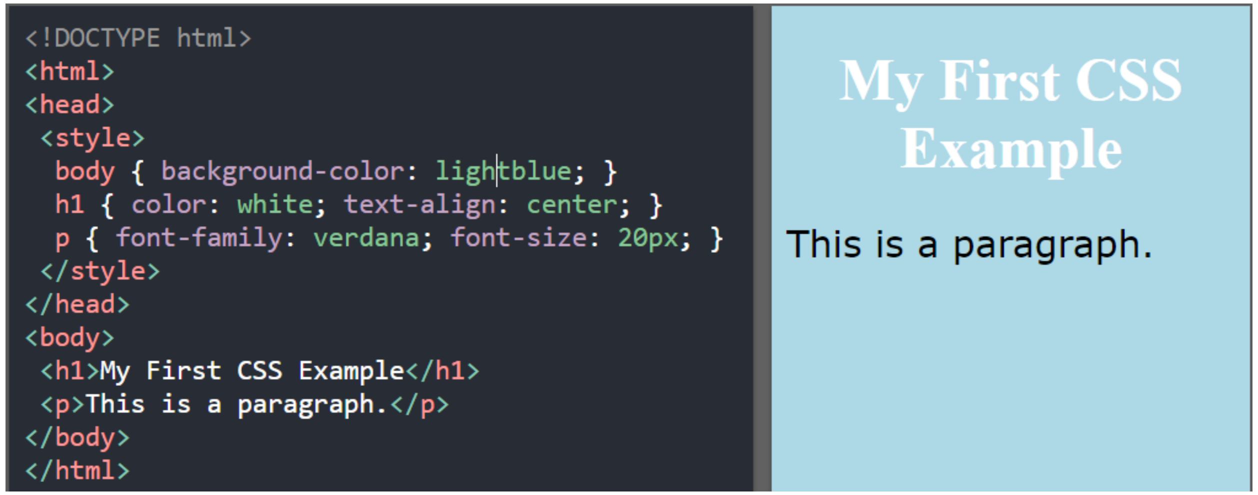
Part III: CSS

CSS - Introduction

C S S
Cascading Style Sheets

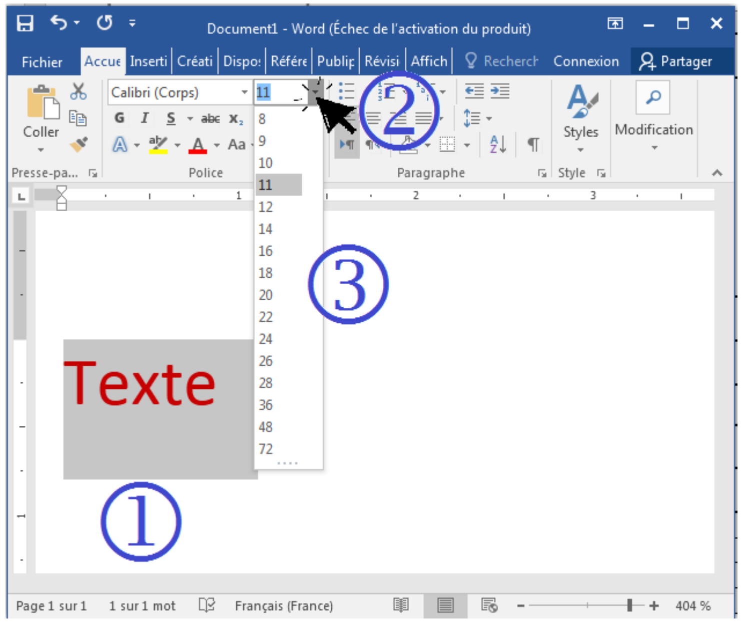


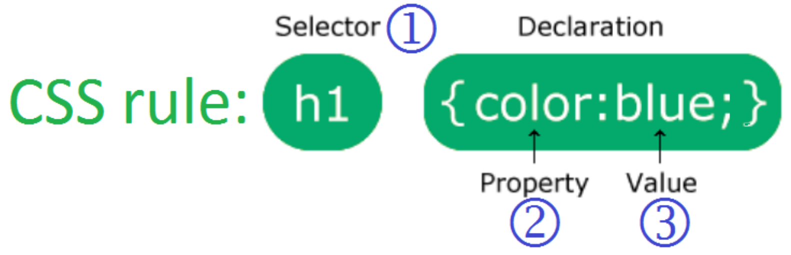
```
<!DOCTYPE html>
<html>
<head>
 <style>
  body { background-color: lightblue; }
  h1 { color: white; text-align: center; }
  p { font-family: verdana; font-size: 20px; }
 </style>
</head>
<body>
 <h1>My First CSS Example</h1>
 <p>This is a paragraph.</p>
</body>
</html>
```

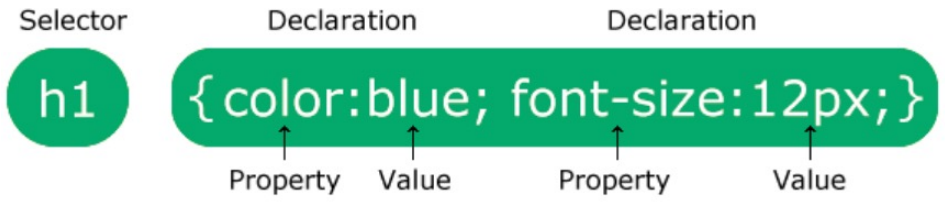
- **CSS** is the second language required to produce web pages, it describes **how** HTML elements should be **displayed**.
- As a **styling language**, CSS is used for decoration, formatting, and layout of web pages.
- With CSS, we can: color text, choose a font, add a background image, draw borders and even round them, make alignments, shifts, adjustments, ... and many other effects and functions.

CSS - Principle

- To format text in a graphic software such as Word, the principle is simple:
  1. **Select** (using mouse) the target part.
  2. Choose the tool (**property**) to apply from the toolbox.
  3. Optionally choose a **value** for the selected property.
- CSS adopts the same principle to format elements (**tags**) in HTML documents, but based on text commands (**rules**):
  1. **Select** the target tag(s) (name, class, ...).
  2. Choose the **property** to apply (name).
  3. Choose the **value** for that property.



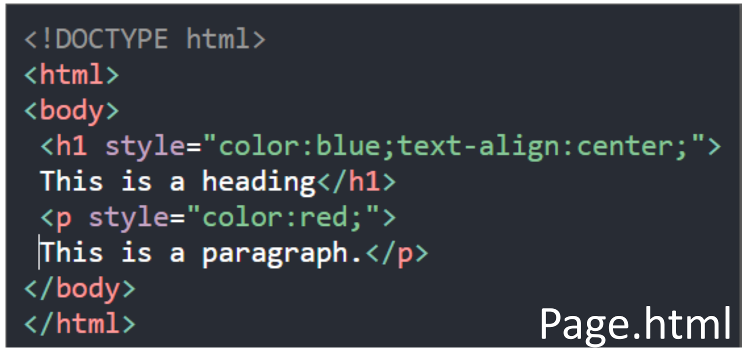




CSS – Where to insert?

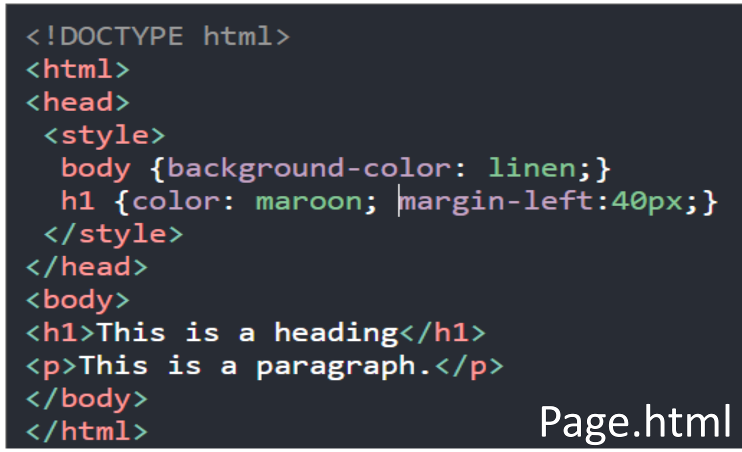
When a browser reads a style sheet (CSS code), it formats the HTML document according to the content of the style sheet. Now, where do we put the CSS code?

- External: CSS code in a separate ".css" file from the HTML document, file linking is ensured by the **<link>** tag (style shared by multiple pages).
- Internal: CSS code within the HTML document inside the **<style>** tag (unique style for a page
- Inline: CSS declarations directly within the tag using the common "**style**" attribute (unique style for a tag).



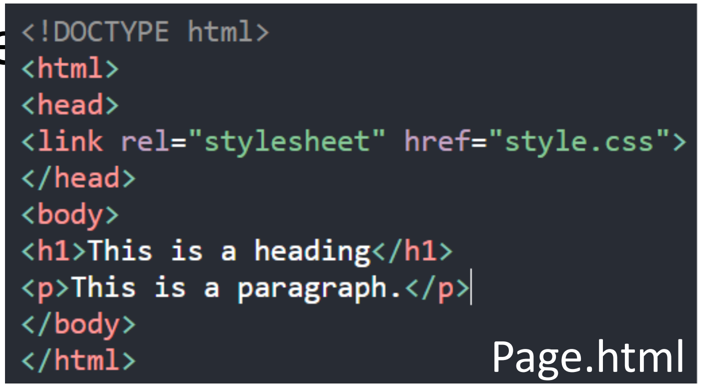
```
<!DOCTYPE html>
<html>
<body>
 <h1 style="color:blue;text-align:center;">
 This is a heading</h1>
 <p style="color:red;">
 This is a paragraph.</p>
</body>
</html>
```

Page.html



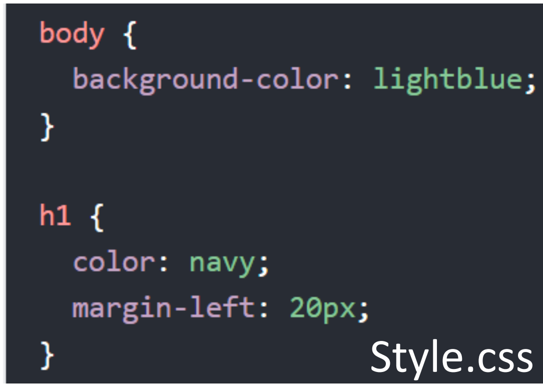
```
<!DOCTYPE html>
<html>
<head>
 <style>
  body {background-color: linen;}
  h1 {color: maroon; margin-left:40px;}
 </style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Page.html



```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="style.css">
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Page.html

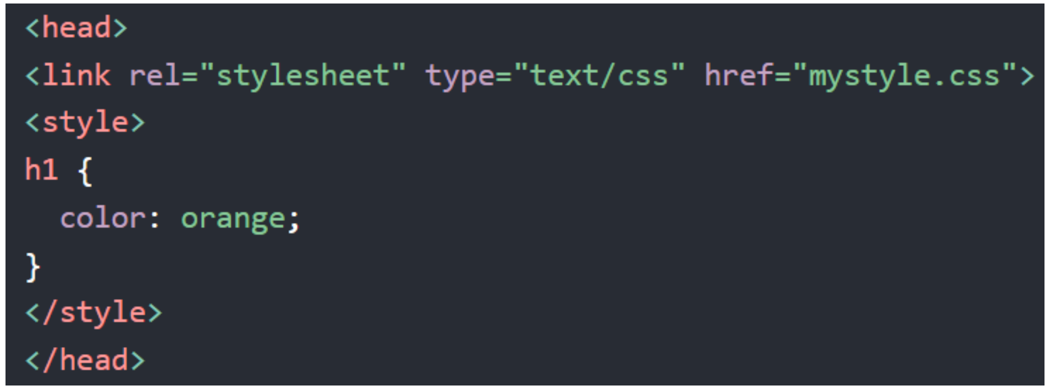


```
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

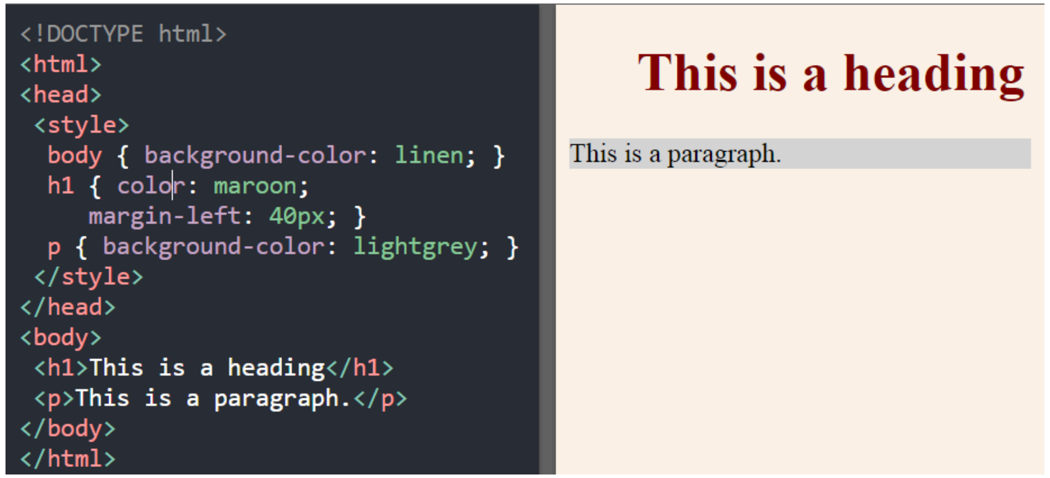
Style.css

CSS – Rules

- If a property of a tag is defined with multiple selectors in several places, the value of the last one read is used.
- Which style to use when multiple styles are specified for a tag? They will "**cascade**" in the following order (taking into account the specificity):
    1. Inline style
    2. Internal and external style
    3. Default style assigned by the browser
- Tags nested within other tags implicitly **inherit** their style if it has not been explicitly assigned.



