

# CSS Essential

*Web application with HTML and CSS*



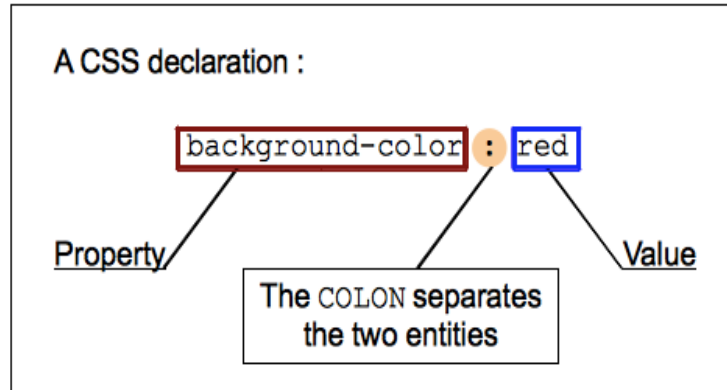
# Lesson Objectives

- Value and Unit
- Box model
- Overflow Content
- Styling Image
- Styling List

## Section 1

# VALUES AND UNITS

- **CSS value:** Every CSS declaration includes a property / value pair. Depending on the property, the value can include a single integer or keyword, to a series of keywords and values with or without units



## ➤ Data type

- ***Textual data type:***
  - Pre-defined keywords as an ident:

```
.box {  
    float: left;  
}
```

## ➤ Data type

### ▪ *Textual data type:*

#### ○ string:

```
1 | .item {  
2 |     grid-area: content;  
3 | }
```

#### ○ url:

```
.box {  
    background-image: url("images/my-background.png");  
}
```

## ➤ Data type

### ▪ *Numeric data types*

- *Integer*: A whole number such as 1024 or -55
- *Number*: A decimal number
- *Dimension*: A number with a unit attached to it
- *Percentage*: A fraction of some other value

➤ **Distance unit (length unit):** There are 2 types of **distance** unit in CSS: Relative and Absolute

❖ ***Absolute length units*** are fixed to a physical length

Unit	Description
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px *	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)



## ➤ Distance unit (length unit):

❖ ***Relative length unit*** specify a length in relation to something else.

Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element

- **Percentage** is a type that represents a fraction of some other value.
- It is always relative to another quantity.
- Each property that allows **percentages** also defines the quantity to which the percentage refers.
- This quantity can be a value of another property of the same element, the value of a property of an ancestor element, a measurement of a containing block, or something else.

- **em** and **rem** are the two relative lengths you are likely to encounter most frequently when sizing anything from boxes to text.
  - **em** unit means *"my parent element's font-size"*: Relative to font size of the parent, in the case of typographical properties like font-size, and font size of the element itself, in the case of other properties like width.
  - **rem** unit means *"The root element's font-size"*: Relative to font size of the root element.

➤ **Colors** in CSS can be specified by the following methods:

- RGB colors
- RGBA colors
- Hexadecimal colors
- HSL colors
- HSLA colors
- Predefined/Cross-browser color names
- With the **currentcolor** keyword

- An **RGB color** value is specified with the `rgb()` function, which has the following syntax: ***rgb(red, green, blue)***
- Each parameter (red, green, and blue) defines the intensity of the color and can be an integer between 0 and 255 or a percentage value (from

```
.one {  
  background-color: rgb(2, 121, 139);  
}  
  
.two {  
  background-color: rgb(197, 93, 161);  
}  
  
.three {  
  background-color: rgb(18, 138, 125);  
}
```

