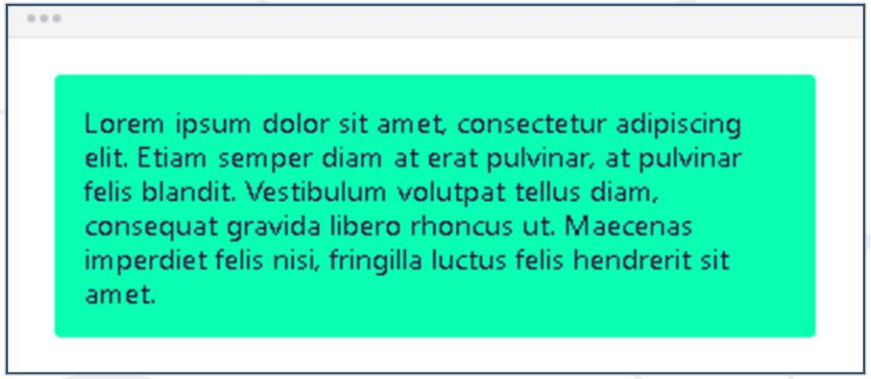
Text Indent

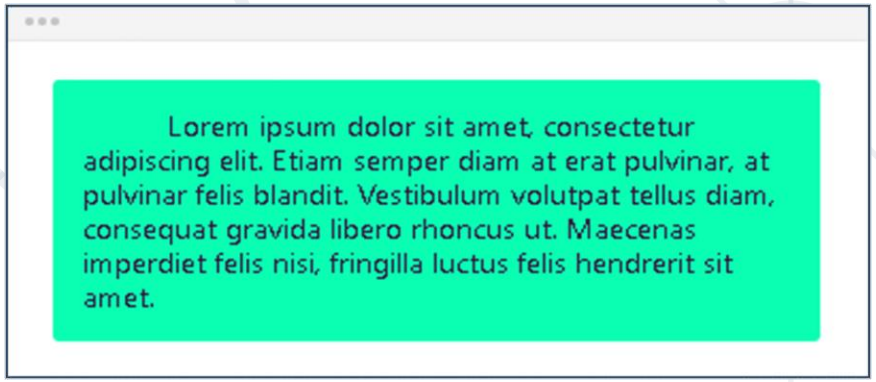
- `text-indent` - defines the indentation of the element's first line of text
  - The text is not indented

```
text-indent: 0;
```

  - The text is indented

```
text-indent: 40px;
```

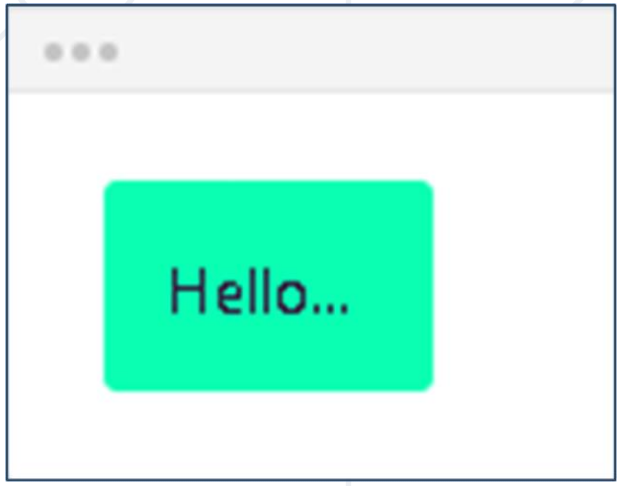




Text Overflow

- `text-overflow` - defines how the hidden text content behaves if it's overflowing
  - `ellipsis` - the overflowing content is replaced by . . .

```
text-overflow: clip;
```



Text Transform

- `text-transform` - specifies how to capitalize text
  - `capitalize` - turns the **first letter** of each word into a capital letter

```
text-transform: capitalize;
```

It Is A Long Established Fact That A Reader Will Be Distracted.

  - `uppercase` - turns all characters to uppercase

```
text-transform: uppercase;
```

IT IS A LONG ESTABLISHED FACT THAT A READER WILL BE DISTRACTED.

  - `lowercase` - turns all characters to lowercase

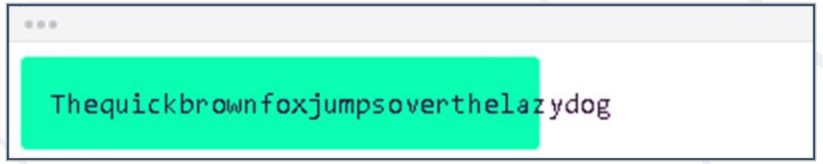
```
text-transform: lowercase;
```

it is a long established fact that a reader will be distracted.

Word Break

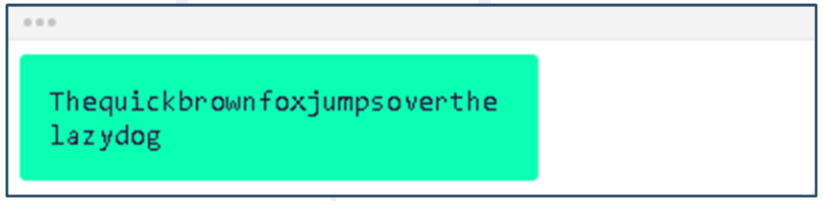
- `word-break` - defines how words should break when reaching the end of line
  - `normal` - words with no space will **NOT** break

```
word-break: normal;
```



  - `break-all` - words with no space will **break** as soon as they reach the end of a line

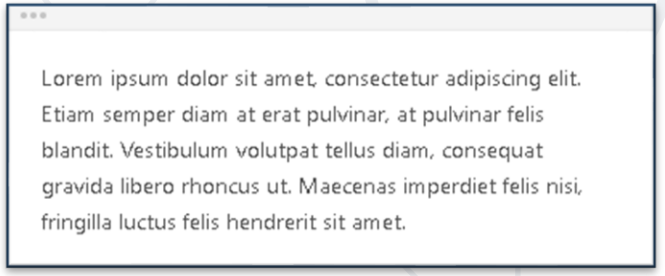
```
word-break: break-all;
```



Text Shadow

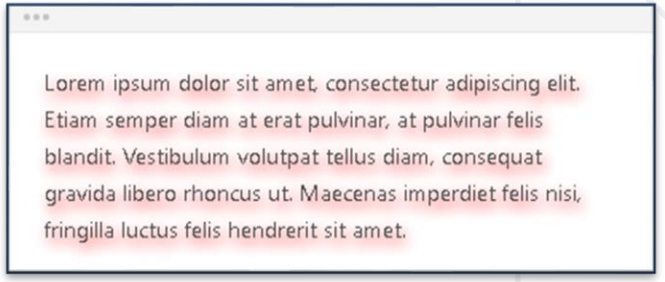
- `Text-shadow` - defines the shadow of the text content
  - None - the text content has **no shadow**

```
p {
    text-shadow: none;
}
```



  - Text-shadow: `<horizontal>` `<vertical>` `<blur>` `<color>`

```
p {
    text-shadow: 2px 4px 10px red;
}
```



Text Color

- 140+ **predefined color names** (e. g. **green**, **red**, **blue**, **coral**, …)

```
color: red;
```

Lorem ipsum

- **Hexadecimal** color code in format **#RGB** or **#RRGGBB**:

```
color: #05ffb0;
```

Lorem ipsum

- Decimal `rgb()` color codes (red, green, blue values):

```
color: rgb(125, 125, 255);
```

Lorem ipsum

- Decimal `rgba()` color codes (red, green, blue, alpha opacity):

```
color: rgba(255, 0, 0, 0.5);
```

Lorem ipsum

Background Color

- `background-color` - defines the color of the background
  - `transparent`

```
background-color: transparent;
```

  - Specify the background color with:
    - HEX
    - RGB/RGBA
    - Named color

```
background-color: navy;
```

Mouse Cursor

- Sets the **mouse cursor** when hovering the element:

```
cursor: pointer;
```

some text

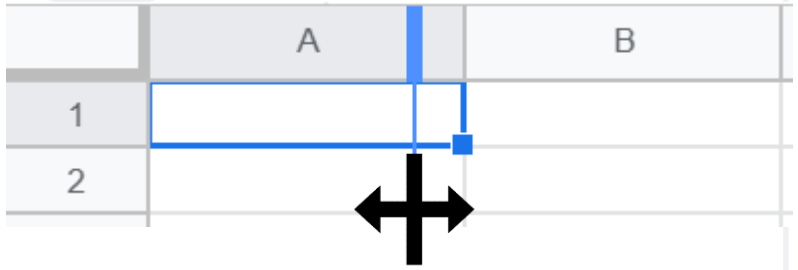
```
cursor: move;
```

move

```
cursor: none;
```

no cursor

```
cursor: col-resize;
```



Outline

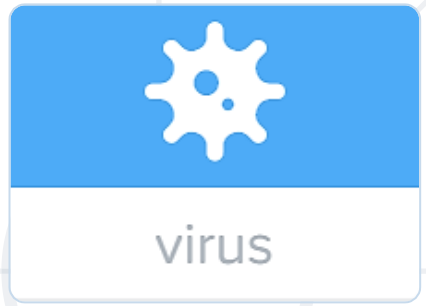
- **Outline**
  - Outline-width - defines the **width** of the element's outlines
  - Outline-style - defines the **style** of the element's outlines
  - Outline-color - defines the **color** of the element's outlines

```
p {
    outline: 4px dotted red;
}
```

Font Awesome Icons

Font Awesome

- **Font Awesome** provides vector icons, emojis, etc.
  - Add the following link inside **<head>**
  - Or import Font Awesome in the **CSS** file
  - Choose an icon → copy the **<i>** element → paste it in your HTML file

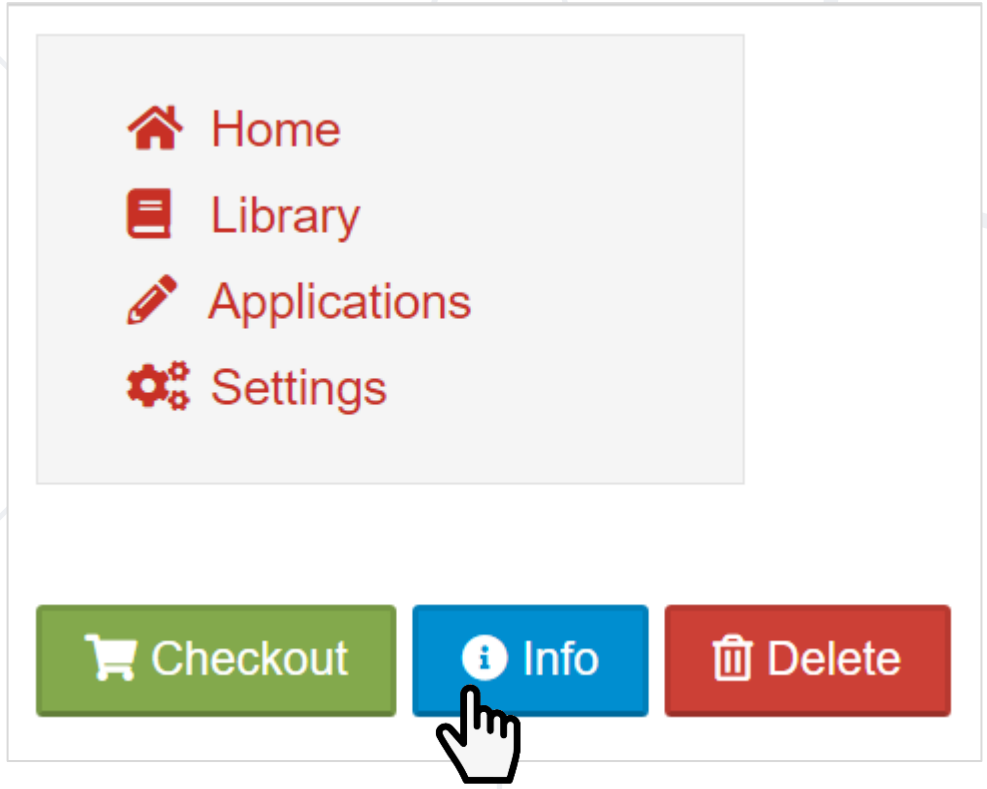



```
<head>
  <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.10.2/css/all.css">
</head>
<body>
  <a href="#"><i class="fa fa-home fa-fw"></i>Home</a>
</body>
```

```
@import 'https://use.fontawesome.com/releases/v5.10.2/css/all.css';
```

Font Awesome Icons – Exercise

- Using **HTML**, **CSS** and **Font Awesome icons** create a **navigation menu** and **buttons** like the following:



- Hints:

```
<i class="fas fa-home"></i>
<i class="fas fa-book"></i>
<i class="fas fa-pencil-alt"></i>
<i class="fas fa-cogs"></i>
<i class="fas fa-shopping-cart"></i>
<i class="fas fa-info-circle"></i>
<i class="far fa-trash-alt"></i>
```

Summary

- What is **Typography**?
- The principles of **readability**
- **CSS properties**: font-family, font-size, font-style, color, background
- **Font Awesome** Icons

Questions?

SoftUni

Software University | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids | Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

createX

Postbank
Решения за твоето утре

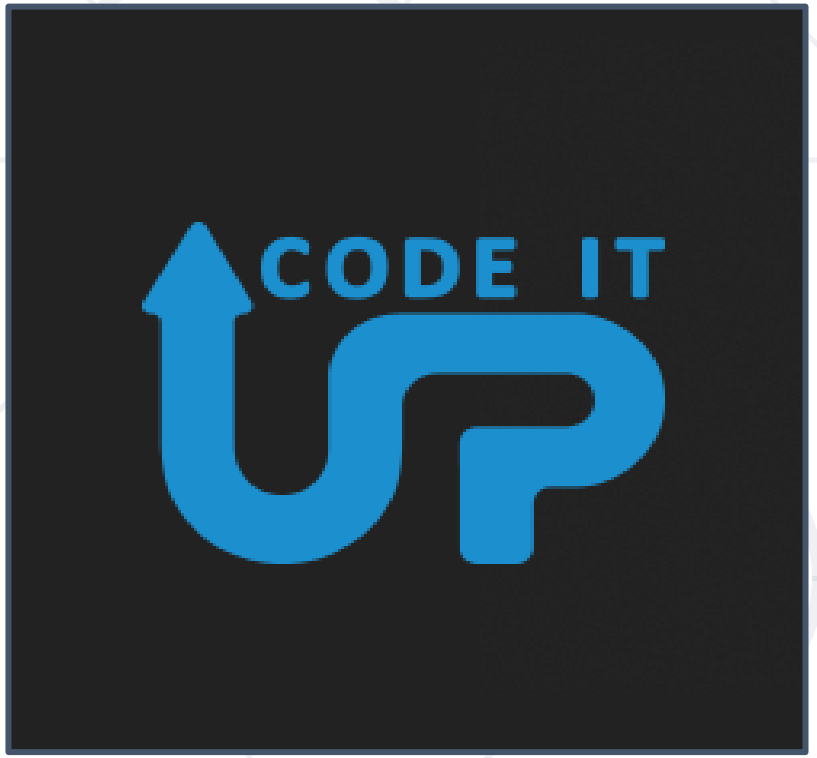
BOSCH

DXC TECHNOLOGY

SmartIT

Educational Partners

Software University



License

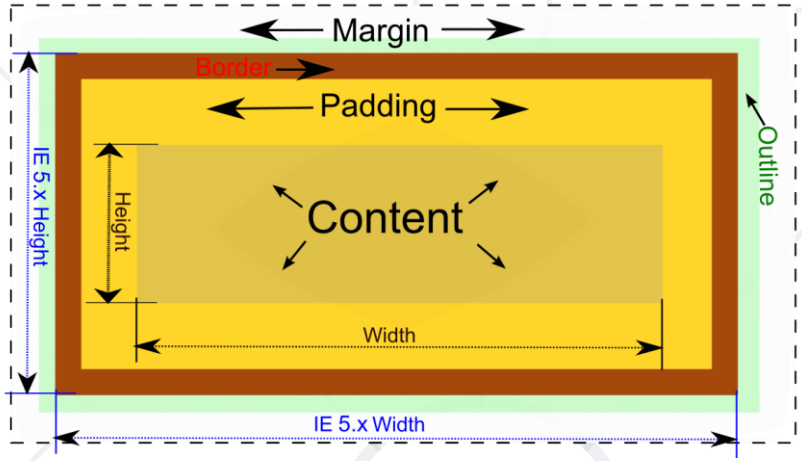
- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

CSS Box Model

Margin, Border, Paddings, Actual Content



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Table of Contents

Have a Question?

Software University

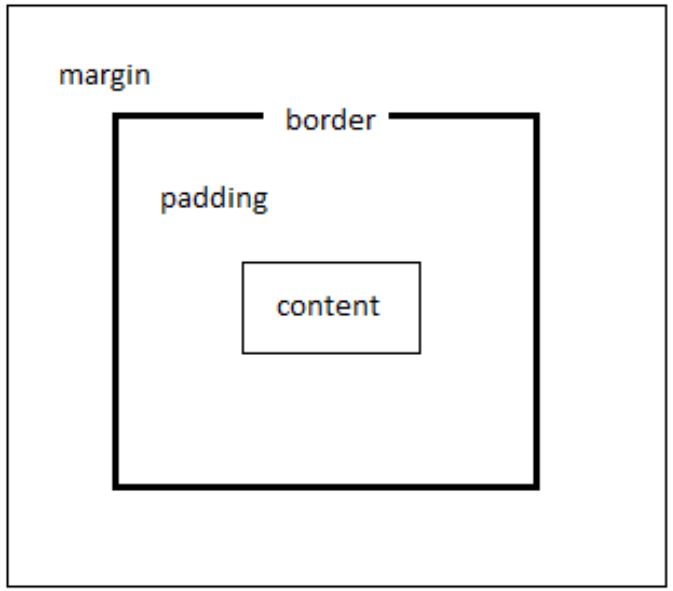
sli.do

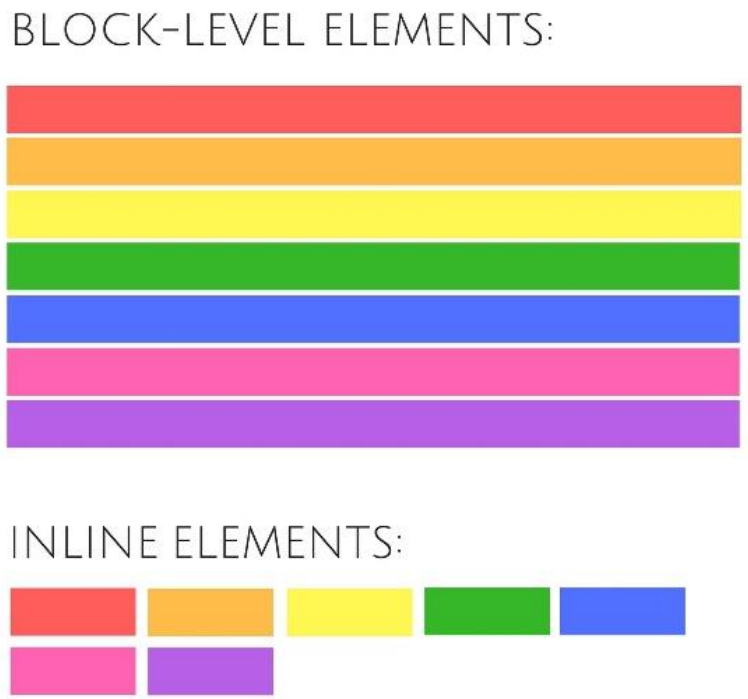
#html-css

What is Box Model?

What is CSS Box Model?

- The **CSS box model** is essentially a **box** that wraps **around every** HTML **element**
  - Visible from the browser Dev Tools: **[F12]**
- HTML elements have:
  - **Content** – the HTML element itself
  - **Padding** – transparent area around the content
  - **Border** – line that goes around the padding
  - **Margin** – transparent area outside the border





Block-level and Inline HTML Elements

Block and Inline Elements

- HTML is made up of various elements that act as the **building blocks** of web pages
- CSS has two different types of boxes - **block** and **inline**
  - **Block** Elements
  - **Inline** Elements
  - **Inline-block** Elements

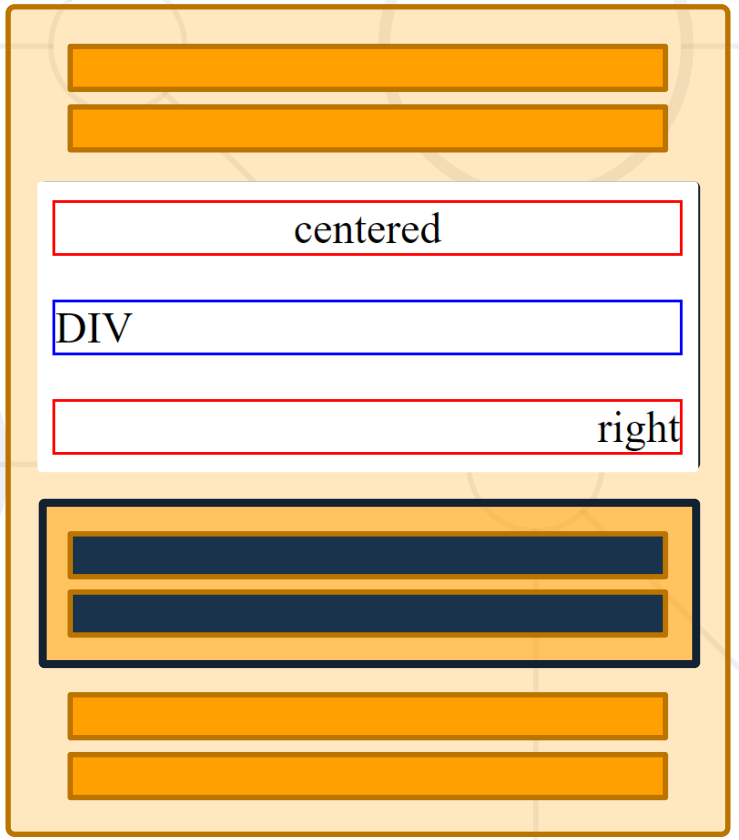
Block Elements

- Block element: starts on a **new line**, and fills up the horizontal space left and right on the web page
- Some examples of block elements are:
  - `main`, `header`, `article`, `section`, `fieldset`, `nav`, `ul`, `ol`, `li`, `form`, `h1-h6`, `p`, `div`

Block Elements – Example

```
<p style="border:1px solid red;
text-align:center">centered</p>
<div style="border:1px solid
blue">DIV</div>
<p style="border:1px solid red;
text-align:right">right</p>
```

display: block



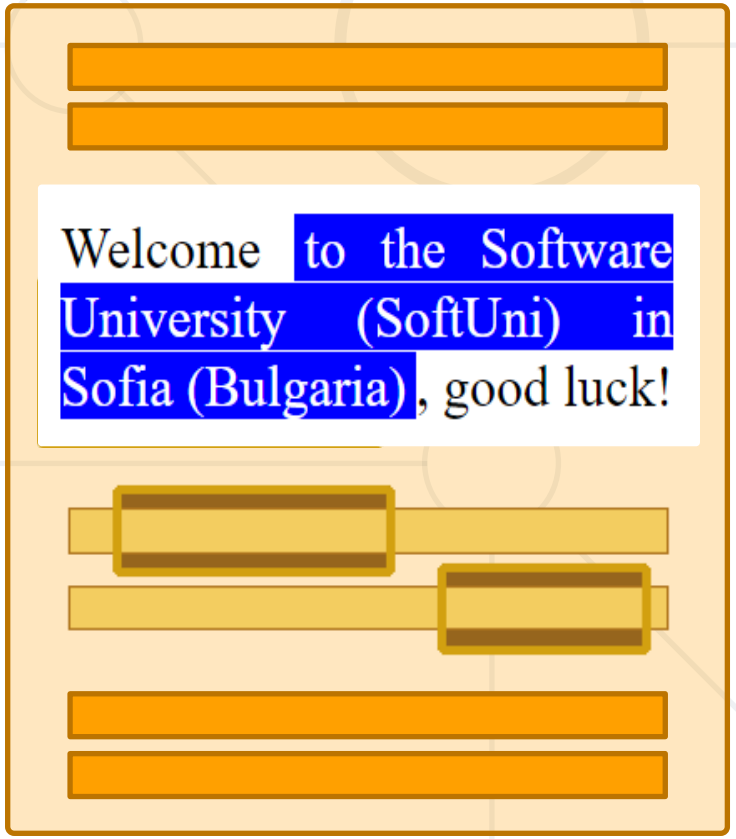
Inline Elements

- Inline element: **don't start** on a new line. They appear on the same line as the content and tags beside them
- Some examples of inline-block elements are:
  - `a`, `label`, `map`, `span`, `strong`, `em`, `i`, `img`, `textarea`, `input`, `button`, `select`
- You can add margins and padding just on **right** and **left** sides of any inline element

Inline Elements – Example

```
<p style="text-align:justify"> Welcome
<span style="color:white;
background:blue; padding-right:3px;
padding-left:3px;">
to the Software University (SoftUni)
in Sofia (Bulgaria)</span>, good
luck!</p>
```

display: inline



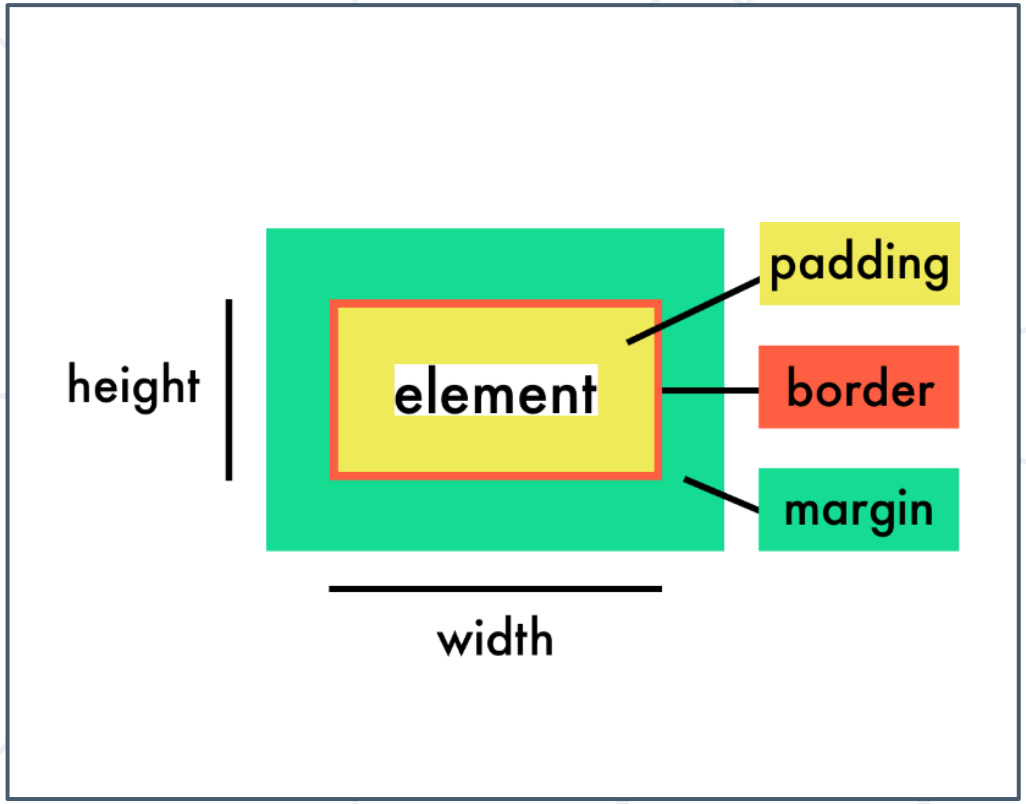
Inline-Block Elements

- Inline-block elements are similar to inline elements
- They can have padding and margins added on **all four sides**
- You have to declare `display: inline-block` in your CSS code
- One common use for using inline-block is for creating navigation links horizontally