```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style>
h1 {
  color: orange;
}
</style>
</head>
```



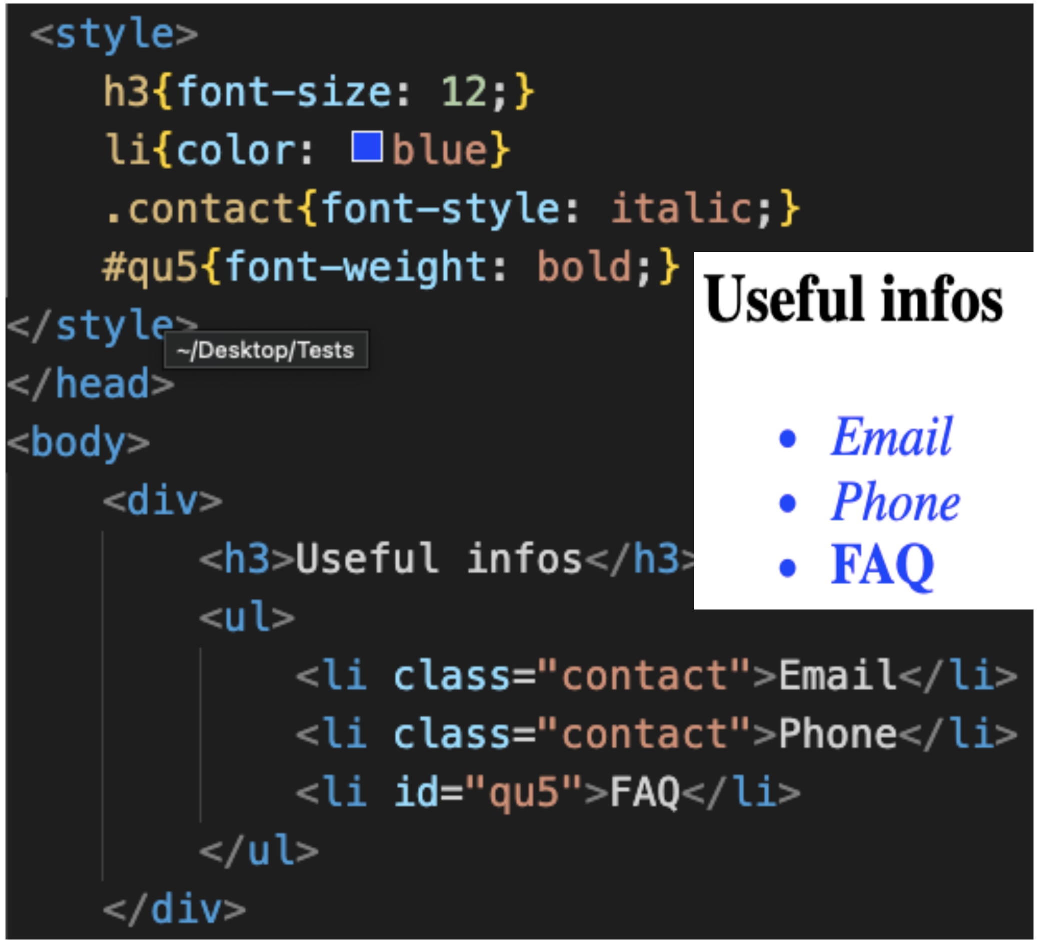
```
<!DOCTYPE html>
<html>
<head>
 <style>
  body { background-color: linen; }
  h1 { color: maroon;
     margin-left: 40px; }
  p { background-color: lightgrey; }
 </style>
</head>
<body>
 <h1>This is a heading</h1>
 <p>This is a paragraph.</p>
</body>
</html>
```

This is a heading

This is a paragraph.

CSS – Selectors

- Selectors? An expression used to find or select the tag(s) to be styled.
- CSS selectors take the form of the following five categories:
- Simple: Use the tag name (p, a, h1, ...), the value of its **class** attribute (.title, .footer, ...), or the value of its **id** attribute (#inp1, #li5, ...) and the universal selector *



```
<style>
    h3{font-size: 12;}
    li{color: ■blue}
    .contact{font-style: italic;}
    #qu5{font-weight: bold;}
</style>
</head>
<body>
    <div>
        <h3>Useful infos</h3>
        <ul>
            <li class="contact">Email</li>
            <li class="contact">Phone</li>
            <li id="qu5">FAQ</li>
        </ul>
    </div>
```

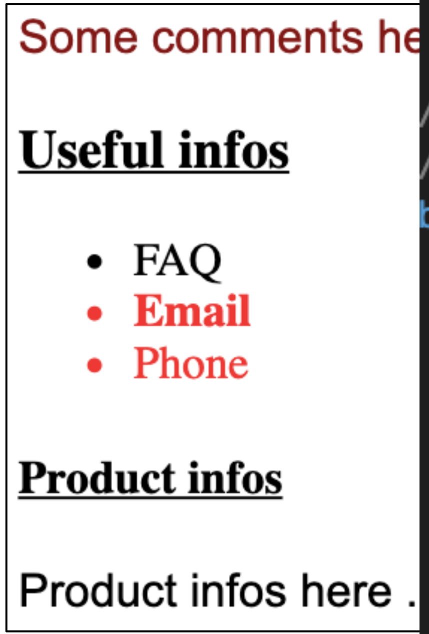
~/Desktop/Tests

Useful infos

- *Email*
- *Phone*
- **FAQ**

CSS – Selectors

- Combined: Use a combination of selectors with multiple operators:
  - (a, b): Select a or b
  - (a b): Select all descendants b of a
  - (a > b): Select all children b of a
  - (a + b): Select the adjacent b following a
  - (a ~ b): Select all adjacent b following a



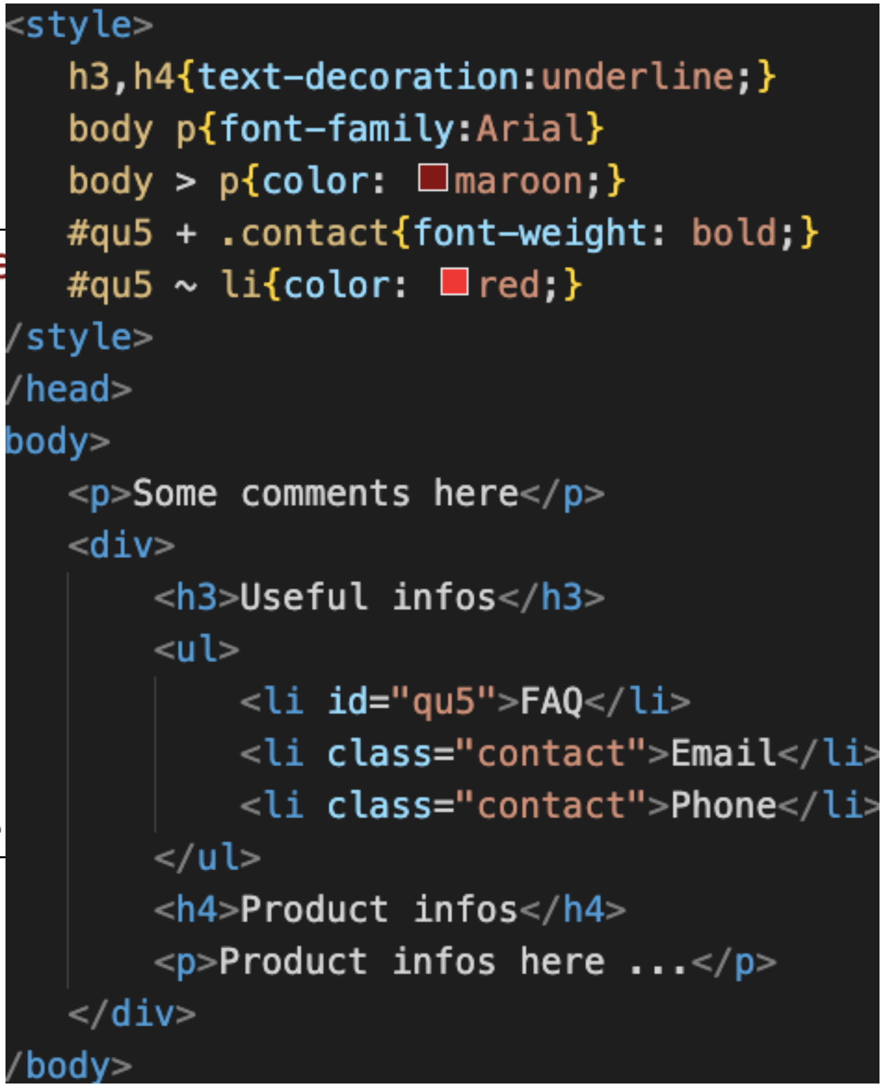
Some comments he

Useful infos

- FAQ
- **Email**
- Phone

Product infos

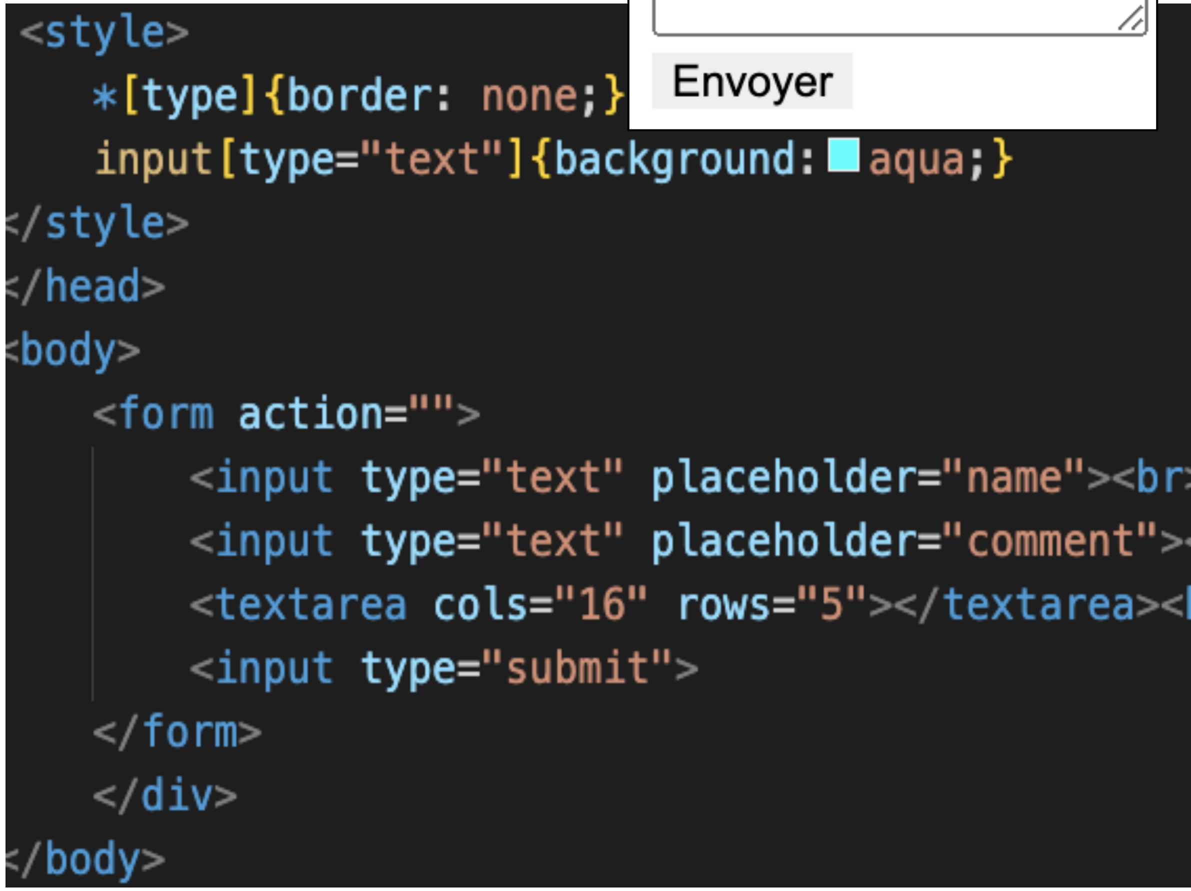
Product infos here .



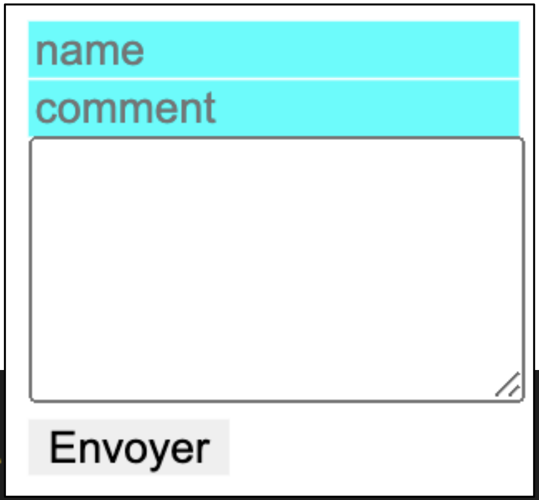
```
<style>
   h3,h4{text-decoration:underline;}
   body p{font-family:Arial}
   body > p{color: ■maroon;}
   #qu5 + .contact{font-weight: bold;}
   #qu5 ~ li{color: ■red;}
/style>
/head>
body>
   <p>Some comments here</p>
   <div>
      <h3>Useful infos</h3>
      <ul>
         <li id="qu5">FAQ</li>
         <li class="contact">Email</li>
         <li class="contact">Phone</li>
      </ul>
      <h4>Product infos</h4>
      <p>Product infos here ...</p>
   </div>
/body>
```

CSS – Selectors

- Attribute selectors: Use the syntax "**selector[attr]** or **selector[attr=val]**" of an attribute as a selection condition.
- You also can use: **[attr*=val], [attr^=val], [attr$=val]**, to select elts whose attr value: contains/starts with/ends with **val** (respectively)

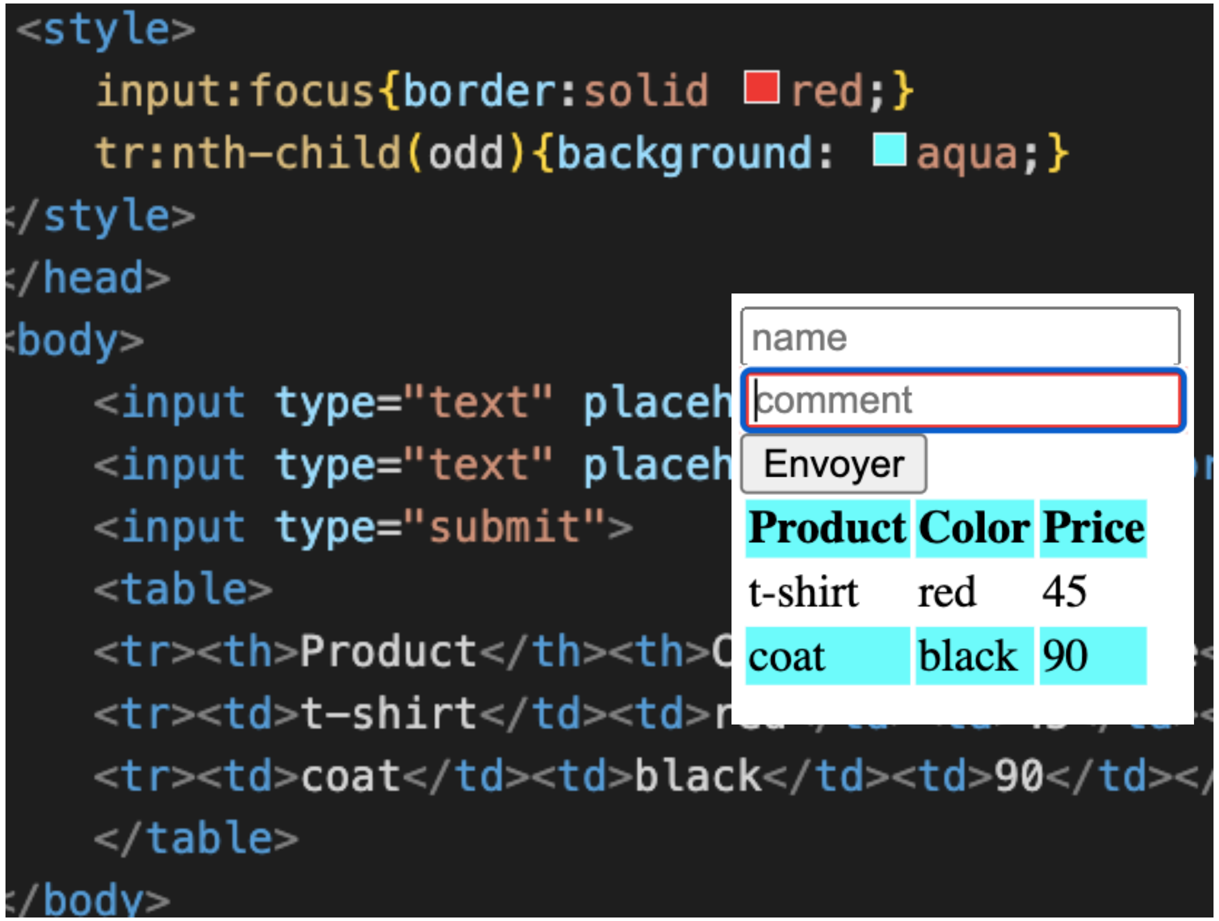