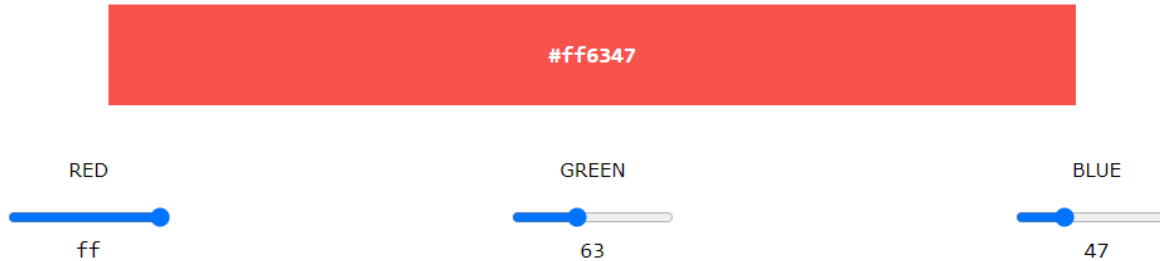
rgb(2, 121, 139)

rgb(197, 93, 161)

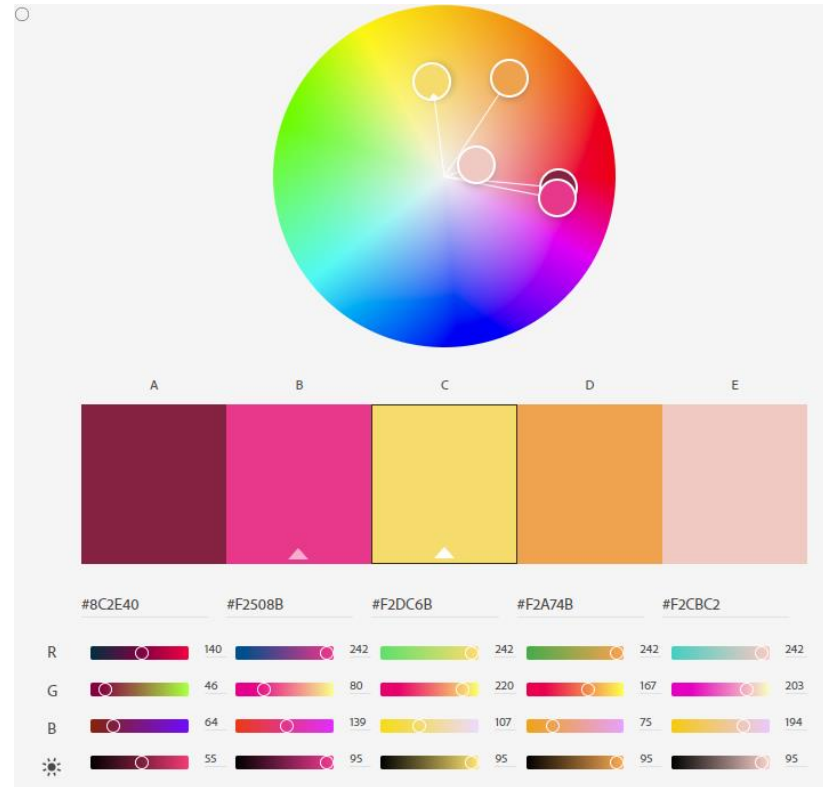
rgb(18, 138, 125)

## ■ **#rrggbb:**

- ✓ rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255)



# Hexadecimal colors



- ***Predefined/Cross-browser color names:*** 140 color names are predefined in the HTML and CSS color specification.

white	black	cyan	darkCyan
red	darkRed	magenta	darkMagenta
green	darkGreen	yellow	darkYellow
blue	darkBlue	gray	darkGray
lightGray			



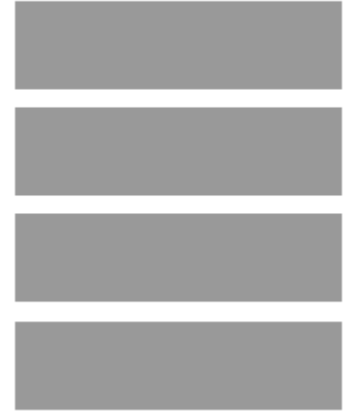
## Section 2

# BOX MODEL

## ➤ **Block box:**

- The box will break onto a new line.
- The width and height properties are respected.
- Padding, margin and border will cause other elements to be pushed away from the box