Inline-Block Elements – Example

```
<div style="text-align:justify;">
   <div style="display:inline-block;
   background:green">green</div>
   <div style="display:inline-block;
   background:red">red block</div>
   …
</div>
```

display: inline-block



CSS Width and Height Dimensions

Width

- **Width** – defines the width of the element
  - **auto** (default)
  - Auto-calculated width
  - **pixels** / **em** / **rem**
  - Fixed width
  - **percentages**
  - Relative to container's width

```
article {
  width: auto;
  background: #8ce;
}
```

Lorem ipsum

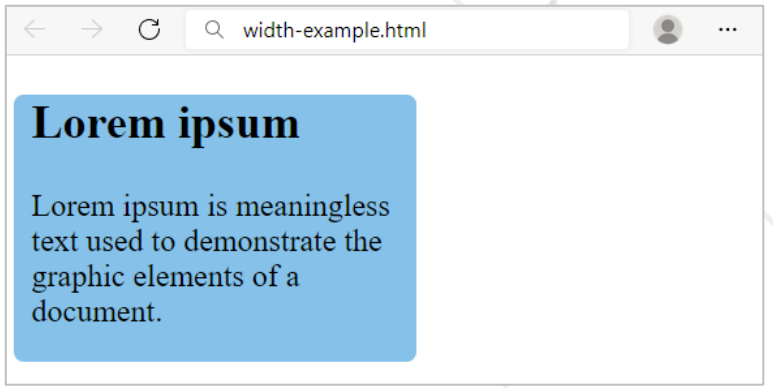
Lorem ipsum is meaningless text used to demonstrate the graphic elements of a document.

```
article {
  width: 240px;
  background: #8ce;
}
```

Lorem ipsum

Lorem ipsum is meaningless text used to demonstrate the graphic elements of a document.

```
article {
  width: 50%;
  background: #8ce;
}
```

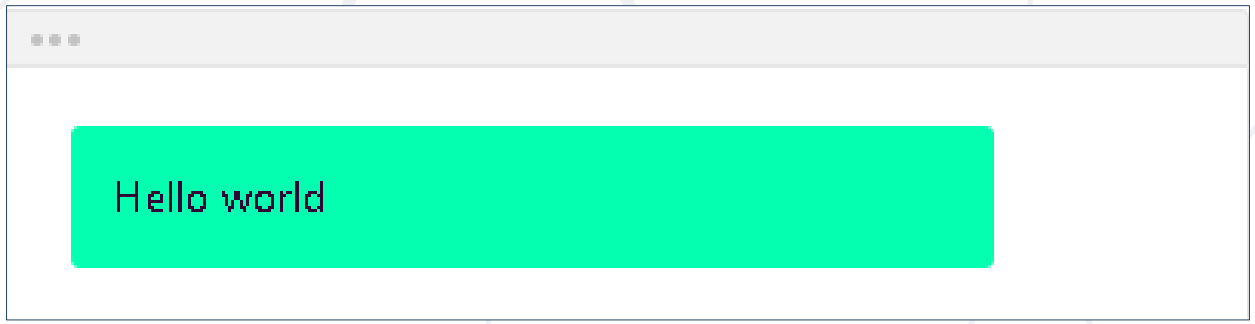


Max-width

- Max-width - defines the **maximum** width the element can be
  - `max-width: none;` - the element has **no limit** in terms of width
  - `max-width: 150px;`
  - `max-width: 2000px;` - you can use numeric values like **pixels**, **(r)em**, **percentages**...
- If the **maximum** width is **larger** than the element's **actual** width, the max width has **no effect**

Min-width

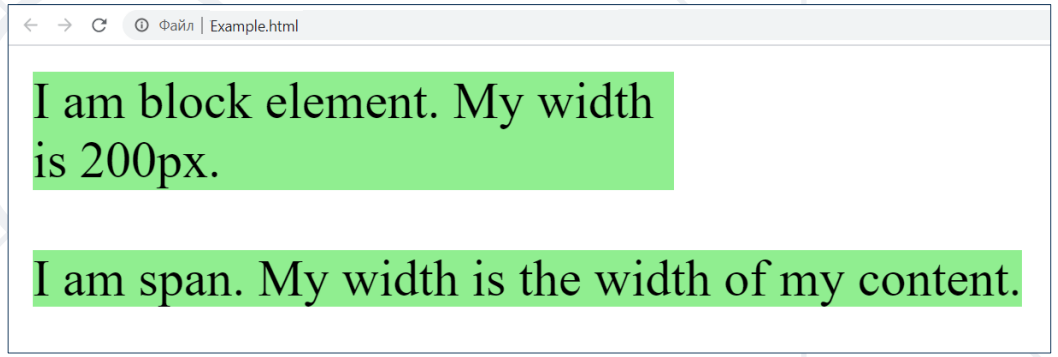
- Min-width - defines the **minimum** width the element
  - `min-width: 300px;` - if the **minimum** width is **larger** than the element's **actual** width, the min width will be applied
- If the **minimum** width is **smaller** than the element's **actual** width, the min width has **no effect** - `min-width: 5px;`



Width – Example

```
<body>
    <div>
        I am block element. My width is 200px.
    </div>
    <span>I am span. My width is the width of my content.</span>
</body>
```

```
div, span {
    width: 200px;
    background-color: lightgreen;
}
```



Height

- **Height** – defines the height of the element
  - **auto** (default)
  - Auto-calculated height

```
article {
  height: auto;
  background: #8ce;
}
```

Lorem ipsum

Lorem ipsum is meaningless text used to demonstrate the graphic elements of a document.

  - numeric values like **px** / **pt** / **em** / **rem** / **%**

```
article {
  height: 100px;
  background: #8ce;
}
```

Lorem ipsum

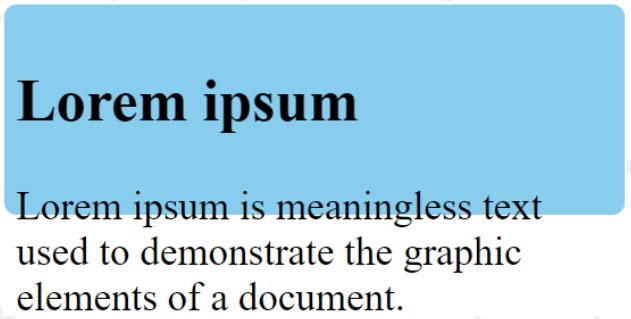
Lorem ipsum is meaningless text used to demonstrate the graphic elements of a document.

Overflow

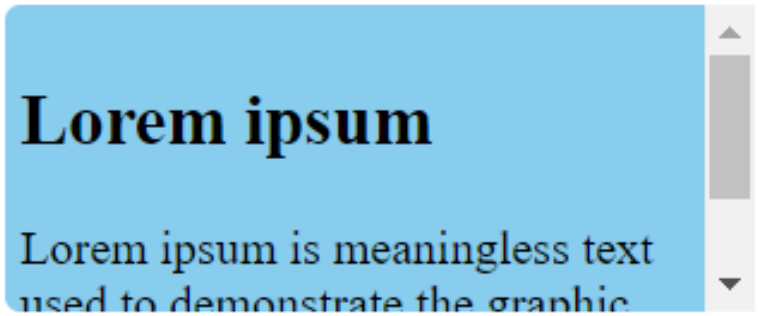
- The **overflow** property specifies what should happen if **content overflows** an element's box
  - visible
  - auto
  - scroll
  - hidden

```
article {
  overflow: visible;
}
```

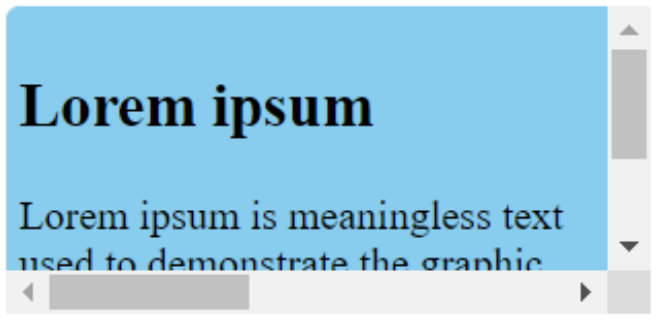
visible



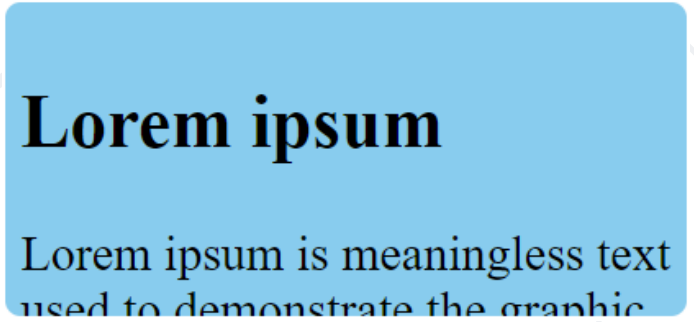
auto



scroll



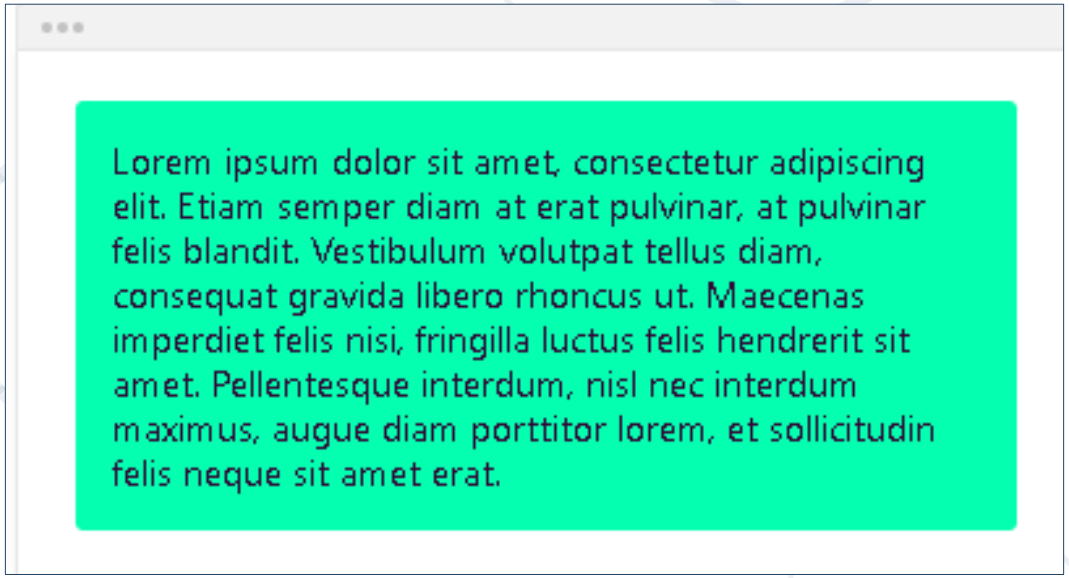
hidden



Max-height

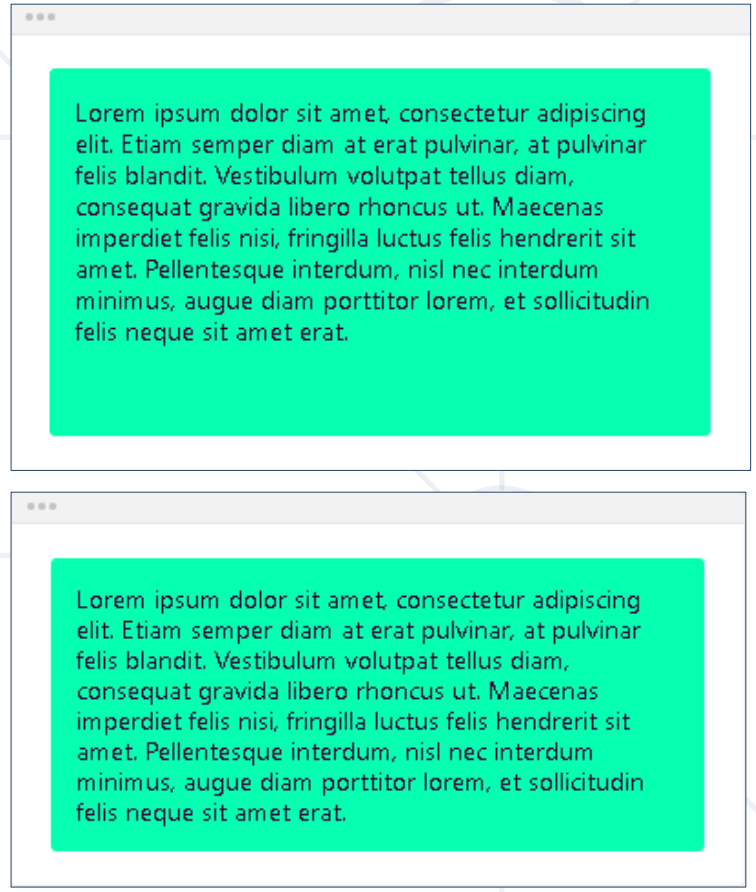
- Max-height - defines the maximum height the element can be
  - `max-height: none;` - the element has **no limit** in terms of height
  - `max-height: 2000px;`

    - if the **maximum** height is **larger** than the element's **actual** height, the max height has **no effect**



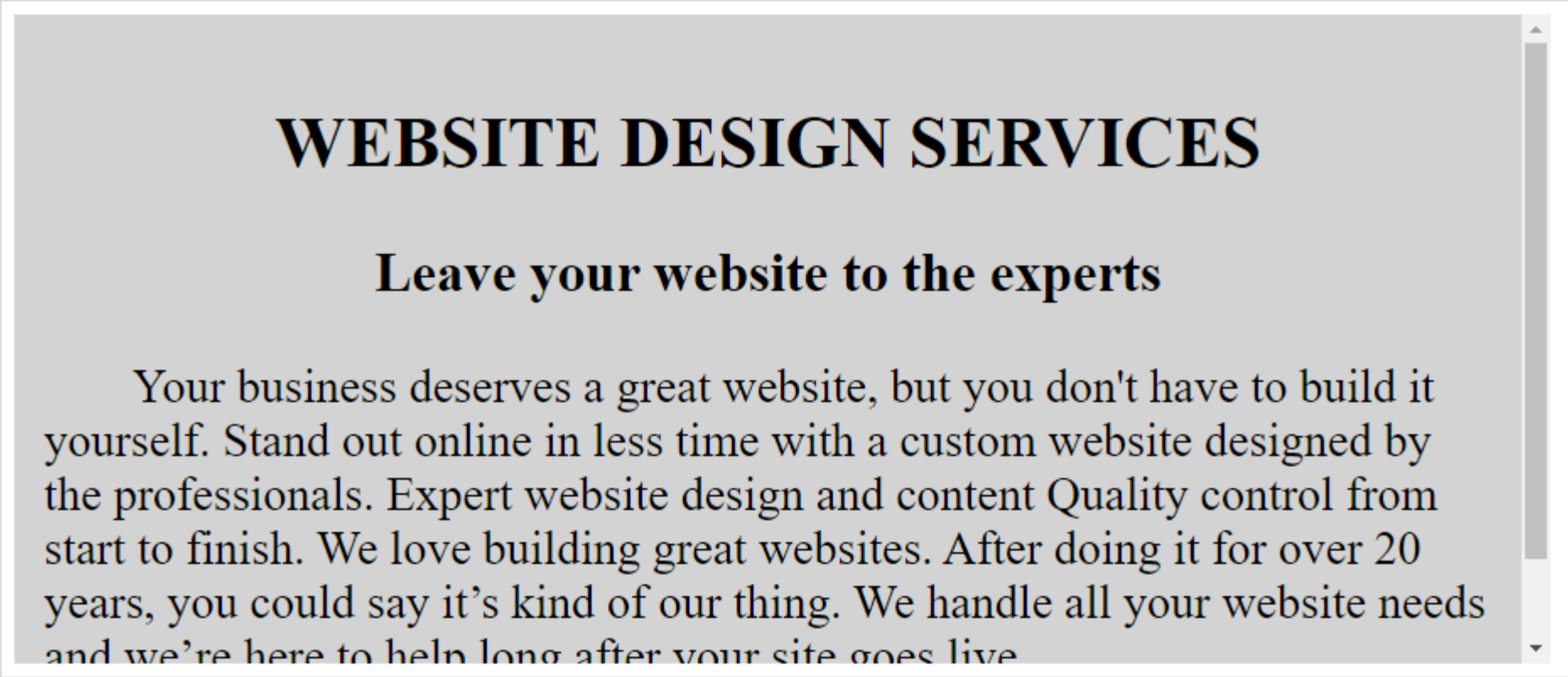
Min-height

- Min-height - defines the minimum height the element
  - `min-height: 200px;` - if the **minimum** height is **larger** than the element's **actual** height, the min height will be applied
  - `min-height: 5px;` - if the **minimum** height is **smaller** than the element's **actual** height, the min height has **no effect**



Scrolling Article – Exercise

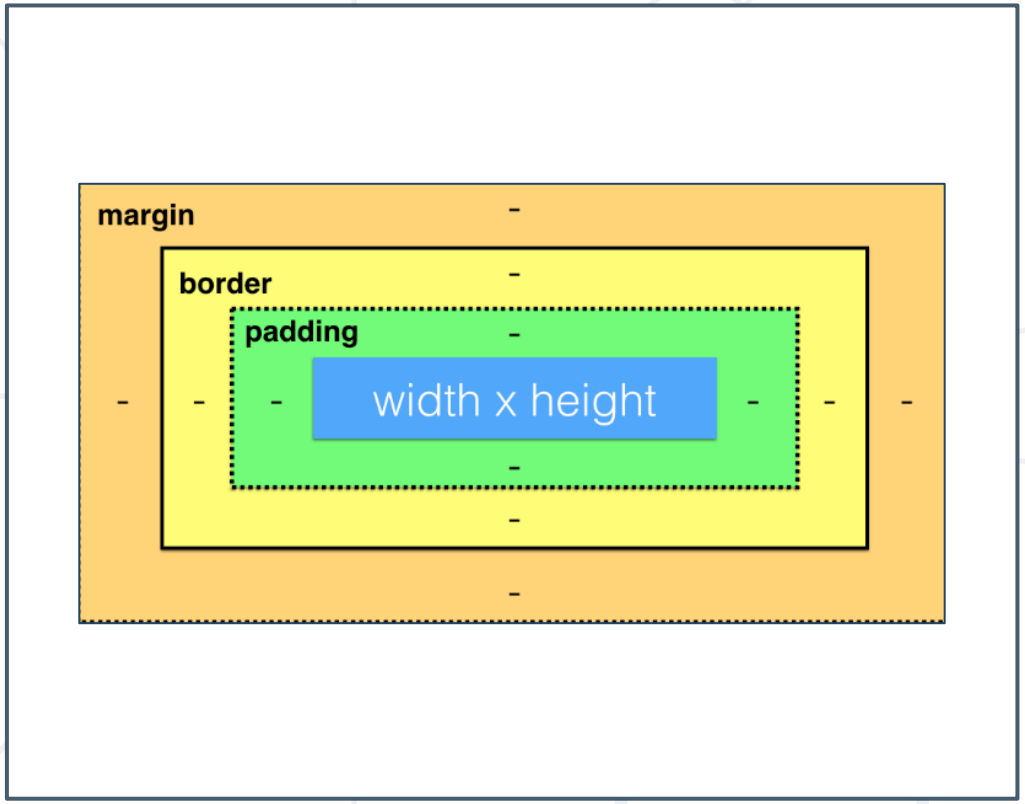
- Using **HTML** and **CSS** create a **section** with **articles** like the following:



Scrolling Article – Solution

- HTML constraints:
  - Use <section> with <article>, which contains: <h2>, <h3> and <p>
- Hints:

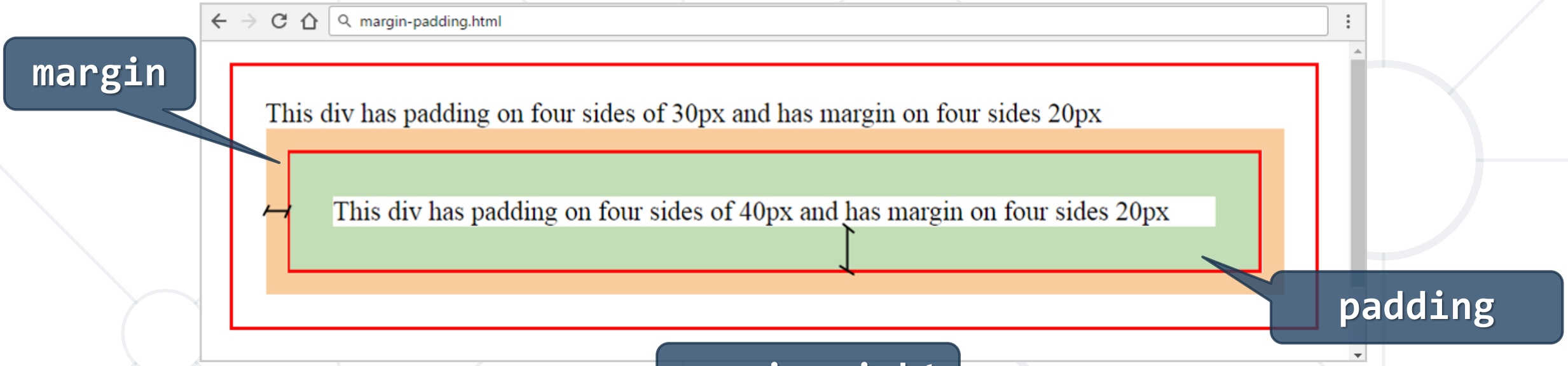
```
article {
  width: 500px;
  height: 200px;
  background-color: lightgray;
  overflow: auto;
  padding: 10px;
}
```



Margin, Padding and Borders

CSS Box Model Layers

Margins and Paddings



margin-right

margin-top

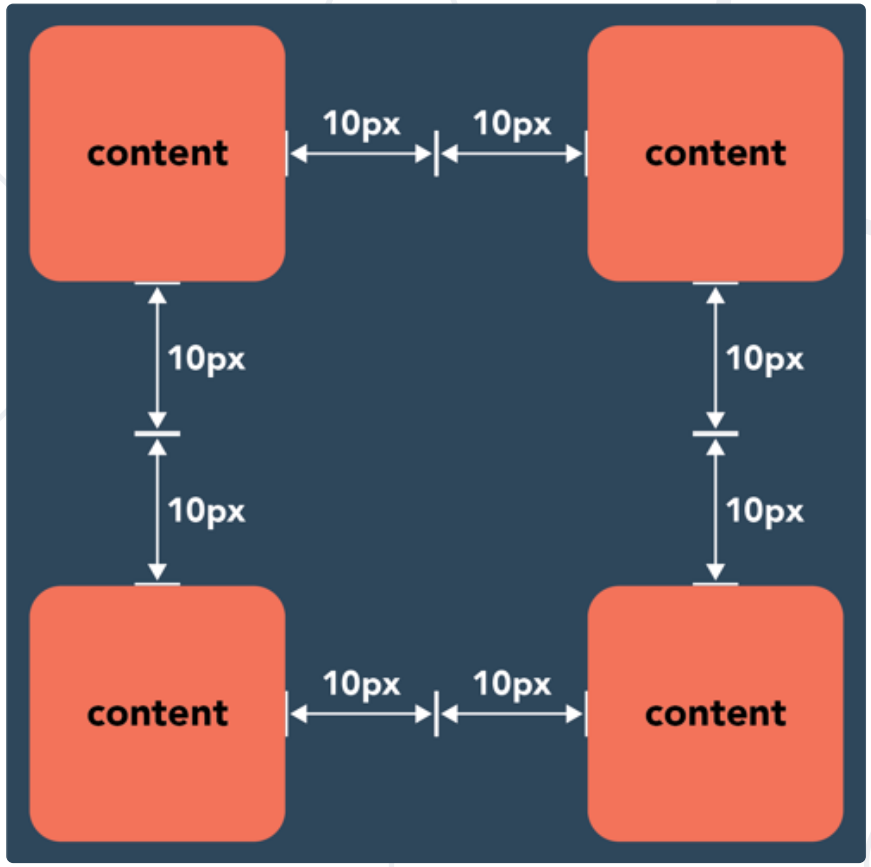
margin-bottom

```
div {
    margin: 20px 20px 20px 20px;
    padding: 30px 30px 30px 30px;
}
```