```
<style>
   *[type]{border: none;}
   input[type="text"]{background: aqua;}
</style>
</head>
<body>
   <form action="">
       <input type="text" placeholder="name"><br
       <input type="text" placeholder="comment">
       <textarea cols="16" rows="5"></textarea><
       <input type="submit">
   </form>
   </div>
</body>
```



CSS – Selectors

- Pseudo-classes: Consider certain states and events that occur to tags to make the selection "**selector:pseudo-class**", such as mouse hover (:hover), link has been visited (:visited), get focus (:focus), elt is clicked (:active), first child of a list (:first-child), nth child of a list (:nth-child(odd), ...

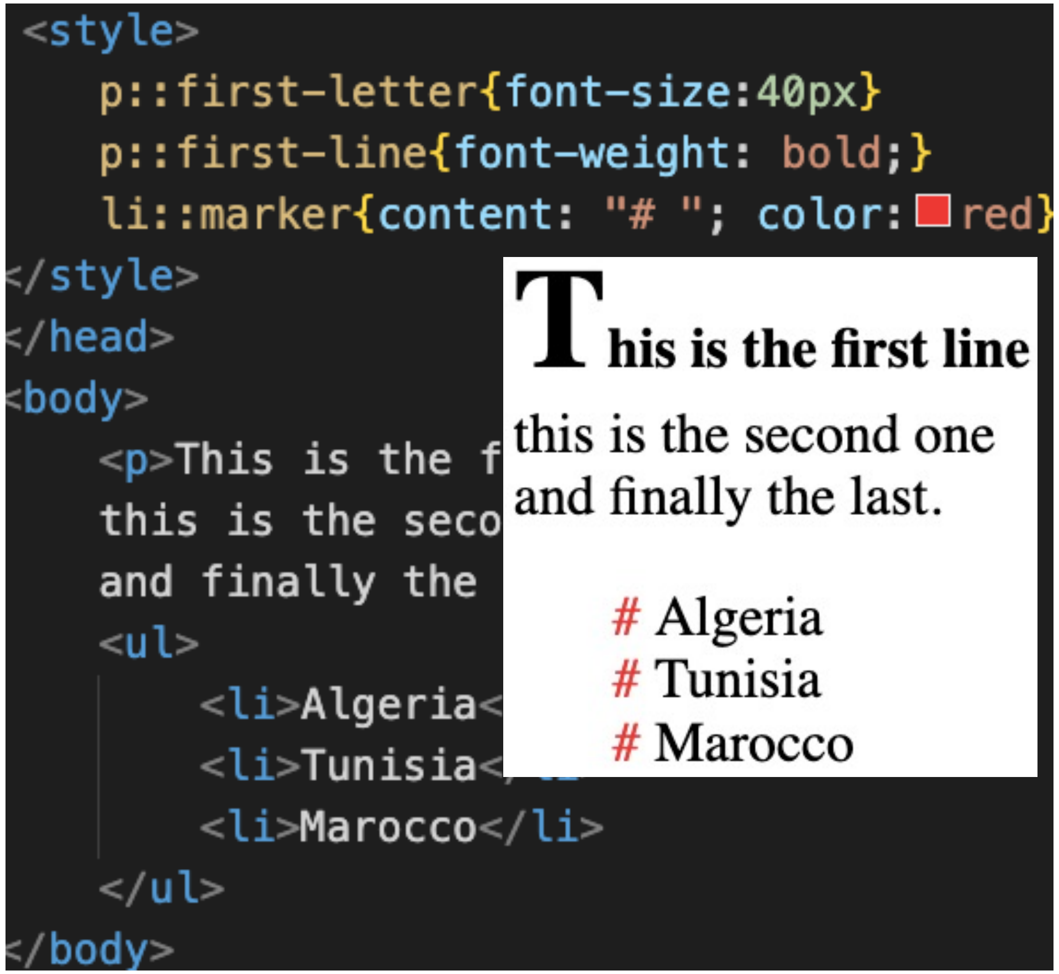


```
<style>
    input:focus{border:solid red;}
    tr:nth-child(odd){background: aqua;}
</style>
</head>
<body>
    <input type="text" placeh
    <input type="text" placeh
    <input type="submit">
    <table>
    <tr><th>Product</th><th>C
    <tr><td>t-shirt</td><td>r
    <tr><td>coat</td><td>black</td><td>90</td><
    </table>
</body>
```

Product	Color	Price
t-shirt	red	45
coat	black	90

CSS – Selectors

- Pseudo-elements: Used to style specific parts of a tag "**selector::pseudo-element**" such as the first letter (::first-letter), first line (::first-line), user selection (::selection), insert content befor/after (::befor/after), input placeholder (::placeholder), list marker (::marker), ...



```
<style>
    p::first-letter{font-size:40px}
    p::first-line{font-weight: bold;}
    li::marker{content: "# "; color:🟥red}
</style>
</head>
<body>
    <p>This is the f
    this is the seco
    and finally the
    <ul>
        <li>Algeria<
        <li>Tunisia</
        <li>Marocco</li>
    </ul>
</body>
```

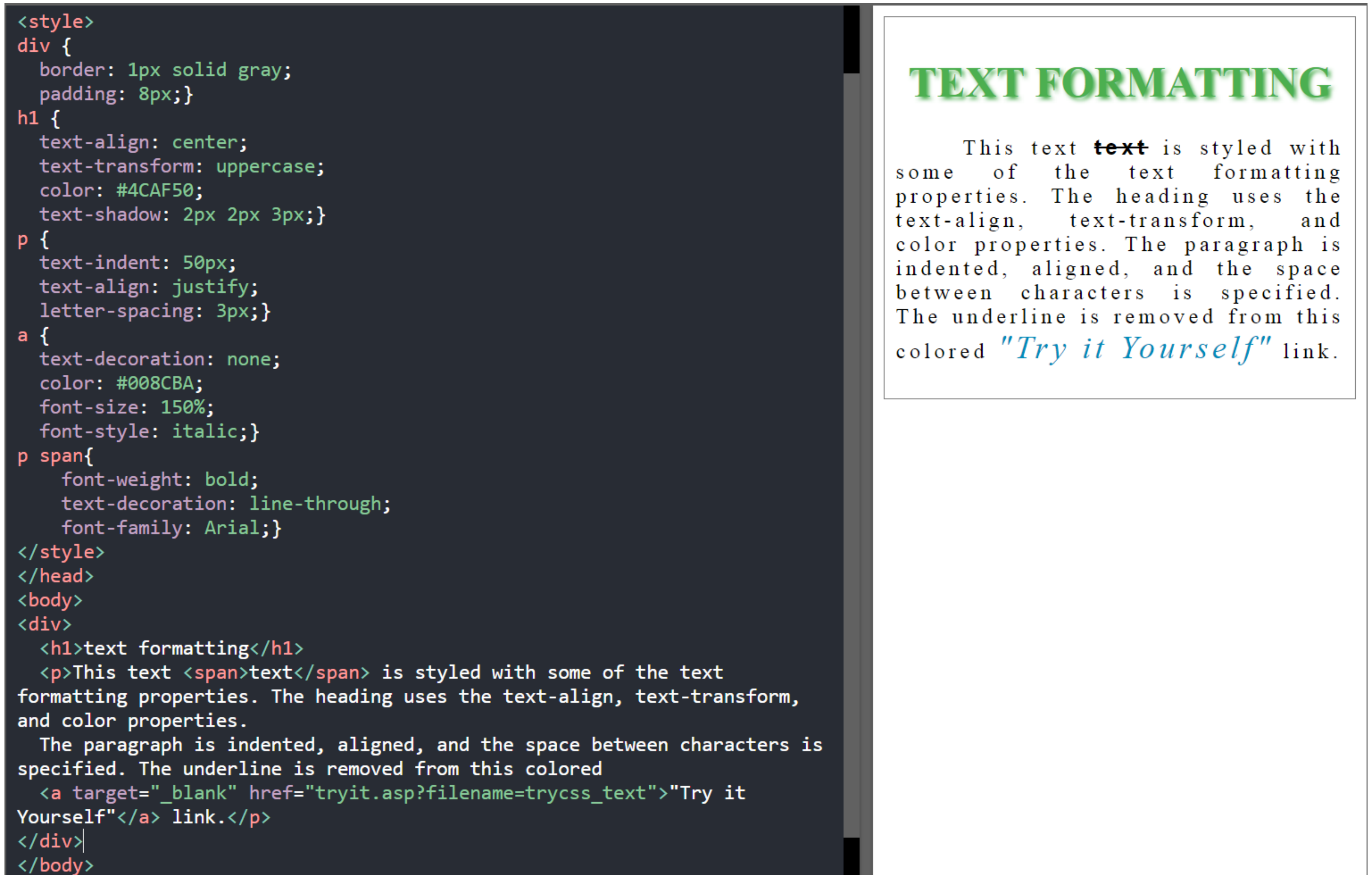
This is the first line
this is the second one
and finally the last.

- # Algeria
- # Tunisia
- # Marocco

CSS -Text

Property	Value	Meaning	Examples
vertical-align	baseline/sub/sup/ …	Vertical alignment	vertical-align: sub;
text-align	left/right/center/justify	Horizontal alignment	text-align: center;
text-decoration	underline/line-through/overline	Underlining	text-decoration: line-through;
text-transform	uppercase/lowercase/capitalize	Uppercase or lowercase	text-transform: capitalize;
text-indent	Value (px/pt/cm/ …)	Indentation of the 1st line	text-indent: 50px;
text-shadow	*h-shad v-shad r-shad color*	Shadow effect: hor ver deg colr	text-shadow: 2px 2px; text-shadow: 2px 2px 3px red;
letter-spacing	Value (px/pt/cm/ …)	Inter-letter spacing	letter-spacing: 5px; letter-spacing: -2px;
word-spacing	Value (px/pt/cm/ …)	Inter-word spacing	word-spacing: 2cm; word-spacing: -2em;
line-height	Value (px/pt/cm/ …)	Line spacing	line-height: 0.8; line-height: 1.8px;

CSS – Text