### **BLOCK:**



## ➤ **Inline box:**

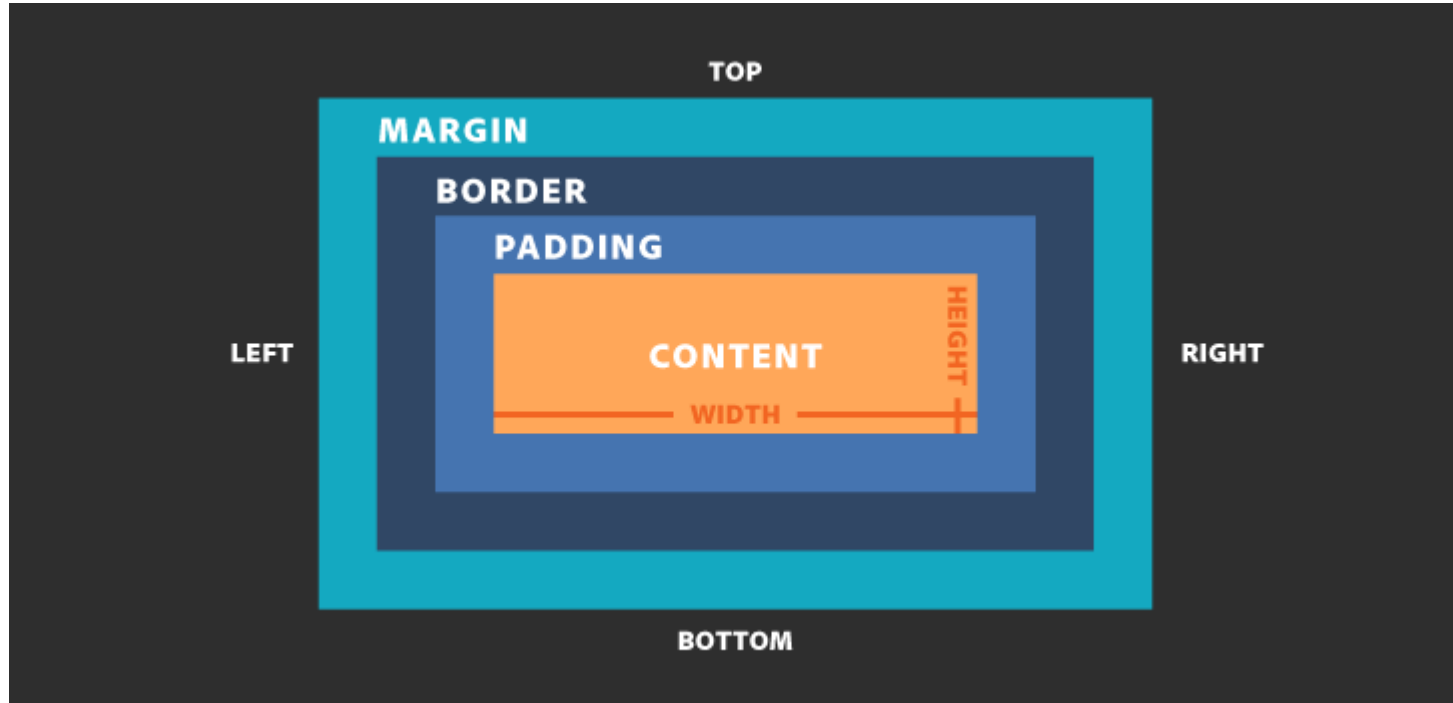
- The box will not break onto a new line.
- The width and height properties will not apply.
- Vertical padding, margins, and borders will apply but will not cause other inline boxes to move away from the box.
- Horizontal padding, margins, and borders will apply and will cause other inline boxes to move away from the box

