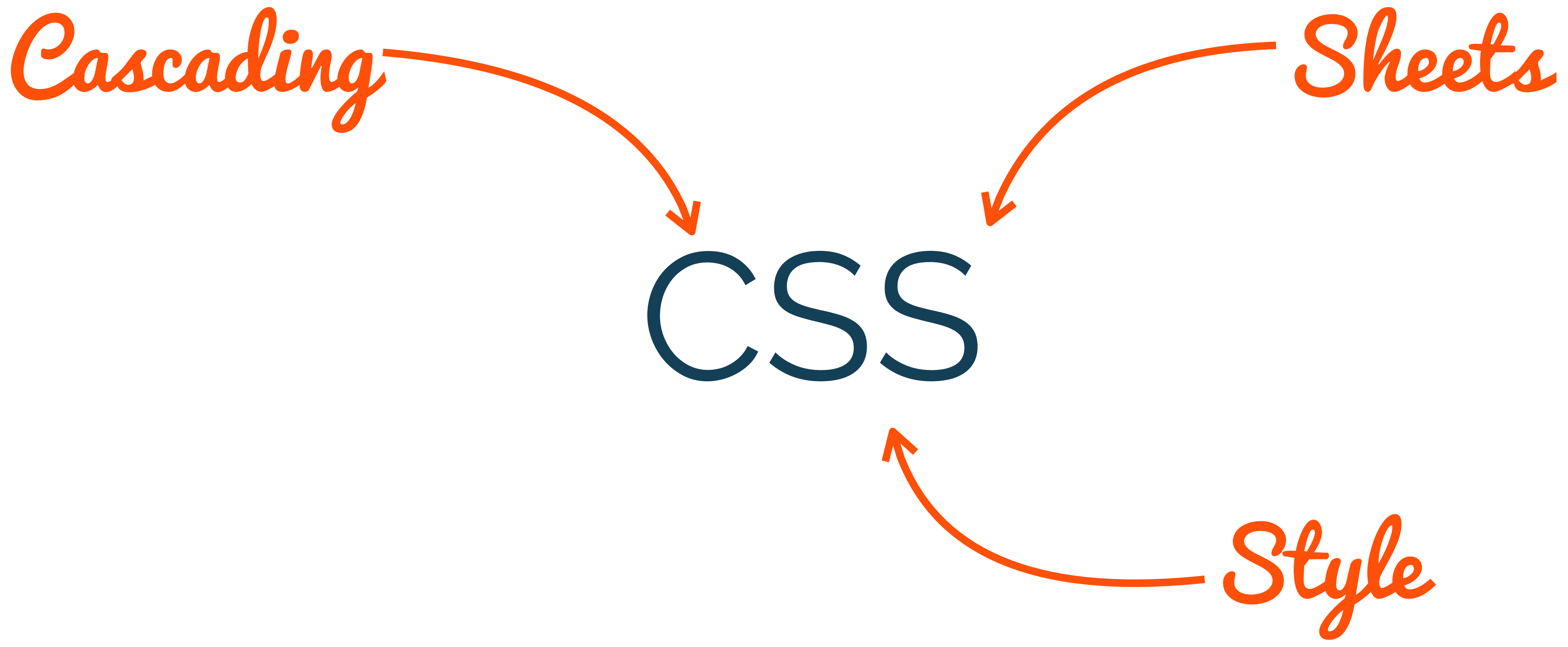
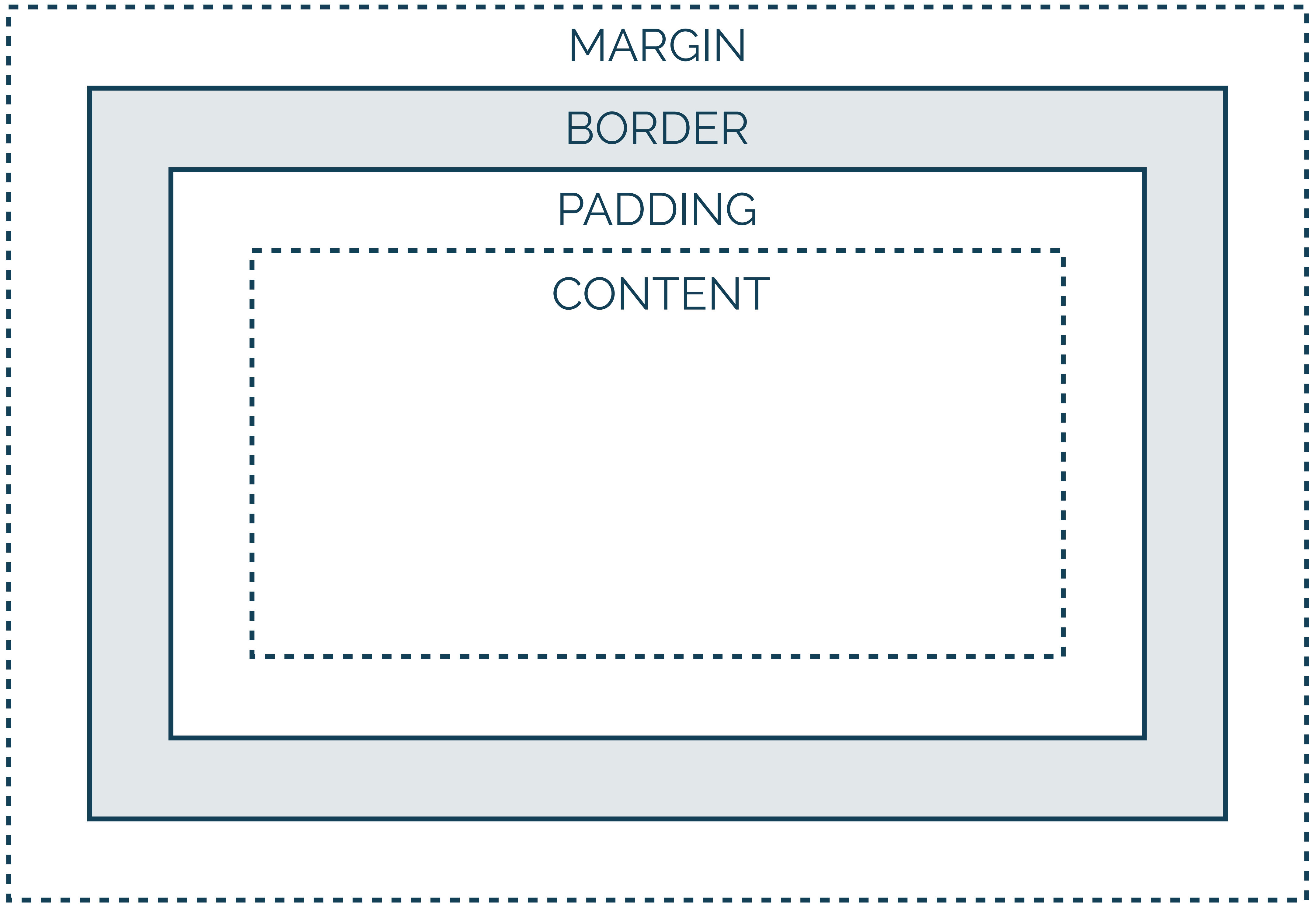