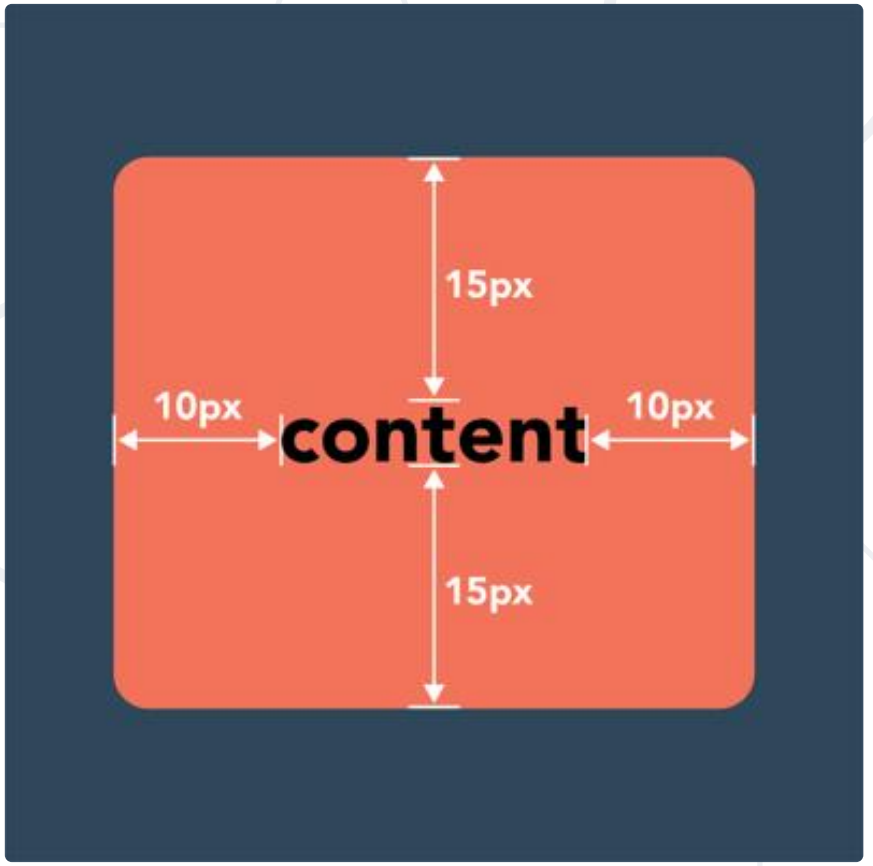
margin-left

Margins and Paddings

- **Margin** – defines the space **outside** the element



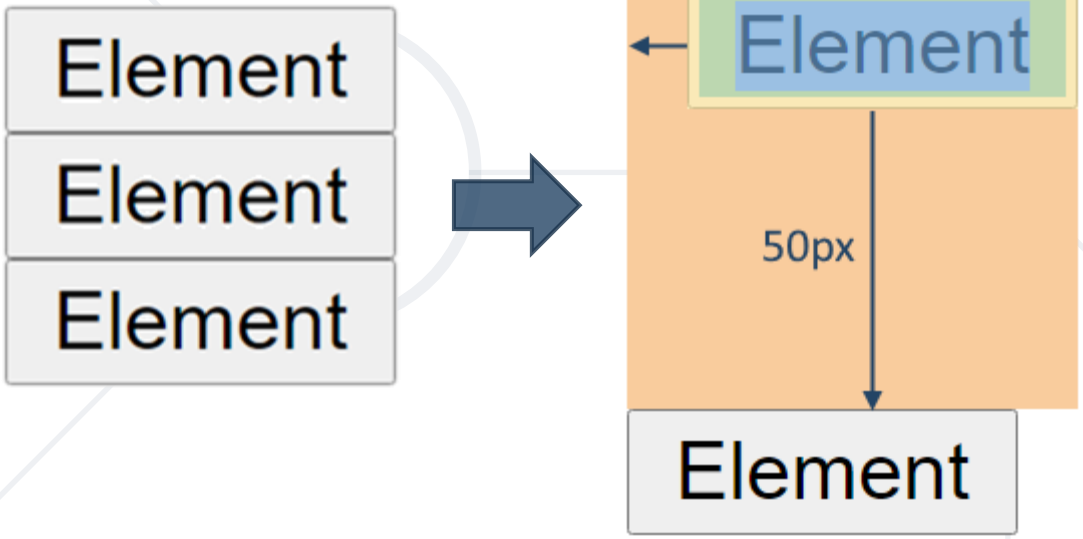
- **Padding** – defines the space **inside** the element



Margin

```
<button class="first">Element</button>
<button class="second">Element</button>
<button class="third">Element</button>
```

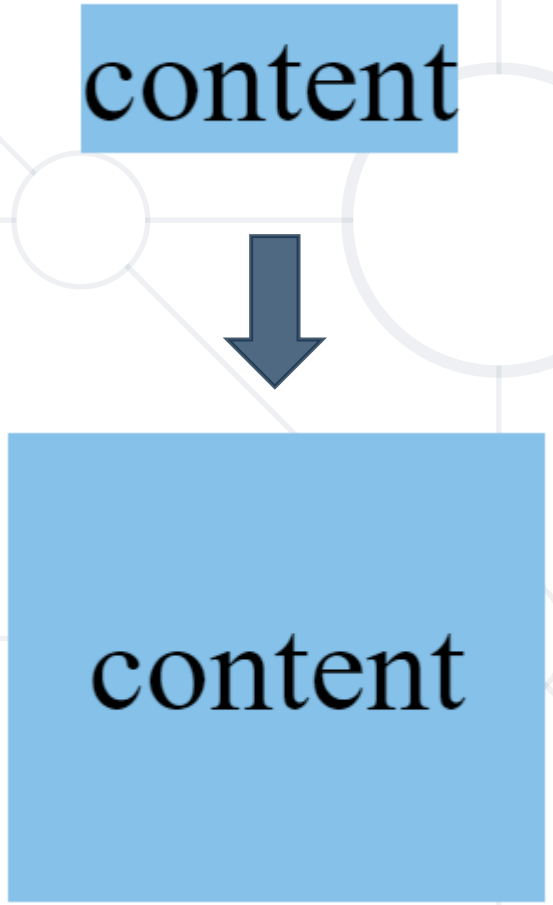
```
button {
  display: block;
}

button.second {
  margin-top: 50px;
  margin-left: 10px;
  margin-bottom: 50px;
  margin-right: 0;
}
```



Padding

```
<div>content</div>
```

```
div {
  background-color: #85c1e9;
  display: inline-block;
  text-align: center;

  padding-top: 20px;
  padding-left: 10px;
  padding-bottom: 20px;
  padding-right: 10px;
}
```



Shorthand Margin / Padding

- Shorthand margin rules:

```
button.second {
  margin: 10px 20px 10px 20px;
}
```

top, right, bottom, left

```
button.second {
  margin: 20px 10px;
}
```

top & bottom, left & right

- Shorthand padding rules:

```
div {
  padding: 5px 10px 8px 15px;
}
```

```
li {
  padding: 5x 10px;
}
```

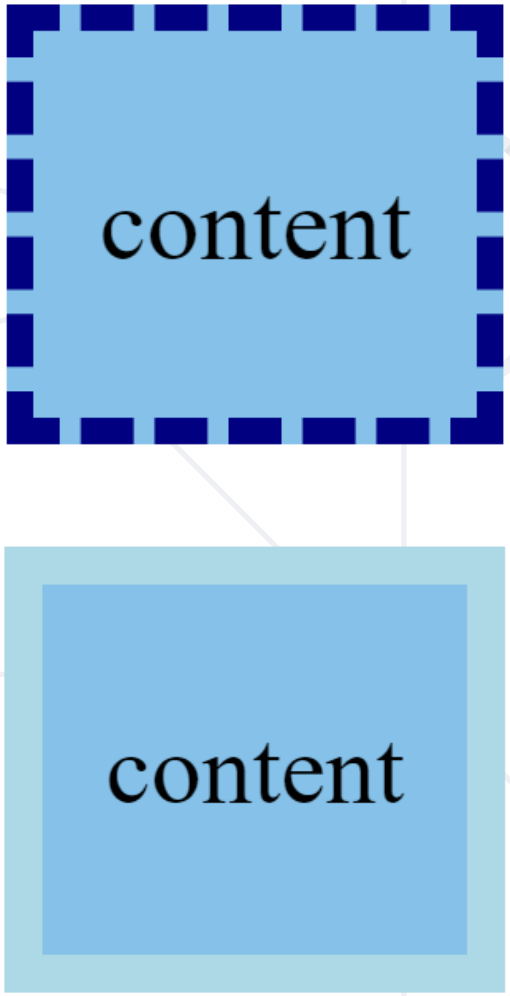
Border

- **Border** – define the style of the borders:
  - **width** (e. g. **1px** / **2px** / **3px**)
  - **style** (e. g. **solid** / **dashed** / **dotted**)
  - **color** (e. g. **blue** / **#eee**)

width style color

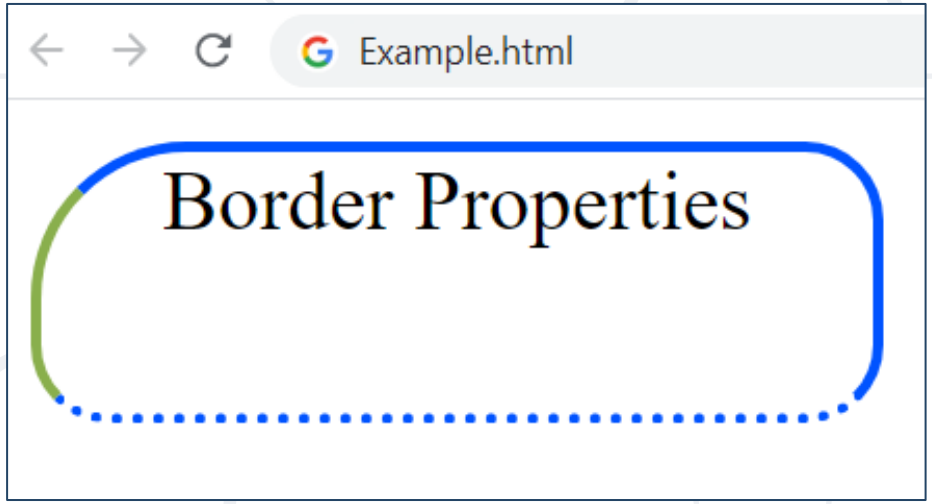
```
border: 4px dashed navy;
```

```
border: 6px solid lightblue;
```

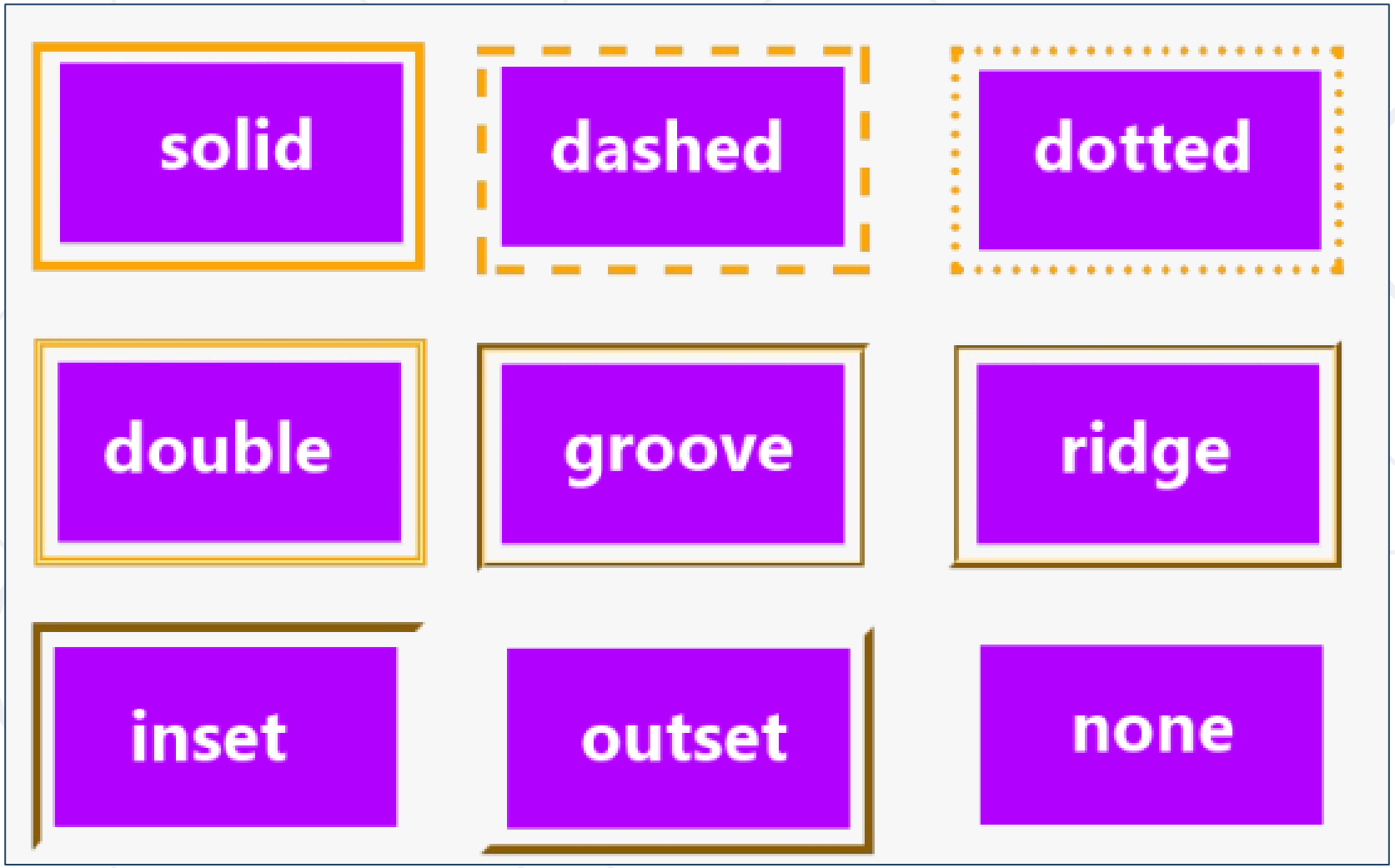


Border Properties

```
div {
  width: 160px;
  height: 50px;
  border-width: 2px;
  border-style: solid;
  border-color: #0053ff;
  border-radius: 15px;
  border-top-left-radius: 30px;
  border-bottom-style: dotted;
  border-left-color: #89af4c;
  text-align: center;
}
```



CSS Borders

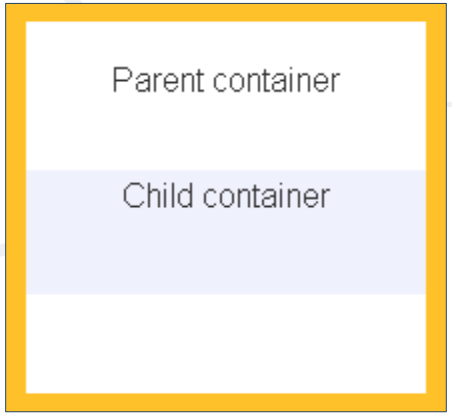


Include the Padding and Border in an Element's Total Width and Height

Box-sizing

- Sets how the total width and height of an element is calculated
  - `content-box` - initial and default value
  - The `width` and `height` properties include the content
  - They **do NOT include** the padding, border and margin

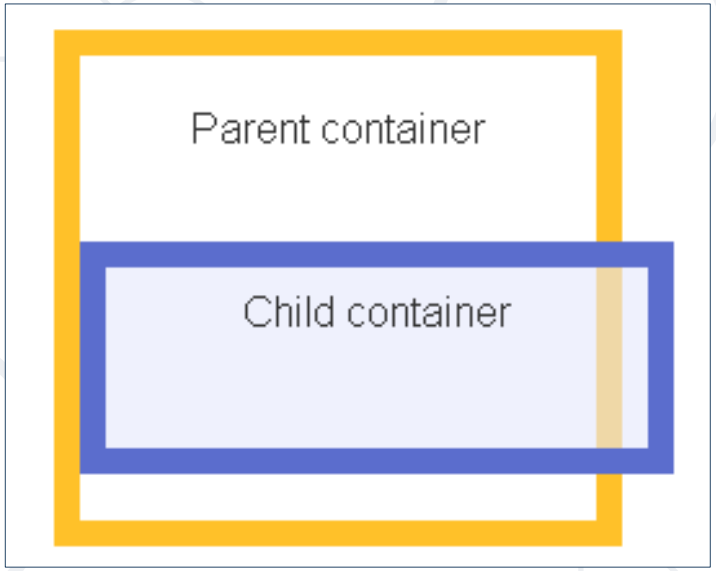
```
div {
  box-sizing: content-box;
  width: 200px;
}
```



Box-sizing

- The full width is: **200px + 2*10px + 2*5px = 230px**

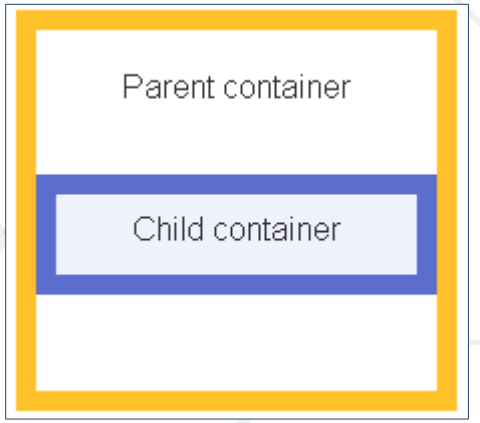
```
div {
   box-sizing: content-box;
   width: 200px;
   border: 10px solid #5b6dcd;
   padding: 5px;
}
```



Box-sizing

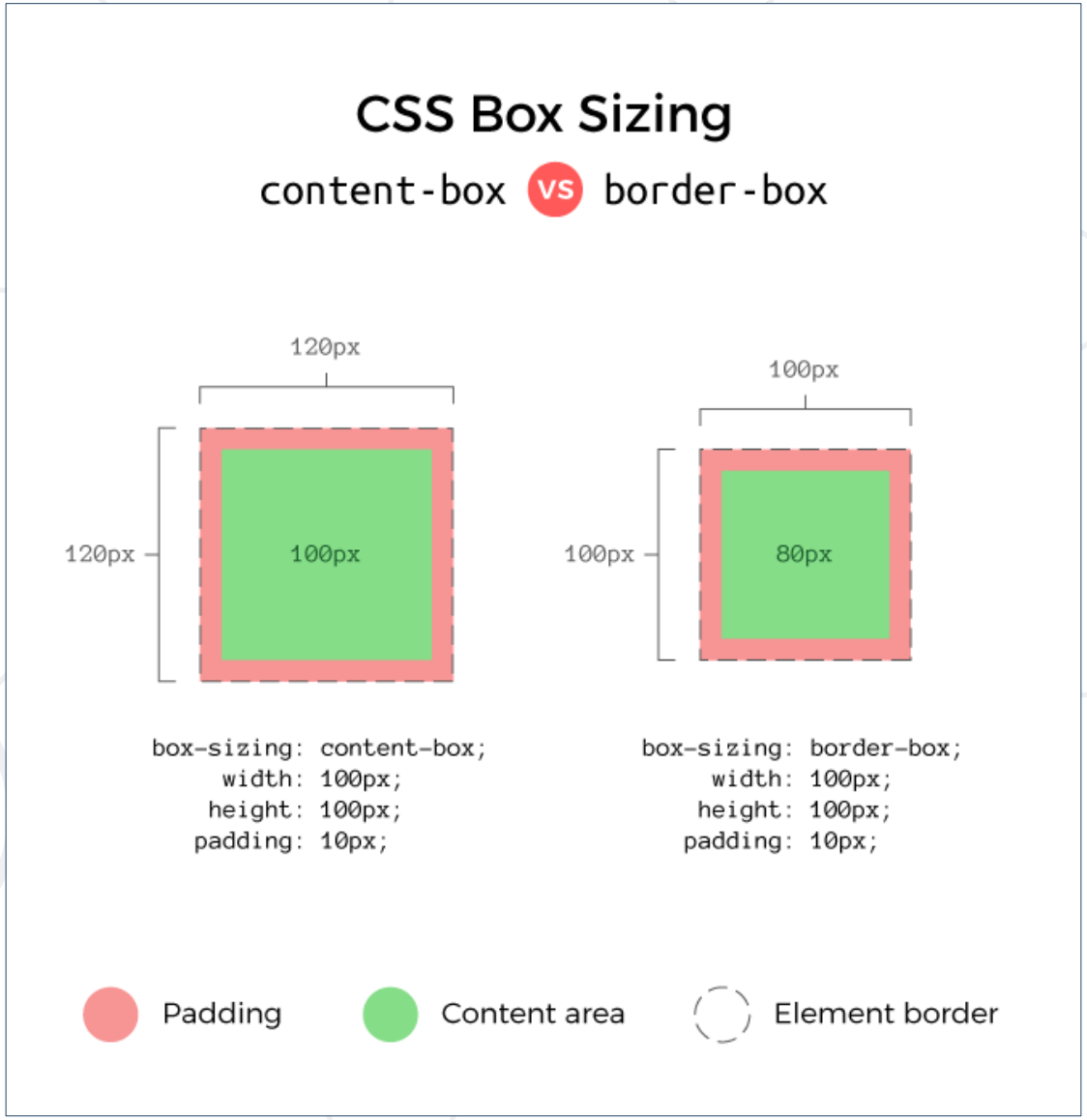
- **border-box** - the **width** and **height** of the element apply to all parts of the element: the **content**, the **padding** and the **borders**

```
box-sizing: border-box;
width: 200px;
border: 10px solid #5b6dcd;
padding: 5px;
```



- The full width is **200px**
- The content width is equal to: **200px - 2*10px - 2*5px = 170px**

Content-box vs Border-box



Universal Box-sizing

- The box-sizing **Reset** takes care of the box-sizing of every element by setting it to border-box using universal CSS selector
- Save your **time** and don't write the same thing **again-and-again**
- Set the "**universal box-sizing**" with inheritance:

```
html {
  box-sizing: border-box;
}

*,
*:before,
*:after {
  box-sizing: inherit;
}
```

Summary

- What is Box Model?
- Width and Height to the elements
- What are the padding, border and margin?
- What is box-sizing?
- How to reset box-sizing?

Questions?

SoftUni

Software University | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids | Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

createX

Postbank
Решения за твоето утре

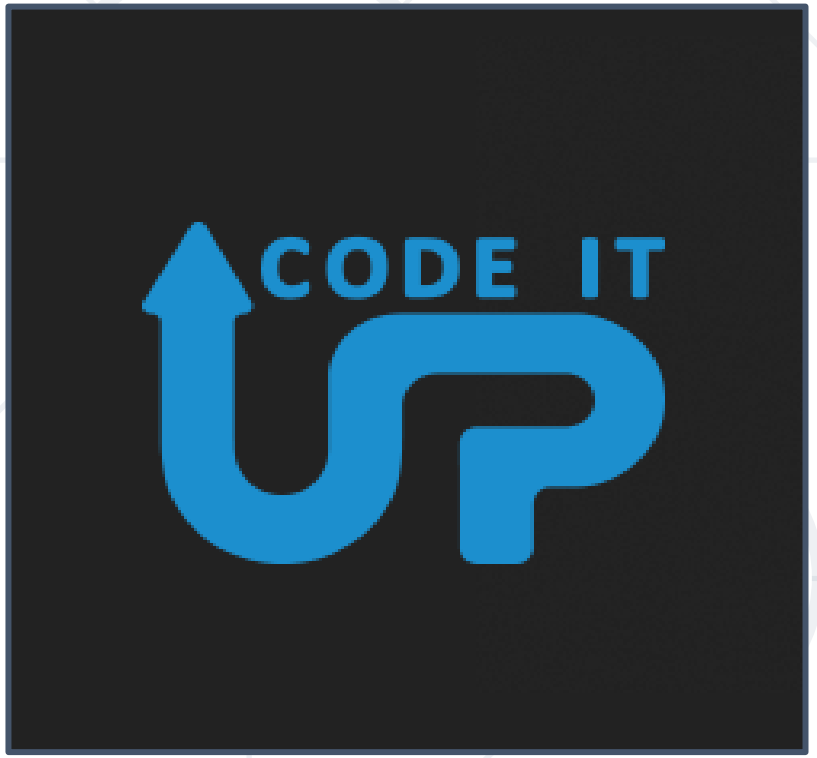
BOSCH

DXC TECHNOLOGY

SmartIT

Educational Partners

Software University



License

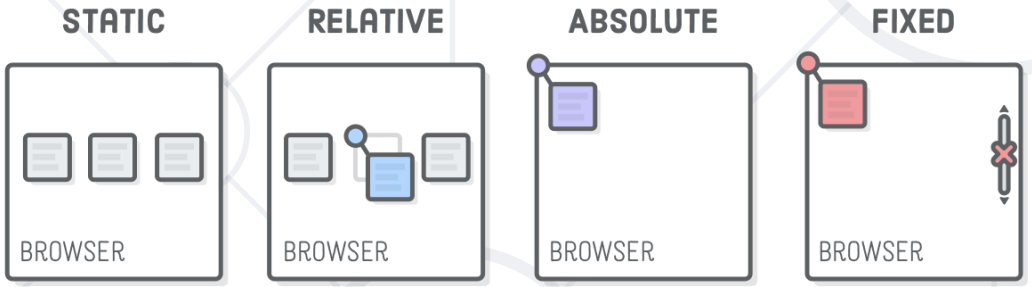
- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

Position & Grid

CSS Positioning



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.org

Table of Contents

Have a Question?