```
<style>
div {
  border: 1px solid gray;
  padding: 8px;}
h1 {
  text-align: center;
  text-transform: uppercase;
  color: #4CAF50;
  text-shadow: 2px 2px 3px;}
p {
  text-indent: 50px;
  text-align: justify;
  letter-spacing: 3px;}
a {
  text-decoration: none;
  color: #008CBA;
  font-size: 150%;
  font-style: italic;}
p span{
    font-weight: bold;
    text-decoration: line-through;
    font-family: Arial;}
</style>
</head>
<body>
<div>
  <h1>text formatting</h1>
  <p>This text <span>text</span> is styled with some of the text
formatting properties. The heading uses the text-align, text-transform,
and color properties.
  The paragraph is indented, aligned, and the space between characters is
specified. The underline is removed from this colored
  <a target="_blank" href="tryit.asp?filename=trycss_text">"Try it
Yourself"</a> link.</p>
</div>
</body>
```

TEXT FORMATTING

This text ~~**text**~~ is styled with some of the text formatting properties. The heading uses the text-align, text-transform, and color properties. The paragraph is indented, aligned, and the space between characters is specified. The underline is removed from this colored *"Try it Yourself"* link.

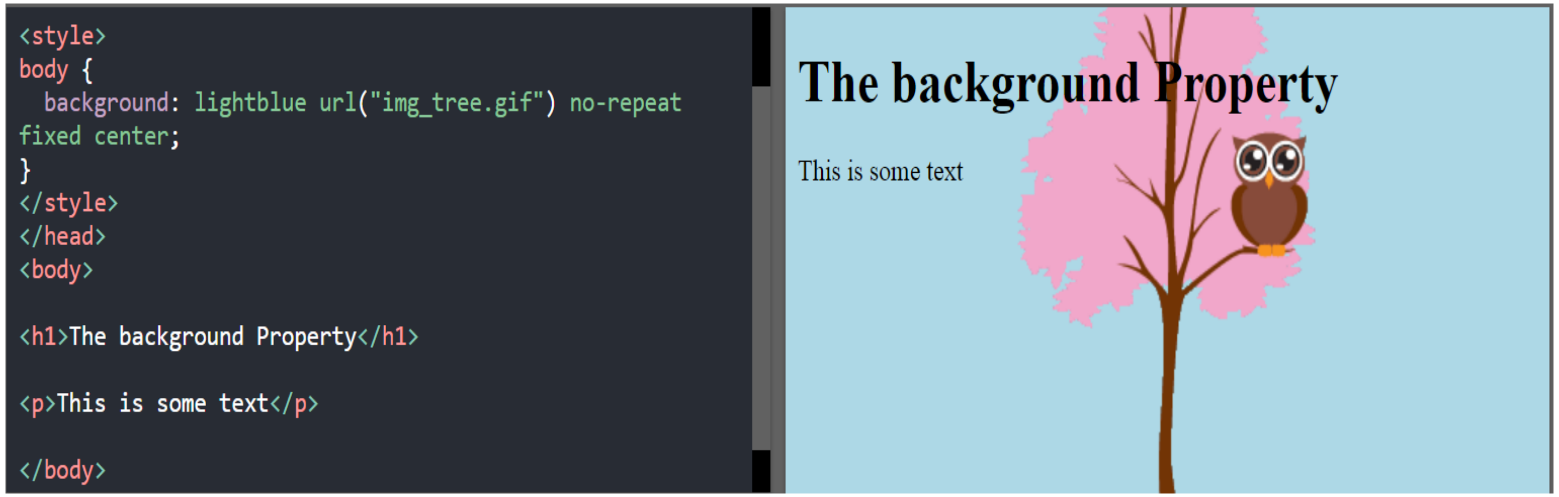
CSS -Text

Property	Value	Meaning	Examples
color	name/HEXcode/RGBcode, …	Coulor	color: lightblue; color: #550047; color: rgb(255,0,0);
font-family	List of font names	Font	font-family: Times; font-family: Tahoma, Times, Arial;
font-size	Value (px/pt/cm/ …)/small/large/ …	Size	font-size: large; font-size: 15px; font-size: 120%;
font-style	italic/oblique/normal	Italic or not	font-style: italic;
font-weight	bold/normal/bolder/lighter/ …	Bold or not	font-weight: bold;
list-style	Type Position Image	Item list style	list-style: square inside url(…); list-style: decimal outside;
direction	rtl/ltr	Direction	direction: ltr;

CSS – Background

Property	Value	Meaning	Examples
background-color	name/HEXcode/RGBcode, …	Color	…: gray; …: #550047; …: rgb(90,0,0);
background-image	url()/linear-gradient()/ …	Image	…: url("paper.gif"); …:linear-gradient(to left,red,blue);
background-repeat	repeat/repeat-x/repeat-y/no-repeat	Image repeat	…: repeat-y;
background-attachment	scroll/fixed/ …	Fixed or scrolled image	…: scroll;
background-position	x y (left,right,center,top,bottom,px,%)	Initial position of the image	…: left top; …: 90px 20px; …: 20% 50%;
background	Spaced Property Values	All background properties	…: blue url("tree.gif") no-repeat fixed center;
opacity	Value (between 0 & 1)	Opacity	…: 0.2;

- Use online tools to choose attractive color palettes like: Coolors, AdobeColor, HueSnap, ColorSafe, etc.



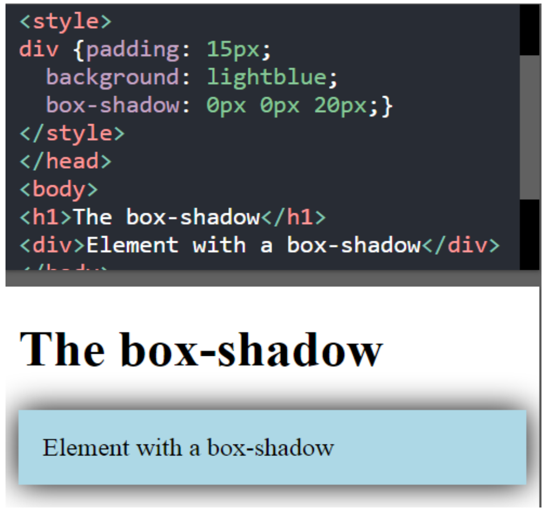
```
<style>
body {
  background: lightblue url("img_tree.gif") no-repeat
fixed center;
}
</style>
</head>
<body>

<h1>The background Property</h1>

<p>This is some text</p>

</body>
```

The background Property

This is some text

CSS – Bordures

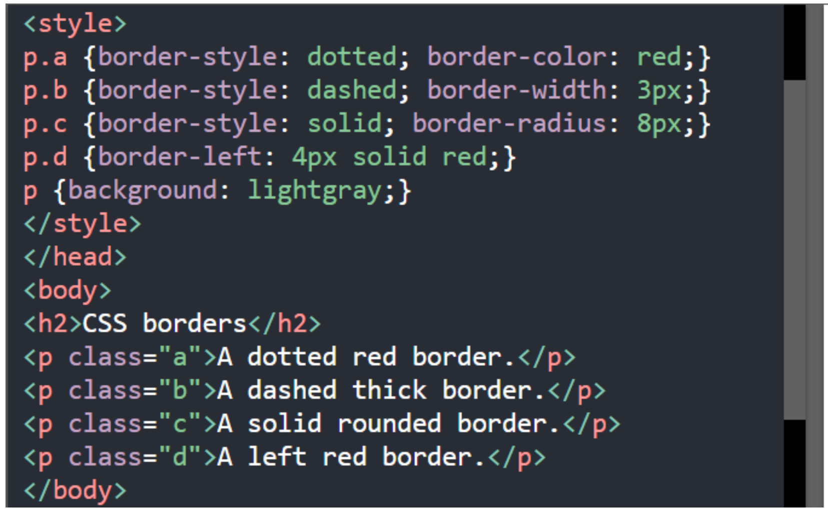
Property	Value	Meaning	Examples
border-color	name/HEXcode/RGBcode, …	Color	…: gray; …: #550047; …: rgb(90,0,0);
border-width	Value (px, pt, cm, …)	Width	…: 3px;
border-style	solid/dotted/double/none/ …	Style	…: double;
border	Spaced property values	All previous properties	…: 5px solid red;
border-radius	Value (px, pt, cm, …)	Rounded corner effect	…: 5px; …: 5%;
border-collapse	collapse/separate	Table Neighboring Borders collapse	…: collapse; …: separate;
box-shadow	offset-hor offset-ver Degree Colr	Shadow defined by 4 parameters	…: 10px 10px 5px green;



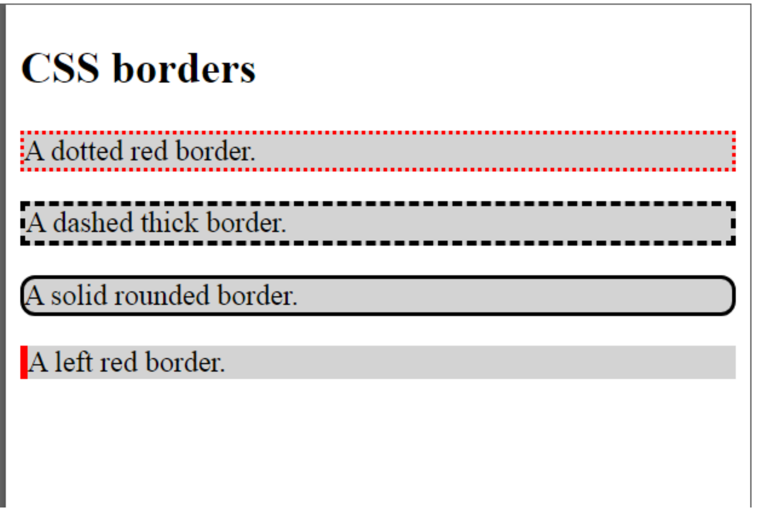
```
<style>
div {padding: 15px;
  background: lightblue;
  box-shadow: 0px 0px 20px;}
</style>
</head>
<body>
<h1>The box-shadow</h1>
<div>Element with a box-shadow</div>
```

The box-shadow

Element with a box-shadow