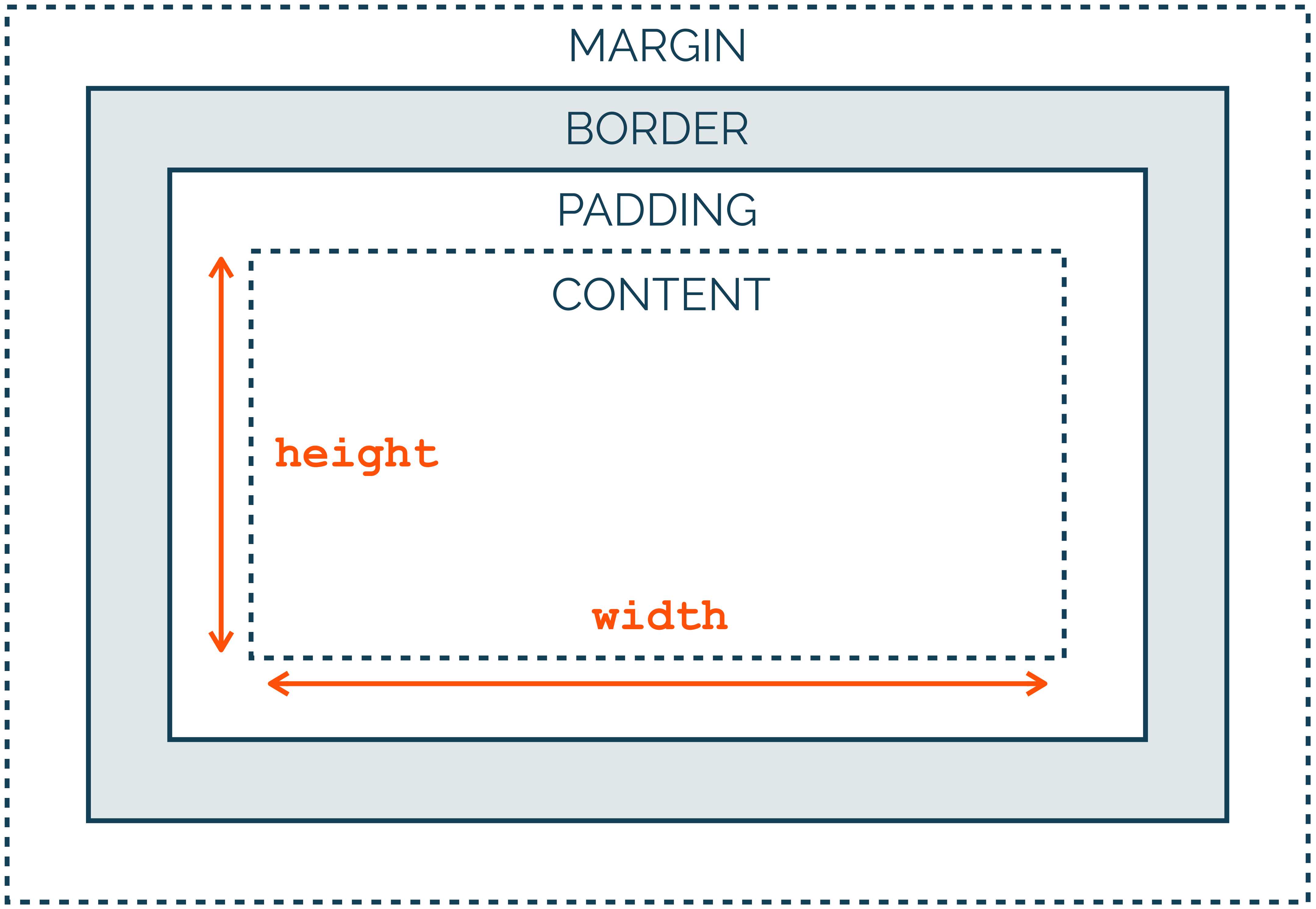