### **INLINE:**



# The alternative CSS Box Model

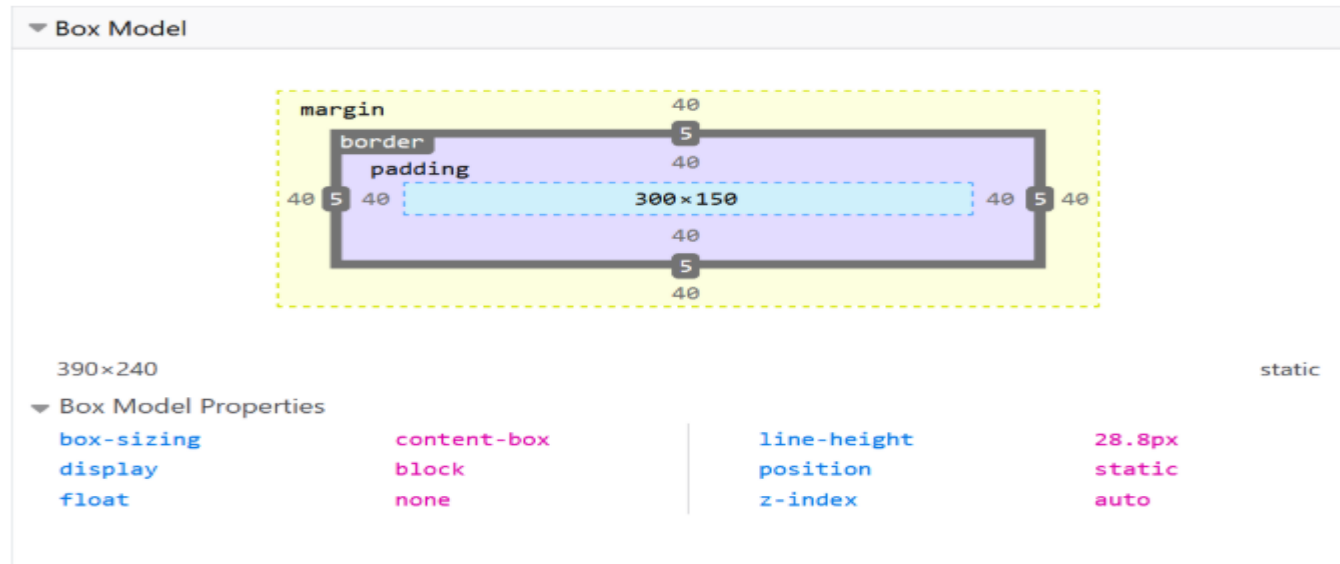
## ➤ Box model



- **Making up a block box in CSS we have:**
  - ***Content box:*** The area where your content is displayed, which can be sized using properties like width and height.
  - ***Padding box:*** The padding sits around the content as white space; its size can be controlled using padding and related properties.
  - ***Border box:*** The border box wraps the content and any padding. Its size and style can be controlled using border and related properties.
  - ***Margin box:*** The margin is the outermost layer, wrapping the content, padding and border as whitespace between this box and other elements. Its size can be controlled using margin and related properties.