Software University

sli.do

#html-css

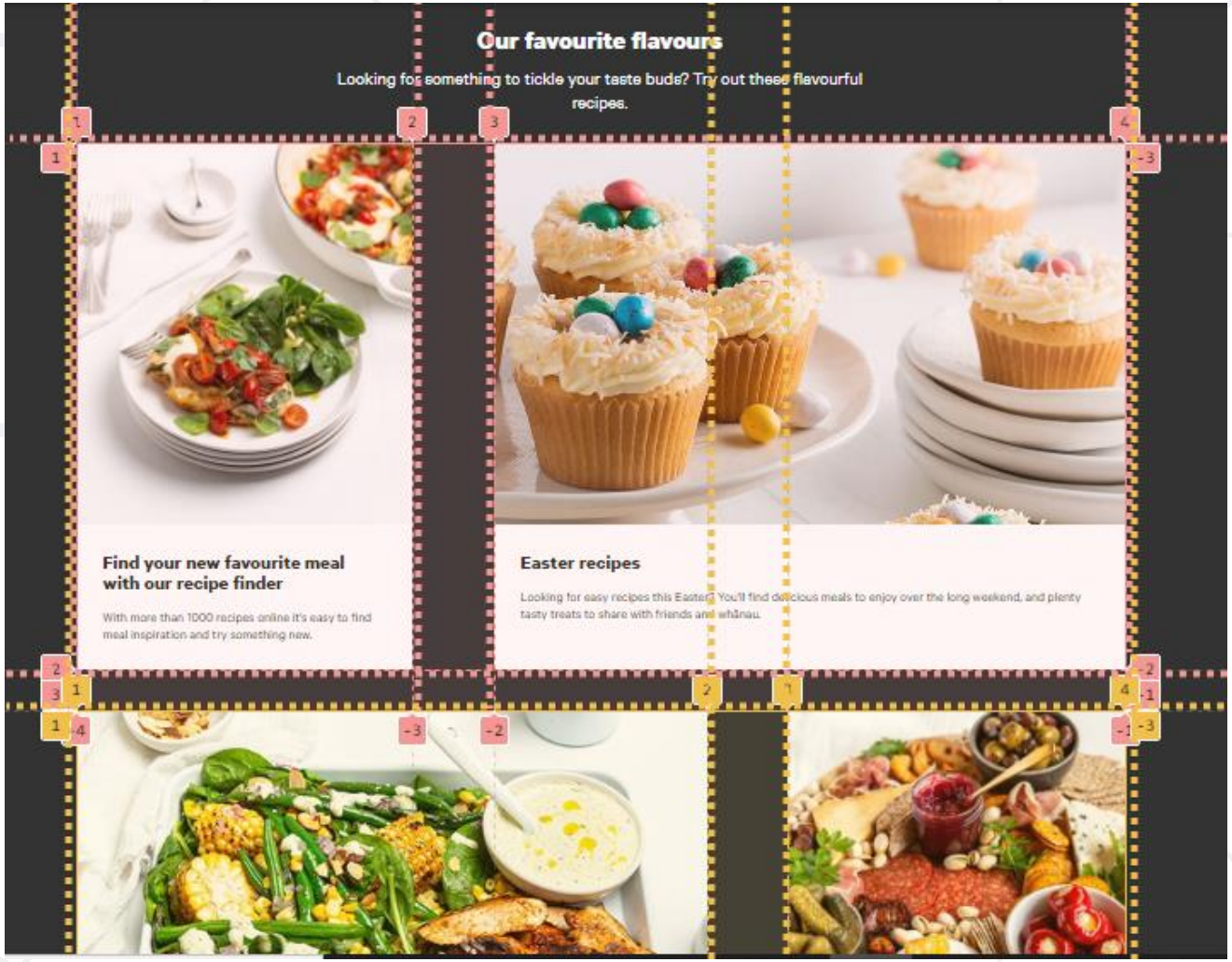
CSS Grid

Modern Layout System for the Web

CSS Grid

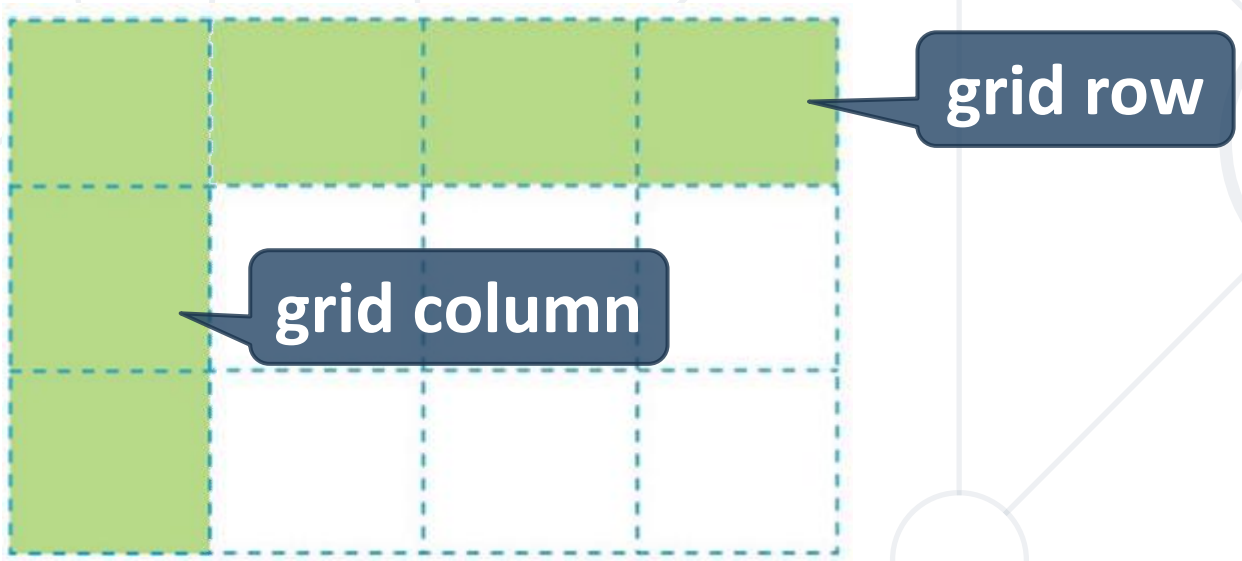
- **CSS grid** is modern CSS layout system for the Web
  - **Simple** and easy to use
  - Very **powerful**

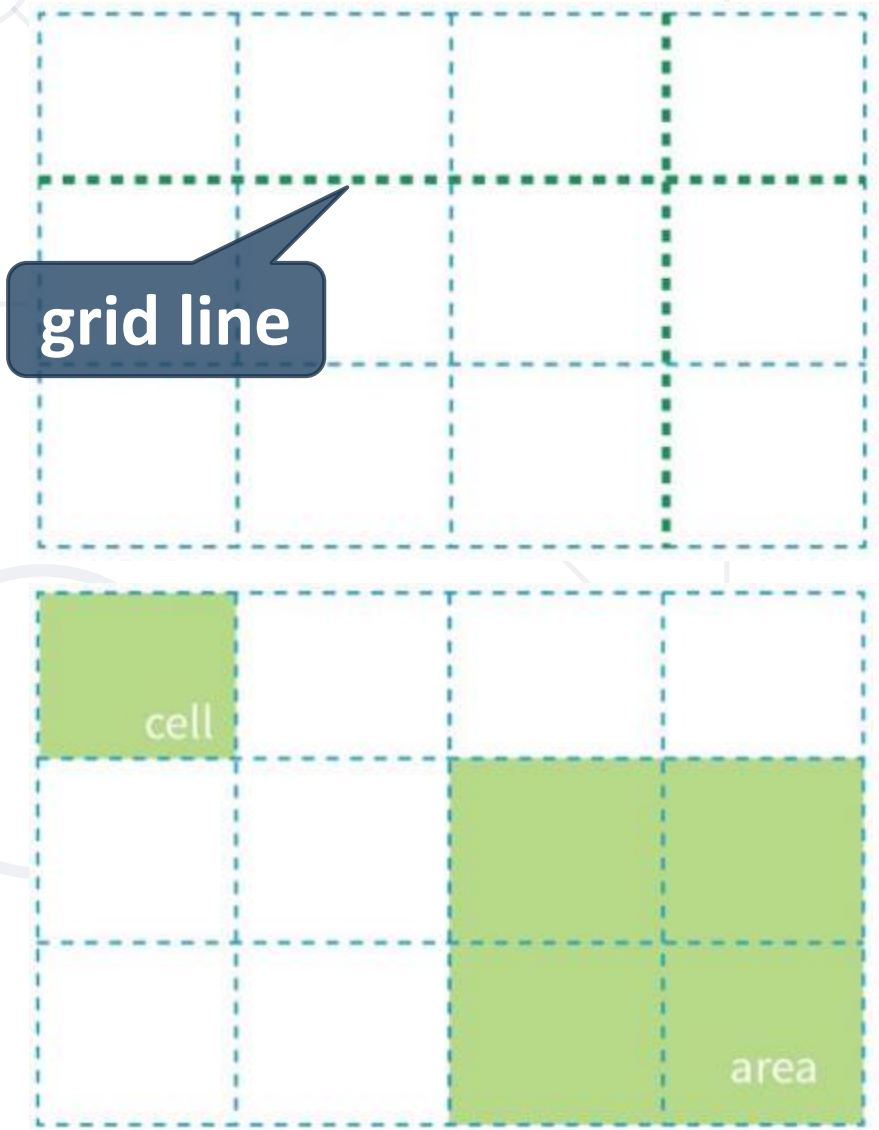
```
.grid-container {
    display: grid;
}
```



Grid Parts

- The primary parts of a grid are:
  - Grid **lines**
  - Grid **cell** and grid **areas**
  - Grid **tracks** (rows or columns)



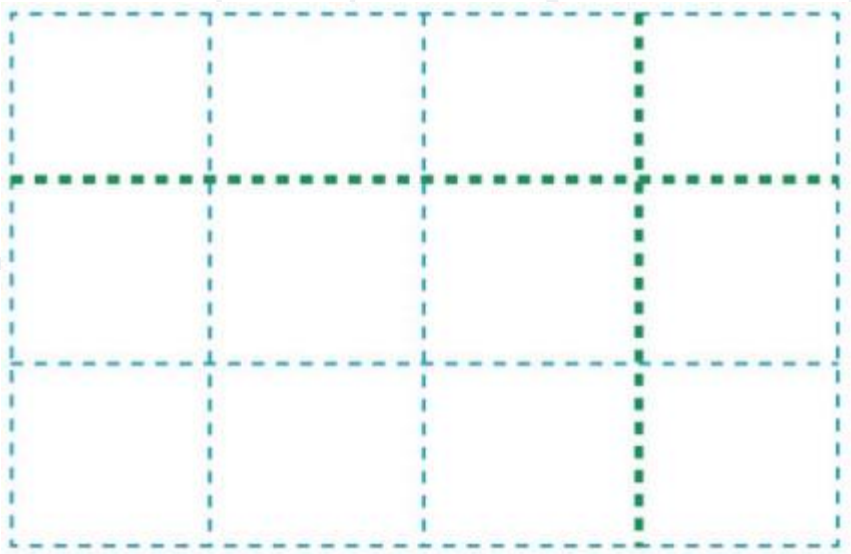


Grid Container

- An HTML element becomes a **grid container** when its display property is set to **grid** or **inline-grid**

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

```
.grid-container {
    display: inline-grid;
}
```

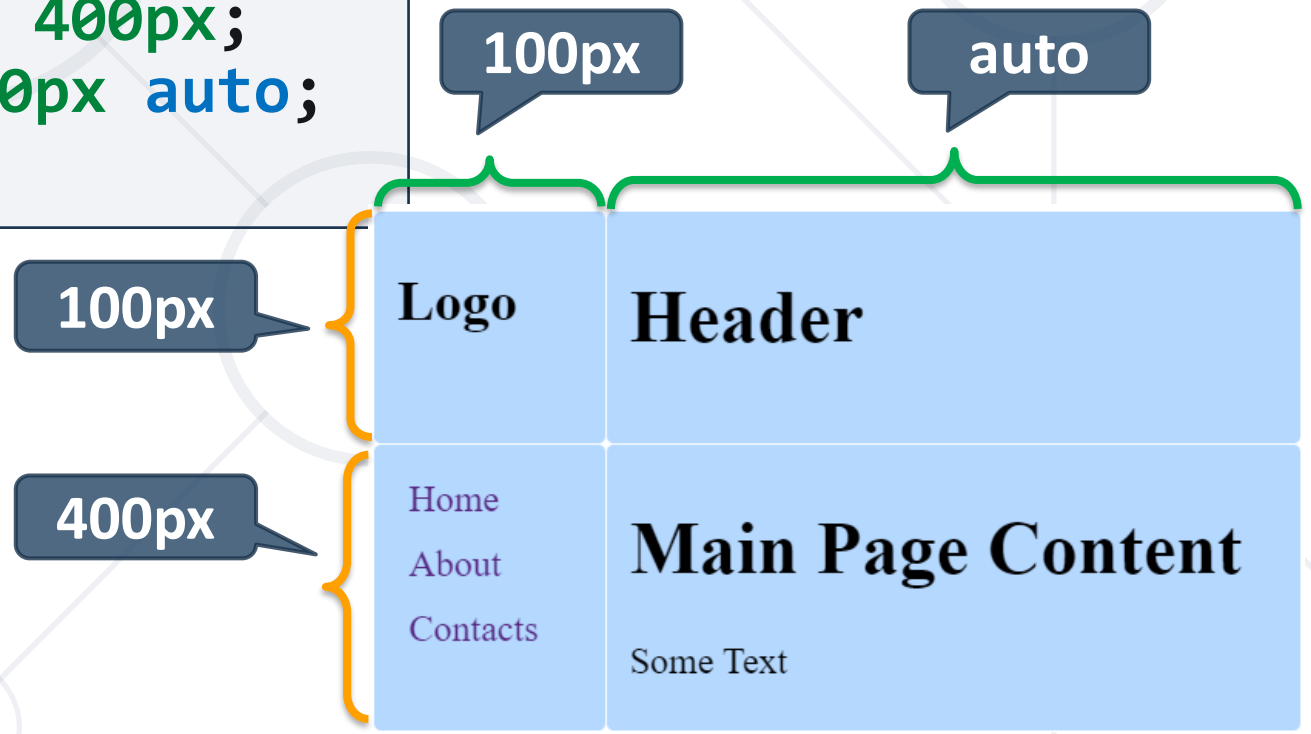


Grid Template

- The grid templates define the **look** of the grid: **count** of the **rows** and **columns** and their **size**

```
body {
   display: grid;
   grid-template-rows: 100px 400px;
   grid-template-columns: 100px auto;
}
```

- This example will create:
  - Two **rows** with **sizes**: **100px** and **400px**
  - Two **columns** with **sizes**: **100px** and **auto** size



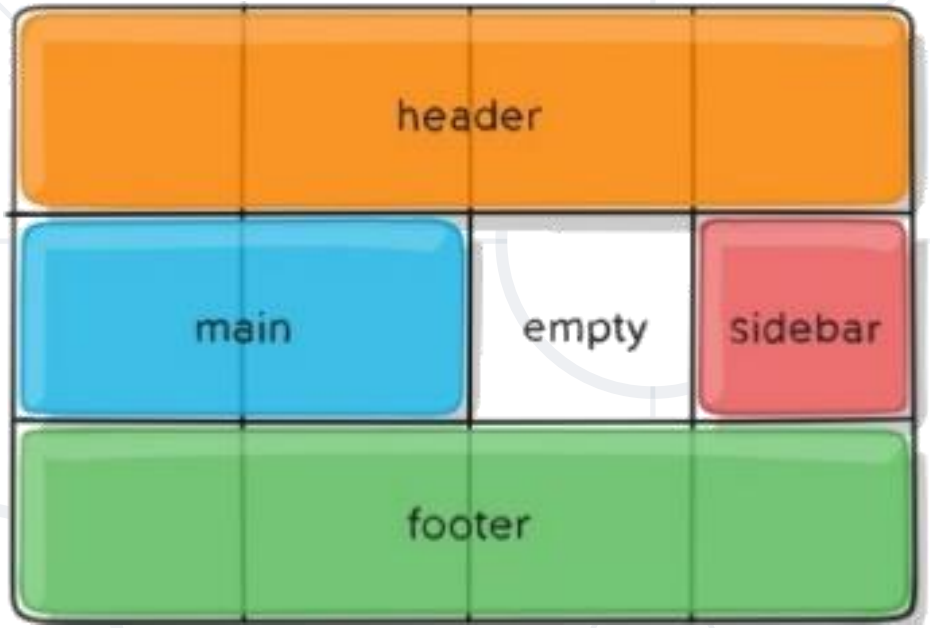
Grid Area and Grid Template Area

- **Grid area** is a rectangular area made up of one or more **adjacent grid cells**
- **Defining** grid areas in CSS:

```
header { grid-area: header; }
```

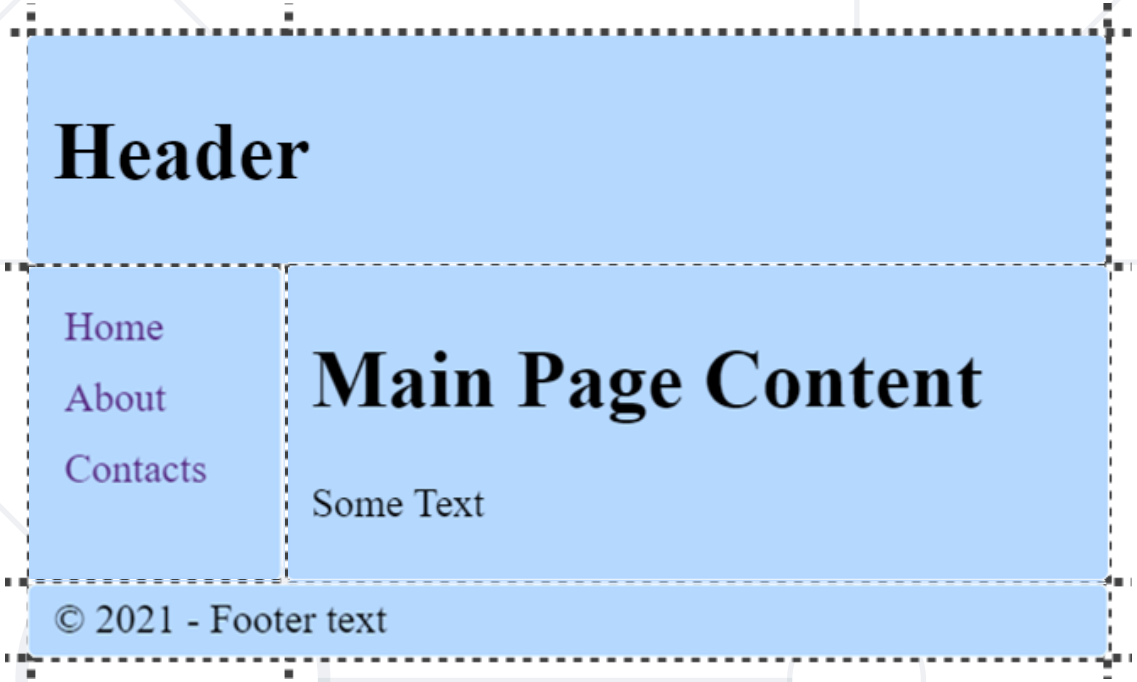
- **Referencing** the grid areas:

```
body { grid-template-areas:
 "header header header header"
 "main main empty sidebar"
 "footer footer footer footer";
```



Grid Area – Examples

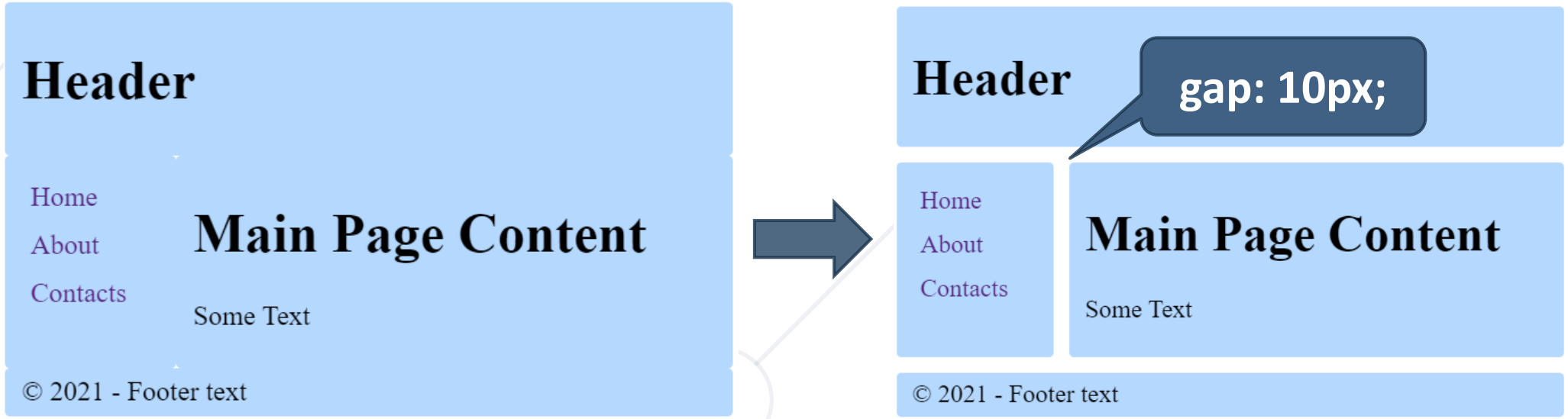
```
body { display: grid;
  grid-template-areas:
    "header header"
    "aside main"
    "footer footer";
  grid-template-columns:
    100px auto;
}

header { grid-area: header; }
aside { grid-area: aside; }
main { grid-area: main; }
footer { grid-area: footer; }
```



Gap

- **Gap** between each cell horizontally and vertically

```
body {
  display: grid;
  grid-template-columns:
    100px auto;
  gap: 10px;
  grid-template-areas:
    "header header"
    "aside main"
    "footer footer";
}
```

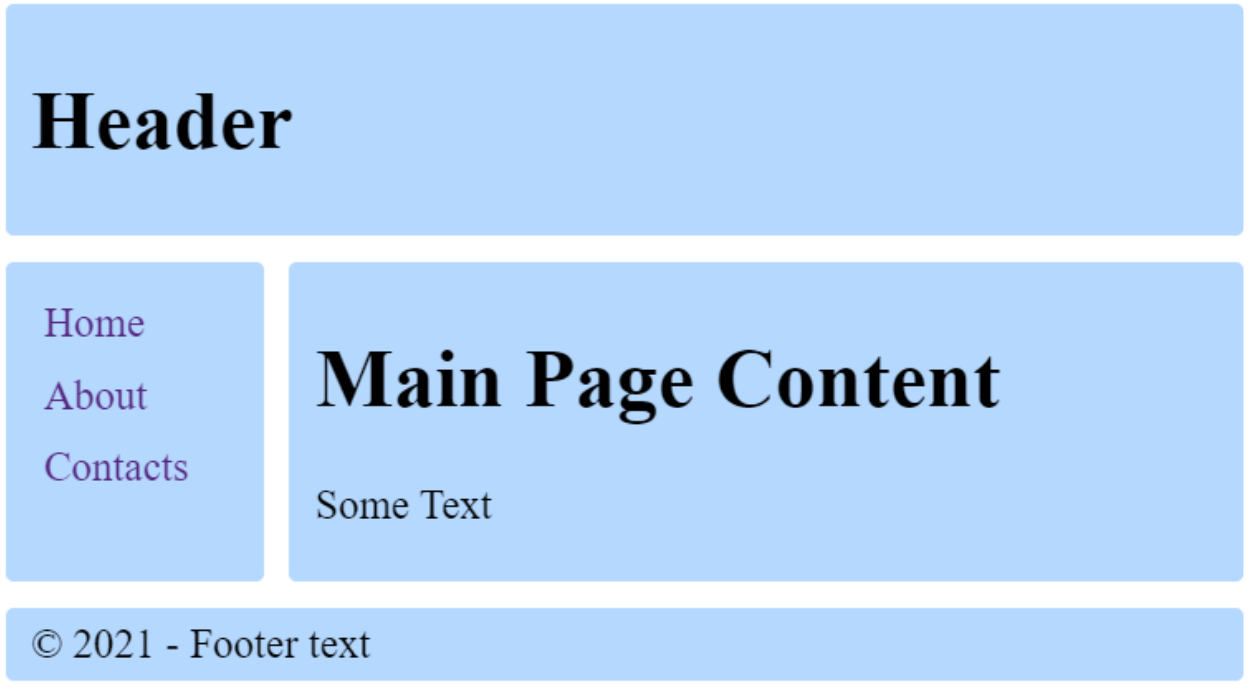


CSS Grid – Example

```
<body>
  <header><h1>Header</h1></header>
  <aside>
    <ul>
      <li>↔</li>
      <li>↔</li>
      <li>↔</li>
    </ul>
  </aside>
  <main>
    <h1>Main Page Content</h1>
    <p> Some Text
  </main>
  <footer>
    <div>© 2021 - Footer text</div>
  </footer>
</body>
```

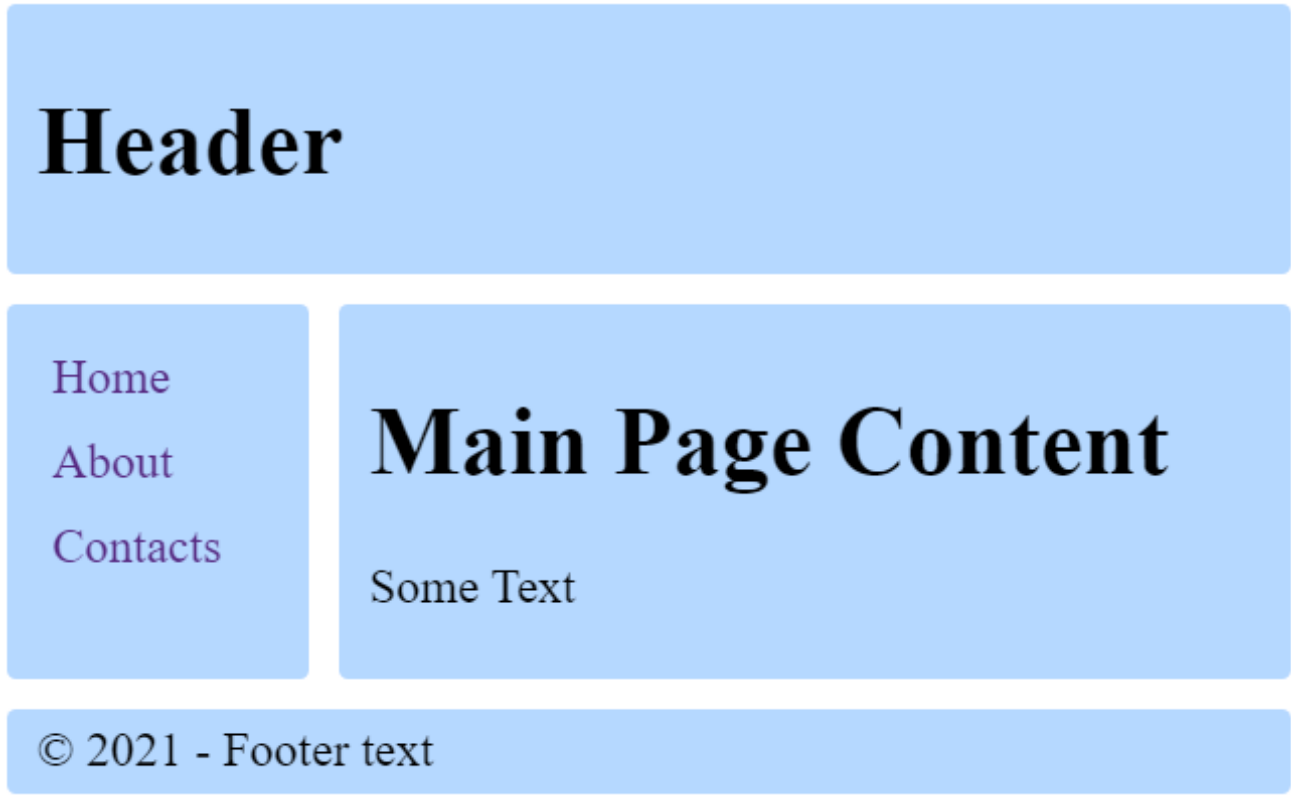
CSS Grid – Example

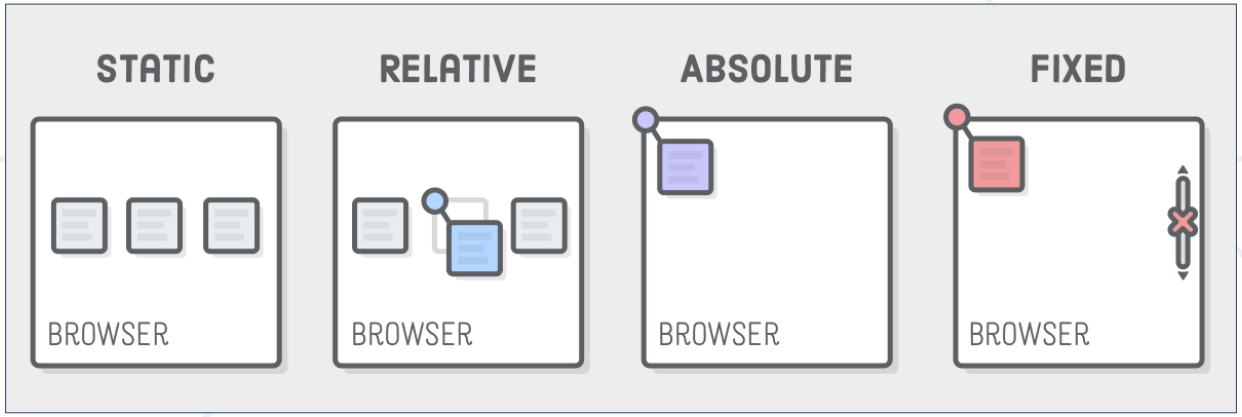
```css
body {
  display: grid;
  grid-template-areas:
    "header header"
    "aside main"
    "footer footer";
  grid-template-columns: 100px auto;
  gap: 10px;
}

header { grid-area: header; }
aside { grid-area: aside; }
footer { grid-area: footer; }
```