Cascading
Sheets
CSS
Style

Box Model



control over white space

Box Model



`width` and **`height`** properties refer to content area

to calculate full-size of the element add padding, border, and margins

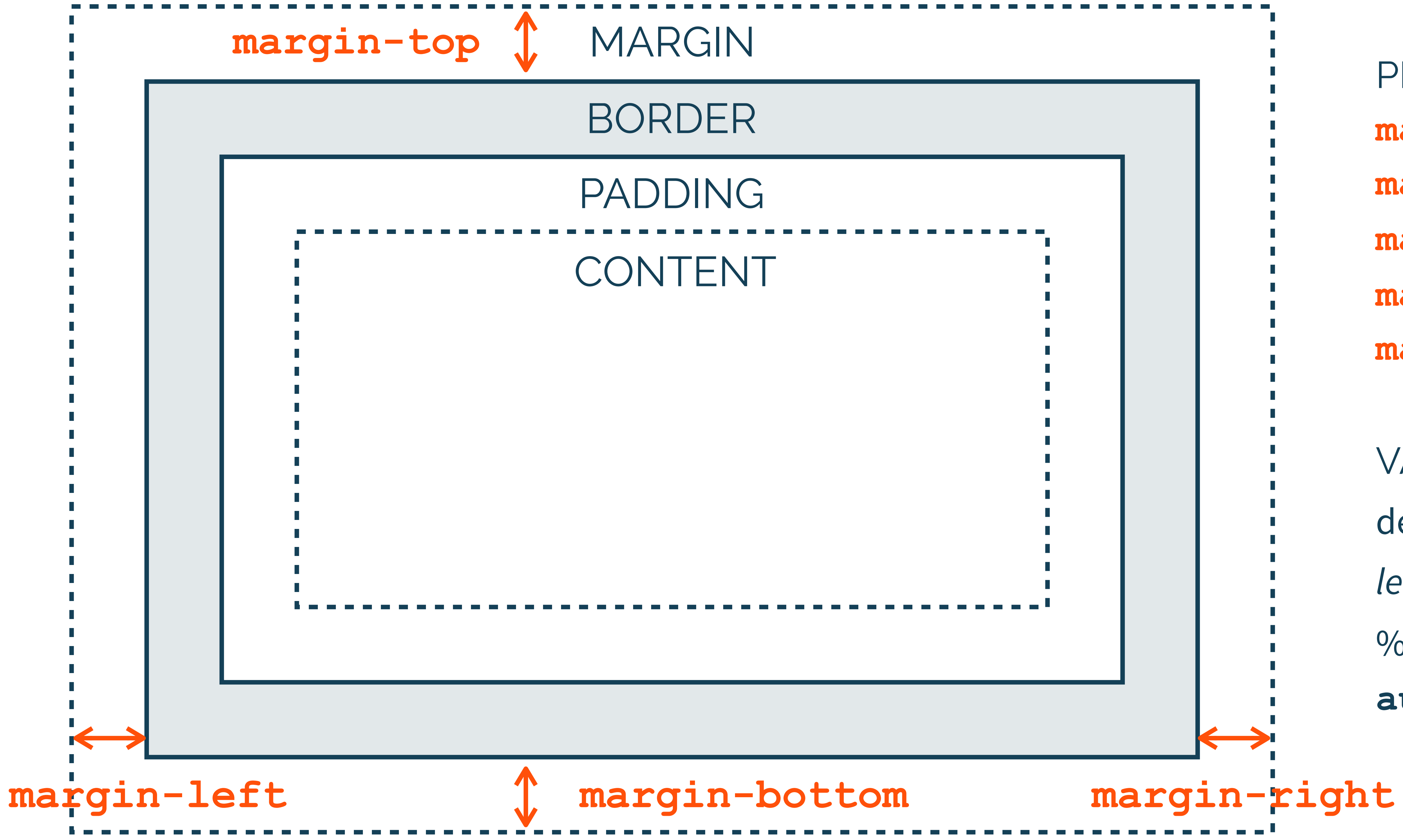
VALUES

default value is **`auto`**

length +/- (px, em, in, cm, pt)

% of parent's width

Box Model: Margin



PROPERTIES

`margin (shorthand)`

`margin-top`

`margin-bottom`

`margin-left`

`margin-right`

VALUES

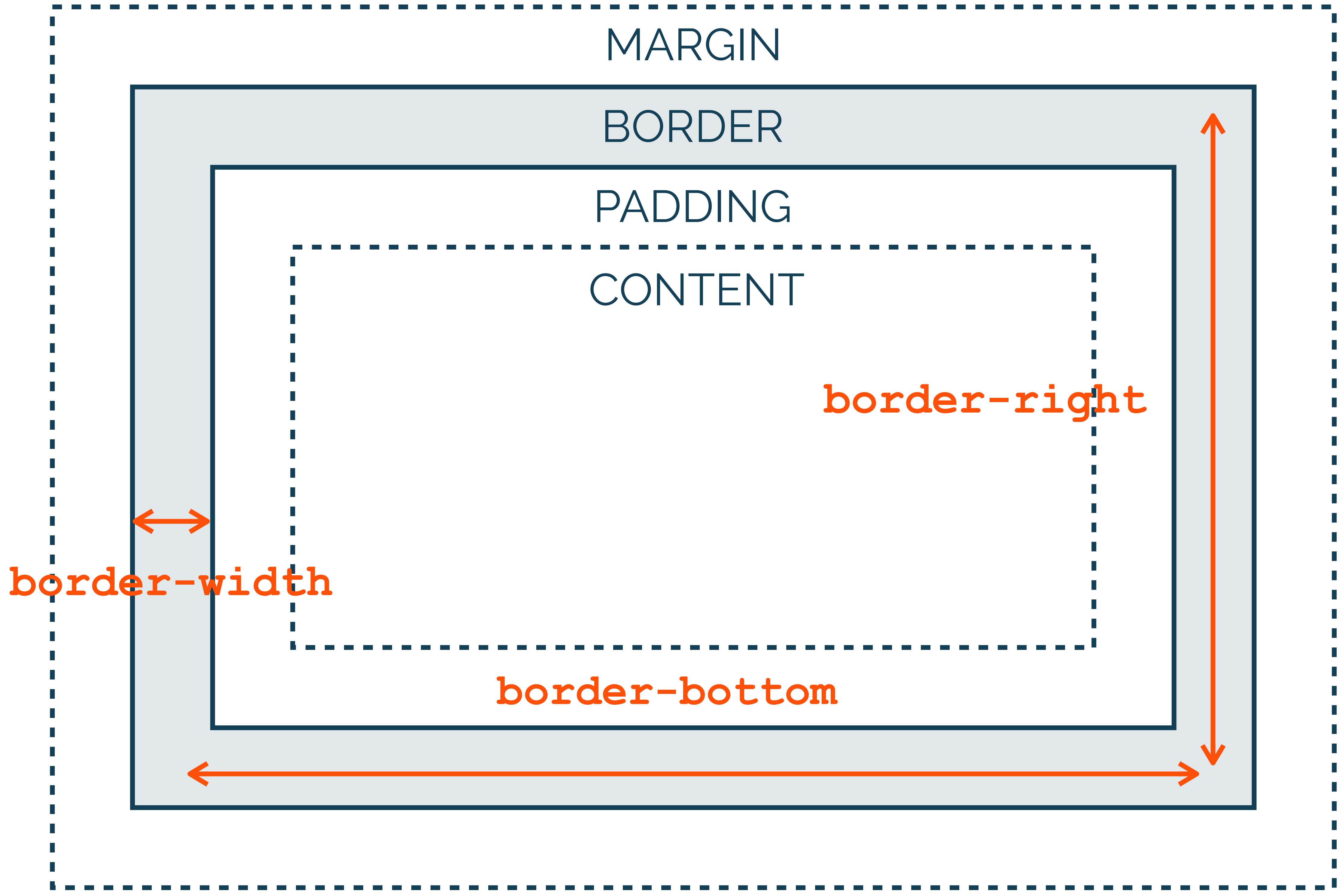
default value is 0

length +/- (px, em, in, cm, pt)

% of parent’s width

`auto`

Box Model: Border



PROPERTIES

border (shorthand)

border-top

border-bottom

border-left

border-right

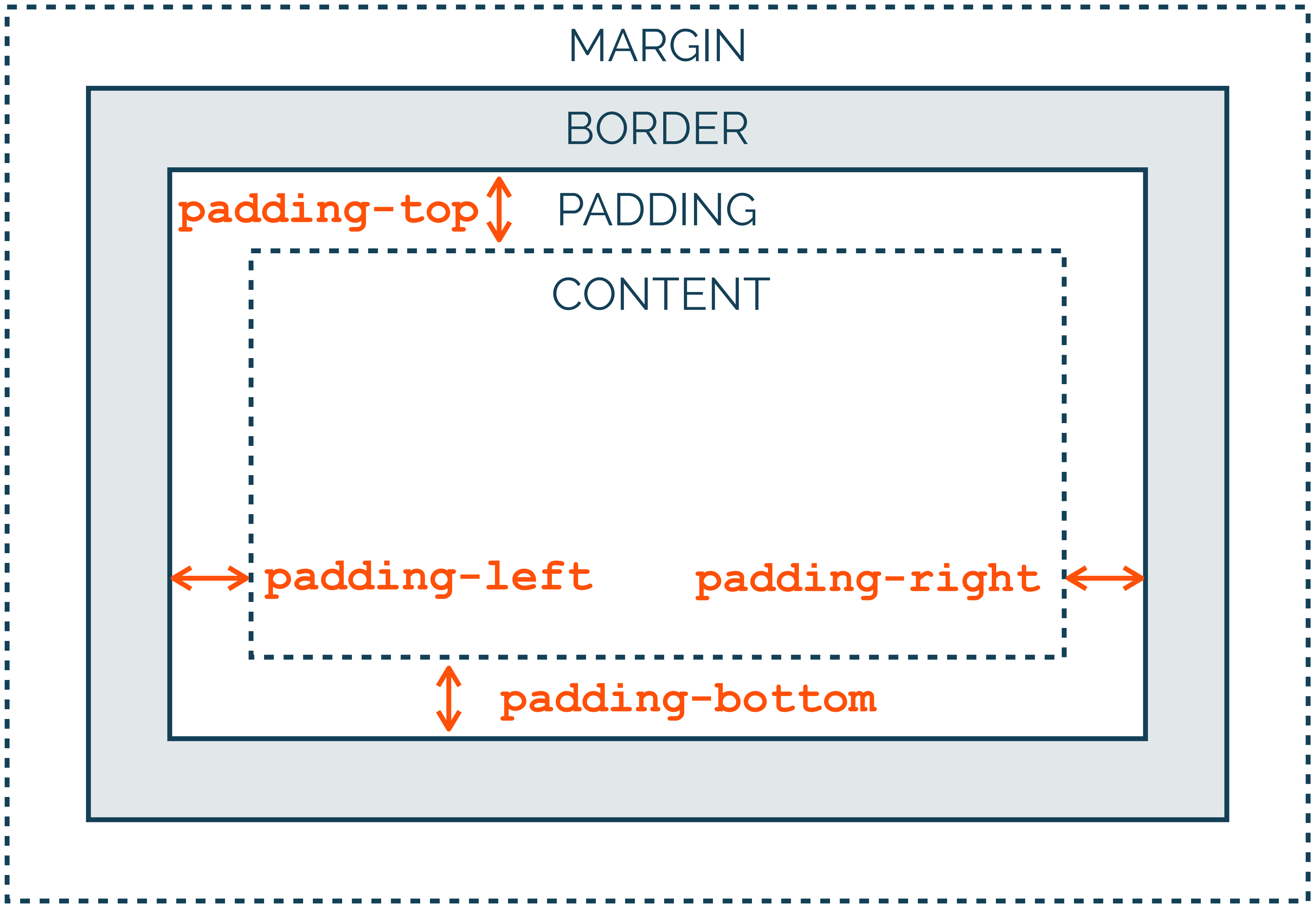
---

border-width

border-style

border-color

Box Model: Padding



PROPERTIES

`padding (shorthand)`

`padding-top`

`padding-bottom`

`padding-left`

`padding-right`

VALUES

default value is 0

length (px, em, in, cm, pt)

% of the element's width

LAYOUT

Block

rendered with preceding and following line breaks (stacked)

line breaks within nested elements collapsed if no other content

`width` of **`auto`** (default) will expand to fill entire width

Inline

rendered on a common baseline or wrap onto a new baseline below

margin, width, height properties don’t affect these elements

can only contain text or other inline elements

www.w3.org/wiki/The_CSS_layout_model_-_boxes_borders_margins_padding

UNITS

absolute (`px`, `in`, `cm`, `pt`) vs relative (`em`, `%`)

`em` relative to the font-size of the element

(or its parent when used to set `font-size`)

be careful when mixing different units

position

VALUE	DESCRIPTION
`static`	default. positioned by the flow model; unaffected by top, bottom, left, right
`fixed`	positioned relative to browser window; will not move when window is scrolled
`relative`	positioned relative to its normal position
`absolute`	positioned relative to the first ancestor where **`position!=static`**

use with
`top`
`bottom`
`left`
`right`

`display`

VALUE	DESCRIPTION
`inline`	default if the element is an inline element (*e.g.*, **`span`**) displays element as inline element
`block`	default if the element is a block-element (*e.g.*, **`div`**) displays element as block element
`table`	element behaves like **`table`** element
`none`	element not displayed (doesn't appear in DOM)

not the same as
`visibility: hidden;`

www.w3schools.com/cssref/pr_class_display.asp

`float`

breaks with the flow model

pushes element to **`left`** or **`right`**, allowing other elements to wrap around it

use **`clear`** (**`left`**, **`right`**, **`both`**) to force other elements below floated ones

often used to flow text around images

Design Challenge:

horizontally center a `<div>`

CODEPEN

SOLUTION

```html
<div class="outer">
  <div class="inner">
  </div>
</div>
```

```css
.outer {
  height: 300px;
  background-color: #144057;
}

.inner {
  width: 100px;
  height: 100px;
  background-color: #B6C4C9;

  margin: 0 auto;
}
```

Design Challenge:

vertically center a `<div>`

CODEPEN

SOLUTION

```
<div class="outer">
  <div class="inner">
  </div>
</div>
```

```
.outer {
   height: 300px;
   background-color: #144057;

   position:relative;
}

.inner {
   width: 100px;
   height: 100px;
   background-color: #B6C4C9;

   position: absolute;
   top: 50%;
   margin-top: -50px;
}
```

known height!

Design Challenge:

vertically center a **`<div>`**

of unknown height

CODEPEN

SOLUTION

```
<div class="table-outer">
  <div class="outer">
    <div class="inner">
    </div>
  </div>
</div>
```

```
.table-outer {
  width: 100%;
  display: table;
}

.outer {
  height: 200px;
  background-color: #144057;

  display: table-cell;
  vertical-align: middle;
}

.inner {
  width: 100px;
  height: 50%;
  background-color: #B6C4C9;
}
```

css tables!

Separation of CONTENT *from* PRESENTATION?

purely presentational html!

```
<div class="table-outer">
  <div class="outer">
    <div class="inner"></div>
  </div>
</div>
```

a lot of HTML suffers from presentational `div` bloat

Separation of CONTENT *from* PRESENTATION?

good in theory, doesn’t always work in practice

DOMs are often cluttered with presentational HTML

Add higher-level design attributes to CSS
(*i.e., CSS3 implemented rounded corners*)

Research: Cascading Tree Sheets (CTS) [Benson et al.]

CSS PREPROCESSORS

languages that extend CSS in meaningful ways

features: variables, nesting, mixins, inheritance

shrinks developer’s codebase and compiles into CSS

popular CSS preprocessors: LESS and SASS

VARIABLES

```
$heading_font:'Source Sans Pro', sans-serif;
$body_font: 'Raleway', sans-serif;
$nav_font: 'Maven Pro', sans-serif;

$text_color: #181818;
$attention_color: #ff500a;

body {
  font-family: $body_font;
  font-size: 14px;
  color: $text_color;
}
…
```

All examples are written in SASS

NESTING

```
.class {
  div {
   font-family: $nav_font;
  }
  a {
    color: $attention_color;
    text-decoration: none;
  }
  li {
    margin-bottom: 10px;
  }
}
```

compiles into

```
.class div {
  font-family: $nav_font;
}
.class a {
  color: $attention_color;
  text-decoration: none;
}
.class li {
  margin-bottom: 10px;
}
```

All examples are written in SASS

MIXINS

```
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}

.small-box { @include border-radius(5px); }
.big-box { @include border-radius(10px); }
```

compiles into

```
.small-box {
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  -ms-border-radius: 5px;
  border-radius: 5px;
}

.big-box {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  -ms-border-radius: 10px;
  border-radius: 10px;
}
```

All examples are written in SASS

JAVASCRIPT

and the Web!

JAVASCRIPT

popular scripting language on the Web, supported by browsers

separate scripting from structure (HTML) and presentation (CSS)

client- and server-side programming

object-oriented, imperative, functional

HOW TO EMBED JS IN HTML

Embed external file

```
<script type=“text/javascript" src=“code.js”></script>
```

Inline in HTML

```
<script type="text/javascript">
<![CDATA[
  Javascript goes here...
]]>
</script>
```

everything inside ignored by parser

Revisiting the Dom

DOM DOCUMENT OBJECT

root node of HTML document

selector properties/methods:

```
document.body
document.getElementById()
document.getElementsByClassName()
document.getElementsByTagName()
```

DOM ELEMENT OBJECT

Element metadata:

```
element.tagName
element.className
element.id
element.attributes
element.innerHTML
```