```
<style>
p.a {border-style: dotted; border-color: red;}
p.b {border-style: dashed; border-width: 3px;}
p.c {border-style: solid; border-radius: 8px;}
p.d {border-left: 4px solid red;}
p {background: lightgray;}
</style>
</head>
<body>
<h2>CSS borders</h2>
<p class="a">A dotted red border.</p>
<p class="b">A dashed thick border.</p>
<p class="c">A solid rounded border.</p>
<p class="d">A left red border.</p>
</body>
```



CSS borders

A dotted red border.

A dashed thick border.

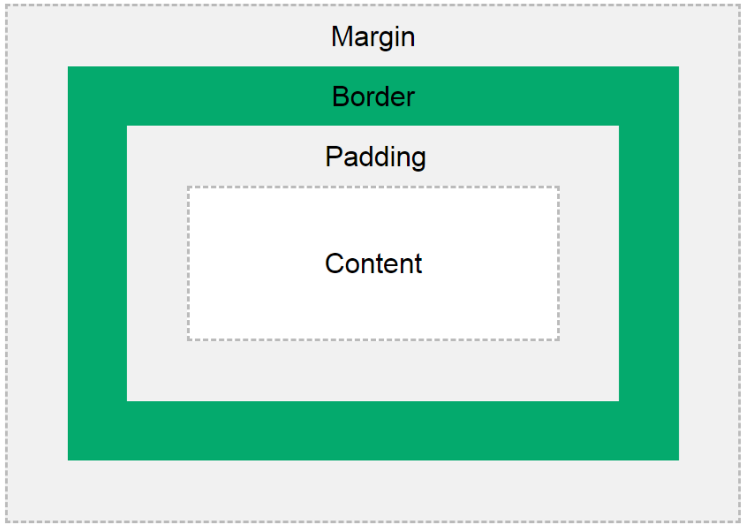
A solid rounded border.

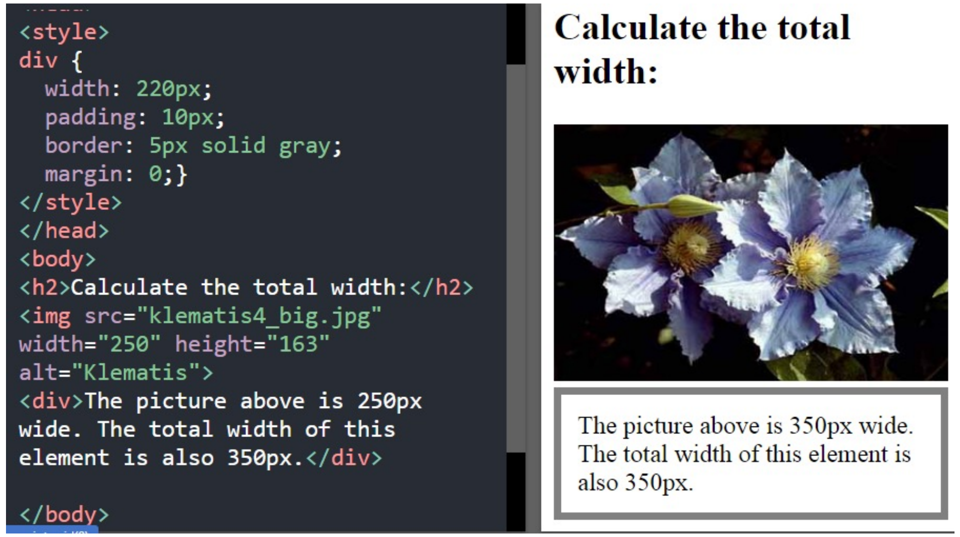
A left red border.

CSS – Box model

Box-model? It's a box wrapping around a tag, consisting of its content size (width, height), its padding, its border (border-width), and its margin.

Propriété	Valeur	Signification	Exemples
margin	Valeur (px, pt, cm, %, auto, …)	Marge intérieure (bordure)	margin: 5% 5% 5% auto;
padding	Valeur (px, pt, cm, %, auto, …)	Marge extérieure (bordure)	padding: 20px 10%; padding: 5px;
width	Valeur (px, pt, cm, %, auto, …)	Largeur (sans marges int/ext et bordure)	width: 12%;
height	Valeur (px, pt, cm, %, auto, …)	Longueur (sans marges int/ext et bordure)	height: 15px;





```
<style>
div {
  width: 220px;
  padding: 10px;
  border: 5px solid gray;
  margin: 0;}
</style>
</head>
<body>
<h2>Calculate the total width:</h2>
<img src="klematis4_big.jpg"
width="250" height="163"
alt="Klematis">
<div>The picture above is 250px
wide. The total width of this
element is also 350px.</div>

</body>
```

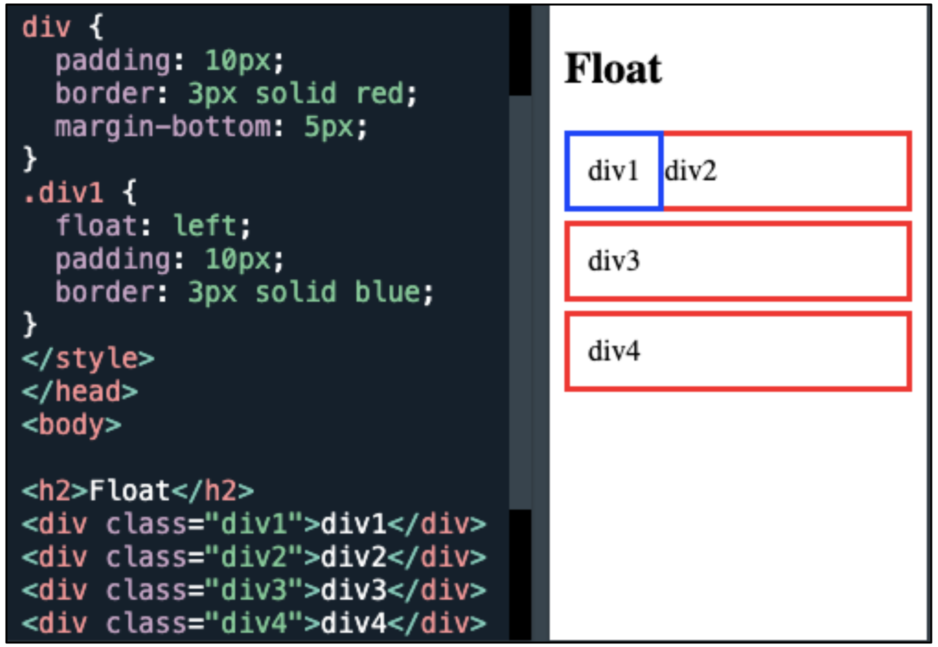
Calculate the total width:

The picture above is 350px wide. The total width of this element is also 350px.

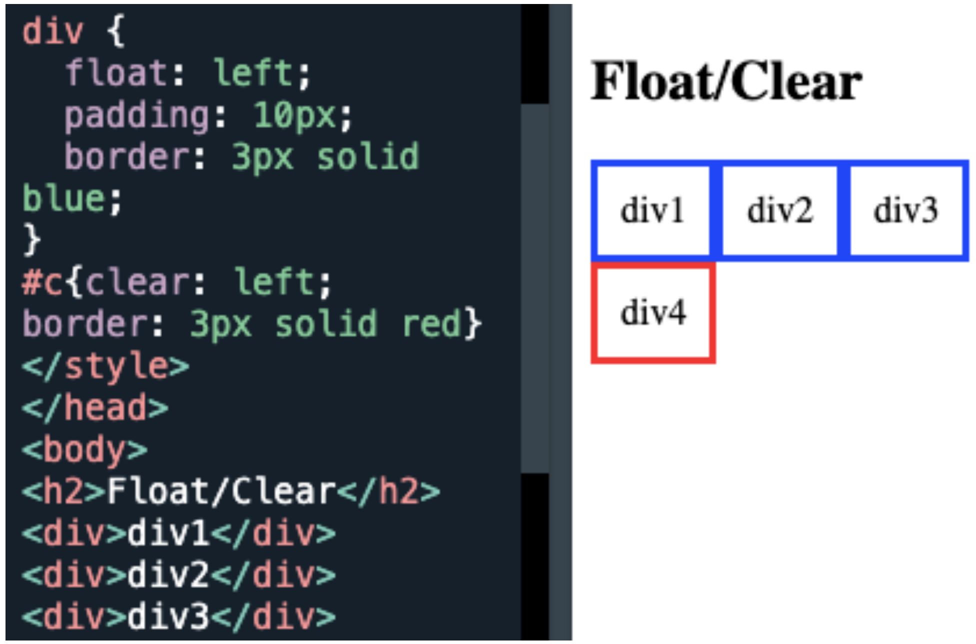
CSS – Positioning(Float)

To control the position, CSS offers several solutions:

- float: defines how an element can float within a container (right, left, none).
  - By default, the element following the floating element sticks (to the right/left). To avoid this, we use the 'clear' property with its values: right, left, both.

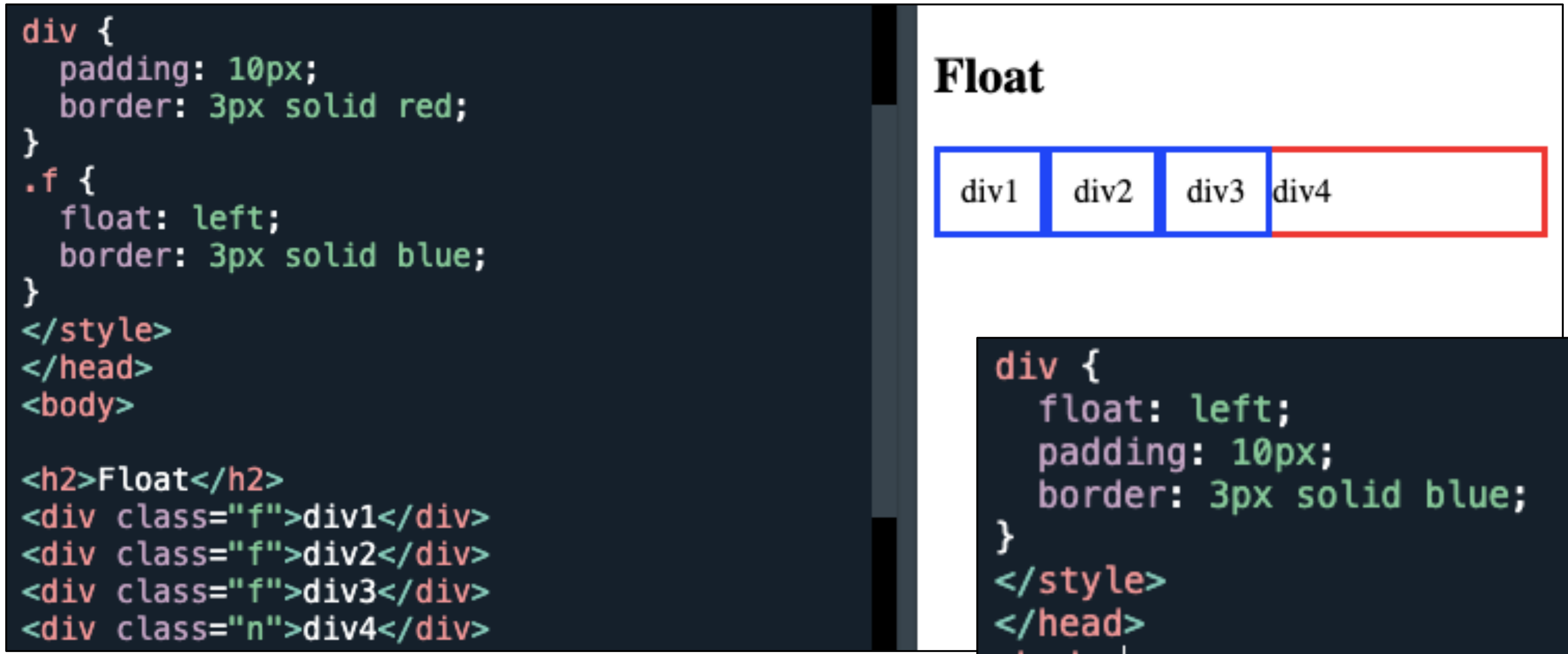


```
div {
  padding: 10px;
  border: 3px solid red;
  margin-bottom: 5px;
}
.div1 {
  float: left;
  padding: 10px;
  border: 3px solid blue;
}
</style>
</head>
<body>