Demo: CSS Grid Site Layout

- https://codepen.io/snakov/pen/jOVJXVN

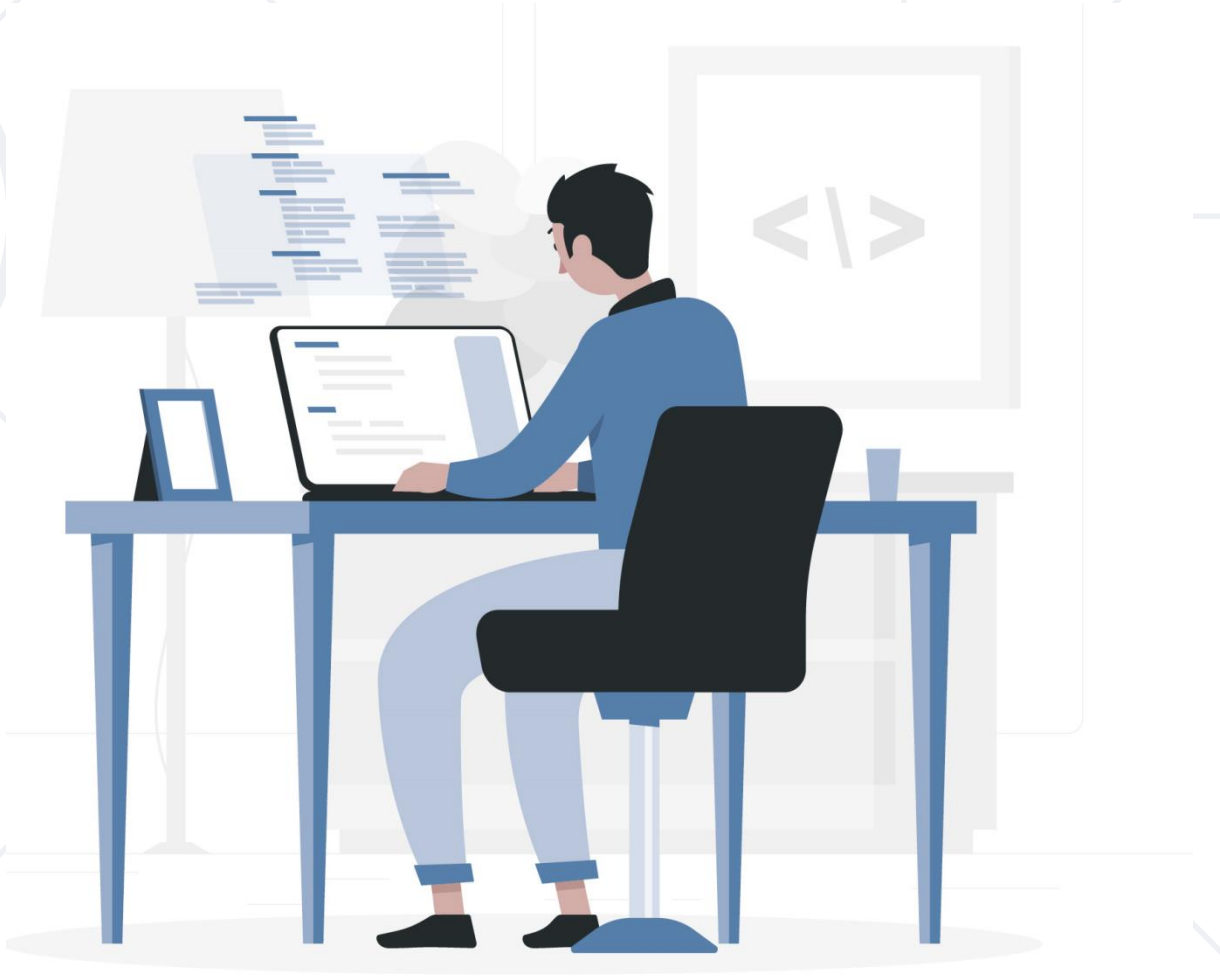




Specifies the Type of Positioning Method Used for an Element

Position

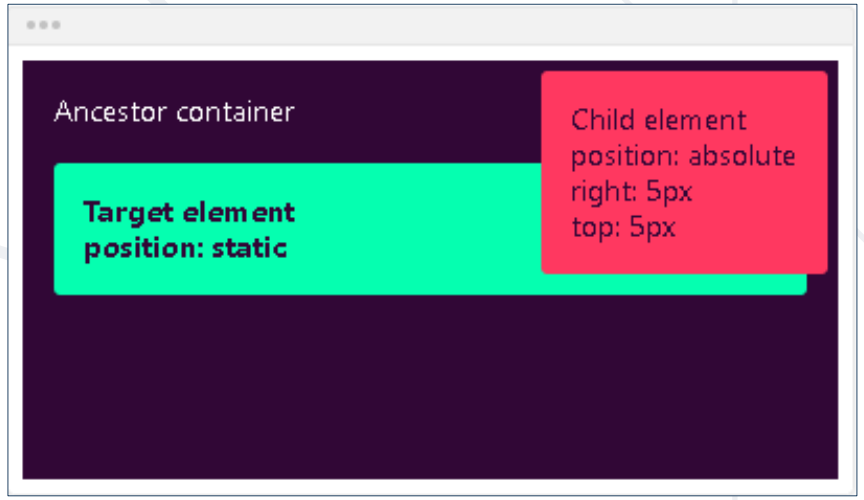
- Position properties:
  - `static`
  - `relative`
  - `absolute`
  - `fixed`
  - `sticky`



Position Static

- Static - the **default** state of every element
  - Puts the element into its **normal position** in the document layout flow
- It will **NOT** react to the following properties: **`top`**, **`bottom`**, **`left`**, **`right`**, **`z-index`**

```
div {
  position: static;
}
```

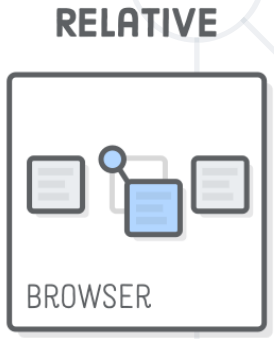


Position Relative

- It looks like static positioning, but once the positioned element has taken its place, you can then modify its final position with the **positional properties**

```
<img src="cup.jpg">
<img class="new" src="new.png">
```

```
img.new {
  position: relative;
  top: -200px;
  right: 150px;
}
```



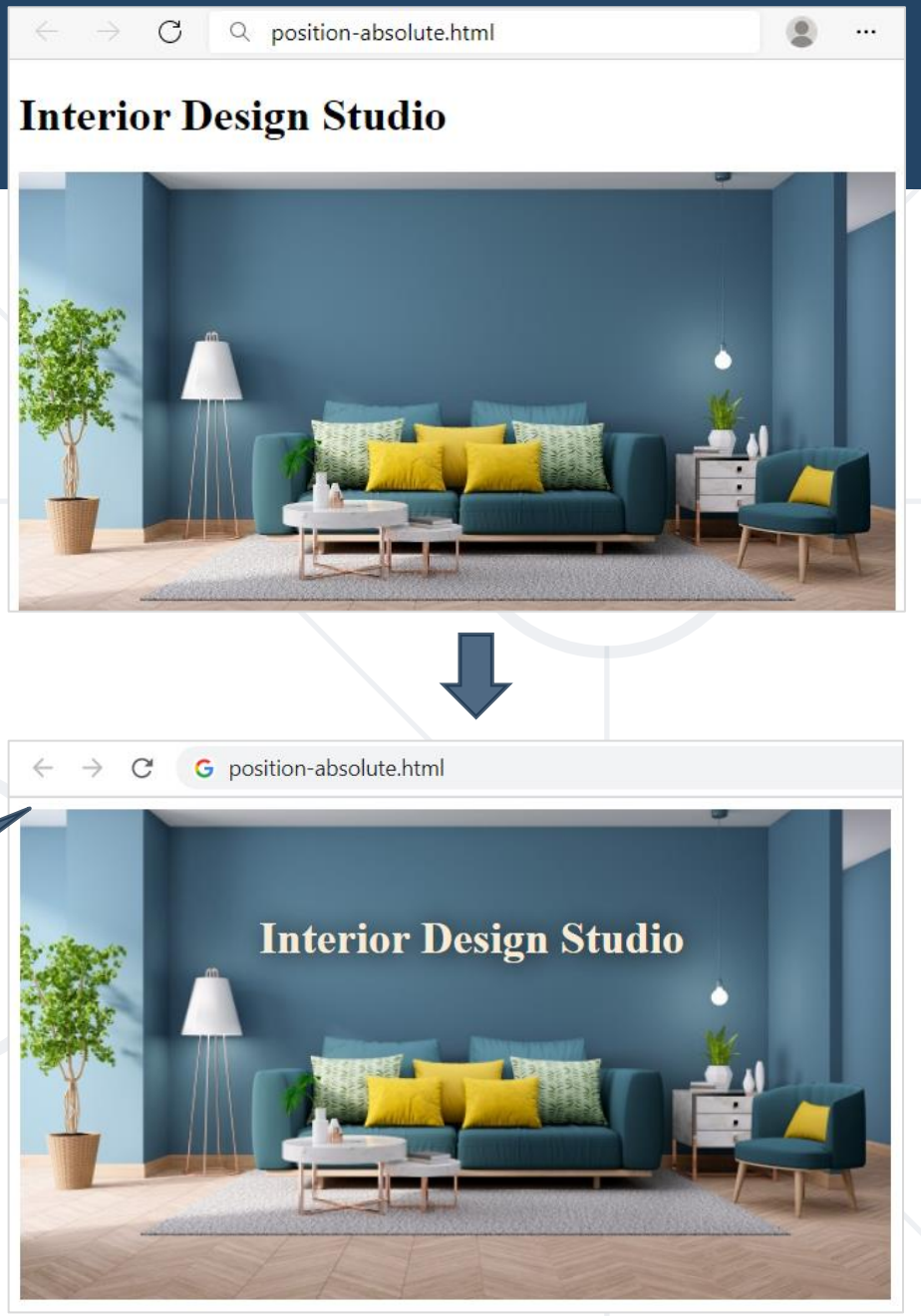
Position Absolute

- Absolute positioning → from the **upper left corner** of the parent

```
<h1>Interior Design Studio</h1>
<img src="livingroom.jpg">
```

```
h1 {
  position: absolute;
  top: 60px;
  left: 180px;
  color: antiquewhite;
  text-shadow: 1px 1px 20px black;
}
```

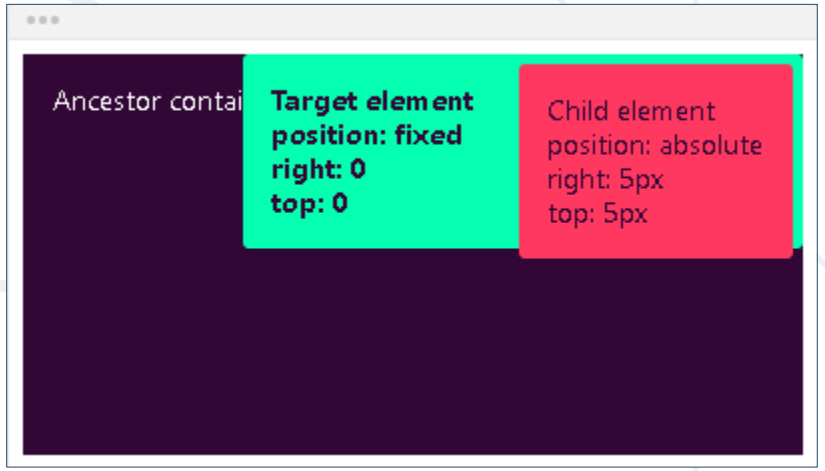
<h1> frees its original place



Position Fixed

- **Fixed** - the element will **NOT remain** in the natural flow of the page
  - **Reacts** to the positional properties
- Positions itself according to the **viewport**

```
div {
  position: fixed;
}
```



Position Sticky

- The element is positioned based on the user's **scroll position**
- A sticky element switches 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`)

Position Sticky – Example

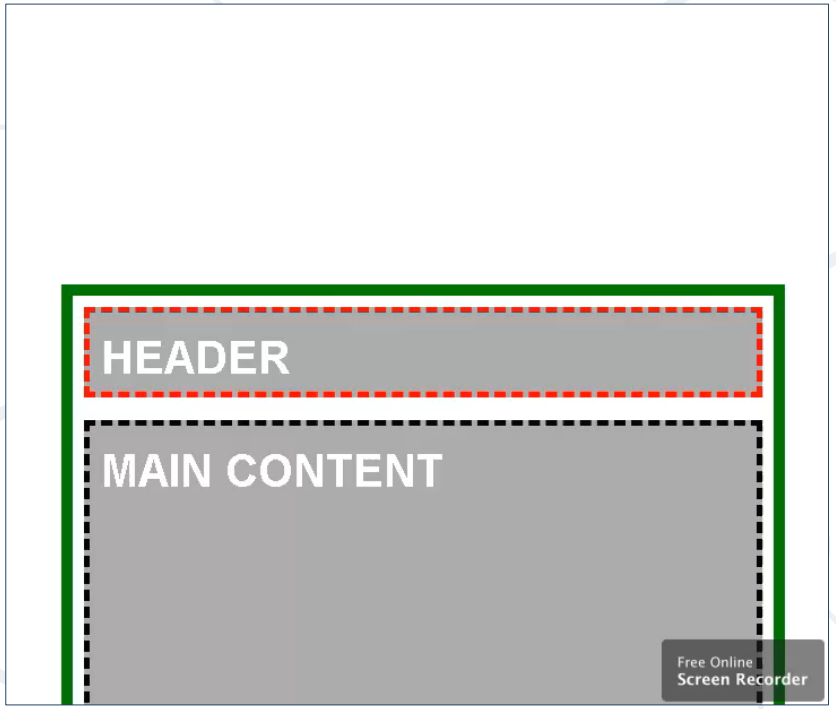
Software University

```
<main class="main-container">
    <header class="main-header">
      HEADER
    </header>
    <div class="main-content">
       MAIN CONTENT
    </div>
    <footer class="main-footer">
      FOOTER
    </footer>
</main>
```

```
.main-container {
  max-width: 600px;
  margin: 0 auto;
  border: 10px solid green;
  padding:10px;
  margin-top:40px;
}
.main-header,
.main-content,
.main-footer {
  padding: 10px;
  background: #aaa;
  border: 5px dashed #000;
  margin: 20px 0;
}
```

Position Sticky – Example

```
.main-content {
  min-height: 1000px;
}

.main-header {
  height: 50px;
  border-color: red;
  position: sticky;
  top: 0;
}
```

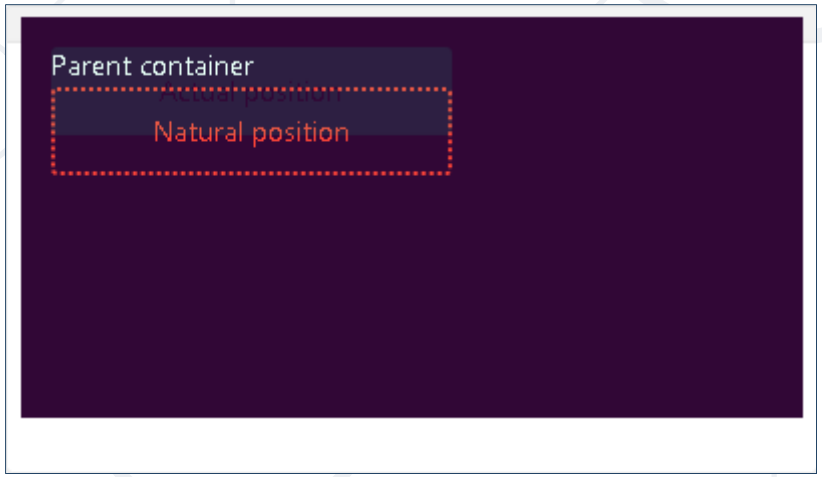


Bottom

- **Bottom** - defines the position of the element according to its bottom edge
  - **`bottom: auto;`** - the element will remain in its **natural** position

Bottom

- If the element is in position **relative**, the element will move **upwards** by the amount defined by the **bottom** value
  - `bottom: 20px;`

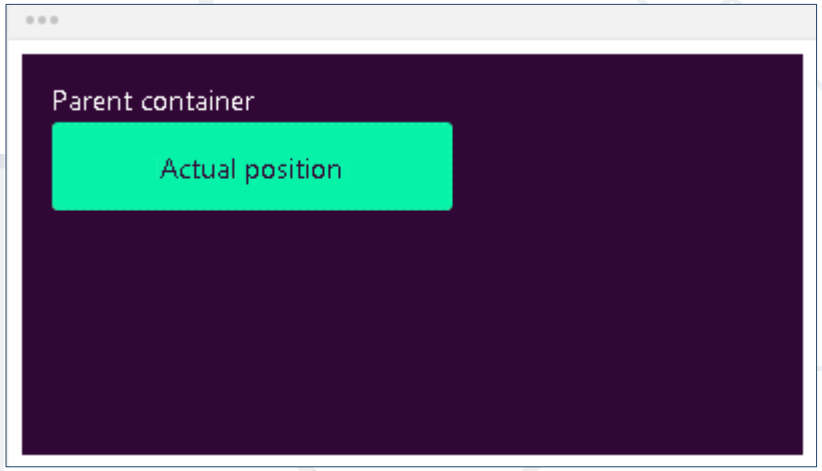


Bottom

- If the element is in position `absolute`, the element will position itself from the `bottom` of the first positioned `ancestor`
  - `bottom: 0;`

Left

- Defines the position of the element according to its left edge
  - `left: auto;` - the element will remain in its **natural** position



Left

- `left: 80px;` - if the element is in position **relative**, the element will move **left** by the amount defined by the left value

Left

- `left: -20px;` - if the element is in position **absolute**, the element will position itself from the **left** of the first positioned **ancestor**

Right

- **Right** - defines the position of the element according to its right edge
  - `right: auto;` - the element will remain in its **natural** position
  - `right: 80px;` - if the element is in position **relative**, the element will move **right** by the amount defined by the right value
  - `right: -20px;` - if the element is in position **absolute**, the element will position itself from the **right** of the first positioned **ancestor**

Top

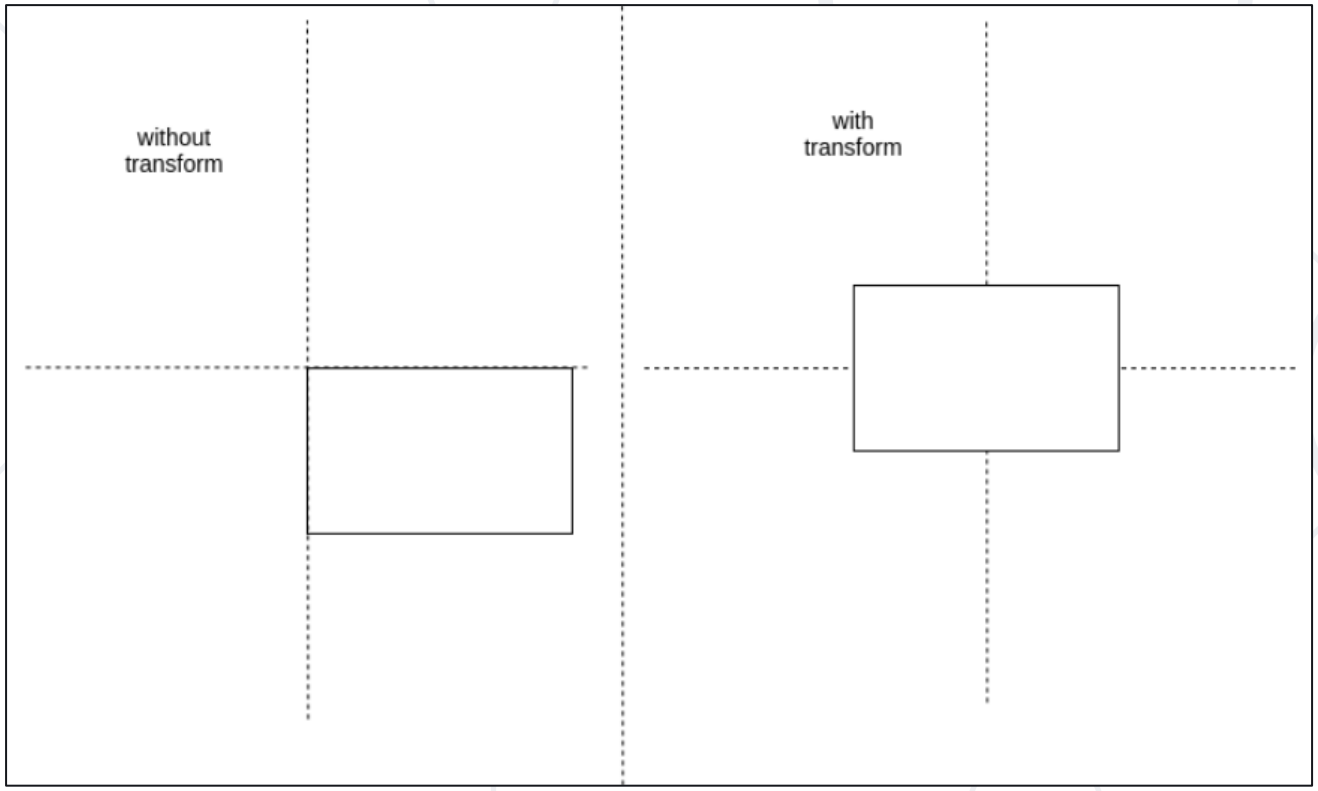
- **Top** - defines the position of the element according to its top edge
  - `top: auto;` - the element will remain in its **natural** position
  - `top: 20px;` - if the element is in position **relative**, the element will move **downwards** by the amount defined by the **top** value
  - `top: 0;` - if the element is in position **absolute**, the element will position itself from the **top** of the first positioned **ancestor**

Center

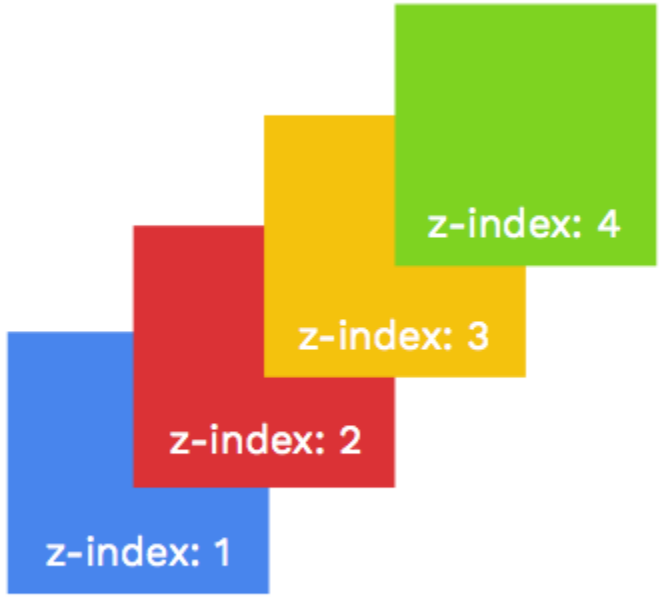
- **Center** - defines the position of the element according to center of the window
  - `position: absolute;`- it will position the element relative to its first positioned parent element
    - If it can't find one, it will be relative to the document
  - `top: 50%; left: 50%;` - the element will step out from the top left corner
  - These properties are set on the child element

Center

- `transform: translate(-50%, -50%);` - it will pull back the item with its half width and height



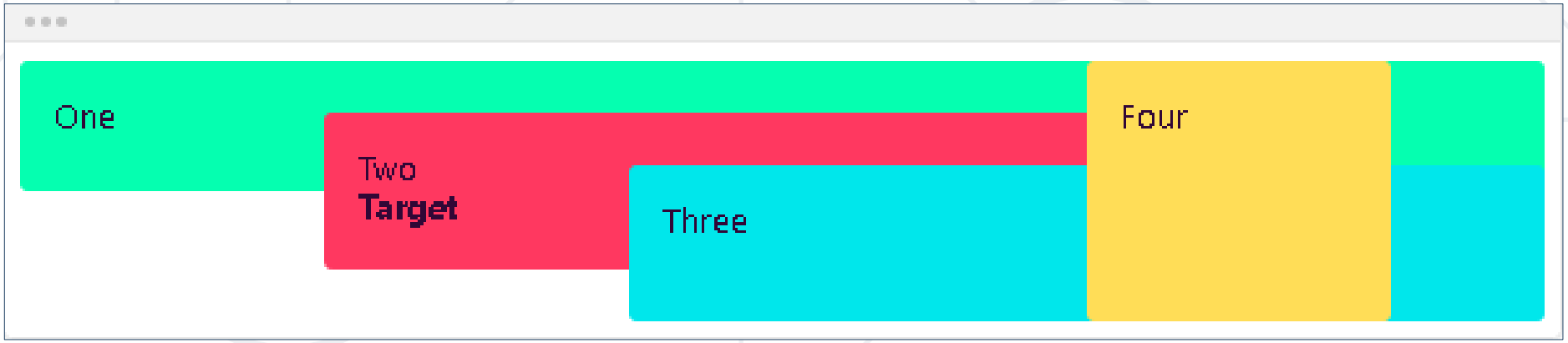

```
.parent {
position: relative;
}
.child {
position: absolute;
top: 50%;
left: 50%;
transform: translate(
-50%, -50%);
}
```



Specifies the Stack order of an Element

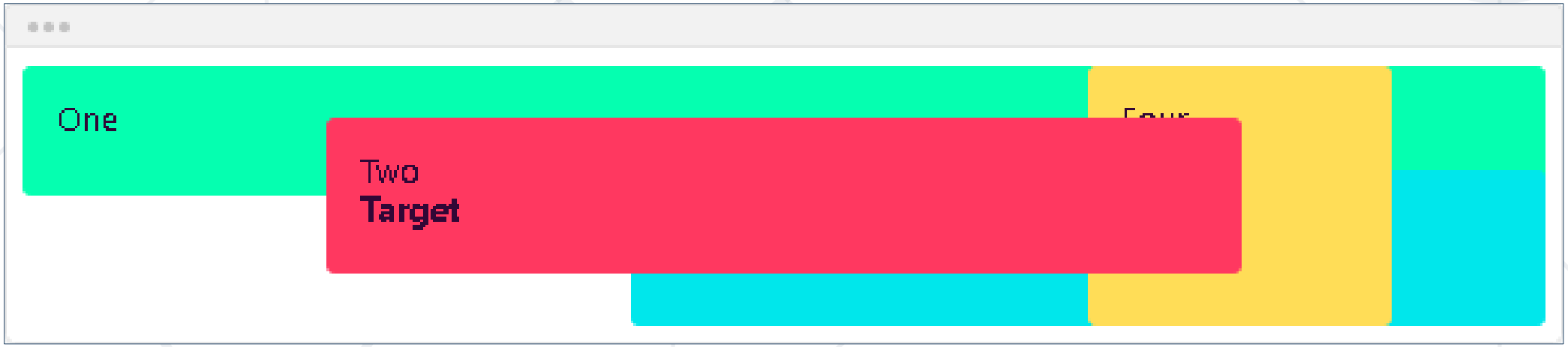
Z-index

- Defines the order of the elements on the **z-axis**. It only works on positioned elements (**anything apart from static**)
  - Default value: `z-index: auto;`
  - The order is defined by the order in the HTML code:



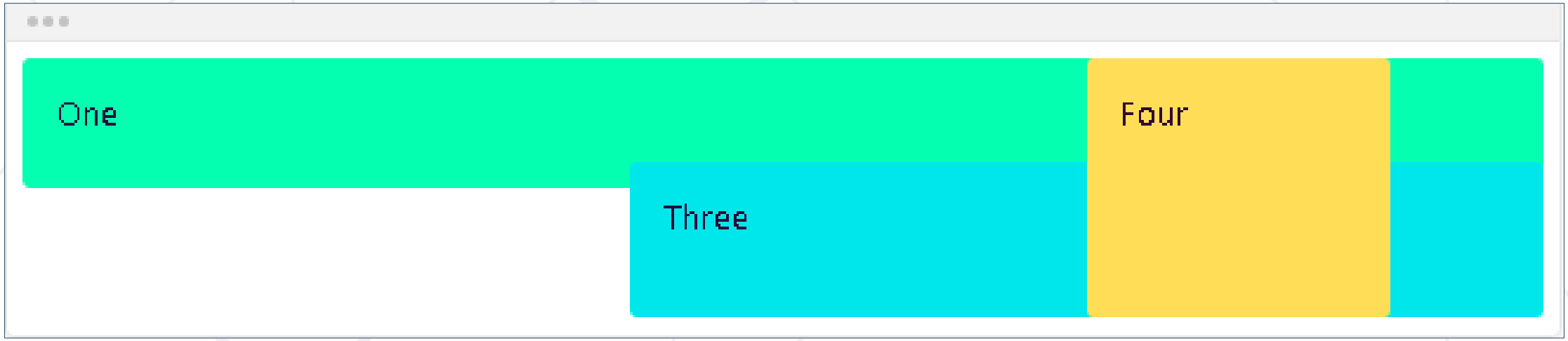
Z-index

- The z-index value is relative to the other ones
- The target element is move in **front** of its siblings
  - `z-index: 1;`



Z-index

- You can use **negative** values
- The target element is moved **behind** its **siblings**
  - `z-index: -1;`



Summary

- CSS grid
  - Lines, cell, areas, tracks
  - grid-area and grid-template-areas
- Positioning properties
- Z-Index

Questions?