Node metadata:

```
element.nodeName
element.nodeType
element.nodeValue
```

www.w3schools.com/jsref/dom_obj_all.asp

DOM ELEMENT OBJECT

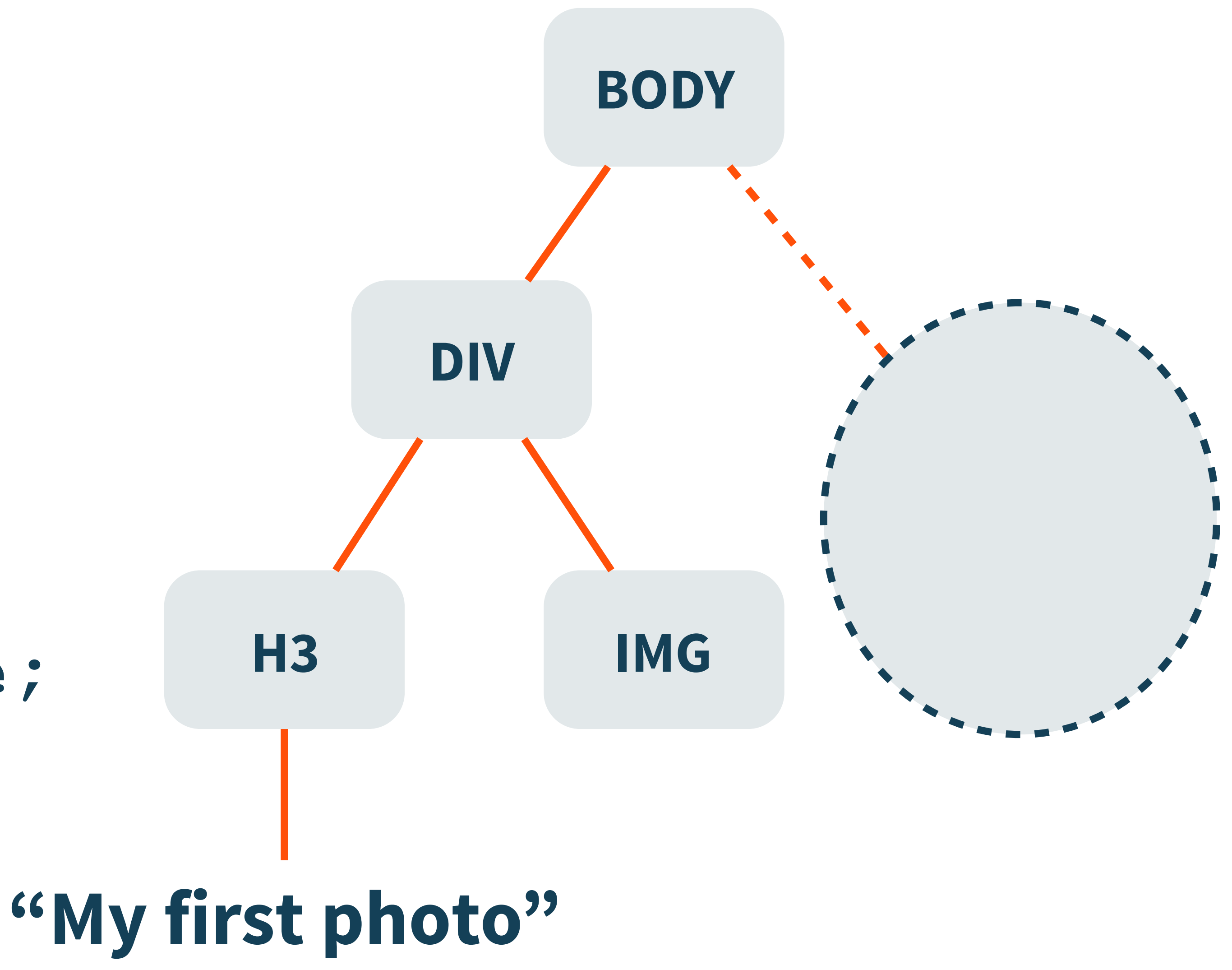
properties for traversing the DOM tree:

```
element.childNodes/element.children
element.parentNode/element.parentElement
element.previousSibling/element.previousElementSibling
element.nextSibling/element.nextElementSibling
```

www.w3schools.com/jsref/dom_obj_all.asp

TRAVERSING THE DOM

```
var body = document.body;
var div = body.children[0];
var h3 = div.children[0];
var textNode = h3.childNodes[0];
var textString = textNode.nodeValue;
```