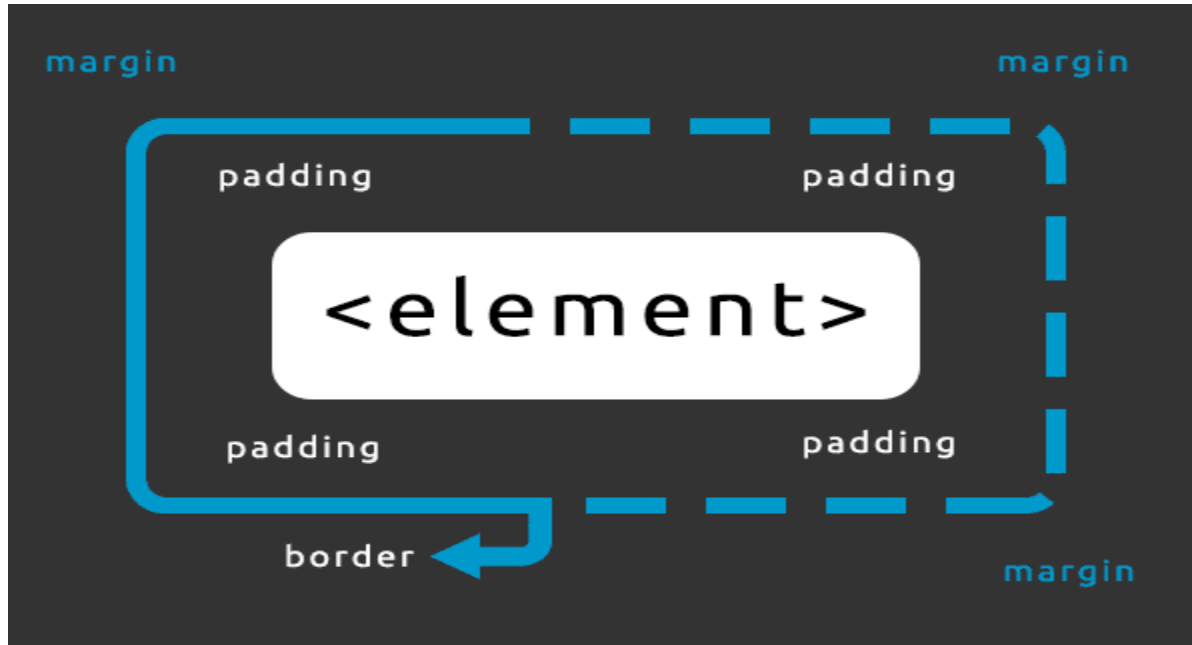
# Use browser DevTools to view the box model

- If you inspect an element in browser DevTools, you can see the size of the element plus its margin, padding, and border. Inspecting an element in this way is a great way to find out if your box is really the size you think it is



# Margins, paddings and borders

## ➤ Take a look



- **Margin:** Is an invisible space around your box. It pushes other elements away from the box. Margins can have positive or negative values. Setting a negative margin on one side of your box can cause it to overlap other things on the page
- We can control all margins of an element at once using the margin property, or each side individually using the equivalent longhand properties:
  - ***margin-top***
  - ***margin-right***
  - ***margin-bottom***
  - ***margin-left***



- **Padding:** Sits between the border and the content area. Unlike margins you cannot have negative amounts of padding, so the value must be 0 or a positive value, it is typically used to push the content away from the border.
- We can control all margins of an element at once using the margin property, or each side individually using the equivalent longhand properties:
  - ***padding-top***
  - ***padding-right***
  - ***padding-bottom***
  - ***padding-left***

- **Border:** is drawn between the margin and the padding of a box.
- To set the properties of each side individually, you can use:
  - ***border-top***
  - ***border-right***
  - ***border-bottom***
  - ***border-left***
- To set the width, style, or color of all sides, use the following:
  - ***border-width***
  - ***border-style*** (*dotted, dashed, solid , double, groove, inset, outset,..* )
  - ***border-color***