SoftUni

Software University | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids | Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

createX

Postbank
Решения за твоето утре

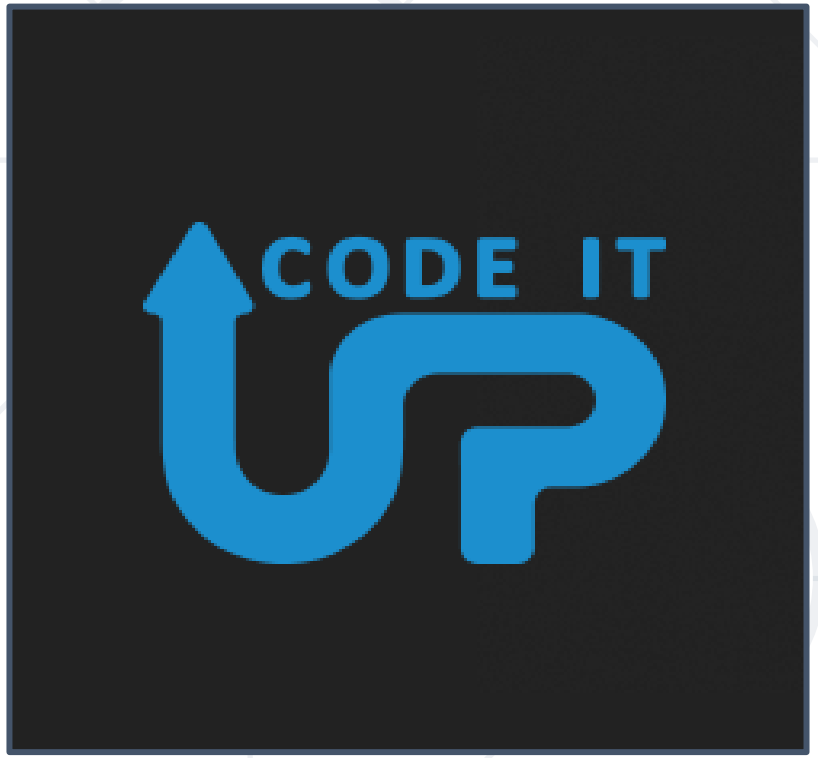
BOSCH

DXC TECHNOLOGY

SmartIT

Educational Partners

Software University



License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

Flexbox

CSS Flexbox Layout Module

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

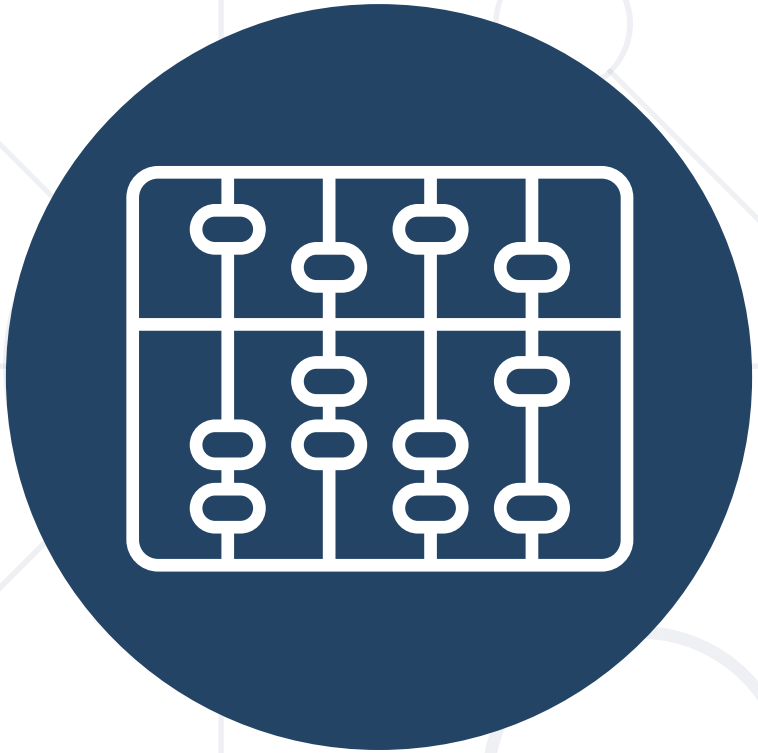
Table of Contents

Have a Question?

Software University

sli.do

#html-css



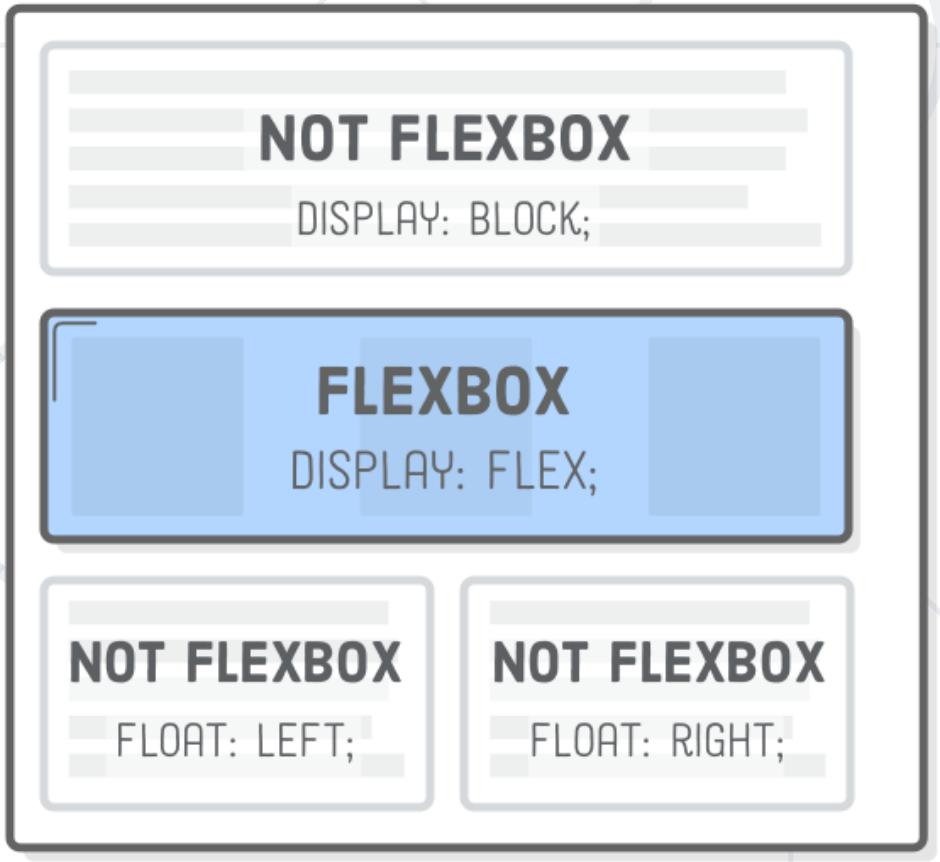
CSS Flexbox Layout Module

What is Flexbox?

- Offers **space distribution** between items in an interface and powerful **alignment** capabilities
- Flexbox is a method for laying out items in **rows** or **columns**
- Items flex to **fill** additional space and **shrink** to fit into smaller spaces

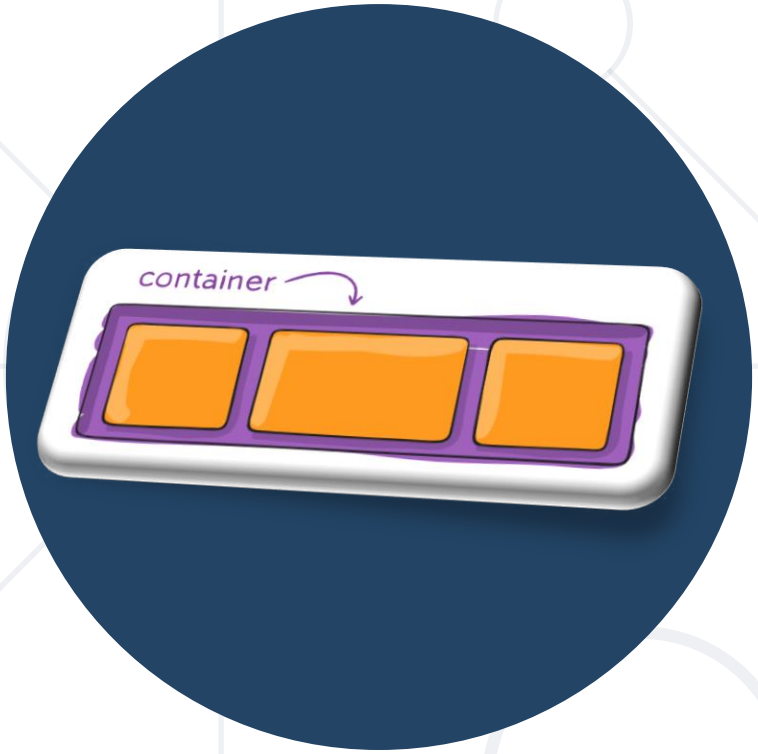
Why Flexbox?

- For a long time, the only reliable cross browser-compatible tools available for creating CSS layouts were **floats** and **positioning**
- These are fine and they work, but in some ways, they are also rather limiting and frustrating



Why Flexbox?

- The following simple layout requirements are either **difficult** or **impossible** to achieve with such tools:
  - **Vertically centering** a block of content inside its parent
  - Making all the children of a container take up an **equal** amount of the available **width/height**
  - Making all columns in a multiple column layout adopt the **same height** even if they contain a different amount of content



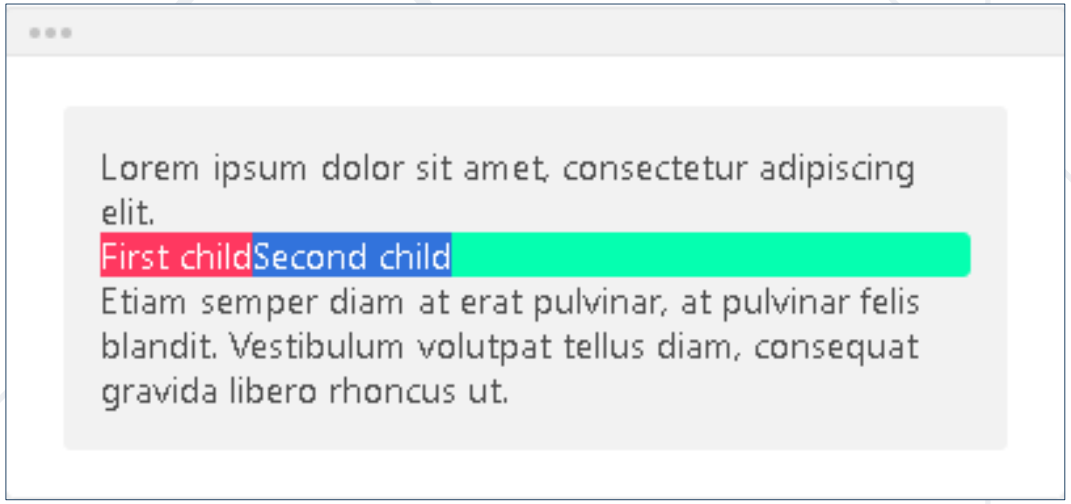
Expands Items to Fill Available Free Space or Shrinks Them to Prevent Overflow

Display – Flex

- The element is turned into a **flexbox** container
- Its child elements will be turned into **flexbox items**

```
<body>
    <p>Lorem ipsum dolor sit amet, con
sectetur adipiscing elit.</p>
    <div class="container">
        <p>First child</p>
        <p>Second child</p>
    </div>
    <p>
        Etiam semper diam at erat pulv
inar, at pulvinar felis blandit. Vesti
bulum volutpat tellus diam,consequat g
ravida libero rhoncus ut.
    </p>
</body>
```

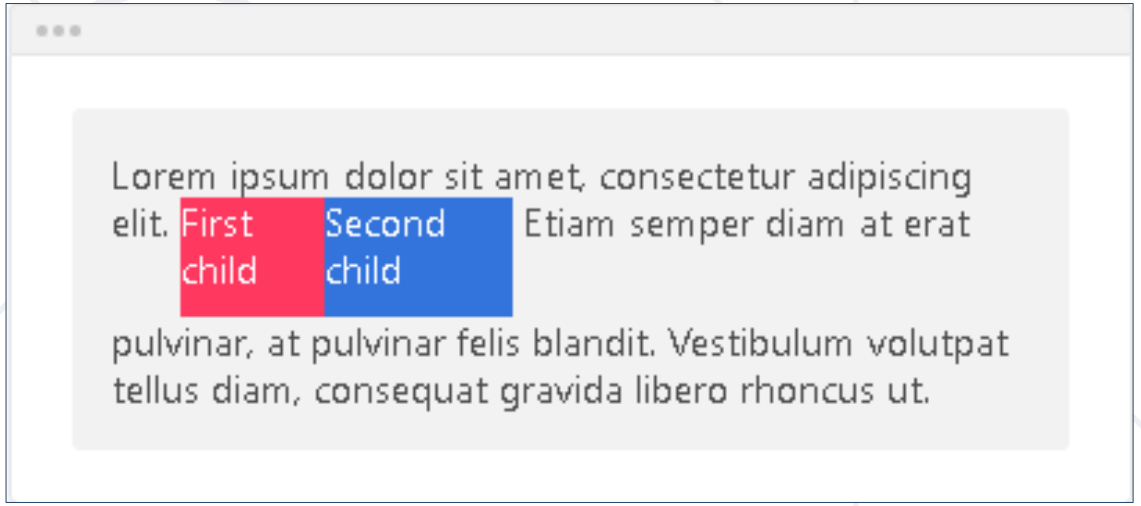
```
.container {
  display: flex;
}
```



Display – Inline-flex

- The element shares properties of both an **inline** and a **flexbox** element:
  - **inline** because the element behaves like simple text, and inserts itself in a block of text
  - **flexbox** because its child element will be turned into flexbox items

```
.container {
  display: inline-flex;
  height: 3em;
  width: 120px;
}
```



Flex Direction

- Defines how flexbox items are ordered within a flexbox container

```
flex-direction: row;
```

  - The flexbox items are ordered the **same way** as the text direction, along the main axis

Flex Direction

- The flexbox items are ordered the **opposite** way as the **text direction**, along the main axis

```
flex-direction: row-reverse;
```

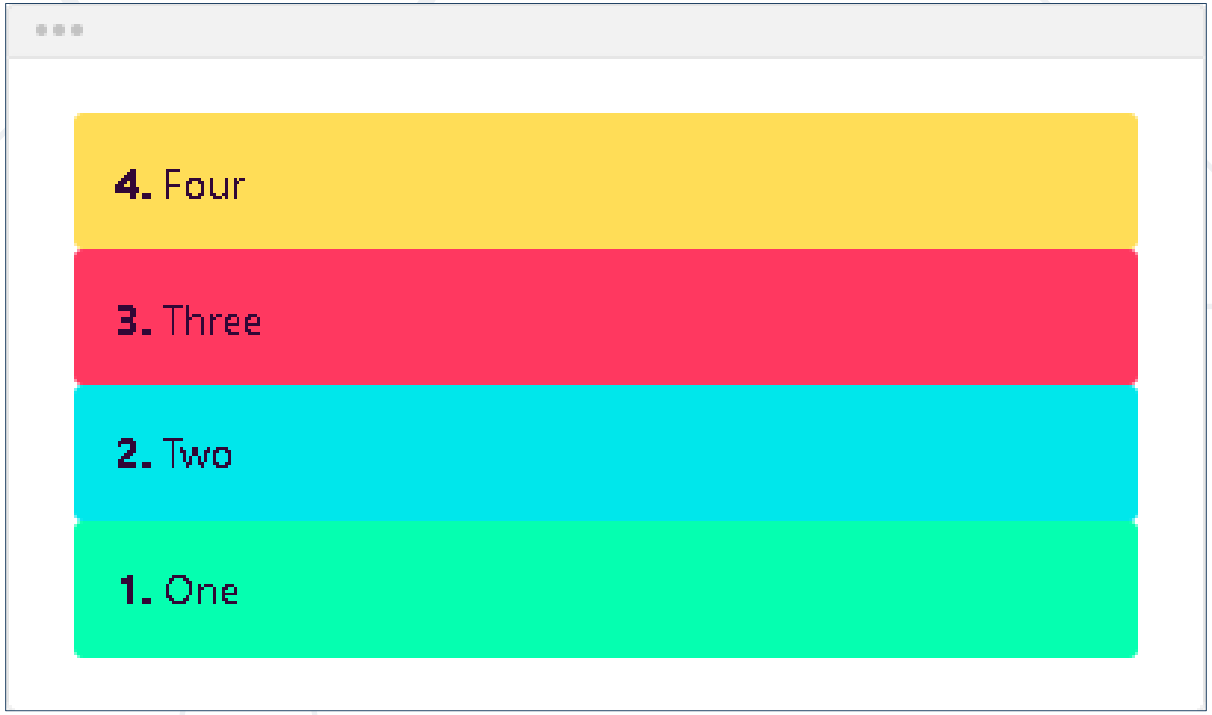
- The flexbox items are ordered the **same** way as the **text direction**, along the **cross axis**

```
flex-direction: column;
```

Flex Direction

- The flexbox items are ordered the **opposite** way as the **text direction**, along the **cross axis**

```
flex-direction: column-reverse;
```



Flex Wrap

- Defines if flexbox items appear on a **single line** or on **multiple lines** within a flexbox container

```
flex-wrap: nowrap;
```

- The flexbox items will remain on a **single line**, no matter what, and will eventually overflow if needed

Flex Wrap

- The flexbox items will be distributed among **multiple lines** if needed

```
flex-wrap: wrap;
```

- The flexbox items will be distributed among **multiple lines** if needed
  - Any additional line will appear **before** the previous one

```
flex-wrap: wrap-reverse;
```

Flex Flow

- `flex-flow` is a shorthand for the `flex-direction` and `flex-wrap` properties
- The default value is `row nowrap`

```
flex-flow: <flex-direction> || <flex-wrap>

.container {
      flex-flow: row wrap;
}
```

Justify Content

- Defines how **flexbox**/**grid** items are aligned according to the **main** axis, within a flexbox container

```
justify-content: flex-start;
```

  - The flexbox items are pushed towards the **start** of the container's main axis

Justify Content

- The flexbox items are pushed towards the **end** of the container's main axis

```
justify-content: flex-end;
```

- The flexbox items are **centered** along the container's main axis

```
justify-content: center;
```

Justify Content

- The remaining space is distributed **between** the flexbox items

```
justify-content: space-between;
```

- The remaining space is distributed **around** the flexbox items: this adds space **before** the first item and **after** the last one

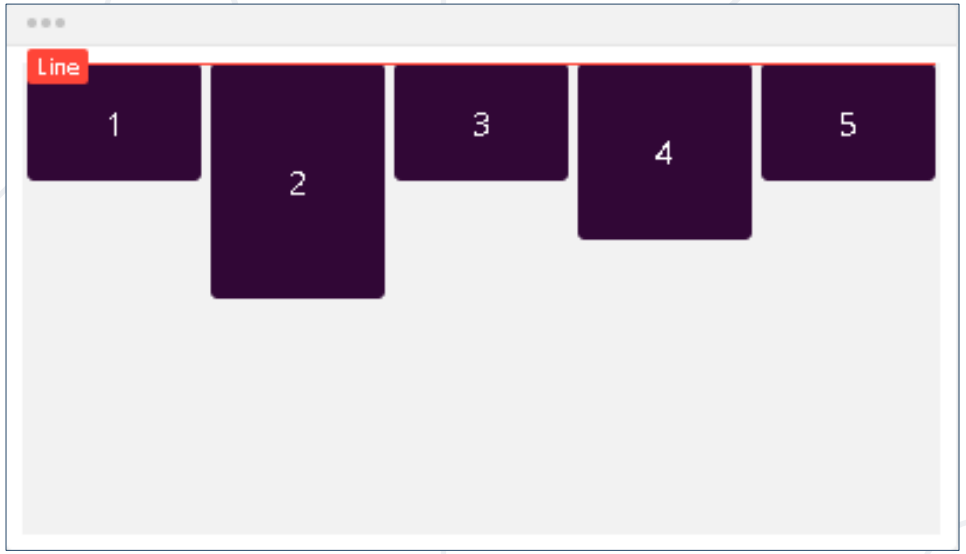
```
justify-content: space-around;
```

Align Items

- Defines how flexbox items are aligned according to the **cross** axis, within a line of a flexbox container

```
align-items: flex-start;
```

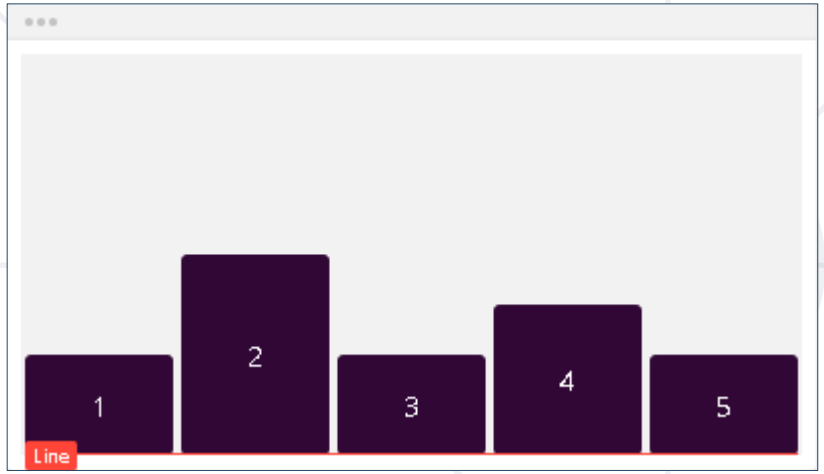
  - The flexbox items are aligned at the **start** of the **cross axis**



Align-items

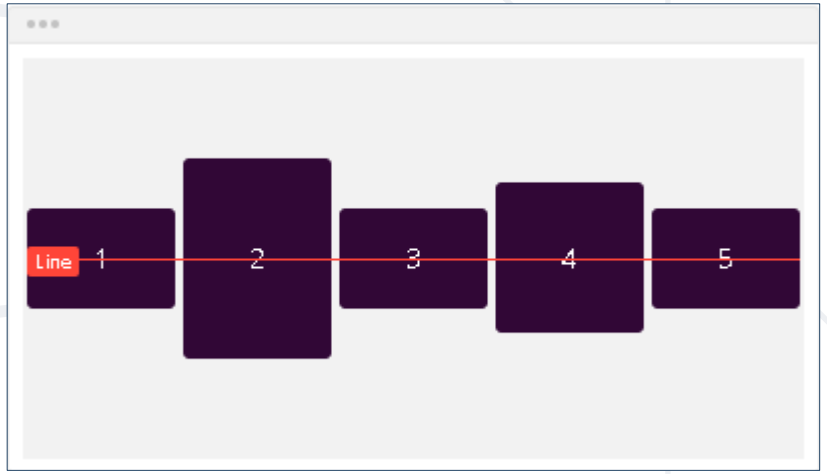
- The flexbox items are aligned at the **end** of the **cross axis**

```
align-items: flex-end;
```



- The flexbox items are aligned at the **center** of the **cross axis**

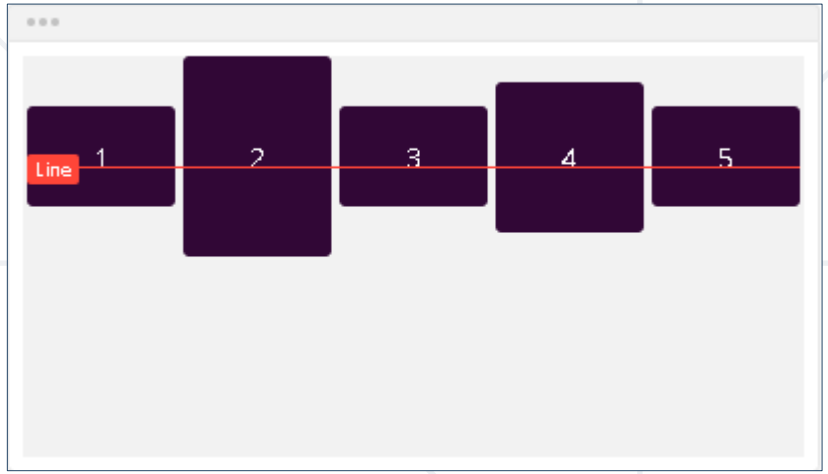
```
align-items: center;
```



Align-items

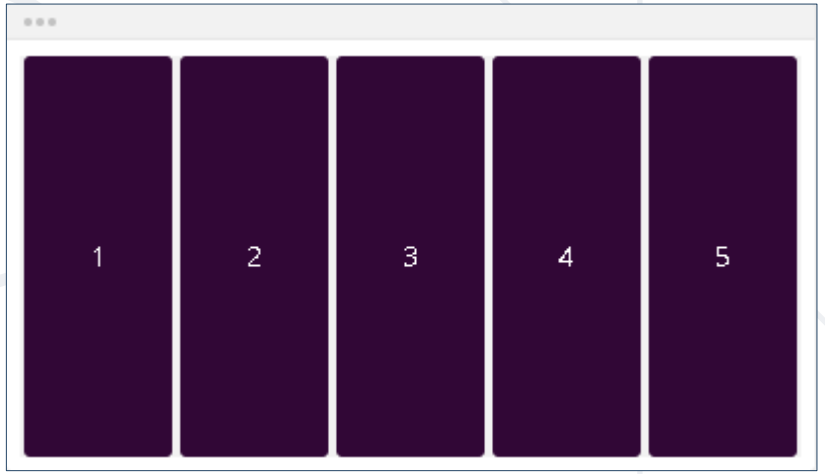
- The flexbox items are aligned at the **baseline** of the **cross axis**

```
align-items: baseline;
```