www.w3schools.com/jsref/dom_obj_all.asp

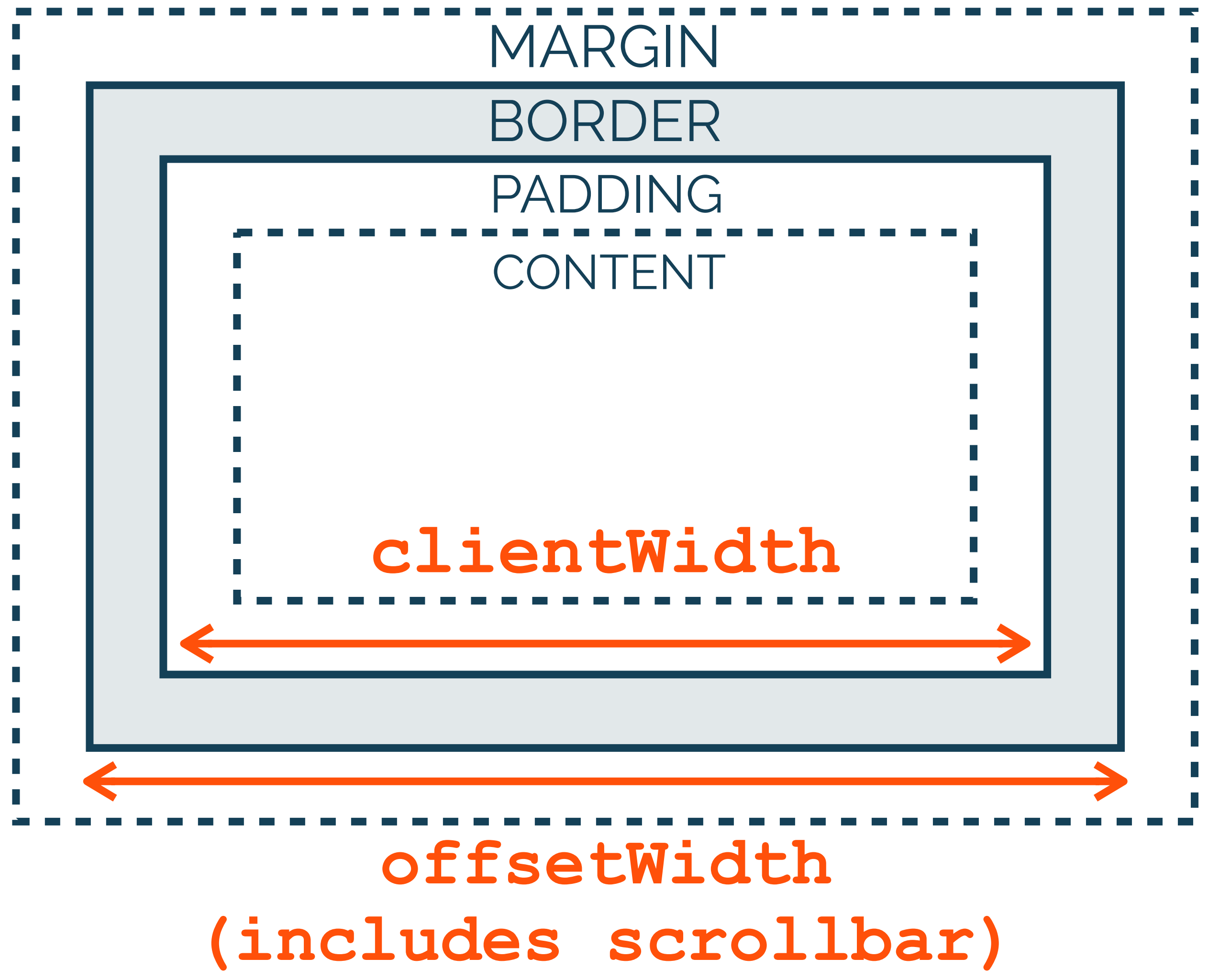
DOM ELEMENT OBJECT

relative to offsetParent

position: **`element.offsetTop`**, **`element.scrollTop`**, ...

dimensions: **`element.clientWidth`**, **`element.offsetWidth`**, ...

style: **`element.style`**



www.w3schools.com/jsref/dom_obj_all.asp

DOM MANIPULATION

programmatically change the structure and modify element properties

```
element.style.backgroundColor = “red”;
element.innerHTML = “<div><h3>Llama!</h3>…</div>”
```

augment DOM structure:

```
element.appendChild(),element.removeChild(),...
```

www.w3schools.com/jsref/dom_obj_all.asp

Events

TYPES OF EVENTS

User: *mouse clicks, mouse moves, key presses*

Browser: *page load/unload*

Network: *responses to AJAX request*

Timer

TIMER EVENTS

```
setTimeout(fn, ms);
```

calls function after specified amount of time (ms)

```
setInterval(fn, ms);
```

calls function at specified intervals (ms) until `clearInterval()` or window is closed

EVENT HANDLERS

also known as listeners

callback functions

specify: what happened, where it happened, and how to handle it

EVENT HANDLERS

DOM LEVEL 0

```
<div onclick="alert('Llama!');">...</div>
```

In HTML

DOM LEVEL 1

```
element.onclick = function(){alert('Llama!');}
```

In Javascript using the DOM

EVENT HANDLERS

DOM LEVEL 2

```
var el = document.getElementById(‘myButton');

el.addEventListener( 'click', function(){

      alert(‘Llama!');});
```

supports multiple handlers per event

CODEPEN