<h2>Float</h2>
<div class="div1">div1</div>
<div class="div2">div2</div>
<div class="div3">div3</div>
<div class="div4">div4</div>
```



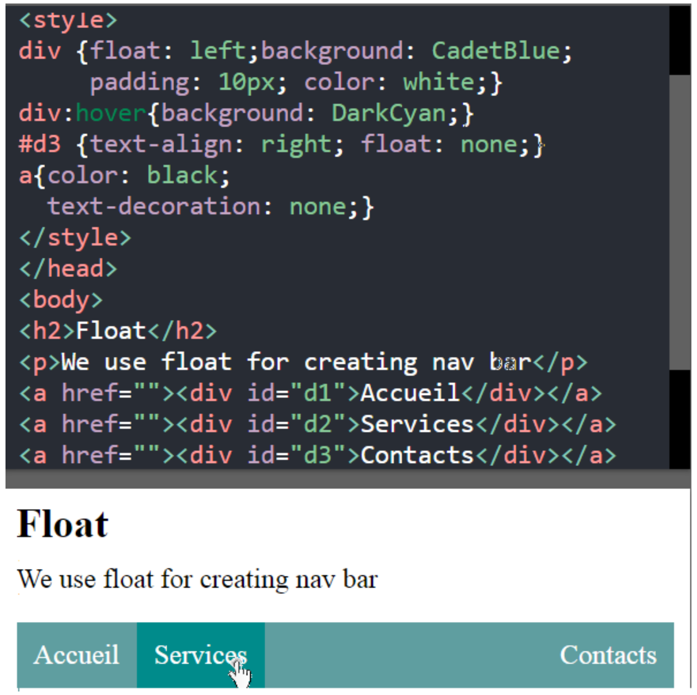
```
div {
  float: left;
  padding: 10px;
  border: 3px solid
blue;
}
#c{clear: left;
border: 3px solid red}
</style>
</head>
<body>
<h2>Float/Clear</h2>
<div>div1</div>
<div>div2</div>
<div>div3</div>
```

Float - Examples



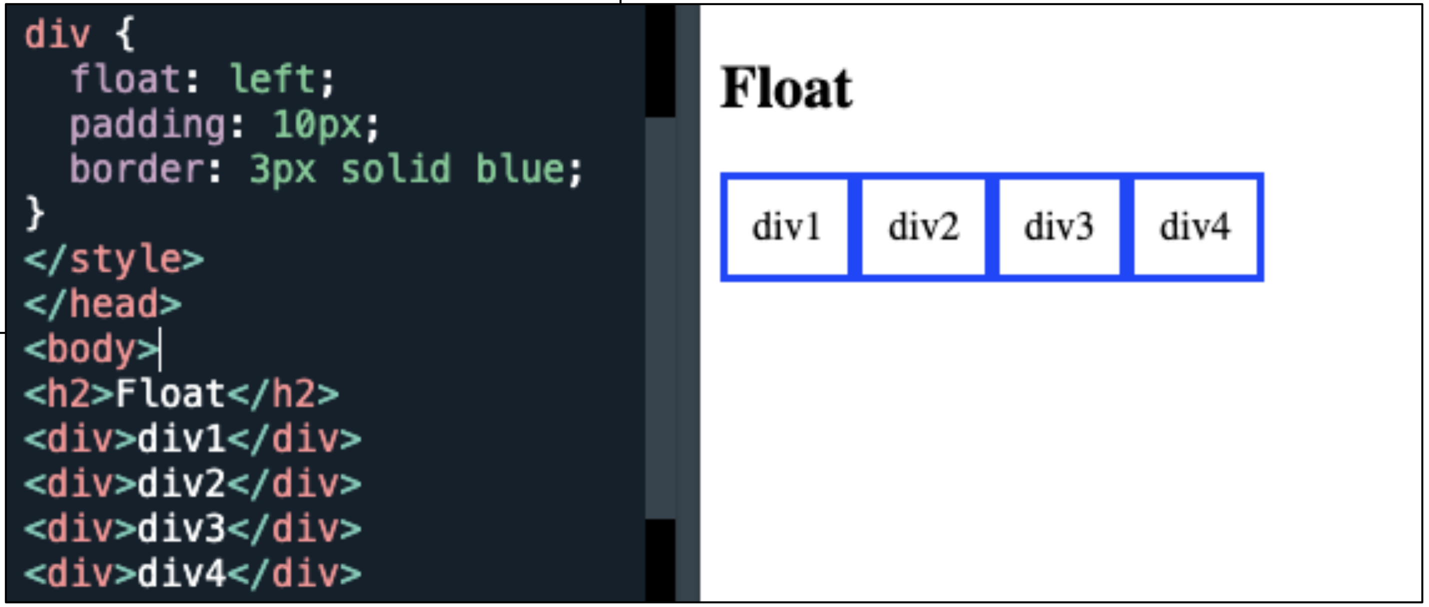
```
div {
  padding: 10px;
  border: 3px solid red;
}
.f {
  float: left;
  border: 3px solid blue;
}
</style>
</head>
<body>

<h2>Float</h2>
<div class="f">div1</div>
<div class="f">div2</div>
<div class="f">div3</div>
<div class="n">div4</div>
```

```
Float

div1 div2 div3 div4
```



```
<style>
div {float: left;background: CadetBlue;
     padding: 10px; color: white;}
div:hover{background: DarkCyan;}
#d3 {text-align: right; float: none;}
a{color: black;
  text-decoration: none;}
</style>
</head>
<body>
<h2>Float</h2>
<p>We use float for creating nav bar</p>
<a href=""><div id="d1">Accueil</div></a>
<a href=""><div id="d2">Services</div></a>
<a href=""><div id="d3">Contacts</div></a>
```

```
Float

We use float for creating nav bar

Accueil Services                    Contacts
```

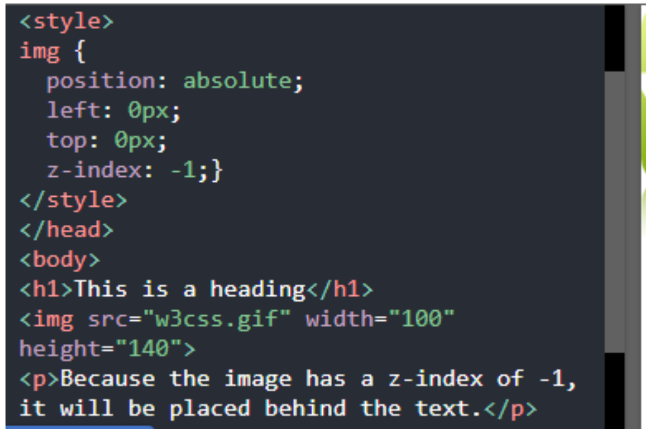


```
div {
  float: left;
  padding: 10px;
  border: 3px solid blue;
}
</style>
</head>
<body>
<h2>Float</h2>
<div>div1</div>
<div>div2</div>
<div>div3</div>
<div>div4</div>
```

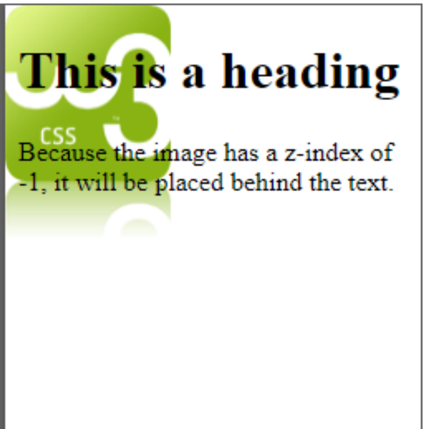
```
Float

div1 div2 div3 div4
```

CSS – Positioning(Position)

- Position: defines the type of positioning to use. This property requires the position properties: **top, bottom, right, left** (value in px, cm, %, ...) which behave relative to the position value:
  - static (default): position according to the normal flow of the page (**top**, **left**, ... have no effect)
  - relative: position relative to its normal position
  - absolute: position relative to the first positioned parent element (or to the body if none)
  - fixed: position relative to the page's window (fixed element)
  - sticky: position **relative** until the element scrolls out of view (top = 0, bottom = 0), then becomes **fixed**
  - z-index: a property defining the stacking order in case of overlapping



```
<style>
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;}
</style>
</head>
<body>
<h1>This is a heading</h1>
<img src="w3css.gif" width="100"
height="140">
<p>Because the image has a z-index of -1,
it will be placed behind the text.</p>
```

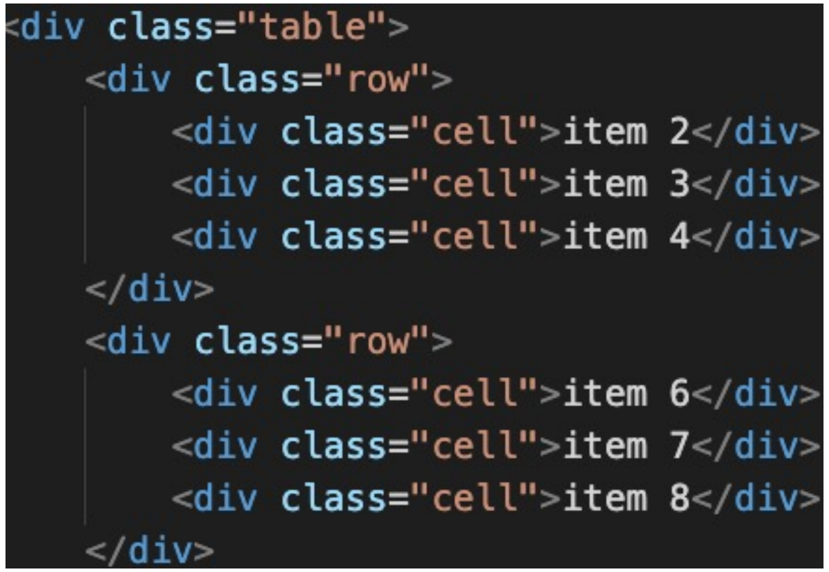


This is a heading

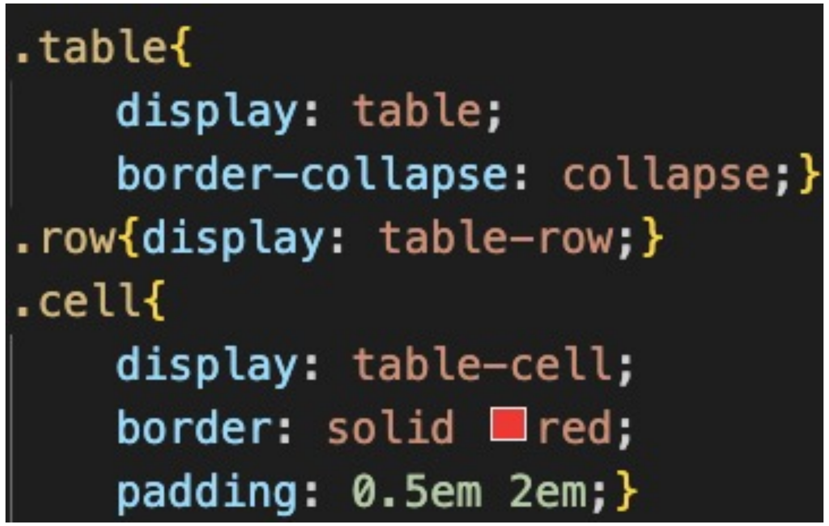
Because the image has a z-index of -1, it will be placed behind the text.

CSS – Positioning(display)

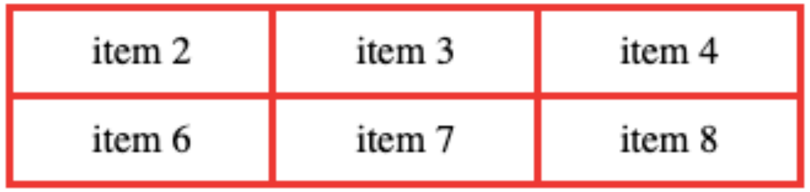
- Display: defines how an element is displayed (inline, block, none, table, ...)
  - inline: no new line break and size to fit content
  - block: new line break and takes up full width
  - inline-block: inline but resizable
  - none: removed
  - table: behaves like a <table> element (use display: table-row/table-cell)



```html
<div class="table">
    <div class="row">
        <div class="cell">item 2</div>
        <div class="cell">item 3</div>
        <div class="cell">item 4</div>
    </div>
    <div class="row">
        <div class="cell">item 6</div>
        <div class="cell">item 7</div>
        <div class="cell">item 8</div>
    </div>
```



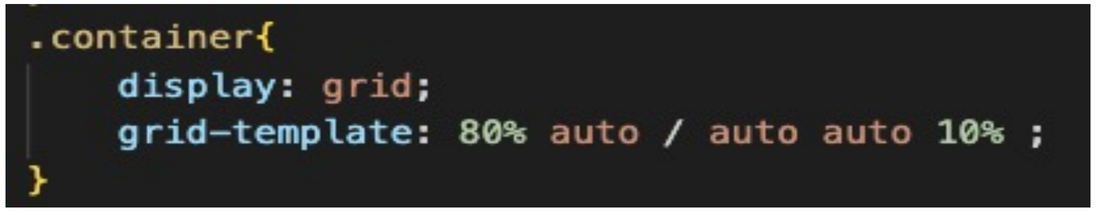
```css
.table{
    display: table;
    border-collapse: collapse;}
.row{display: table-row;}
.cell{
    display: table-cell;
    border: solid 🟥red;
    padding: 0.5em 2em;}
```



item 2	item 3	item 4
item 6	item 7	item 8

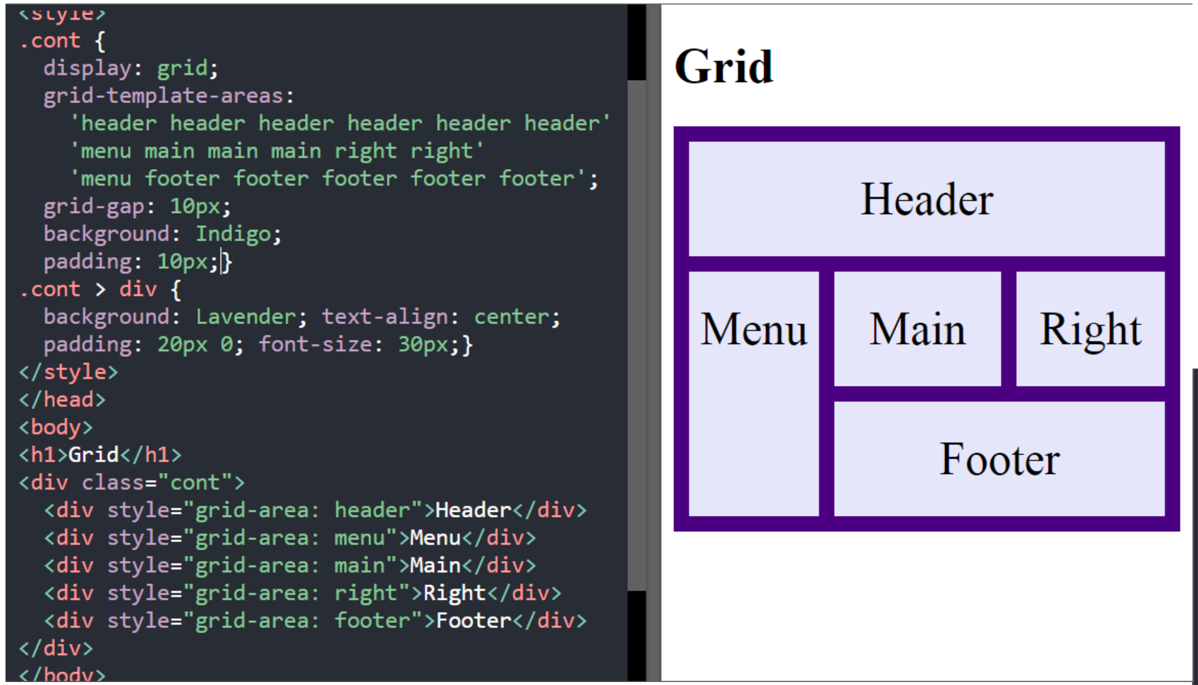
CSS – Positioning(display)

- Display: defines how an element is displayed (inline, block, none, table, ...)
  - flex: behaves as a flexible container (**uni-directional**)
    - **Container**: in addition to display: **flex**, other properties can be used: flex-direction (row/col), justify-content (align main axis), align-content (align cross axis), align-items (aligt all items), ...
    - **Items**: several properties can be used: flex (grow, shrink, basis), order, align-self (one item)...
  - grid: behaves as a grid container (**bi-directional**)
    - **Container**: in addition to display: **grid**, other properties can be used: grid-template, justify-content (aligt along the main axis), align-content (aligt along the cross axis), gap (row/col)...
    - **Items**: several properties can be used: grid-area (name or rowS/colS/rowE/colE), ...

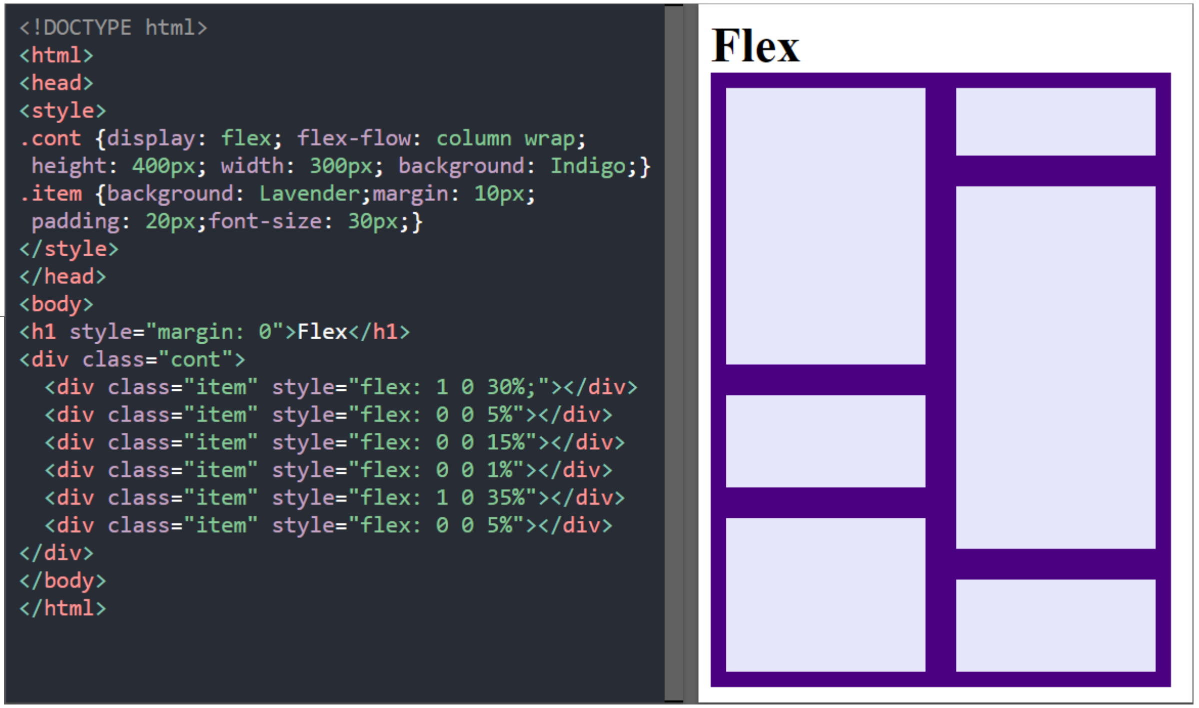


```
.container{
    display: grid;
    grid-template: 80% auto / auto auto 10% ;
}
```

CSS – Positioning(display)



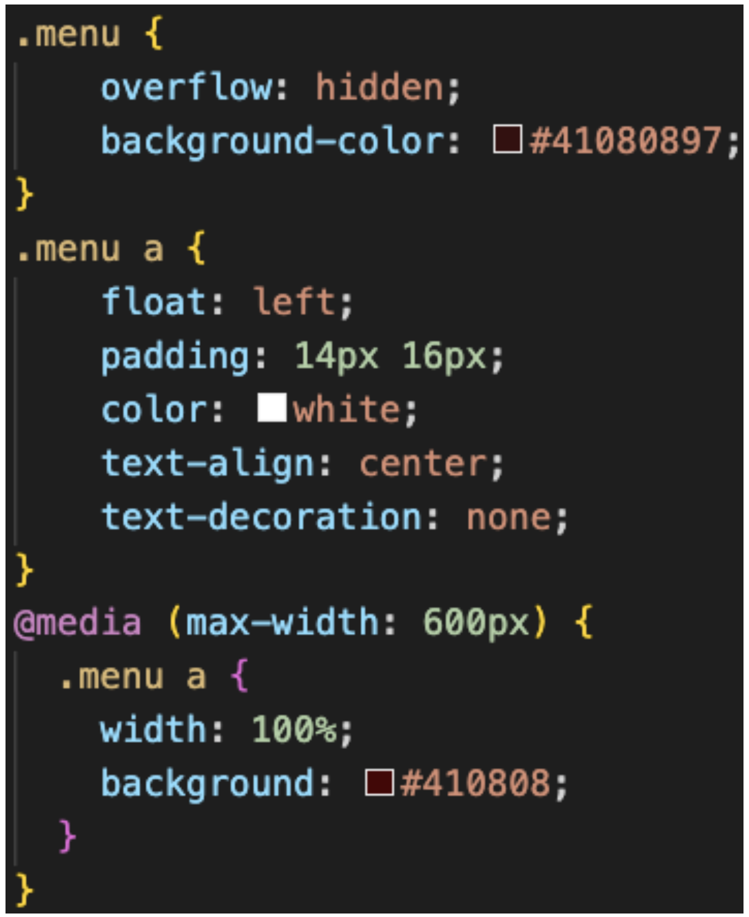
```
<style>
.cont {
  display: grid;
  grid-template-areas:
    'header header header header header header'
    'menu main main main right right'
    'menu footer footer footer footer footer';
  grid-gap: 10px;
  background: Indigo;
  padding: 10px;}
.cont > div {
  background: Lavender; text-align: center;
  padding: 20px 0; font-size: 30px;}
</style>
</head>
<body>
<h1>Grid</h1>
<div class="cont">
  <div style="grid-area: header">Header</div>
  <div style="grid-area: menu">Menu</div>
  <div style="grid-area: main">Main</div>
  <div style="grid-area: right">Right</div>
  <div style="grid-area: footer">Footer</div>
</div>
</body>
```



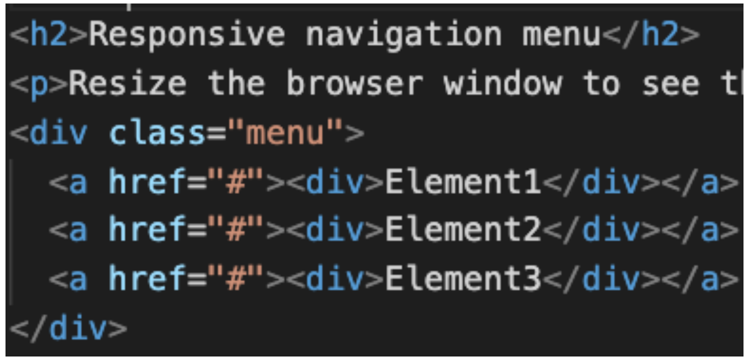
```
<!DOCTYPE html>
<html>
<head>
<style>
.cont {display: flex; flex-flow: column wrap;
 height: 400px; width: 300px; background: Indigo;}
.item {background: Lavender;margin: 10px;
 padding: 20px;font-size: 30px;}
</style>
</head>
<body>
<h1 style="margin: 0">Flex</h1>
<div class="cont">
  <div class="item" style="flex: 1 0 30%;"></div>
  <div class="item" style="flex: 0 0 5%"></div>
  <div class="item" style="flex: 0 0 15%"></div>
  <div class="item" style="flex: 0 0 1%"></div>
  <div class="item" style="flex: 1 0 35%"></div>
  <div class="item" style="flex: 0 0 5%"></div>
</div>
</body>
</html>
```

CSS – Media queries

- @media rules are making possible to define a tailored styles for different media types (laptops, tablets, phones, …)
- Syntax (adapted):
  @media [**not**|**only** *mediatype* and] (*mediafeature*:value)$^+${
    CSS-Code;
  }
  - Mediatypes: all, print, screen
  - Mediafeature: orientation, width, height, min/max-width, min/max-height



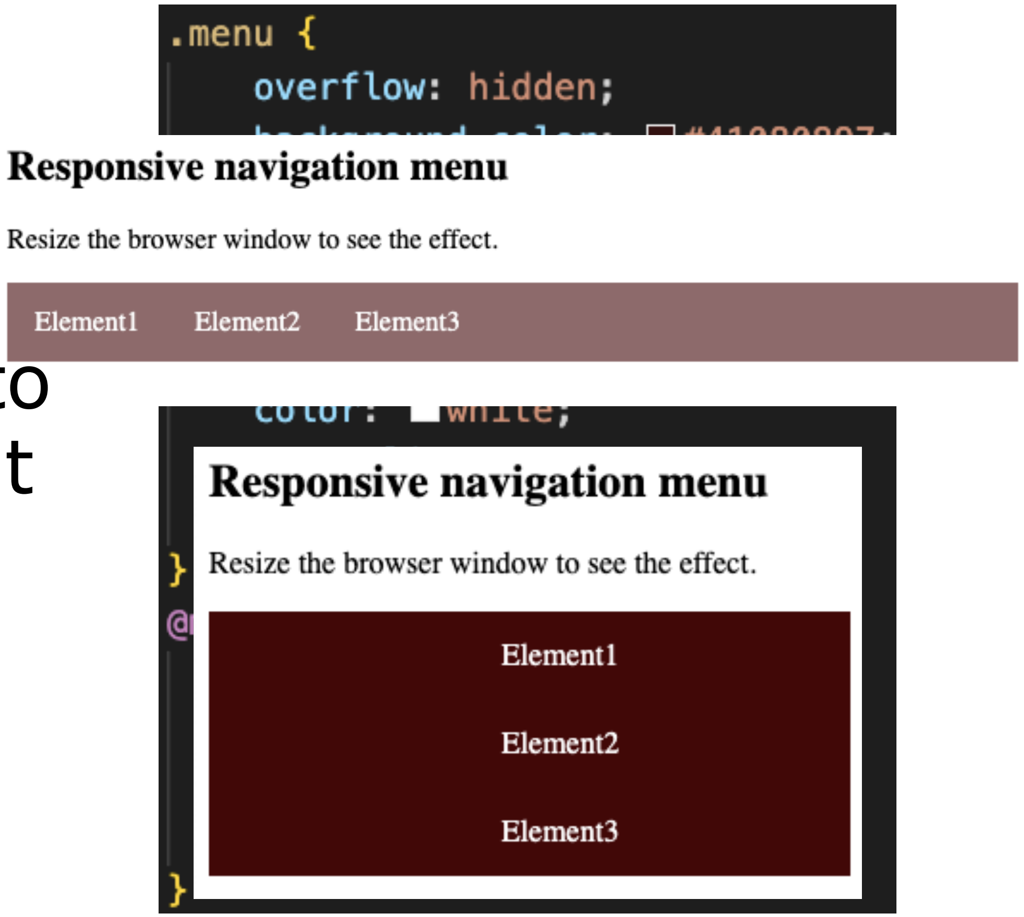
```css
.menu {
    overflow: hidden;
    background-color: #41080897;
}
.menu a {
    float: left;
    padding: 14px 16px;
    color: white;
    text-align: center;
    text-decoration: none;
}
@media (max-width: 600px) {
  .menu a {
    width: 100%;
    background: #410808;
  }
}
```



```html
<h2>Responsive navigation menu</h2>
<p>Resize the browser window to see t
<div class="menu">
  <a href="#"><div>Element1</div></a>
  <a href="#"><div>Element2</div></a>
  <a href="#"><div>Element3</div></a>
</div>
```

CSS – Media queries

- @media rules are making possible to define a tailored styles for different media types (laptops, tablets, phones, …)
- Examples:
  - @media only screen and (max-width:600px){body{display:flex;}}
  - @media screen and (max-width:900px and min-width){h3{font-size:20px}}
  - @media screen and (orientation:portrait){div{float:non;}}



```
.menu {
    overflow: hidden;
```

Responsive navigation menu

Resize the browser window to see the effect.

Element1 Element2 Element3

```
    color: white;
```

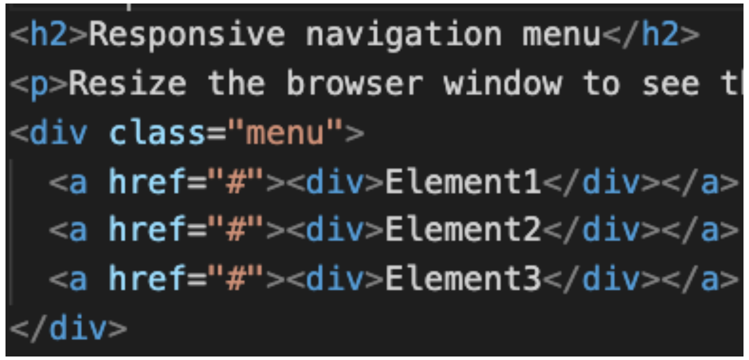
Responsive navigation menu

Resize the browser window to see the effect.

Element1

Element2

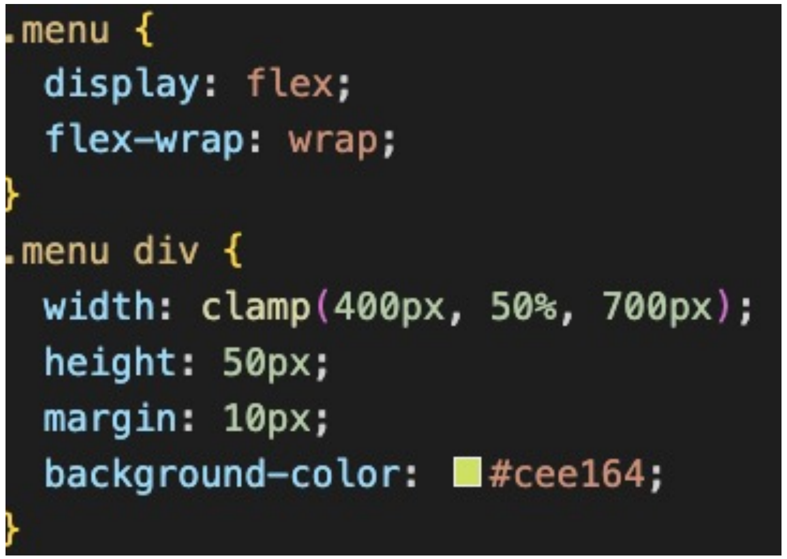
Element3



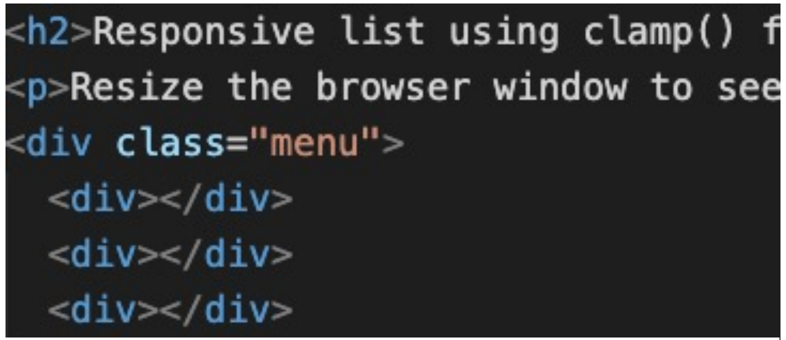
```
<h2>Responsive navigation menu</h2>
<p>Resize the browser window to see t
<div class="menu">
  <a href="#"><div>Element1</div></a>
  <a href="#"><div>Element2</div></a>
  <a href="#"><div>Element3</div></a>
</div>
```

CSS – Clamp() function

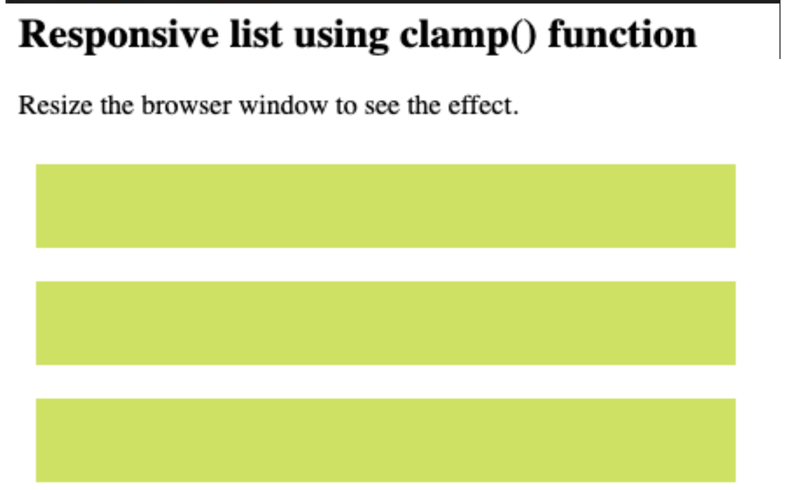
- **clamp()** function is used within CSS rules to define a flexible range for a particular property, ensuring that it adapts smoothly within the specified limits
- Syntaxe:
  cssProperty: **clamp**(minVal, preferedVal, maxVal);
  - width: clamp(100px, 50%, 200px);
  - font-size: clamp(16px, 2vw, 24px);
  - margin-left: clamp(200px, 20vw, 300px);



```
.menu {
  display: flex;
  flex-wrap: wrap;
}
.menu div {
  width: clamp(400px, 50%, 700px);
  height: 50px;
  margin: 10px;
  background-color: 🟩#cee164;
}
```



```
<h2>Responsive list using clamp() f
<p>Resize the browser window to see
<div class="menu">
  <div></div>
  <div></div>
  <div></div>
```

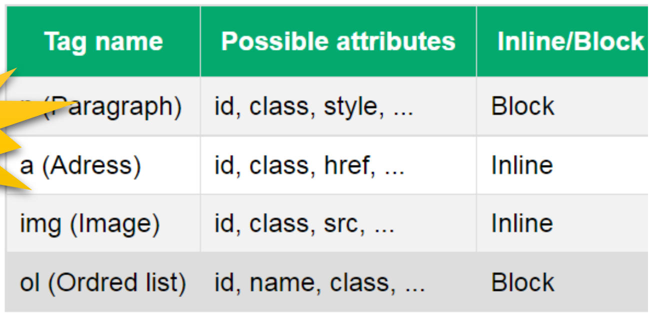


Responsive list using clamp() function

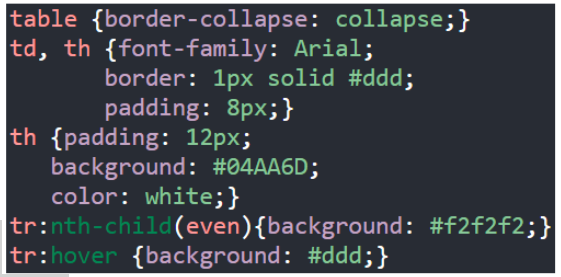
Resize the browser window to see the effect.

CSS – Examples

Table vivante

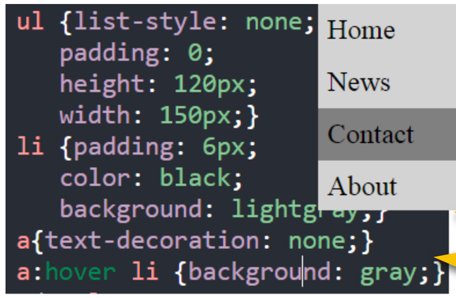


Tag name	Possible attributes	Inline/Block
p (Paragraph)	id, class, style, ...	Block
a (Adress)	id, class, href, ...	Inline
img (Image)	id, class, src, ...	Inline
ol (Ordred list)	id, name, class, ...	Block

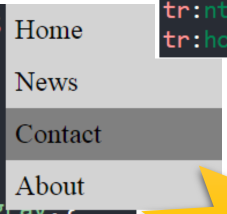


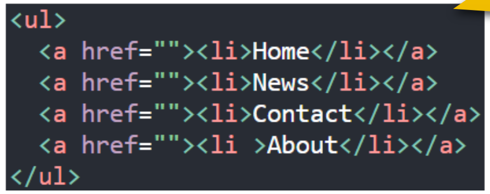
```
table {border-collapse: collapse;}
td, th {font-family: Arial;
       border: 1px solid #ddd;
       padding: 8px;}
th {padding: 12px;
   background: #04AA6D;
   color: white;}
tr:nth-child(even){background: #f2f2f2;}
tr:hover {background: #ddd;}
```

Barre de menu verticale



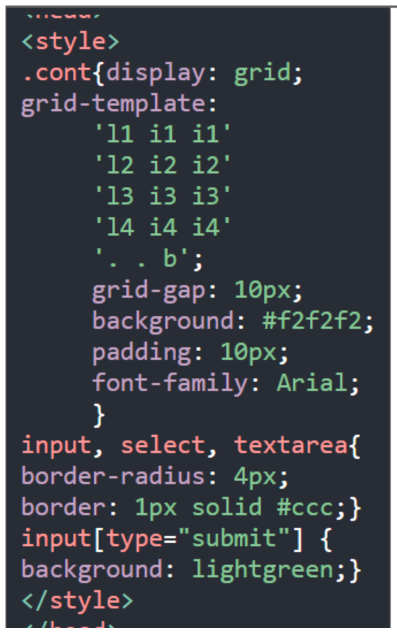
```
ul {list-style: none;
   padding: 0;
   height: 120px;
   width: 150px;}
li {padding: 6px;
   color: black;
   background: lightgray;}
a{text-decoration: none;}
a:hover li {background: gray;}
```



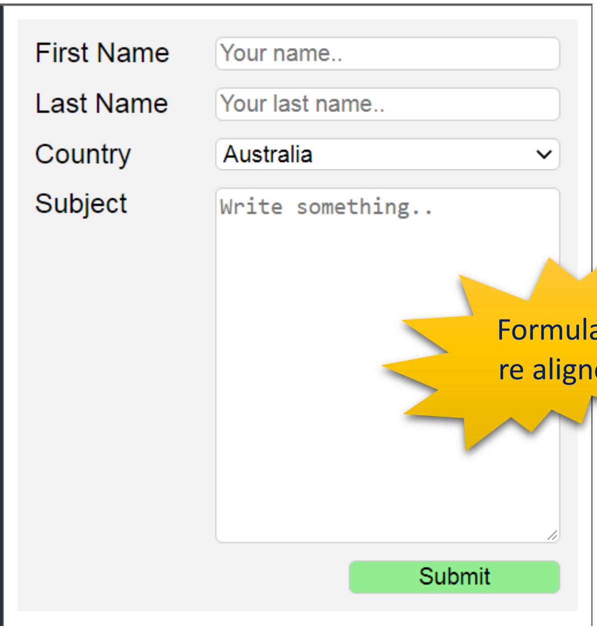


```
<ul>
  <a href=""><li>Home</li></a>
  <a href=""><li>News</li></a>
  <a href=""><li>Contact</li></a>
  <a href=""><li >About</li></a>
</ul>
```

Formulaire aligné



```
<style>
.cont{display: grid;
grid-template:
     'l1 i1 i1'
     'l2 i2 i2'
     'l3 i3 i3'
     'l4 i4 i4'
     '. . b';
     grid-gap: 10px;
     background: #f2f2f2;
     padding: 10px;
     font-family: Arial;
     }
input, select, textarea{
border-radius: 4px;
border: 1px solid #ccc;}
input[type="submit"] {
background: lightgreen;}
</style>
```



First Name	Your name..
Last Name	Your last name..
Country	Australia
Subject	Write something..

CSS – Responsive web design

- Web pages can be accessed using computers, tablets, and phones. They should be enjoyable and easy to use, regardless of the used device!
- Responsive web design (RWD)? set of techniques for adapting web pages to different devices

```css
img {max-width: 100%;height: auto;}
```

  - Assign relative dimensions (%) that adapt to the device window size
  - Instruct the browser to control the page dimensions and scaling via the viewport meta tag

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

  - Use Media Queries (rule: conditionally applied properties) to assign different styles to different devices and dimensions

```css
@media only screen and (max-width: 600px) {
  body {background-color: lightblue;}}
@media only screen and (min-width: 600px) {
  body {background-color: lightgreen;}}
```

  - Utilize flexible layouts (Flex, Grid, ...)
  - Utilize responsive frameworks (W3.CSS, Bootstrap, ...)