- The flexbox items will stretch across the whole **cross axis**

```
align-items: stretch;
```



Align Content

- Defines how each line is **aligned** within a flexbox container
- It only applies if **`flex-wrap: wrap`** is present, and if there are multiple lines of flexbox items

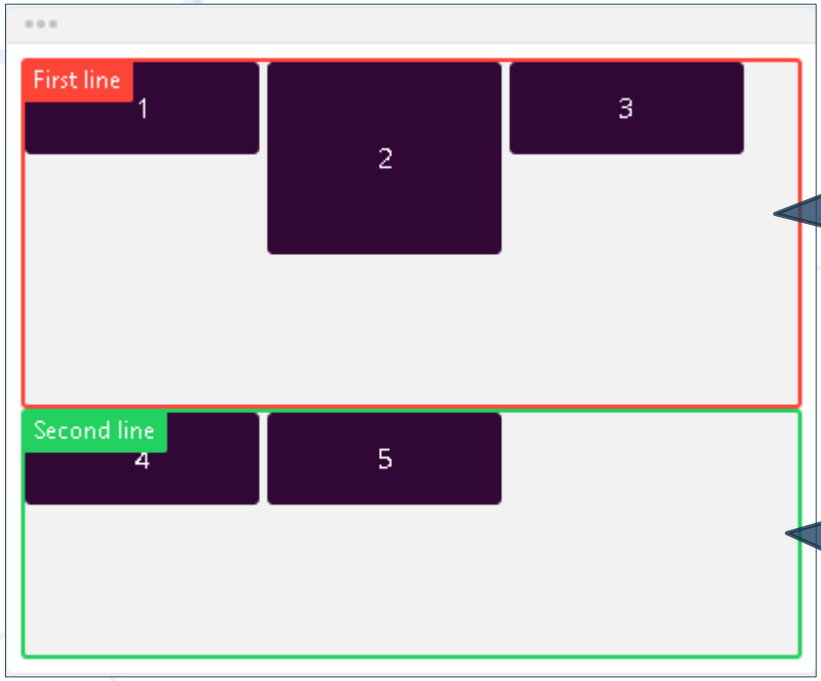
```
align-content: stretch;
```

  - Each line will stretch to **fill** the remaining space

Align Content: Stretch Example

- The first line is **100px** high
- The second line is **50px** high
- The remaining space is **150px** and it is distributed equally amongst the two lines

The container is 300px high
All boxes are 50px high
The second box is 100px high



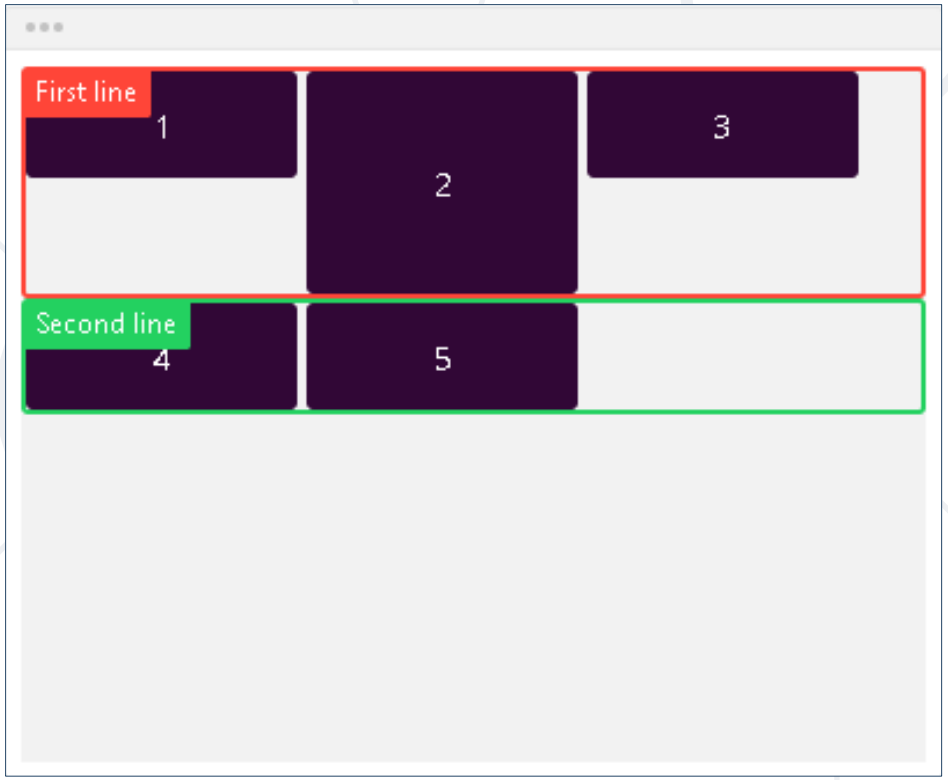
The first line is 175px high

The second line is 125px high

Align Content

- Each line will only fill the space it **needs**
- They will all move towards the **start** of the flexbox container's cross axis

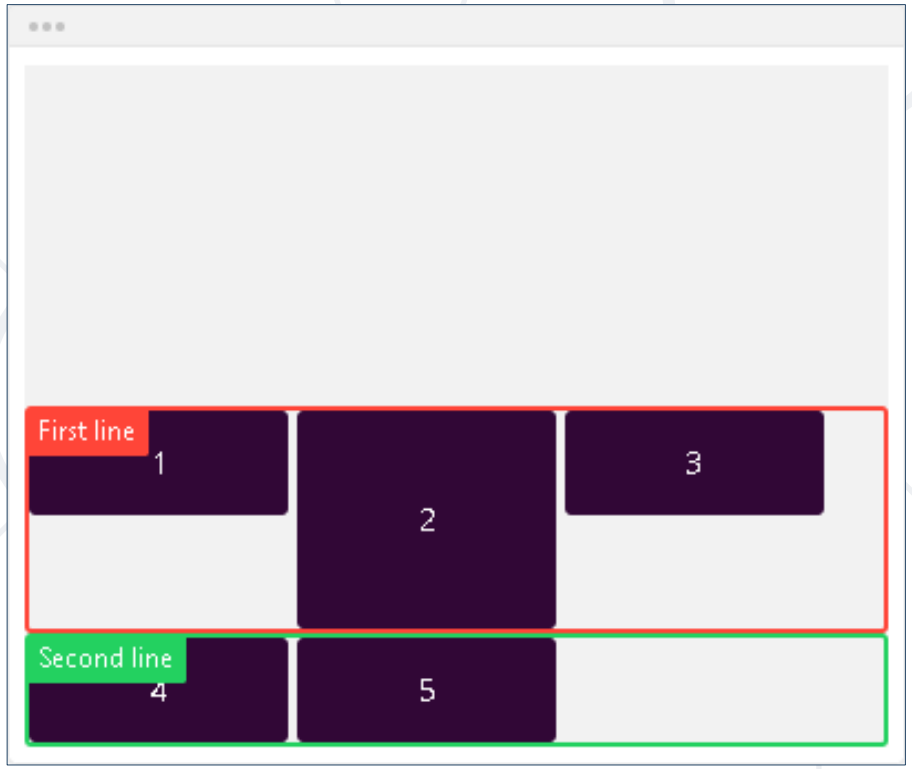
```
align-content: flex-start;
```



Align Content

- Each line will only fill the space it **needs**
- They will all move towards the **end** of the flexbox container's cross axis

```
align-content: flex-end;
```



Align Content

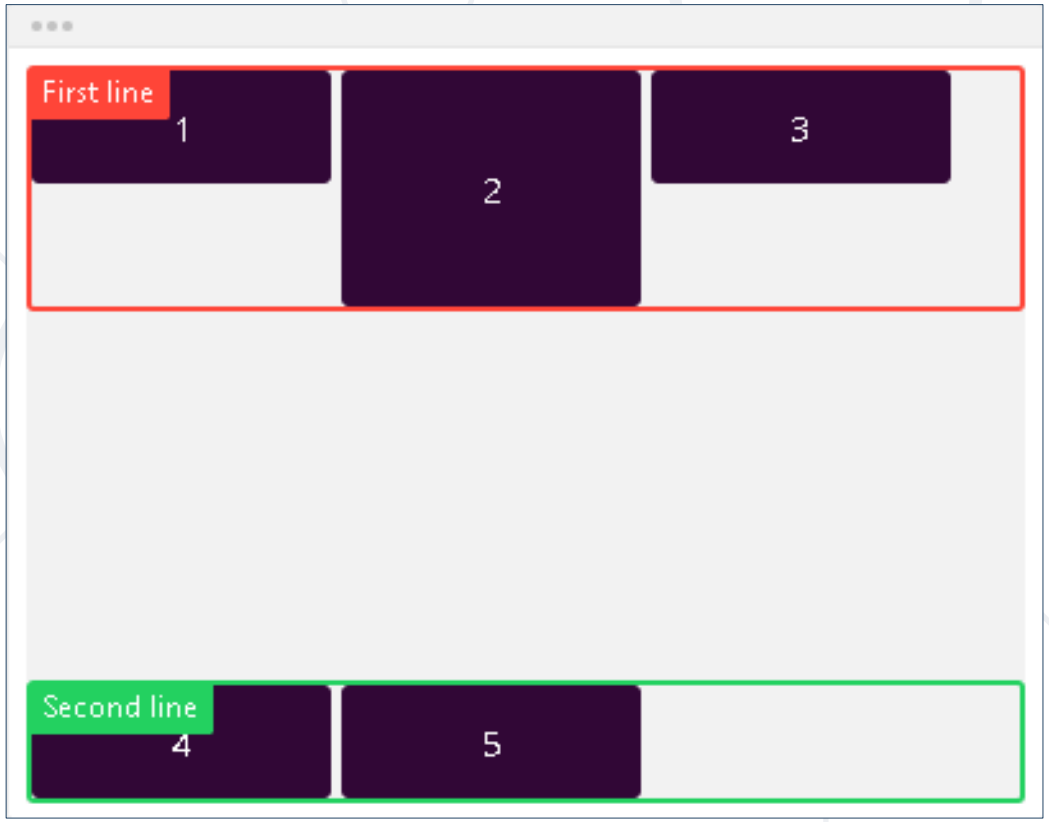
- Each line will only fill the space it **needs**
- They will all move towards the **center** of the flexbox container's cross axis

```
align-content: center;
```

Align Content

- Each line will only fill the space it **needs**
- The **remaining** space will appear **between** the lines

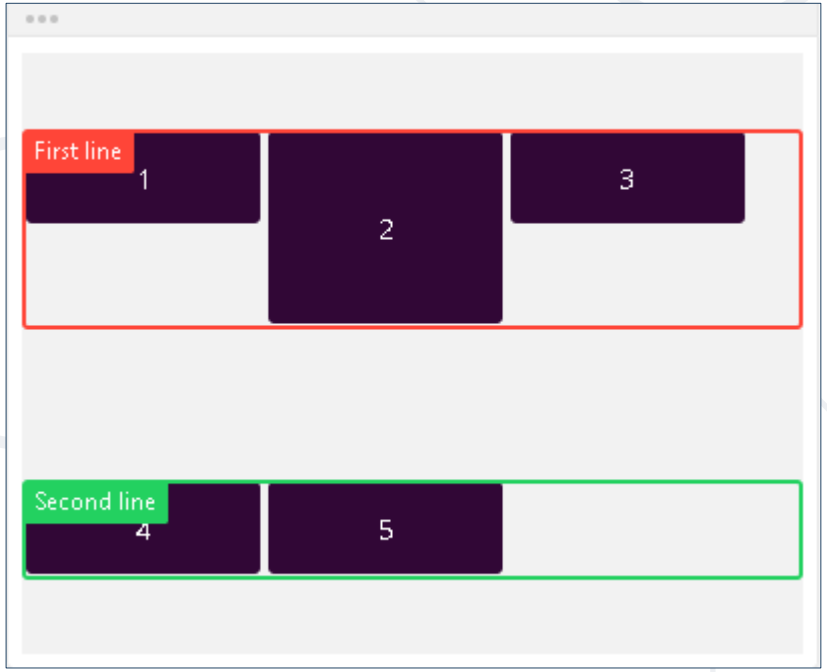
```
align-content: space-between;
```

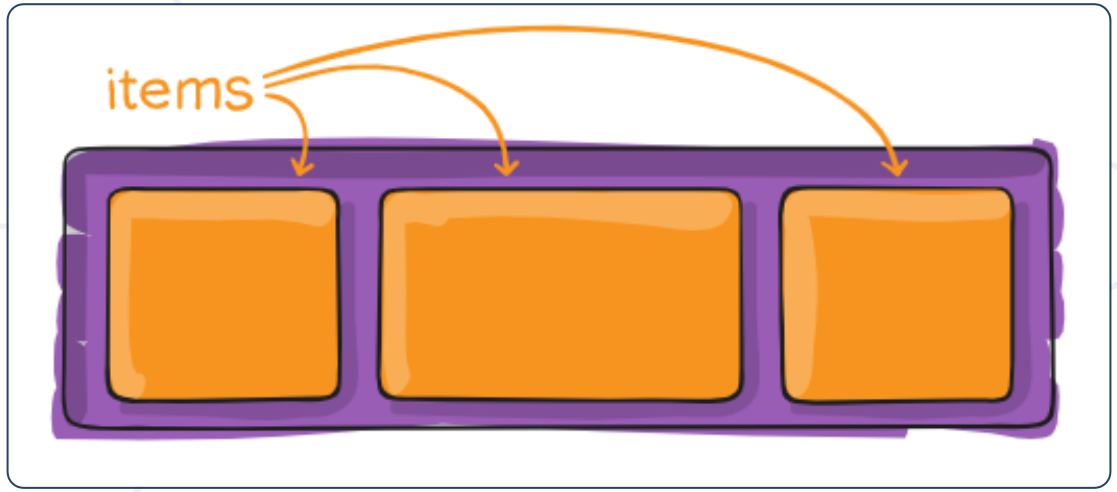


Align Content

- Each line will only fill the space it **needs**
- The **remaining** space will be distributed equally **around** the lines: before the first line, between the two, and after the last one

```
align-content: space-around;
```





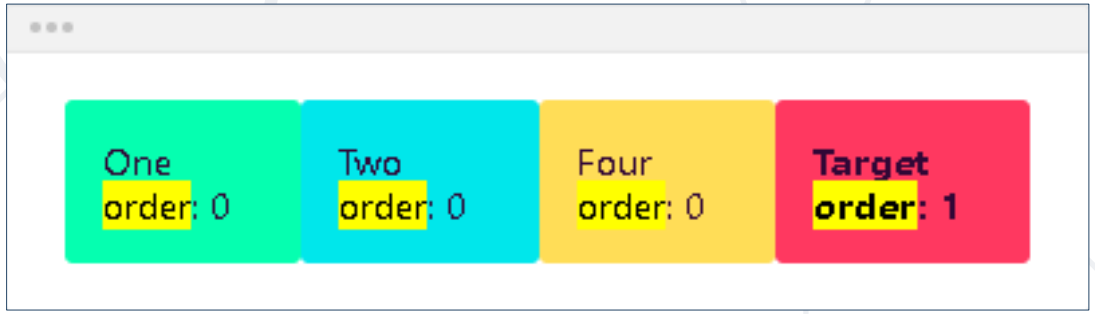
Expands items to Fill Available Free Space or Shrinks Them to Prevent Overflow

Order

- Order - defines the order of a flexbox item
  - The order of the flexbox items is the one defined in the **HTML code**
- The order is **relative** to the flexbox item's **siblings**
- The final order is defined when all individual flexbox item order values are considered

```
order: 0;
```

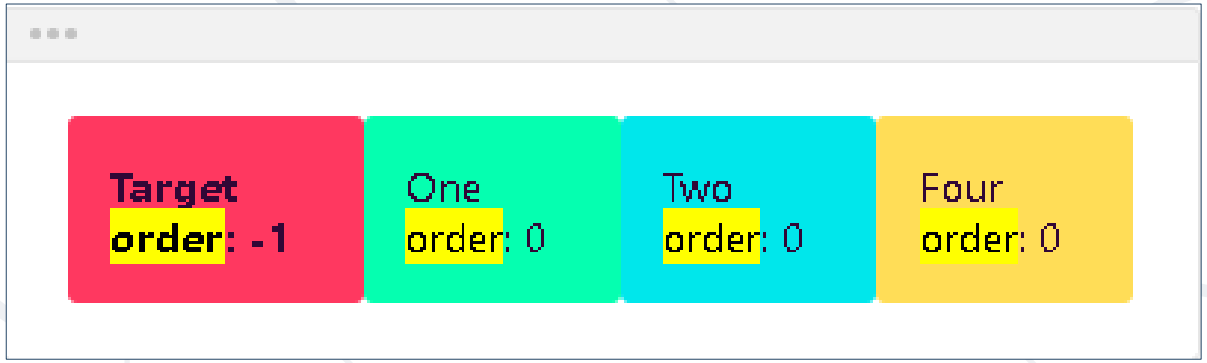

```
order: 1;
```



Order

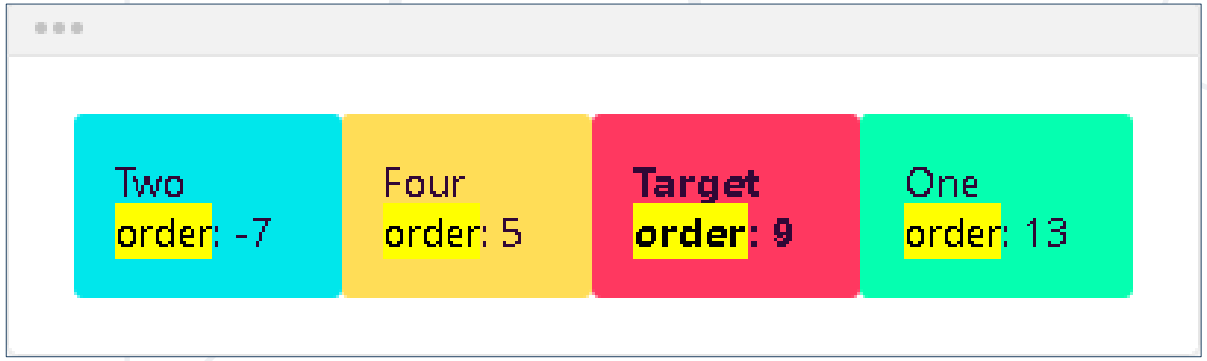
- You can use **negative** values

```
order: -1;
```



- You can set a **different** value for each flexbox item

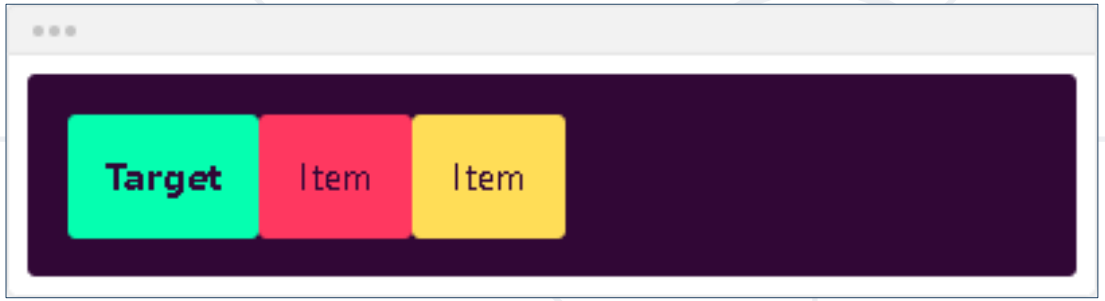
```
order: 9;
```



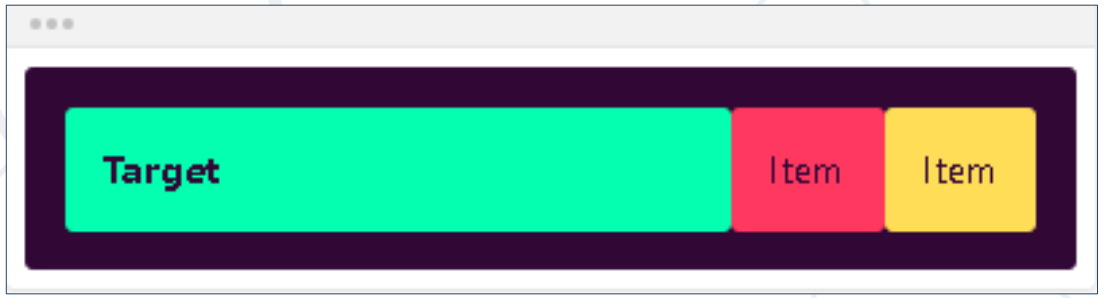
Flex Grow

- Defines how much a flexbox item should **grow** if there's space available
  - The element will **NOT** grow if there's space available
  - It will only use the space it needs
- The element will **grow** by a factor of 1
- It will fill up the remaining space if no other flexbox item has a **`flex-grow`** value

```
flex-grow: 0;
```




```
flex-grow: 1;
```

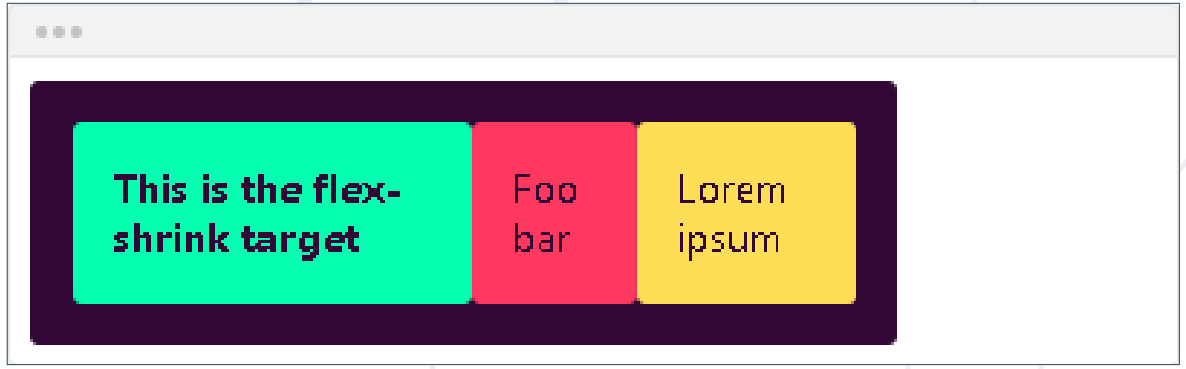


Flex Shrink

- Defines how much a flexbox item should **shrink** if there's **NOT enough** space available

```
flex-shrink: 1;
```

  - If there's **NOT enough** space available in the container's main axis, the element will **shrink** by a factor of **1**, and will wrap its content



Flex Shrink

- The element will **NOT** shrink it will retain the width it needs, and **NOT** wrap its content
- Its siblings will shrink to give space to the target element.
  - Because the target element will **NOT** wrap its content, there is a chance for the flexbox container's content to **overflow**

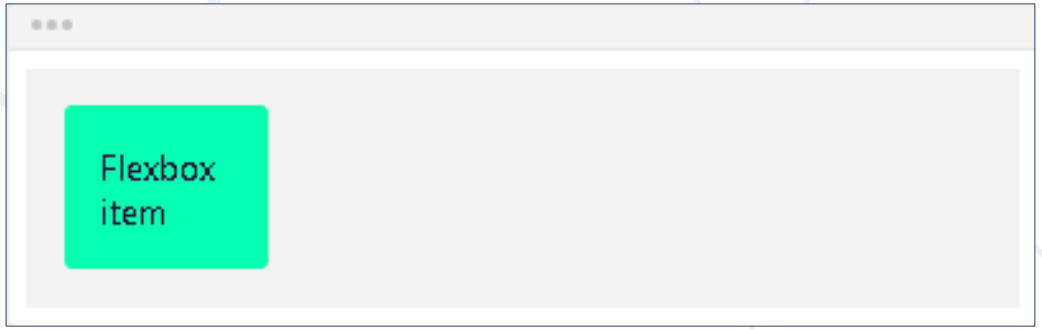
```
flex-shrink: 0;
```

Flex Basis

- Defines the **initial size** of a flexbox item
  - The element will be automatically sized based on its **content**, or on any **height** or **width value** if they are defined
- You can define **pixel** or **(r)em** values
- The element will wrap its content to avoid any overflow

```
flex-basis: auto;
```


```
flex-basis: 80px;
```



Flex

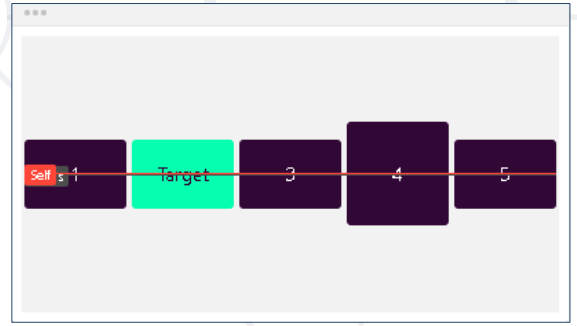
- Flex is the shorthand for:
  - `flex-grow`
  - `flex-shrink`
  - `flex-basis`
- The default value is `0 1 auto`

```
.item {
    flex: <flex-grow> <flex-shrink> <flex-basis>
}
```

Align Self

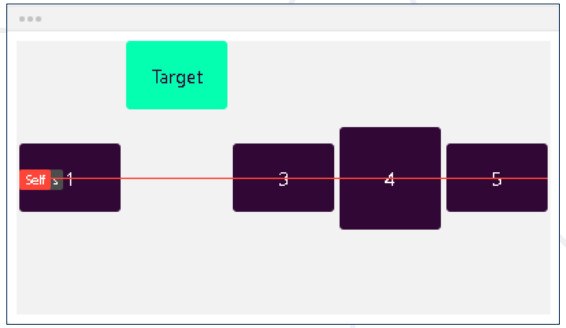
- Works like `align-items`, but applies only to a **single** flexbox item, instead of all of them
  - The target will use the value of `align-items`

```
align-self: auto;
```



- The container has `align-items: center`
- The target has `align-self: flex-start`

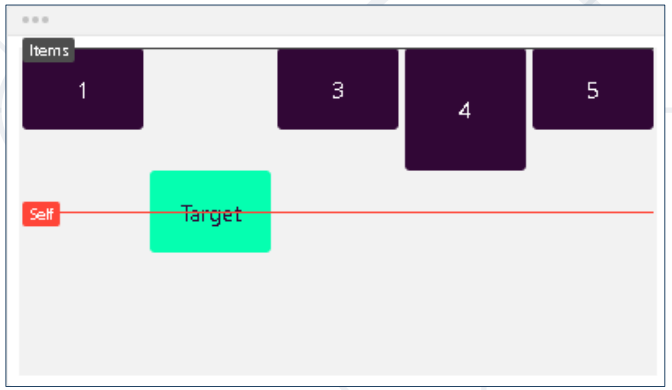
```
align-self: flex-start;
```



Align Self

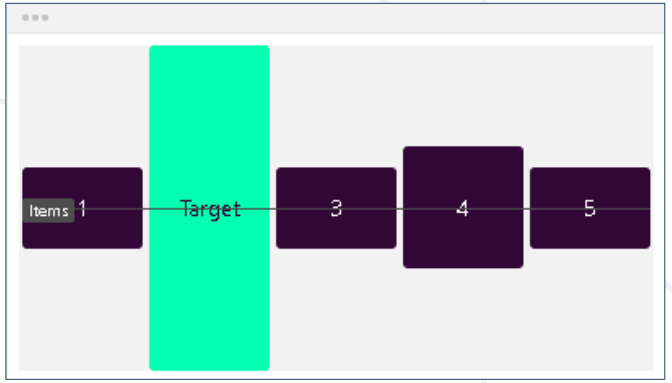
- The container has align-items: `flex-start`
- The target has `align-self: center`

```
align-self: center;
```



- The container has `align-items: center`
- The target has `align-self: stretch`

```
align-self: stretch;
```



Summary

- What is **Flexbox**?
  - Why **Flexbox**?
- Properties for the Parent: `display`, `direction`, `wrap`, `justify`, `align`
- Properties for the children: `order`, `shrink`, `align`

Questions?