- The border-radius property is used to add rounded borders to an element

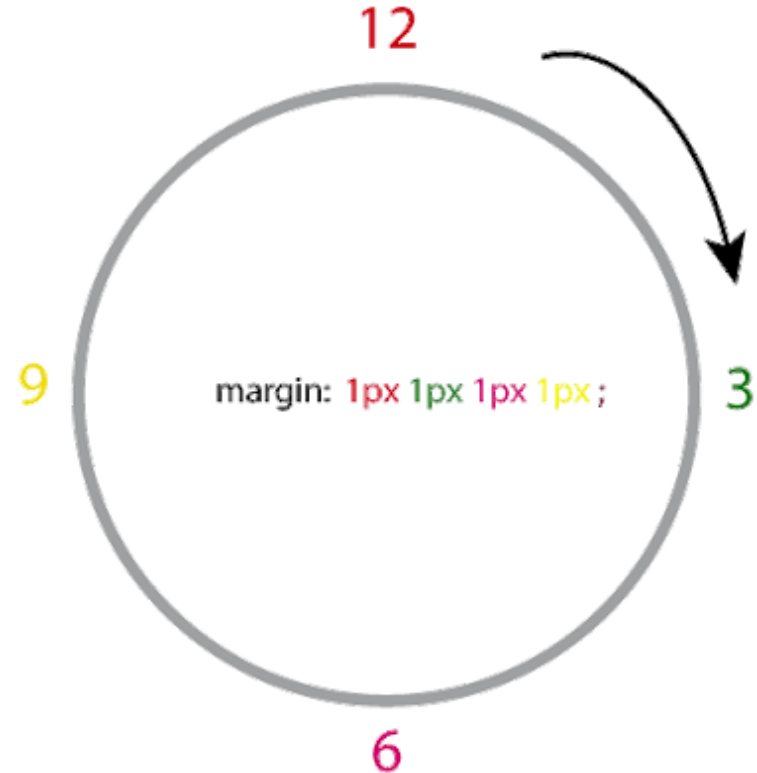
Normal border

Round border

Rounder border

Roudest border

- To shorten the code, it is possible to specify all the margin/ or padding/ or border properties in one property
  - ✓ 1 value: all sides
  - ✓ 2 values: top-bottom, right-left
  - ✓ 4 values: top-right-bottom-left
  - ✓ 3 values: top, right-left, bottom



## Section 3

# OVERFLOW CONTENT

# What is overflow?

- **Overflow:** Is a css property sets what to do when an element's content is too big to fit in its block formatting context.
- This demo below is **default value** of overflow property:

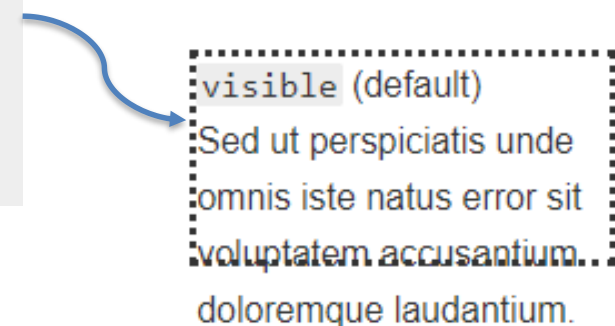
```
overflow: visible;
```

Michaelmas term lately over,  
and the Lord Chancellor  
sitting in Lincoln's Inn Hall.  
Implacable November  
weather. As much mud in the  
streets as if the waters had  
but newly retired from the  
face of the earth.

## ➤ Value:

- **visible** (default): Content is not clipped and may be rendered outside the padding box.

```
1 p {  
2   width: 12em;  
3   height: 6em;  
4   border: dotted;  
5   overflow: visible; /* content is not clipped */  
6 }
```

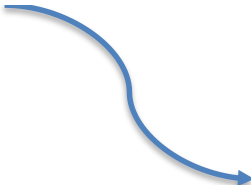


visible (default)  
Sed ut perspiciatis unde  
omnis iste natus error sit  
voluptatem accusantium  
doloremque laudantium.

## ➤ Value:

- ***hidden***: Content is clipped if necessary to fit the padding box. No scrollbars are provided, and no support for allowing the user to scroll

```
1 | p { overflow: hidden; /* no scrollbars are provided */ }
```



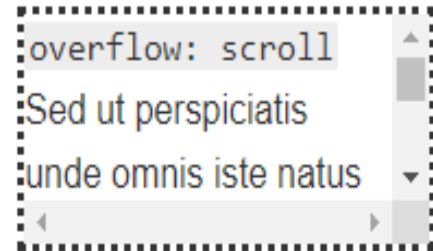
overflow: hidden  
Sed ut perspiciatis unde  
omnis iste natus error sit  
voluntatem accusantium...



## ➤ Value:

- **scroll**: Content is clipped if necessary to fit the padding box. Browsers always display scrollbars whether or not any content is actually clipped, preventing scrollbars from appearing or disappearing as content changes

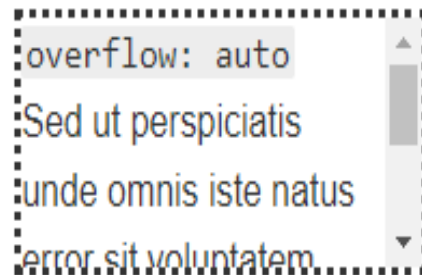
```
1 | p { overflow: scroll; /* always show scrollbars */ }
```



## ➤ Value:

- **auto**: Depends on the user agent. If content fits inside the padding box, it looks the same as visible, but still establishes a new block formatting context. Desktop browsers provide scrollbars if content

```
1 | p { overflow: auto; /* append scrollbars if necessary */ }
```



## Section 4

# STYLING IMAGE

- Use the **border-radius** property to create rounded images

```
img {  
  border-radius: 8px;  
}
```



```
img {  
  border-radius: 50%;  
}
```



➤ **Responsive images** will automatically adjust to fit the size of the screen

- Step1: Add HTML

```

```

- Step 2: Add CSS

If you want the image to scale both up and down on responsiveness, set the CSS width property to 100% and height to auto

```
.responsive {  
  width: 100%;  
  height: auto;  
}
```

## ➤ Responsive images

If you want an image to scale down if it has to, but never scale up to be larger than its original size, use `max-width: 100%`

```
.responsive {  
  max-width: 100%;  
  height: auto;  
}
```

If you want to restrict a responsive image to a maximum size, use the `max-width` property, with a pixel value of your choice

```
.responsive {  
  width: 100%;  
  max-width: 400px;  
  height: auto;  
}
```

- **Center an image:** To center an image, set left and right margin to auto and make it into a block element

```
img {  
  display: block;  
  margin-left: auto;  
  margin-right: auto;  
  width: 50%;  
}
```

- **Transparent image:** The ***opacity*** property can take a value from 0.0 - 1.0. The lower value, the more transparent



opacity 0.2



opacity 0.5



opacity 1  
(default)

## Section 5

# STYLING LIST



- **Bullet styles:** The list-style-type property specifies the type of list item marker.

```
1  ol {  
2    list-style-type: upper-roman;  
3  }
```



- I. Toast pitta, leave to cool, then slice down the edge.
- II. Fry the halloumi in a shallow, non-stick pan, until browned on both sides.
- III. Wash and chop the salad
- IV. Fill Pitta with salad, humous, and fried halloumi.

## ➤ **Bullet styles:** some value of list-style-type property

- disc
- circle
- square
- decimal
- decimal-leading-zero
- lower-roman
- lower-greek

▪ ...

- **Bullet position:** The list-style-position property specifies the position of the list-item markers.

- list-style-position: out

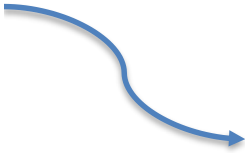
- |   |
|---|
| • Coffee - A brewed drink prepared from roasted coffee beans... |
| • Tea   |
| • Coca-cola   |

- list-style-position: inside

- |   |
|---|
| • Coffee - A brewed drink prepared from roasted coffee beans... |
| • Tea   |
| • Coca-cola   |

- **Using a custom bullet image:** The list-style-image property specifies an image as the list item marker

```
ul {  
  list-style-image: url('sqpurple.gif');  
}
```



- Coffee
- Tea
- Coca Cola

- **list-style shorthand:** The three properties mentioned above can all be set using a single shorthand property, list-style

```
1 | ul {  
2 |   list-style-type: square;  
3 |   list-style-image: url(example.png);  
4 |   list-style-position: inside;  
5 | }
```

Could be replaced by this

```
1 | ul {  
2 |   list-style: square url(example.png) inside;  
3 | }
```

# Lesson Summary



# Thank you