SoftUni

Software University | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids | Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

createX

Postbank
Решения за твоето утре

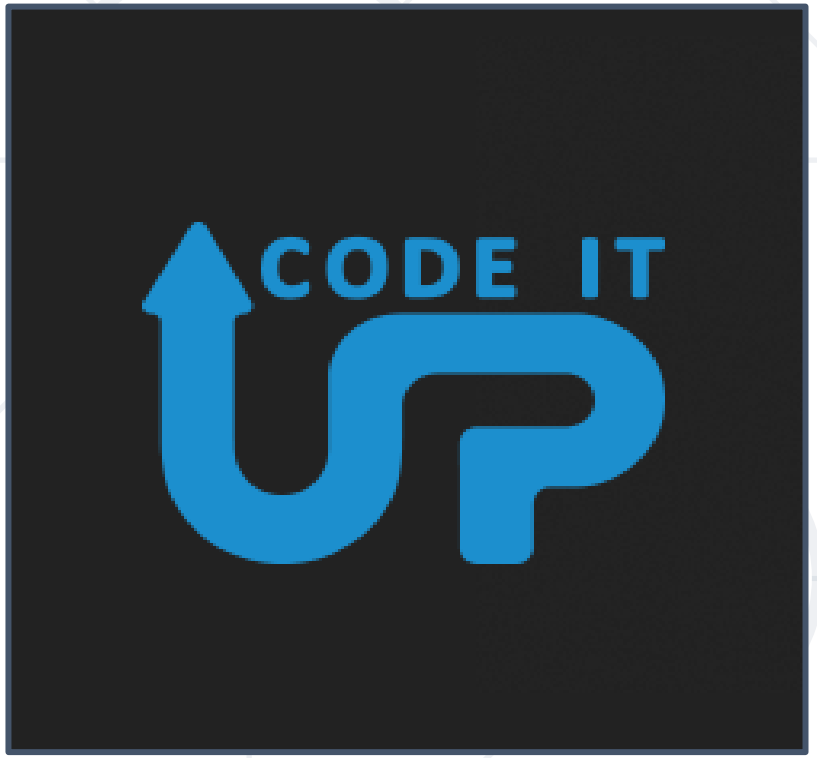
BOSCH

DXC TECHNOLOGY

SmartIT

Educational Partners

Software University



License

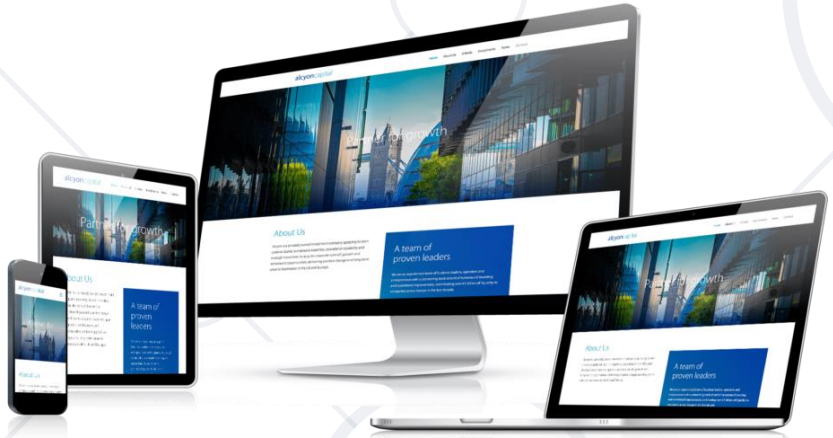
- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

Media Queries

Responsive Web Design



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Table of Contents

Have a Question?

Software University

sli.do

#html-css

Resize, Hide, Shrink, or Enlarge A website to Make It Look Good On All Devices

What is Responsive Web Design?

- **Automatically resize**, **hide**, **shrink**, or **enlarge**, a website, to make it look good on **all devices** (desktops, tablets, and phones)
- Setting the Viewport:
  - Add the following <**meta**> element:

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

- This will give the browser instructions on how to control the page's **dimensions** and **scaling**

Components of Responsive Design

- Responsive website design consists of the following three main components:
  - **Flexible layouts**
    - Using a flexible grid to create the website layout
    - That will dynamically resize to any width
  - **Media queries**
    - Allow designers to specify different styles for specific browser and device circumstances

Components of Responsive Design

- **Flexible Media**
  - Makes media (images, video and other format) scalable

Benefits of Using a Responsive Website

Software University

- Increased **traffic** from mobile users
- Lower **cost** and website maintenance
- Provides a seamless **User Experience (UI)**
- Adapts easily to any screen size
- Improves your **SEO** efforts

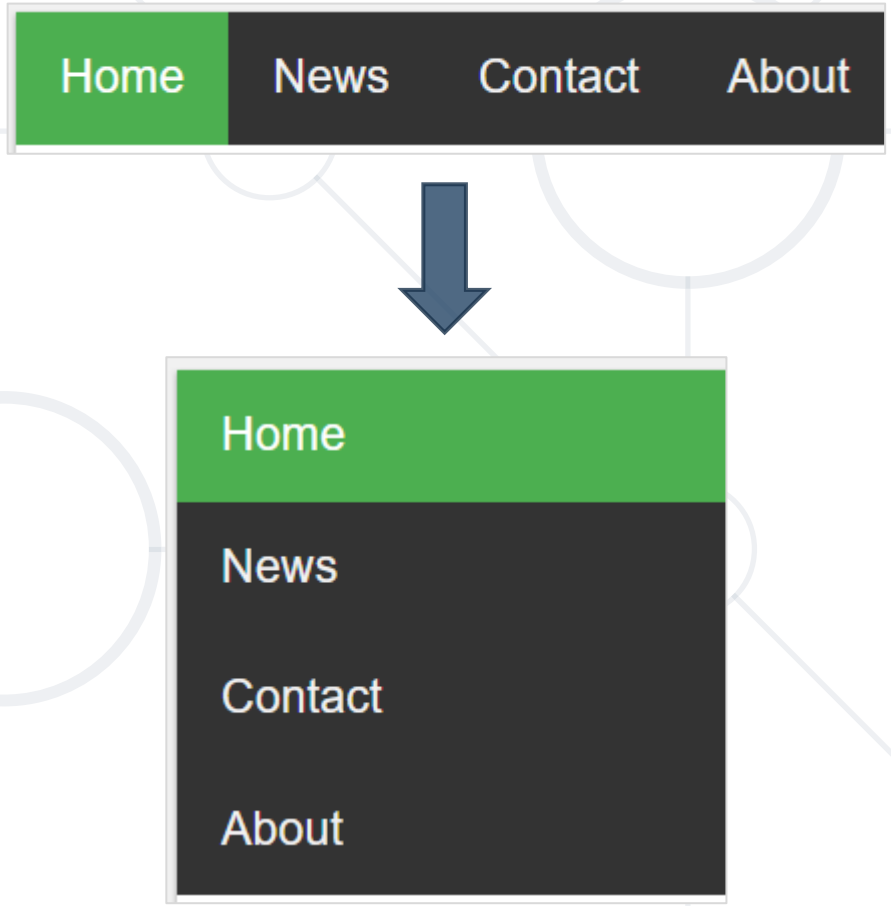
Why Responsive Website Design Works?

Software University

- **Google** prioritizes responsive websites
- **73%** of total **eCommerce** revenue comes from mobile
- **94%** of people **judges** websites on responsive web design
- Almost **63%** of all Internet **access** is done through the phone
- **85%** of adults own a **smart phone**
- First **impressions** are **94%** design-related
- **85%** of people want **Mobile-Friendly** websites
- Responsive design integrates **social media**

What is a Media Query?

- Media Queries are a feature of CSS that enable webpage content to adapt to different screen sizes and resolutions
- They are a fundamental part of responsive web design and are used to customize the appearance of websites for multiple devices



Media Query Syntax

- A media query consists of a **media type** and can contain one or more **expressions**, which resolve to either **true** or **false**

```
@media (max-width: 600px) {
   .menu a { display: block }
}
```

if (width <= 600px), apply these CSS rules

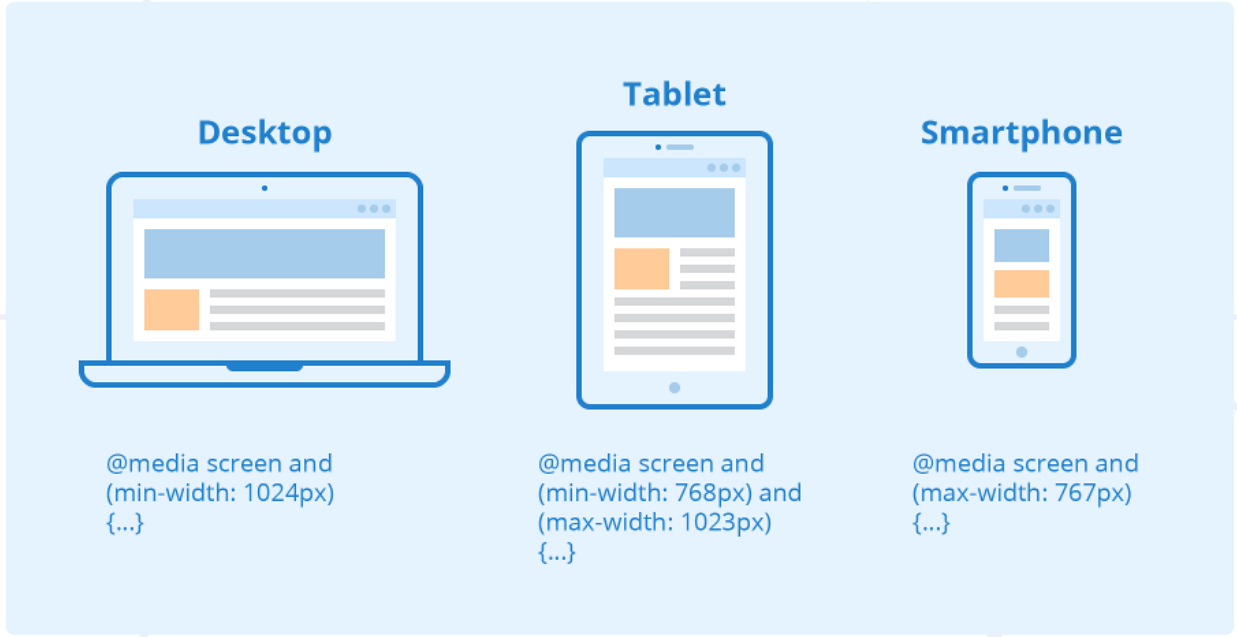
- The result of the query is **true** if the specified media type matches the type of device the document is being displayed on
- Unless you use the **not** or **only** operators, the media type is **optional**, and the all type will be implied

Media Query Syntax

- A **media type**, which tells the browser what kind of media this code is for (e.g. print, or screen)
- A **media feature rule** - test that must be passed for the contained CSS to be applied
- A set of **CSS rules** that will be applied if the test passes and the media type is correct

Media Queries

- Media queries in CSS3 look at the **capability** of the device
- Media queries can be used to check many things, such as:
  - width and height of the **viewport**
  - width and height of the **device**
  - **orientation** (is the tablet/phone in **landscape** or **portrait** mode?)
  - **resolution**



Introduction to Media Types

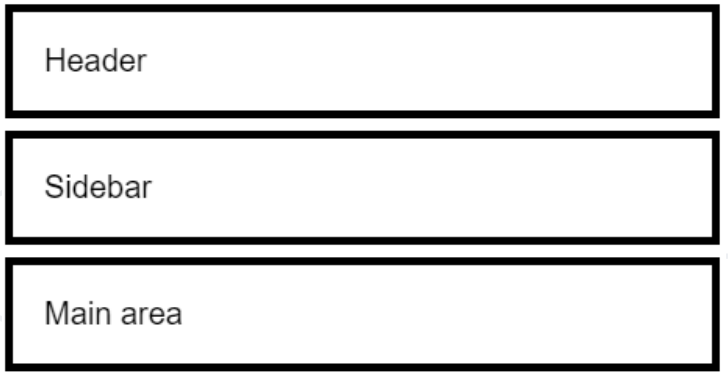
CSS Media Types

- Media Types describe the general category of a given device:
  - `all` - used for all media type devices
  - `print` - used for printers
  - `screen` - used for computer screens, tablets, smart-phones etc.

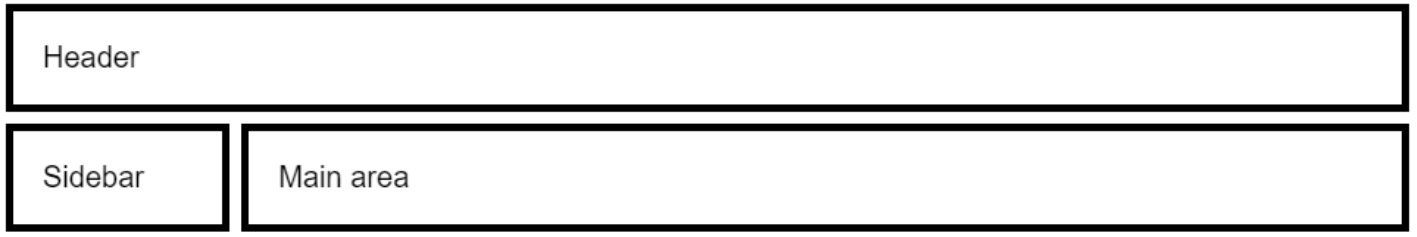
Media Queries + CSS Grid – Example

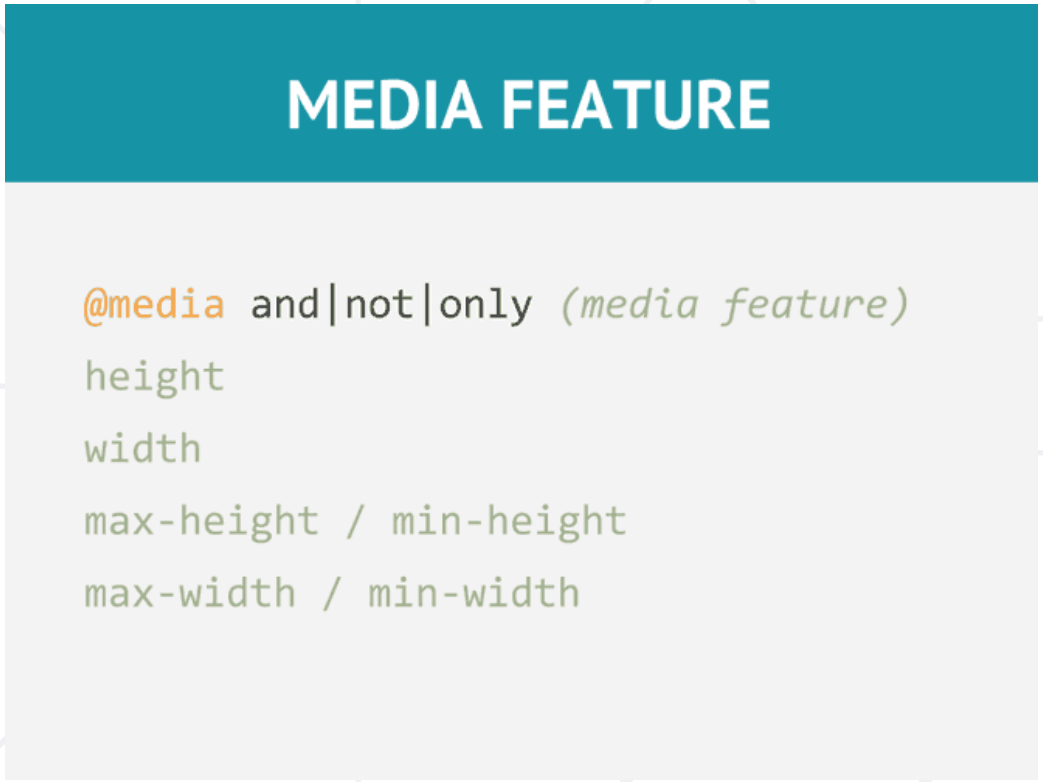
```
.container {
  grid-template-areas:
    "header"
    "sidebar"
    "main";
}
@media (min-width: 600px) {
.container {
  grid-template-areas:
    "header  header"
    "sidebar main";
  }
}
```

Width < 600px



Width >= 600px





Rule Used In media queries to Apply Different Styles for Different media types/devices

Media Feature Rules

- After specifying the type, you can then target a media feature with a **rule**
  - Width and height - we can apply CSS if the viewport is above or below a certain width, using **width**

```
@media screen and (width: 600px) {
    body {
    color: red;
  }
}
```

Media Feature Rules

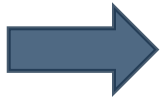
- We can apply CSS if the viewport is under or above an exact width - using **min-width**, **max-width**

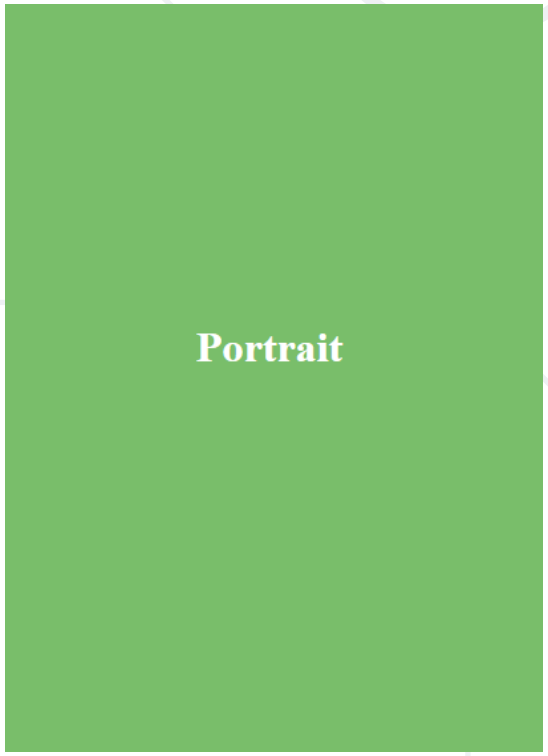
```
@media screen and (max-width: 400px) {
    body {
      color: blue;
   }
}
```

Media Feature Rules

- Orientation - allows to test for `portrait` or `landscape` mode
  - **Landscape** – when window is **wider than** its **height**
  - **Portrait** – when window is **higher** than its **wide**

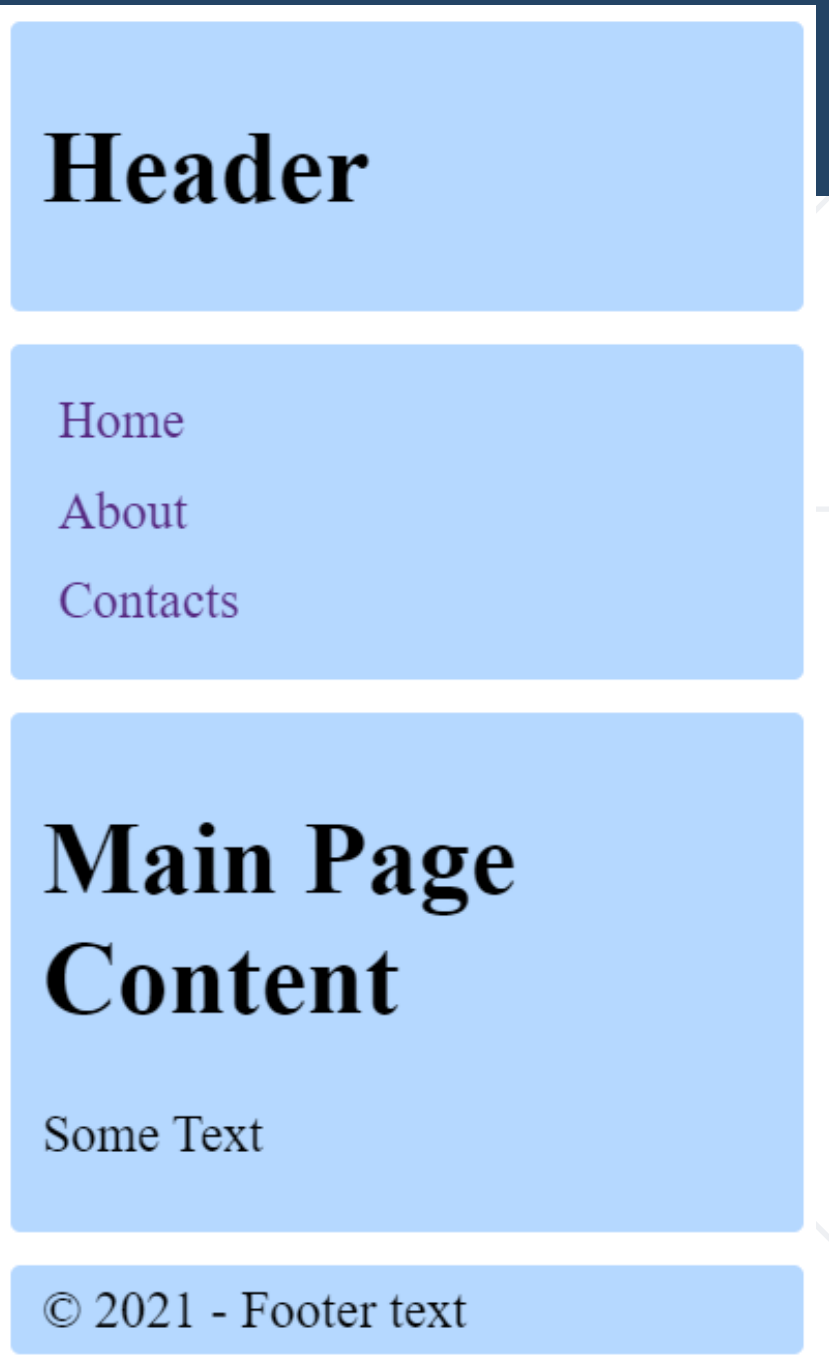
```
@media (orientation: portrait) {
   .portrait { display: block; }
   .landscape { display: none; }
}
```





CSS Grid Responsive Layout

```css
@media (max-width: 500px) {
  body {
    display: grid;
    grid-template-areas:
      "header header"
      "aside aside"
      "main main"
      "footer footer";
  }
}
```



Demo: CSS Grid Site Layout

- https://codepen.io/snakov/pen/MWbxZOy

Media Queries Conditions

Logical Operators: and | or

```
@media (min-width: 600px) and (max-width: 900px) {
   .example {
     /* Styles! */
   }
}
```

Screen width between 600px and 900px

```
@media (max-width: 900px), (min-width: 1100px) {
   .example {
     /* Styles! */
   }
}
```

OR

CSS @Import Rule

Summary

- What is **Responsive Web Design**?
- What are **Media Queries**?
- Media **Types**
- Media **Feature Rules**

Questions?

SoftUni

Software University | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids | Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

createX

Postbank
Решения за твоето утре

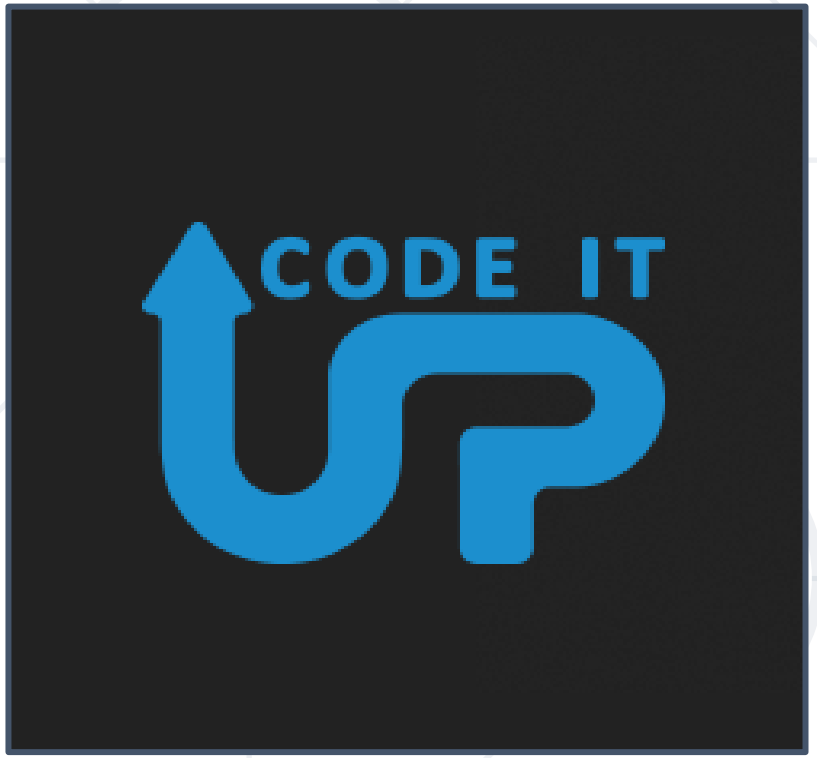
BOSCH

DXC TECHNOLOGY

SmartIT

Educational Partners

Software University



License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg