



CROSS X COUNTRY

FROST-PROOF FUNDAMENTALS

FROST-PROOF FUNDAMENTALS

Slope Style

Selectors

Cascade Order

Floating Left & Right

SLOPE STYLE

Adding CSS:

- ✗ Inline style attribute
- ✗ In the <head>
- ✗ External <link>

SLOPE STYLE

Adding CSS:

- ✗ Inline style attribute

```
<h1 style="color: #98c7d4;">CSS Cross Country</h1>
```

!

SLOPE STYLE

Adding CSS:

- ✗ In the <head>

```
<head>
  <style>
    h1 { color: #98c7d4; }
  </style>
</head>
```

!

SLOPE STYLE

Adding CSS:

- ✗ External <link>

```
<head>
  <title>CSS Cross Country</title>
  <link rel="stylesheet" href="styles.css" />
</head>
```



FROST-PROOF FUNDAMENTALS

Slope Style

Selectors

Cascade Order

Floating Left & Right

SELECTORS

Primary DOM selectors:

- ✖ Element selector
- ✖ Class selector
- ✖ ID selector

ADVANCED SELECTORS

Learn more about advanced
selectors like child & sibling:

[go.codeschool.com/advanced-
css-selectors](http://go.codeschool.com/advanced-css-selectors)

SELECTORS

Primary DOM selectors:

- ✗ Element selector

```
<h1 class="intro" id="header">Nice and Toasty</h1>
```



```
h1 {  
    color: #aba4ac;  
    margin-bottom: 10px;  
}
```

SELECTORS

Primary DOM selectors:

- ✗ Class selector

```
<h1 class="intro" id="header">Nice and Toasty</h1>
```

```
.intro {  
  color: #aba4ac;  
  margin-bottom: 10px;  
}
```

SELECTORS

Primary DOM selectors:

- ✗ ID selector

```
<h1 class="intro" id="header">Nice and Toasty</h1>
```

```
#header {  
    color: #aba4ac;  
    margin-bottom: 10px;  
}
```

SELECTORS

Compound selectors:

```
<h1 class="intro" id="header">Nice and Toasty</h1>
```

```
h1#header {  
    color: #aba4ac;  
    margin-bottom: 10px;  
}
```

FROST-PROOF FUNDAMENTALS

Slope Style

Selectors

Cascade Order

Floating Left & Right

CASCADE ORDER

Style priority is determined by position in site:

- ✗ External <link>
- ✗ In the <head>
- ✗ Inline style attribute
- ✗ Using !important !



Increasing Priority

CASCADE ORDER

Priority is also dependent on position in document:

```
.intro {  
  color: #444245;  
}  
.intro {  
  color: #dddadd;  
}
```



*the second color definition for
.downhill overrides the first*

CASCADE ORDER

Non-conflicting properties will be combined:

```
.intro {  
  color: #dddadd;  
}  
.intro {  
  margin-bottom: 5px;  
  width: 900px;  
}
```

```
.intro {  
  color: #dddadd;  
  margin-bottom: 5px;  
  width: 900px;  
}
```

FROST-PROOF FUNDAMENTALS

Slope Style

Selectors

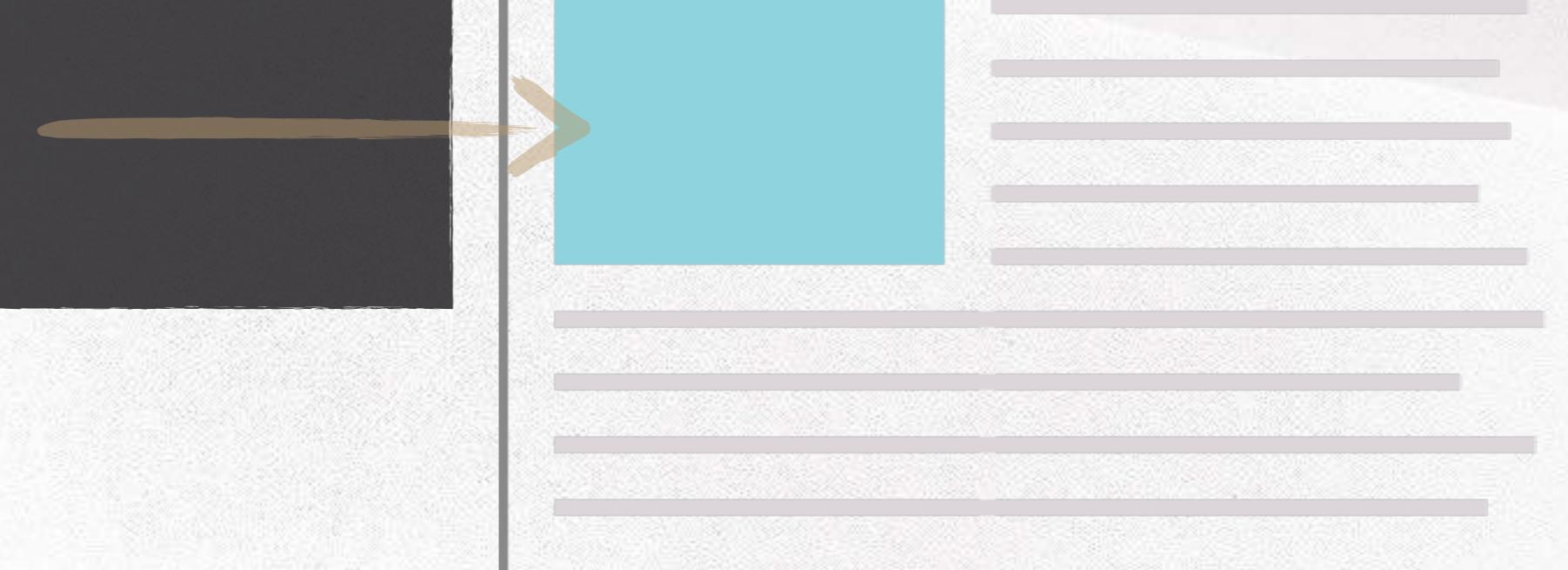
Cascade Order

Floating Left & Right

FLOATING LEFT & RIGHT

```
<article>
  
  <p>To successfully ski, simply do not fall.</p>
</article>
```

```
img {
  float: left;
}
```

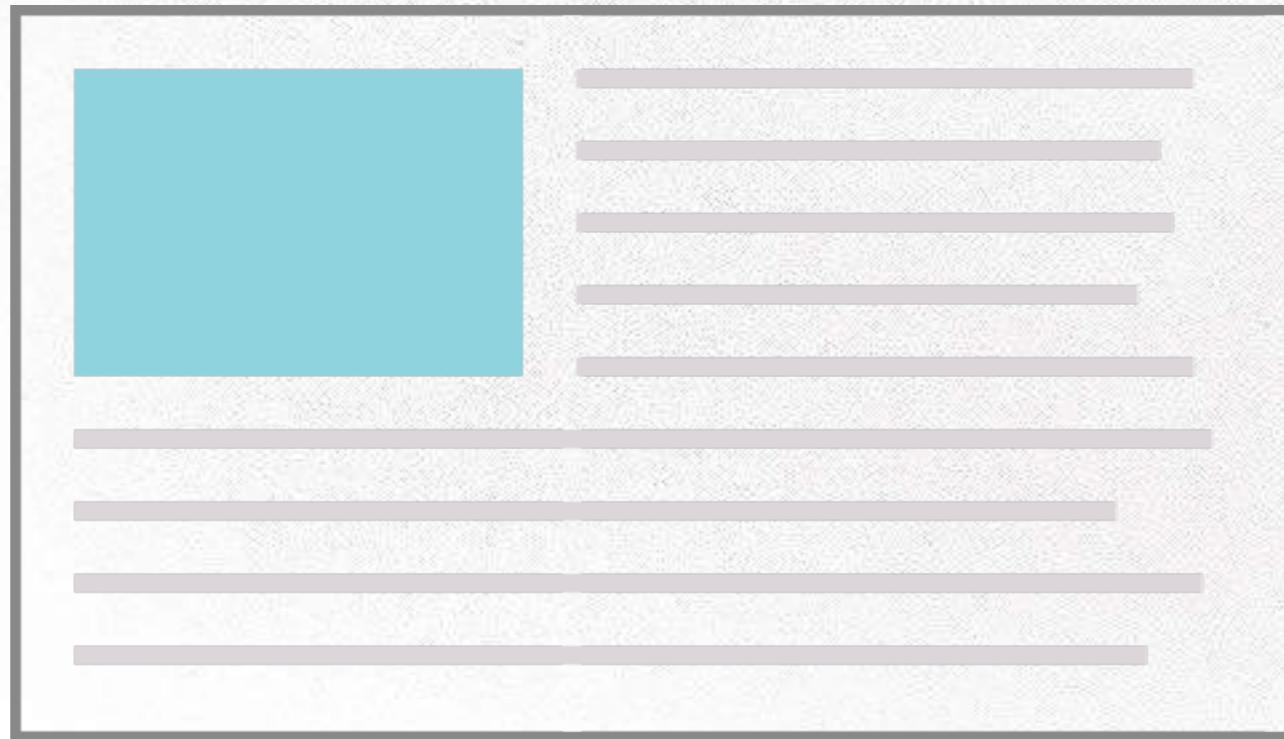


FLOATING LEFT & RIGHT

Removes elements from the document flow and moves them to a specified edge

- × Other content within the parent element will wrap around floats

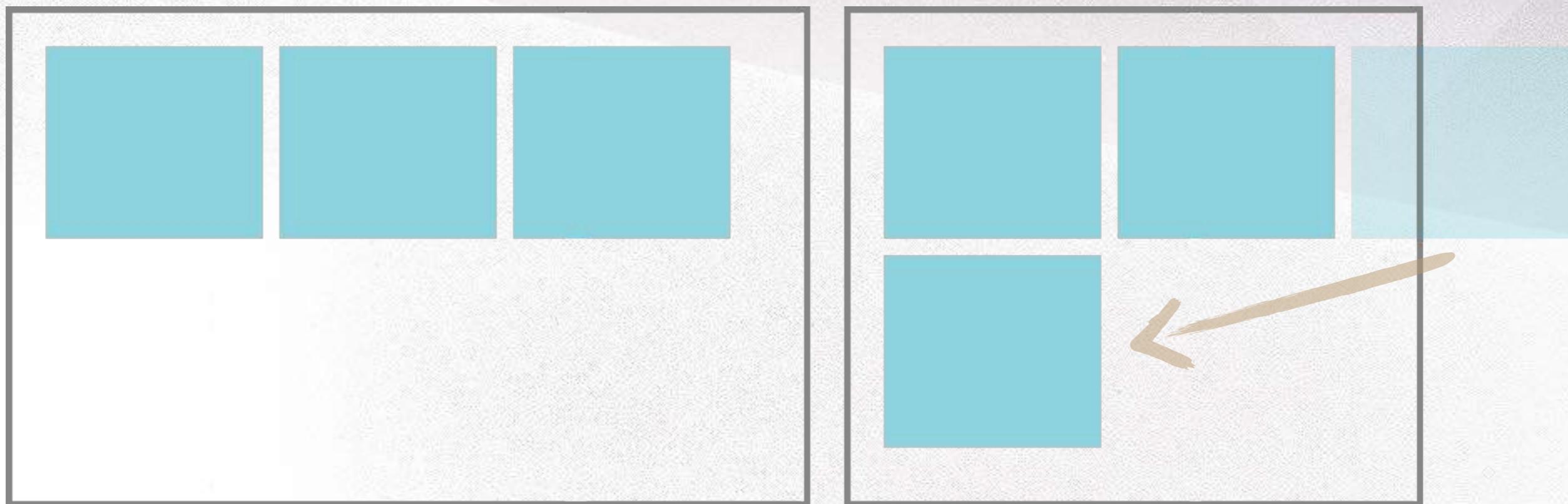
```
float: left / right / none
```



FLOATING LEFT & RIGHT

Stacking order:

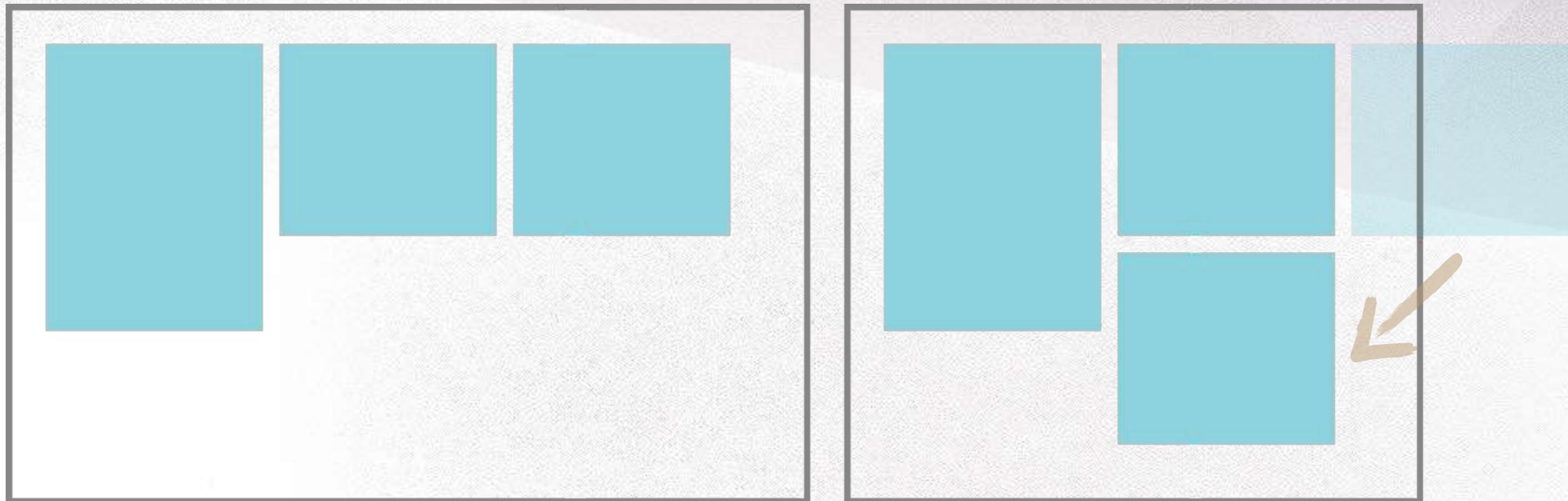
- × Floated elements stack up to the parent edge, then move down to the next available edge



FLOATING LEFT & RIGHT

Stacking order:

- Take care with elements that have differing heights - the first available edge isn't always below



FLOATING LEFT & RIGHT

```
<article>
  
  
</article>
```

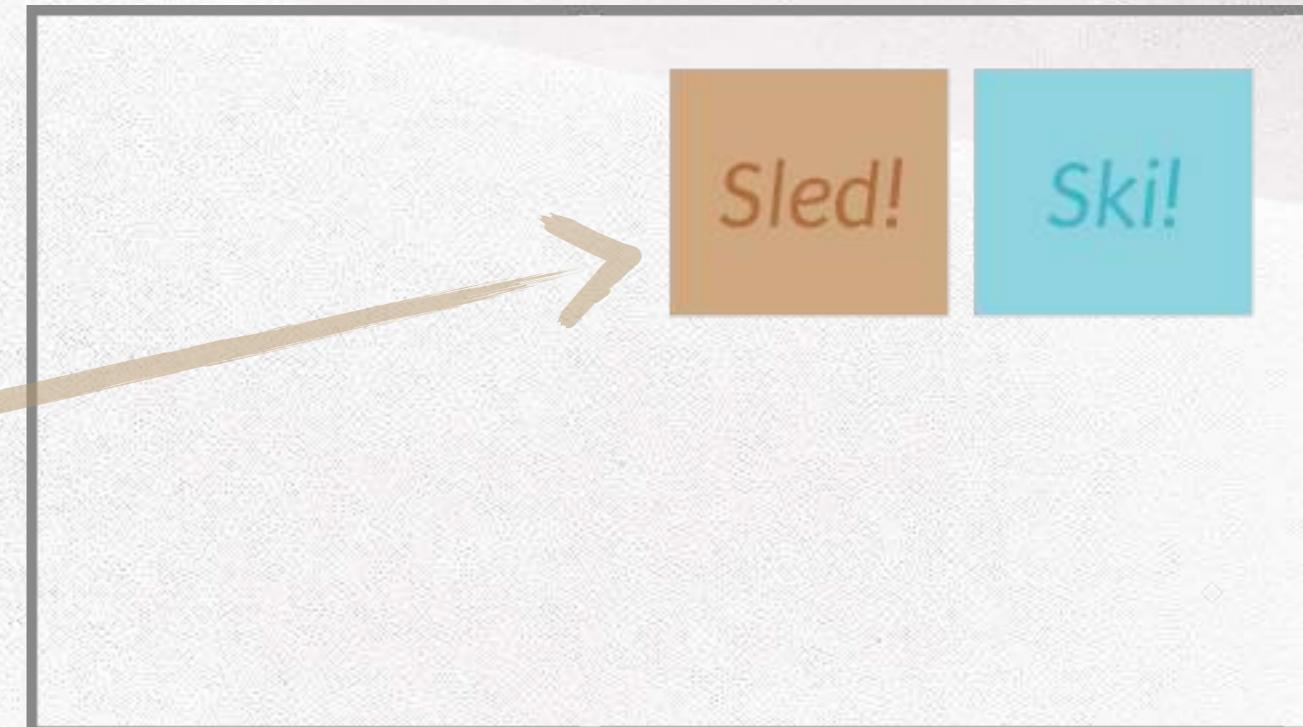
```
.ski {
  float: right;
}
.sled {
  float: left;
}
```



FLOATING LEFT & RIGHT

```
<article>
  
  
</article>
```

```
.ski {
  float: right;
}
.sled {
  float: right;
}
```





CROSS X COUNTRY

**CLEAR
CARVING**

CLEAR CARVING

Clearing Floats

Inheritance & Specificity

CLEARING FLOATS

Clearing is necessary if:

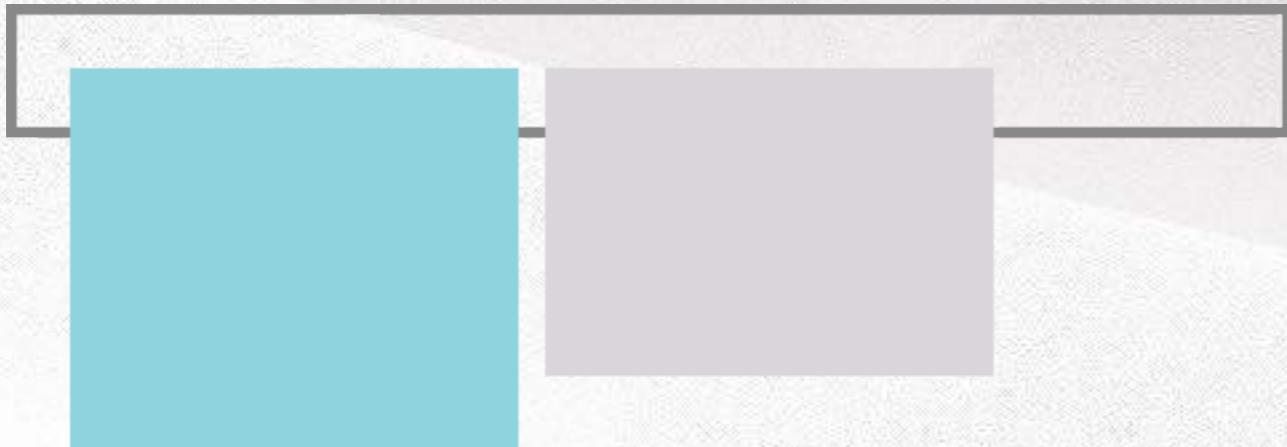
- ✗ Floated items can be taller than non-floated content



CLEARING FLOATS

Clearing is necessary if:

- ✗ All children are floating



CLEARING FLOATS

Common float-clearing methods:

- ✗ Clear with a subsequent element
- ✗ Manual clearing
- ✗ The clearfix

```
clear: left / right / both
```

CLEARING FLOATS

Common float-clearing methods:

- ✗ Clear with a subsequent element

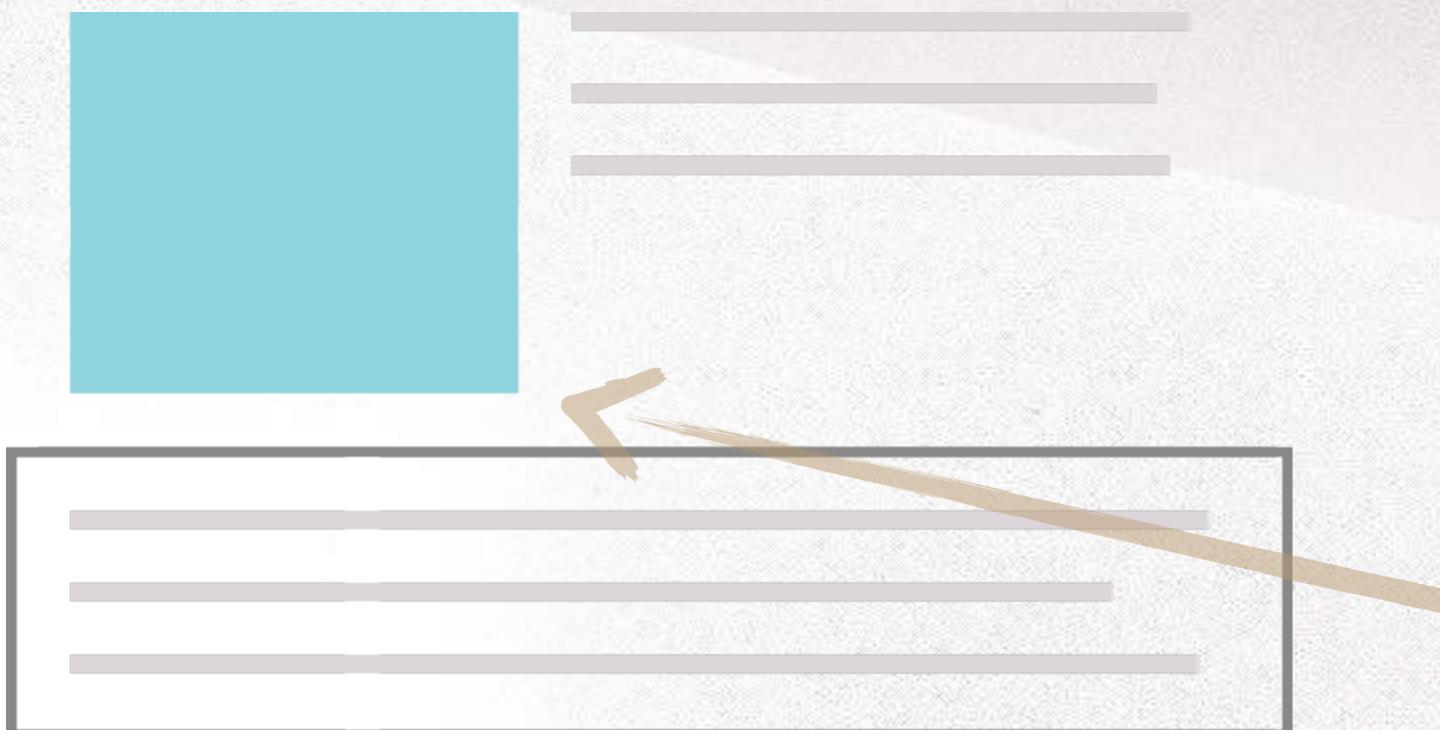
```
<div>
  
  <p>To successfully ski, simply do not fall.</p>
</div>
<div class="intro">
  <p>Whee!</p>
</div>
```

```
img {
  float: left;
}
.intro {
  clear: both;
}
```

CLEARING FLOATS

Common float-clearing methods:

- × Clear with a subsequent element

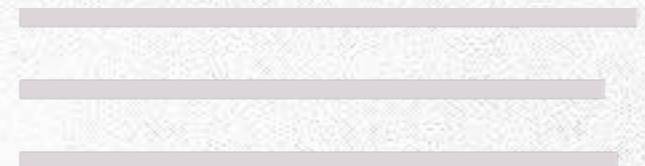
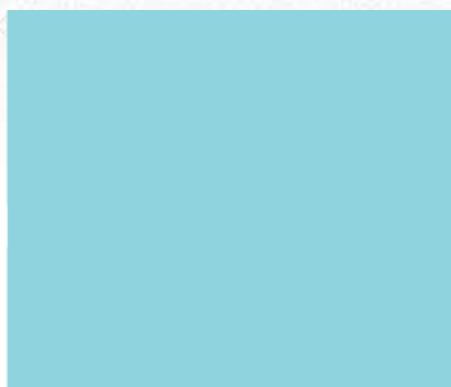


```
img {  
  float: left;  
}  
.intro {  
  clear: both;  
}
```

CLEARING FLOATS

Common float-clearing methods:

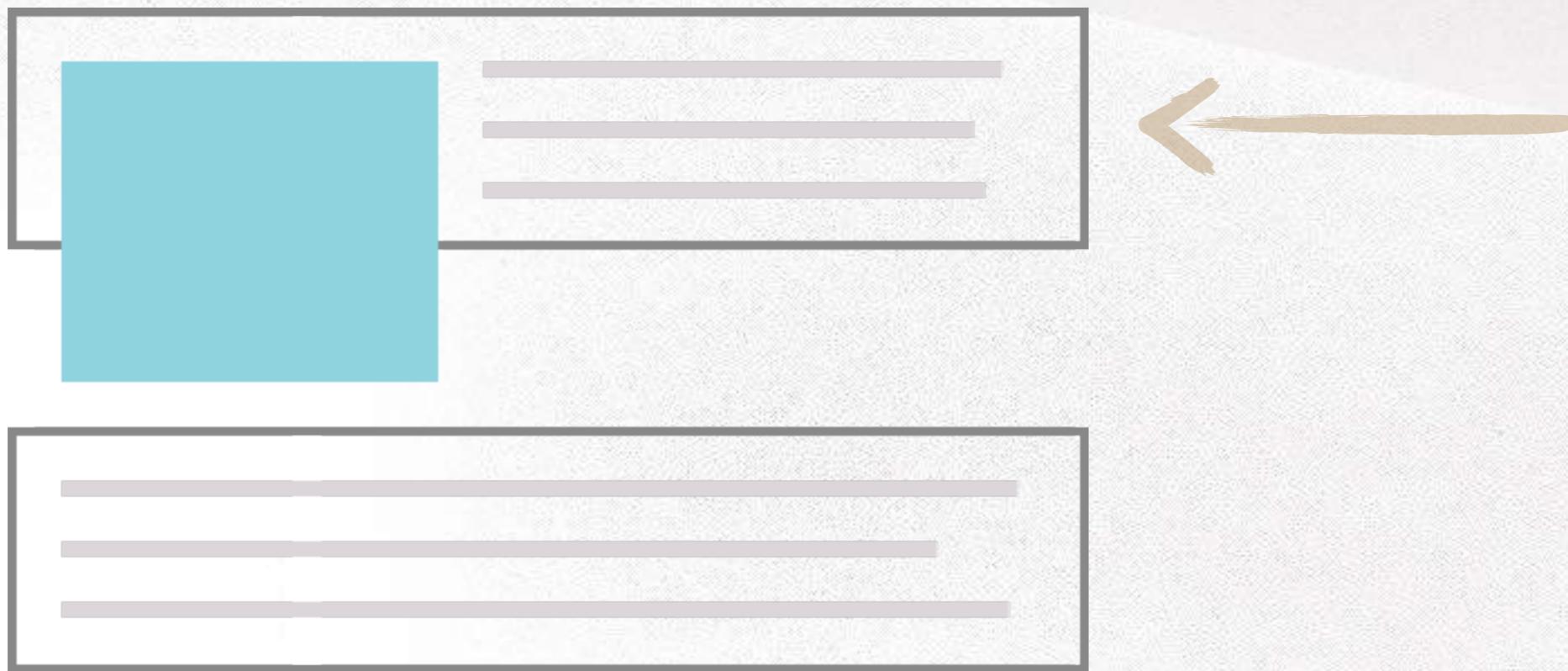
- ✗ Clear with a subsequent element
 - *Requires sequence to stay intact - breaks if things move*
 - *Background / border do not extend*



CLEARING FLOATS

Common float-clearing methods:

- ✗ Clear with a subsequent element
 - *Requires sequence to stay intact - breaks if things move*
 - *Background / border do not extend*

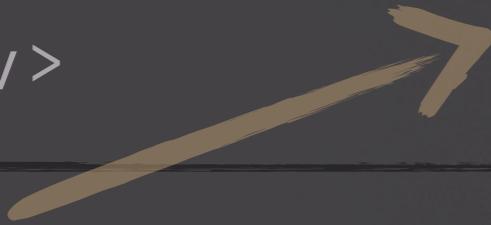


CLEARING FLOATS

Common float-clearing methods:

- ✗ Manual clearing

```
<div>
  
  <p>To successfully ski, simply do not fall.</p>
  <div class="clear"></div>
</div>
```

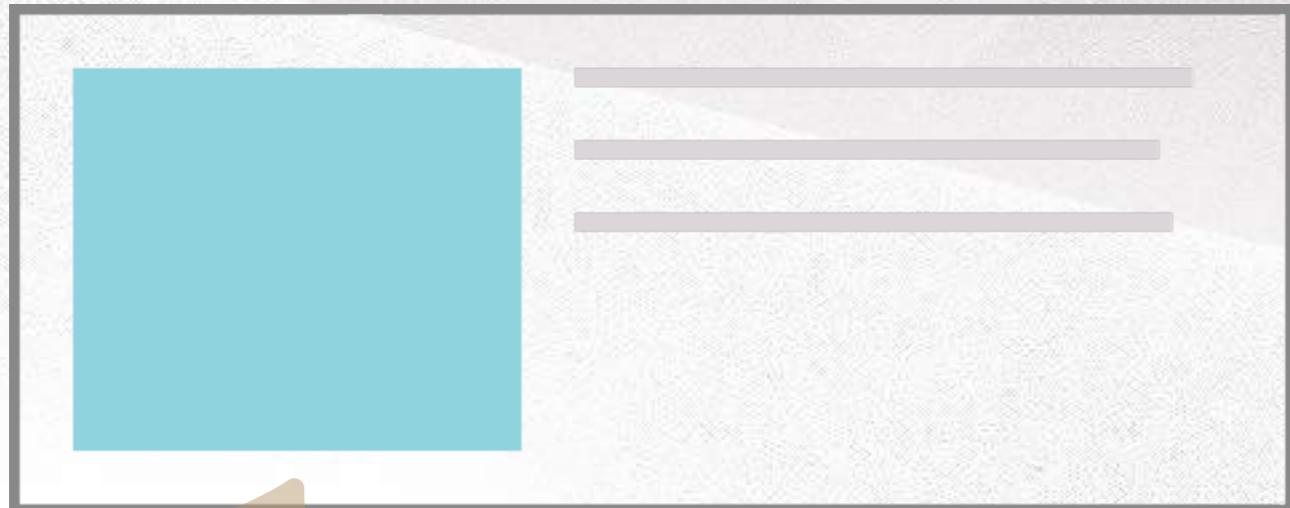


```
.clear {
  clear: both;
}
```

CLEARING FLOATS

Common float-clearing methods:

- ✗ Manual clearing

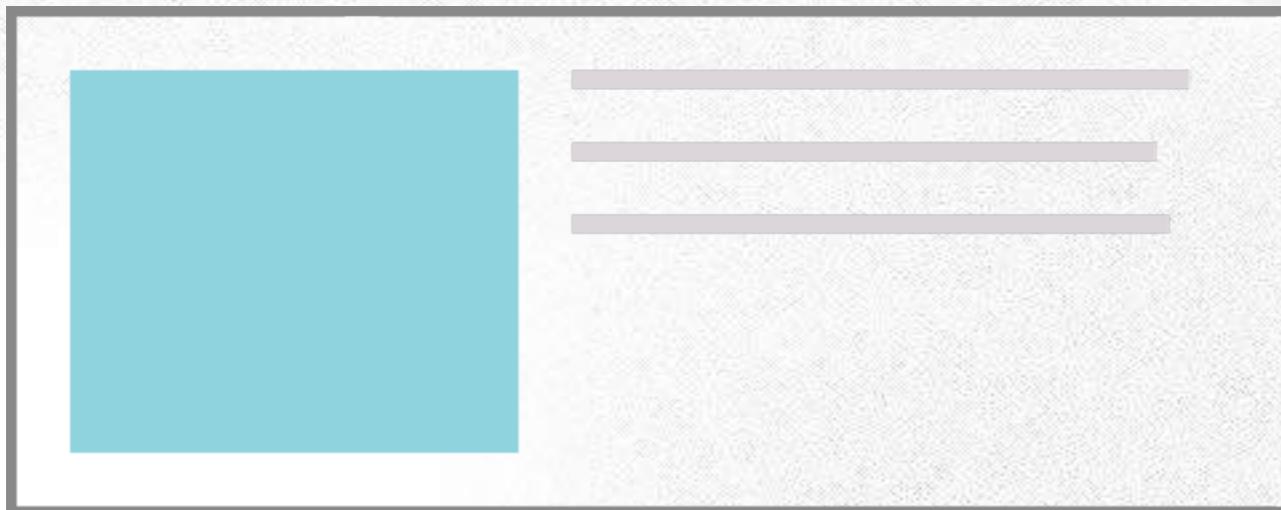


```
.clear {  
  clear: both;  
}
```

CLEARING FLOATS

Common float-clearing methods:

- ✗ Manual clearing
 - *Requires an empty element*
 - *Might not be necessary later*



CLEARING FLOATS

Common float-clearing methods:

- ✗ The clearfix

```
.group:before,  
.group:after {  
  content: " ";  
  display: table;  
}  
.group:after {  
  clear: both;  
}  
.group {  
  zoom: 1; /* IE6&7 */  
}
```

- Originally developed by Tony Aslett
- Refined version by Nicholas Gallagher
- When used on the parent, the children will be self-cleared

CLEARING FLOATS

Common float-clearing methods:

- × Theclearfix

```
<div class="group">
  
  <p>To successfully ski, simply do not fall.</p>
</div>
```



CLEAR CARVING

Clearing Floats

Inheritance & Specificity

INHERITANCE & SPECIFICITY

Nested elements automatically inherit parent styles:

```
<article class="featured">  
  <p>Dang, it's cold up here!</p>  
</article>
```

```
.featured {  
  color: #aba4ac;  
}
```



*the paragraph content above
will inherit this color*

INHERITANCE & SPECIFICITY

Selectors can be nested to override parent properties:

```
<article class="featured">  
  <p>Dang, it's cold up here!</p>  
</article>
```

```
.featured {  
  color: #0000ff;  
}  
.featured p {  
  color: #ffff;  
}
```

*nesting the element selector
overrides the color on .featured*

INHERITANCE & SPECIFICITY

Dealing with specificity:

```
<div id="content" class="featured">  
  <p>Break out the cocoa.</p>  
</div>
```

```
#content {  
  color: #555;  
}  
.featured {  
  color: #ccc;  
}
```

INHERITANCE & SPECIFICITY

The priority of a selector (specificity):

inline styles?

0 , 0 ,
↓ ↑
of ID selectors

of class selectors

0 , 0
↓ ↑
of element selectors

INHERITANCE & SPECIFICITY

```
p { color: #fff; }
```

0, 0, 0, 1

```
.intro { color: #98c7d4; }
```

0, 0, 1, 0

```
#header { color: #444245; }
```

0, 1, 0, 0

```
<h1 style="color: #000;">Mogul</h1>
```

1, 0, 0, 0

```
p { color: #fff !important; }
```



INHERITANCE & SPECIFICITY

```
.intro p.article { color: #fff; } 0,0,2,1
```

```
.intro ul li.active { color: #98c7d4; } 0,0,2,2
```

```
#header { color: #444245; } 0,1,0,0
```

STICKING TO CLASSES

Learn more about why using IDs sparingly will decrease complexity:
go.codeschool.com/id-selectors

INHERITANCE & SPECIFICITY

Dealing with specificity:

```
<section id="content">
  <p class="featured">Free coffee with ski rental.</p>
  <p>Choose from 48 different ski colors.</p>
</section>
```

```
#content p {          0,1,0,1
  color: #000;
}
.featured {           0,0,1,0
  color: #777;
}
```



INHERITANCE & SPECIFICITY

Dealing with specificity:

```
<section id="content">
  <p class="featured">Free coffee with ski rental.</p>
  <p>Choose from 48 different ski colors.</p>
</section>
```

```
#content p {
  color: #000;
}
.featured {
  color: #777 !important;
}
```



INHERITANCE & SPECIFICITY

Dealing with specificity:

```
<section id="content">
  <p class="featured">Free coffee with ski rental.</p>
  <p>Choose from 48 different ski colors.</p>
</section>
```

```
#content p {          0,1,0,1
  color: #000;
}
#content .featured { 0,1,1,0
  color: #777;
}
```





CROSS X COUNTRY

BOX BINDINGS



BOX BINDINGS

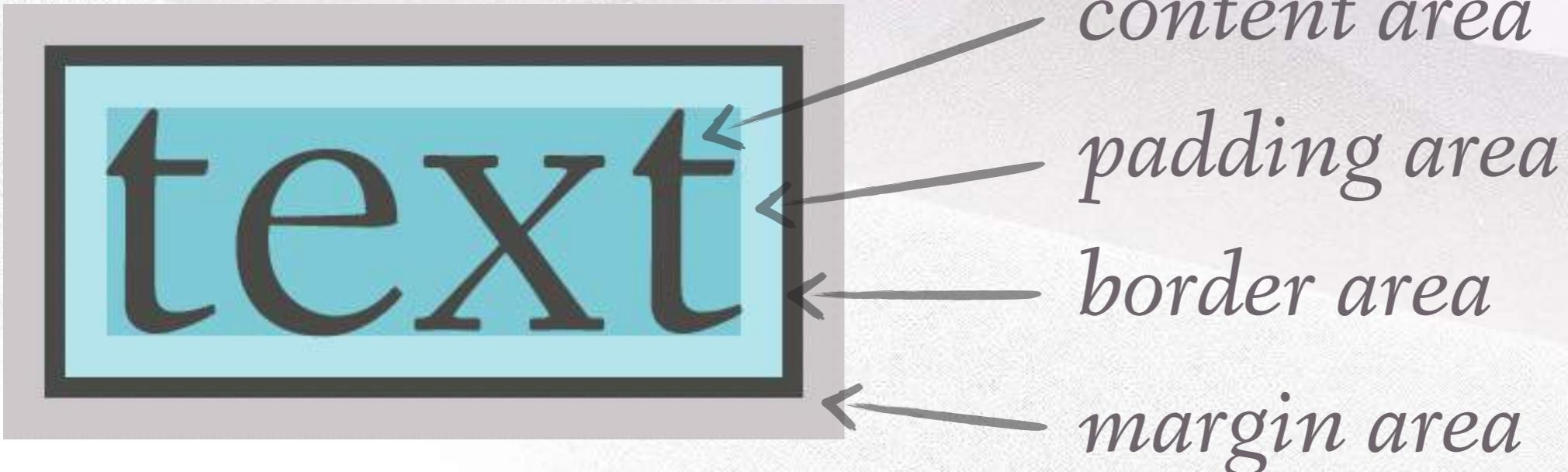
The Box Model

Positioning

Z-Index

THE BOX MODEL

An *imaginary* diagram that outlines each DOM element:



THE BOX MODEL

Width:

- × Total calculated box width =
content width + padding width + border width

```
.downhill {  
    border: 5px solid #fff;  
    padding-left: 10px;  
    padding-right: 5px;  
    width: 100px;  
}
```



100px *content width*

15px *padding width*

+ 10px *border width*

125px *box width*



THE BOX MODEL

Width:

- When adapting a design, you'll need to calculate the content width

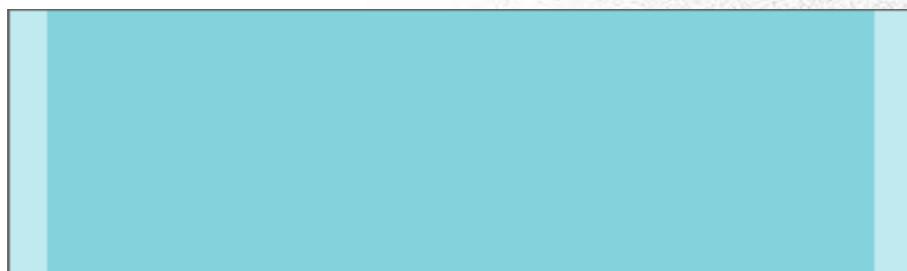
```
.downhill {  
  border: 1px solid #fff;  
  padding-left: 14px;  
  padding-right: 14px;  
  width: ? } ←
```

340px *box width (design)*

28px *padding width*

- 2px *border width*

310px *content width*



THE BOX MODEL

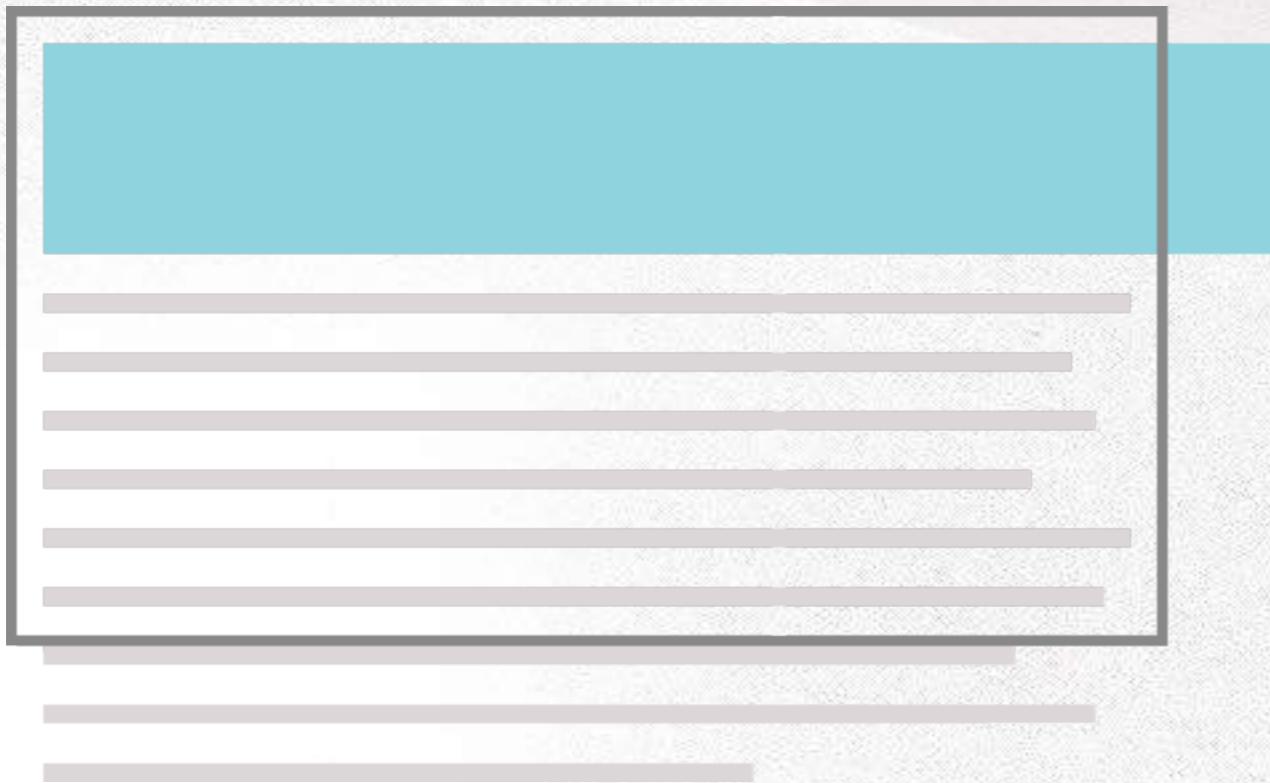
The overflow property:

```
overflow: visible / auto / hidden / scroll
```

THE BOX MODEL

The overflow property:

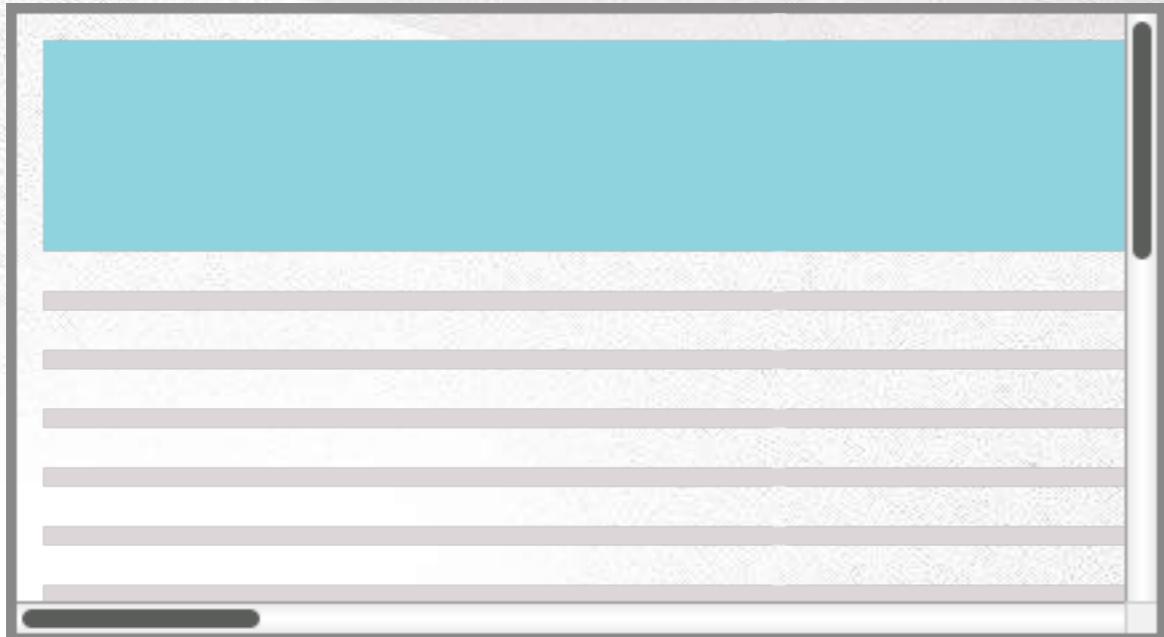
- × `visible` - the default value, which allows content to extend beyond container boundaries



THE BOX MODEL

The overflow property:

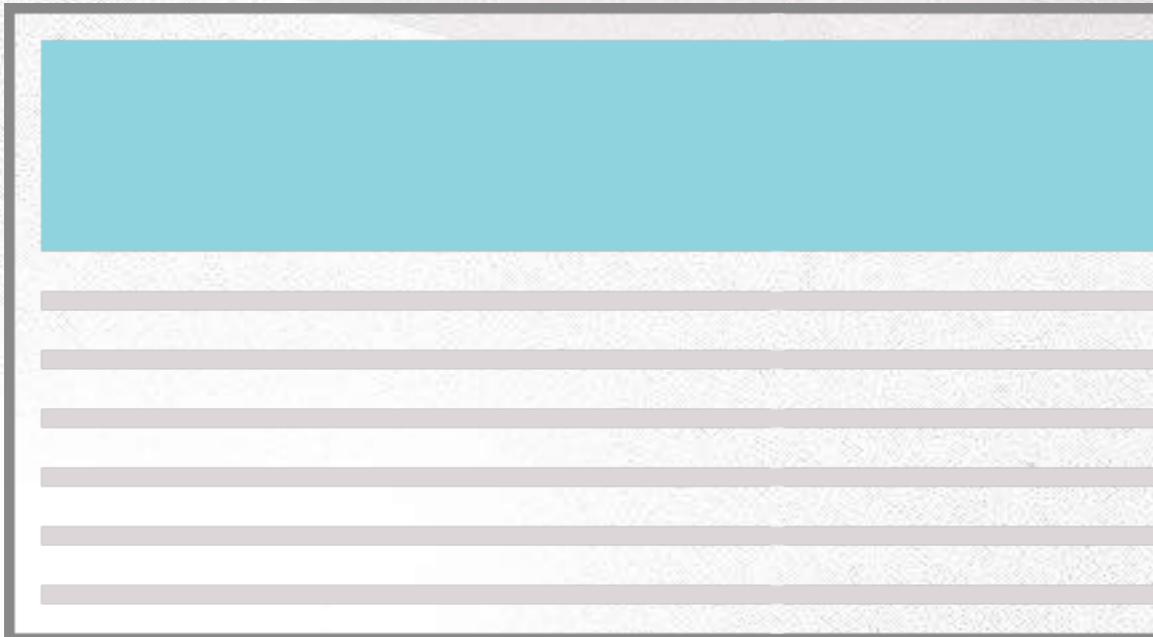
- × auto - adds a scrollbar as needed when content overflows



THE BOX MODEL

The overflow property:

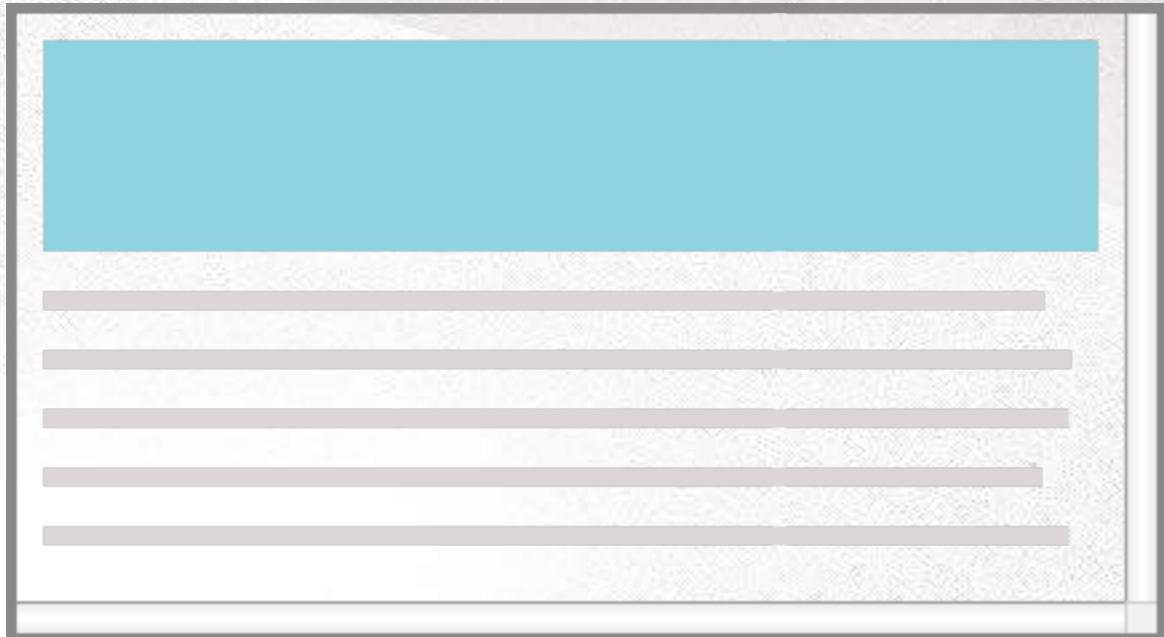
- × hidden - hides content that extends beyond the container



THE BOX MODEL

The overflow property:

- × scroll - adds a scrollbar at all times, even if unneeded



BOX BINDINGS

The Box Model

Positioning

Z-Index

POSITIONING

Elements have a position value of static by default:

position: static / relative / absolute / fixed

- ✖ Using a value other than static causes an object to become a *positioned element*
- ✖ Positioned elements may use the top, left, bottom, and right properties for placement

POSITIONING

Relative positioning:

- × Renders in the normal flow, then shifted via positioning properties

```
<article>
  <h2>Sven's SnowshoeX2</sup></h2>
</article>
```

```
sup {
  font-size: 75%;
  line-height: 0;
  vertical-align: baseline;
  position: relative;
  top: -0.5em;
}
```

Link #1

Sven's SnowshoeX²



POSITIONING

Absolute positioning:

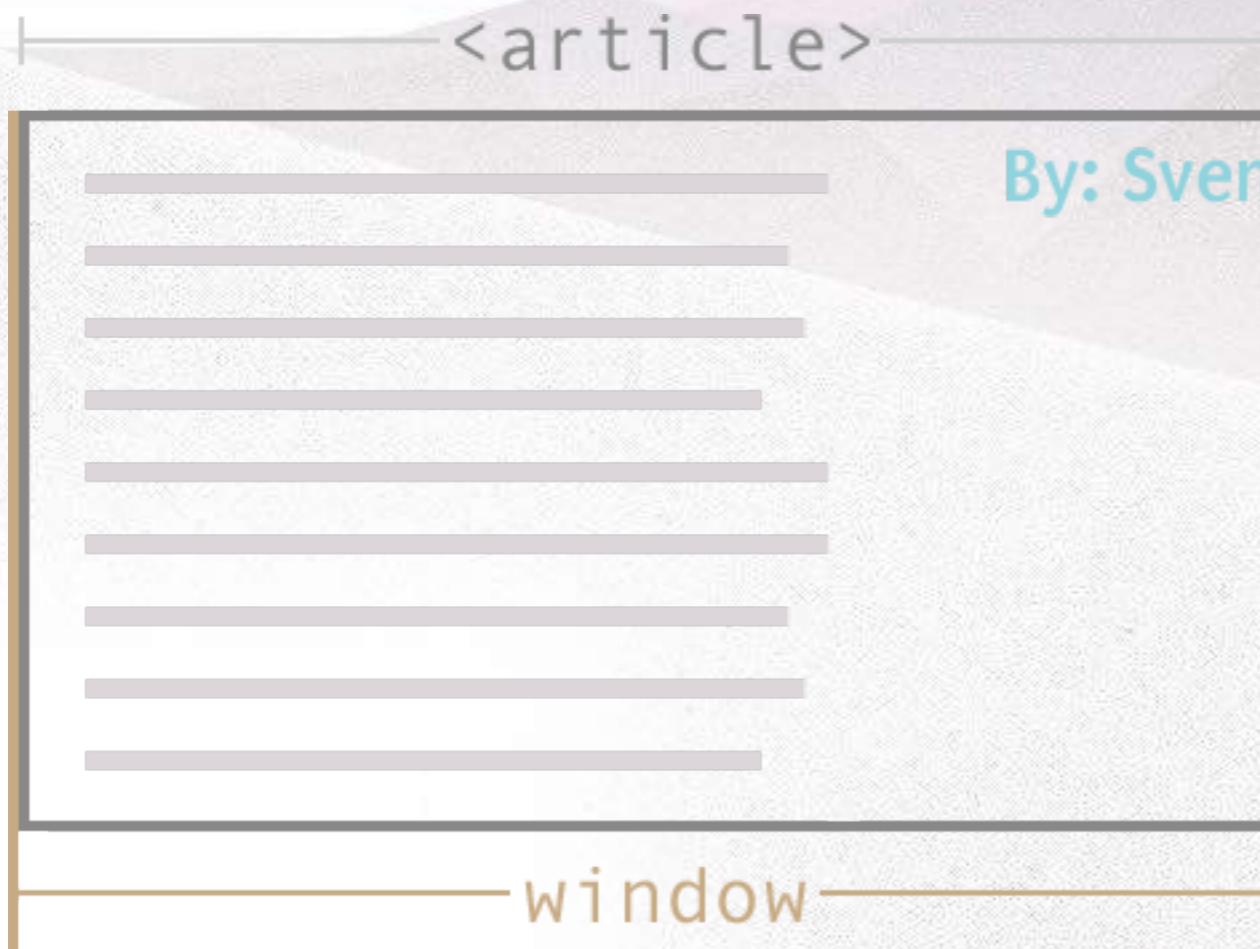
- × Takes an element out of the normal flow for manual positioning

```
<article>
  <h2>New Snowshoes</h2>
  <h3>By: Sven</h3>
  <p>This season's hot styles, available now!</p>
</article>
```

```
h3 {
  position: absolute;
  right: 10px;
  top: 10px;
}
```

POSITIONING

Absolute positioning:



```
h3 {  
  position: absolute;  
  right: 10px;  
  top: 10px;  
}
```

POSITIONING

Absolute positioning:



```
h3 {  
    position: absolute;  
    right: 10px;  
    top: 10px;  
}
```

POSITIONING

Absolute positioning:



```
article {  
  position: relative;  
}  
h3 {  
  position: absolute;  
  right: 10px;  
  top: 10px;  
}
```

POSITIONING

Fixed positioning:

- × Affixes an element to a specific place in the window, where it will stay regardless of scrolling

```
.ski {  
  position: fixed;  
  right: 10px;  
  top: 10px;  
}
```



BOX BINDINGS

The Box Model

Positioning

Z-Index

Z-INDEX

Manually adjusting overlap:

- ✗ No z-index or equal z-index = overlap determined by placement in DOM

```
<article>
  
  
</article>
```

```
.ski, .sled {
  z-index: 1;
}
```



Z-INDEX

Manually adjusting overlap:

- × Higher values appear above lower values

```
<article>
  
  
</article>
```

```
.ski {
  z-index: 1;
}
```



Z-INDEX

Manually adjusting overlap:

- ✗ Elements must be positioned for z-index to take effect. Use relative if you're not interested in moving the object

STACKING ORDER

Learn more about how floats, positioning and z-index stacks:

[go.codeschool.com/
understanding-z-index](http://go.codeschool.com/understanding-z-index)



CROSS X COUNTRY

GROOMING YOUR CODE



GROOMING YOUR CODE

Staying DRY

Display Types

Centering

STAYING DRY

D - Don't

R - Repeat

Y - Yourself

STAYING DRY

Suppose you want to include a font property in a number of text-related elements:

```
<body>
  <div class="container">
    <h1>Sven's Snowshoe Superstore</h1>
    <p>Our snowshoes are so stylish!</p>
    <a class="sale" href="/sale">View our sales</a>
  </div>
</body>
```

STAYING DRY

```
h1 {  
    font-family: Arial, sans-serif;  
}  
  
p {  
    font-family: Arial, sans-serif;  
}  
  
.sale {  
    font-family: Arial, sans-serif;  
}
```

```
body {  
    font-family: Arial, sans-serif;  
}
```

```
<body>  
    <div class="conta  
        > <h1>Sven's Snows  
        > <p>Our snowshoes  
        > <a class="sale"  
    </div>  
</body>
```



STAYING DRY

You can also target another parent element:

```
.container {  
    font-family: Arial, sans-serif;  
}
```

```
<body>  
    <div class="container">  
        <h1>Sven's Snowshoe Superstore</h1>  
        <p>Our snowshoes are so stylish!</p>  
        <a class="sale" href="/sale">View our sales</a>
```

STAYING DRY

Selector combination:

```
p {  
    font-size: 12px;  
}
```

```
.ski_lift {  
    font-size: 12px;  
}
```

```
h6 {  
    font-size: 12px;  
}
```

```
p, .ski_lift, h6 {  
    font-size: 12px;  
}
```

STAYING DRY

Selector abstraction:

```
.submit { border: 1px solid #000; cursor: pointer;  
text-transform: uppercase;  
}  
  
.next_button { border: 1px solid #000; cursor: pointer;  
text-transform: uppercase;  
}
```

!

```
<input type="submit" class="submit" />  
<a href="next_page.html" class="next_button" >Next</a>
```

STAYING DRY

DRY this up by abstracting your the CSS declarations into one class:

```
.button { border: 1px solid #000; cursor: pointer;  
text-transform: uppercase;  
}
```

```
<input type="submit" class="button"/>  
<a href="next_page.html" class="button">Next</a>
```

STAYING DRY

What if the buttons are only mostly the same?

```
.button {  
background: #fff;  
border: 1px solid #000;  
color: #333;  
cursor: pointer;  
}
```

```
<input type="submit" class="button" />  
<a href="next_page.html" class="button">Next</a>
```

STAYING DRY

```
.button {  
    background: #fff;  
    border: 1px solid #000;  
    color: #333;  
    cursor: pointer;  
}  
.submit {  
    background: #555;  
}
```

*order
matters
here*

```
<input type="submit" class="button submit" />  
<a href="next_page.html" class="button">Next</a>
```

STAYING DRY

Style shorthand - margin:

```
.ski_poles { margin-top: 15px; margin-left: 20px;  
margin-right: 10px; margin-bottom: 0; }
```

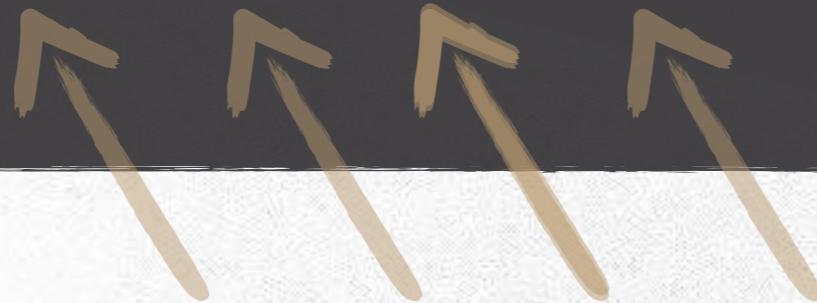
Can be *re-written* as:

```
.ski_poles {  
    margin: 15px 10px 0 20px; /* top right bottom left */  
}
```

STAYING DRY

Top - Right - Bottom - Left:

```
.ski_poles {  
  margin: 15px 10px 0 20px; /* top right bottom left */  
}
```



STAYING DRY

Selected CSS property shorthands:

```
font: 16px/18px bold italic sans-serif;  
/* size/line-height weight style family */  
  
background: #000 url(image.jpg) no-repeat center top;  
/* color image repeat x-pos y-pos */  
  
list-style: disc inside none;  
/* style position image */  
  
margin or padding: 0 10px 0 10px / 0 10px 0 / 0 10px;  
/* top right bottom left / top right&left bottom / top&bottom right&left */  
  
border: 3px solid #ccc;  
/* width style color */
```

GROOMING YOUR CODE

Staying DRY

Display Types

Centering

DISPLAY TYPES

Display:

```
display: none / block / inline / inline-block
```

DISPLAY TYPES

```
<div class="container">
  <h1>Sven's Snowshoe Superstore</h1>
  <p>Our snowshoes are so stylish!</p>
  <a class="sale" href="/sale">View our sales</a>
</div>
```

```
.sale {
  float: left;
  width: 100%;
  padding: 0 10px;
  margin: 5px 0;
  text-align: center;
}
```

Sven's Snowshoe Superstore

Our snowshoes are so stylish!

View our sales

DISPLAY TYPES

```
<div class="container group">
  <h1>Sven's Snowshoe Superstore</h1>
  <p>Our snowshoes are so stylish!</p>
  <a class="sale" href="/sale">View our sales</a>
</div>
```

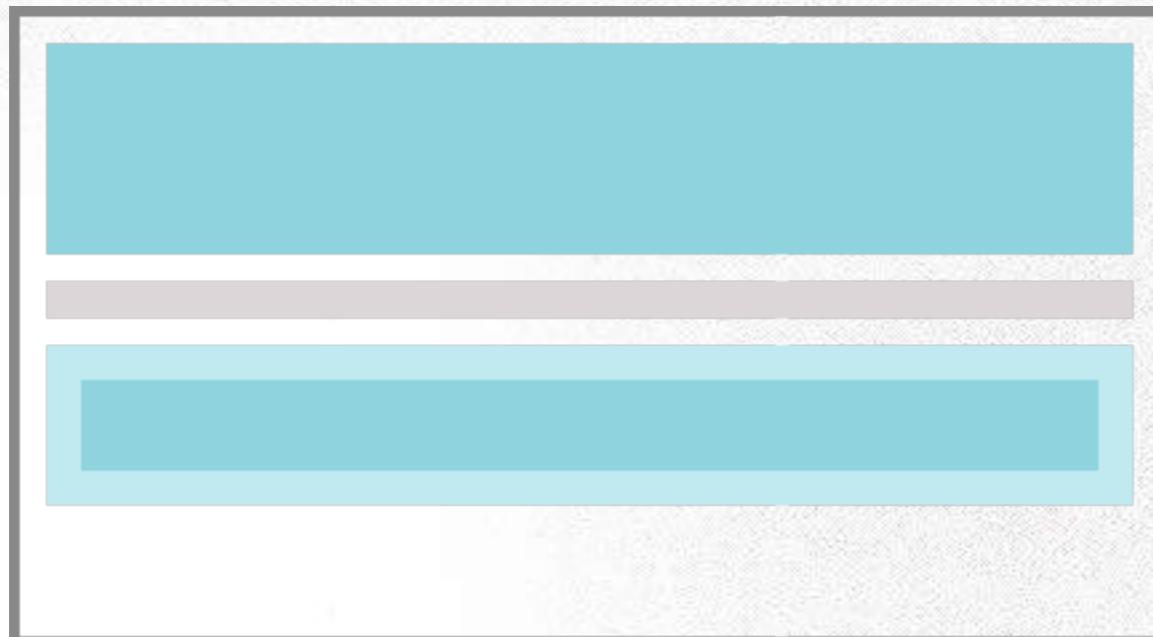
```
.sale {
  float: left;
  width: 100%;
  padding: 0 10px;
  margin: 5px 0;
  text-align: center;
}
```



DISPLAY TYPES

Block elements:

- × Stretch the full width of their container
- × Behave as though there is a line break before and after the element
- × Full box model can be manipulated



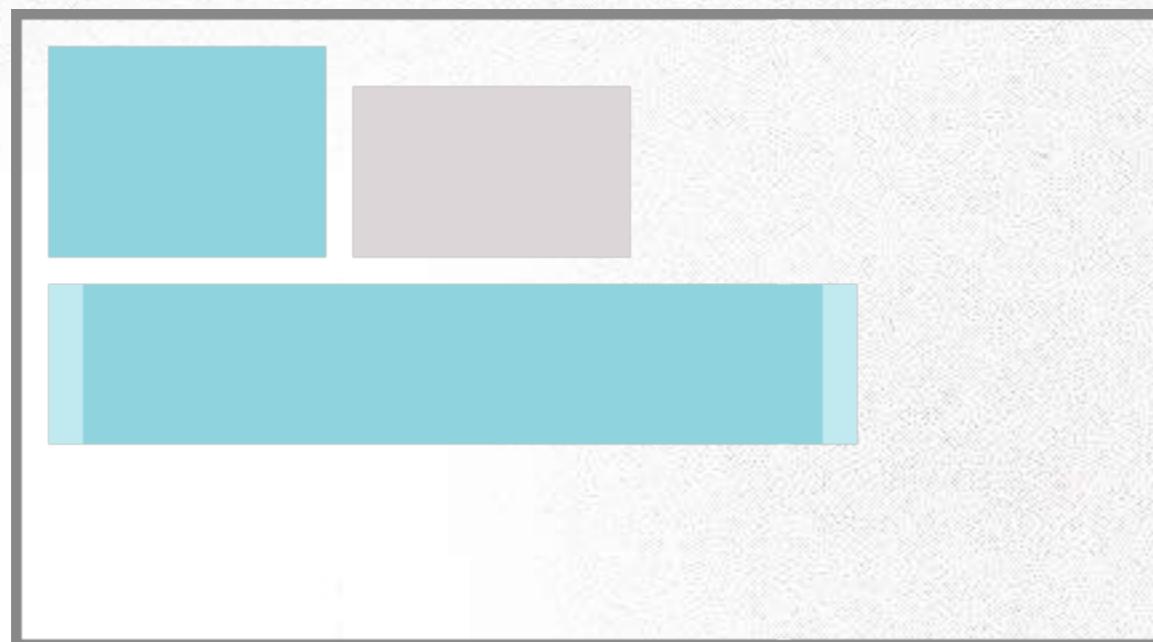
BLOCK ELEMENTS

Tags that are block-level by default:
`<div>`, `<p>`, ``, ``, `` and
`<h1>` through `<h6>`.

DISPLAY TYPES

Inline elements:

- × Typically found within block-level elements
- × Only take up the space of the content inside
- × Do not generate a line break before and after the content



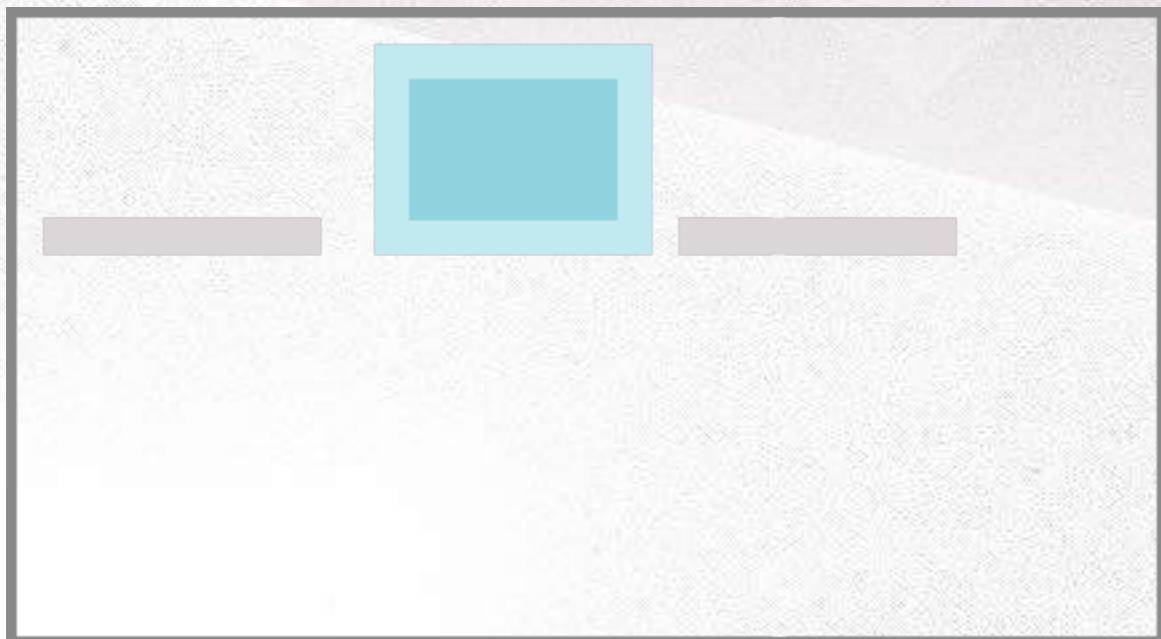
INLINE ELEMENTS

Tags that are inline by default include ``, `<a>`, ``, ``, and ``.

DISPLAY TYPES

Inline-block

- × Same flow as an inline element but behave as a block element



DISPLAY TYPES

```
<div class="container group">
  <h1>Sven's Snowshoe Superstore</h1>
  <p>Our snowshoes are so stylish!</p>
  <a class="sale" href="/sale">View our sales</a>
</div>
```

```
.sale {
  float: left;
  width: 100%;
  padding: 0 10px;
  margin: 5px 0;
  text-align: center;
}
```



DISPLAY TYPES

```
<div class="container group">
  <h1>Sven's Snowshoe Superstore</h1>
  <p>Our snowshoes are so stylish!</p>
  <a class="sale" href="/sale">View our sales</a>
</div>
```

```
.sale {
  display: block; ←
  padding: 0 10px;
  margin: 5px 0;
  text-align: center;
}
```

DISPLAY TYPES

```
<div class="container">
  <h1>Sven's Snowshoe Superstore</h1>
  <p>Our snowshoes are so stylish!</p>
  <a class="sale" href="/sale">View our sales</a>
</div>
```

```
.sale {
  display: block;
  padding: 0 10px;
  margin: 5px 0;
  text-align: center;
}
```



GROOMING YOUR CODE

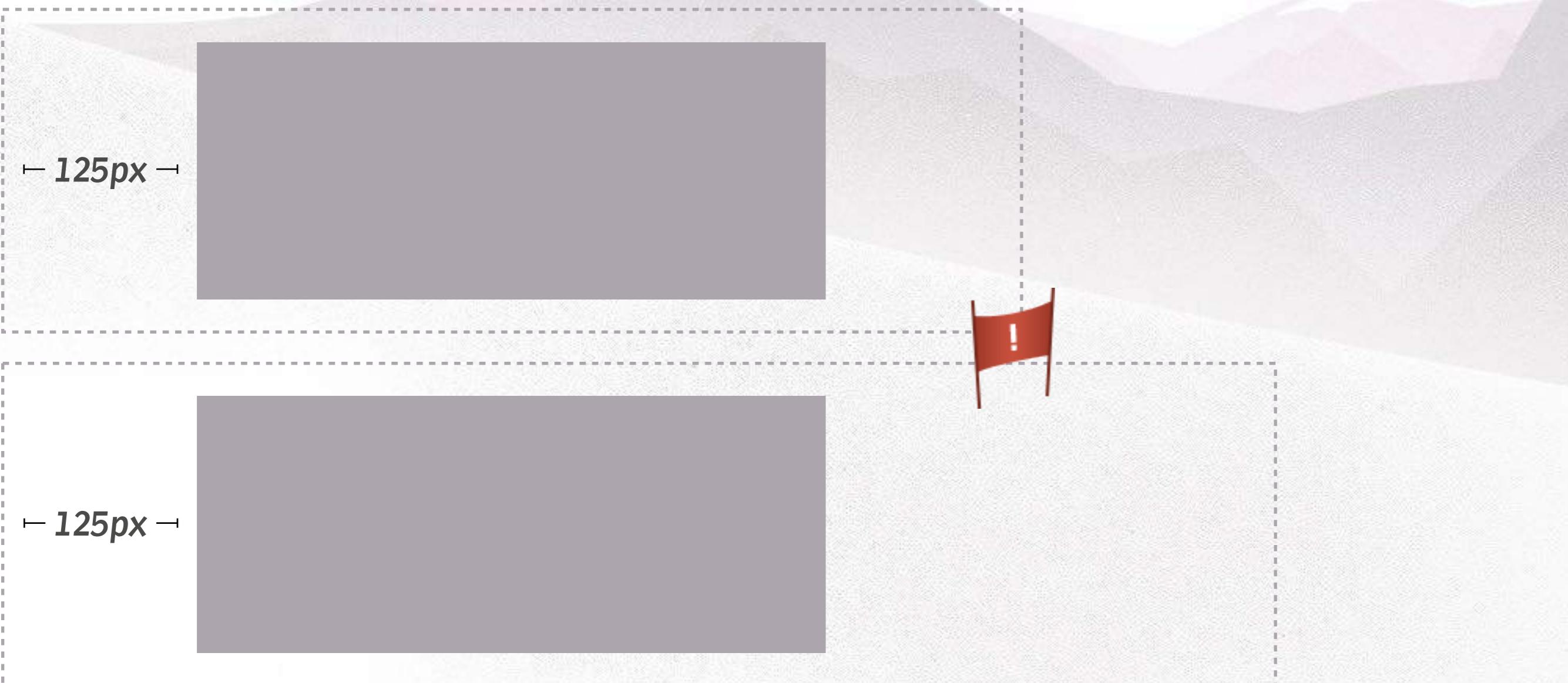
Staying DRY

Display Types

Centering

CENTERING

Centering a block-level element:



CENTERING

Centering a block-level element:



- ✗ Define a width, and the element width must be less than that of the parent container
- ✗ `margin: 0 auto;`

CENTERING

Centering inline and inline-block elements:



SVEN'S SNOWSHOE SUPERSTORE

✗ `text-align: center`



CROSS X COUNTRY

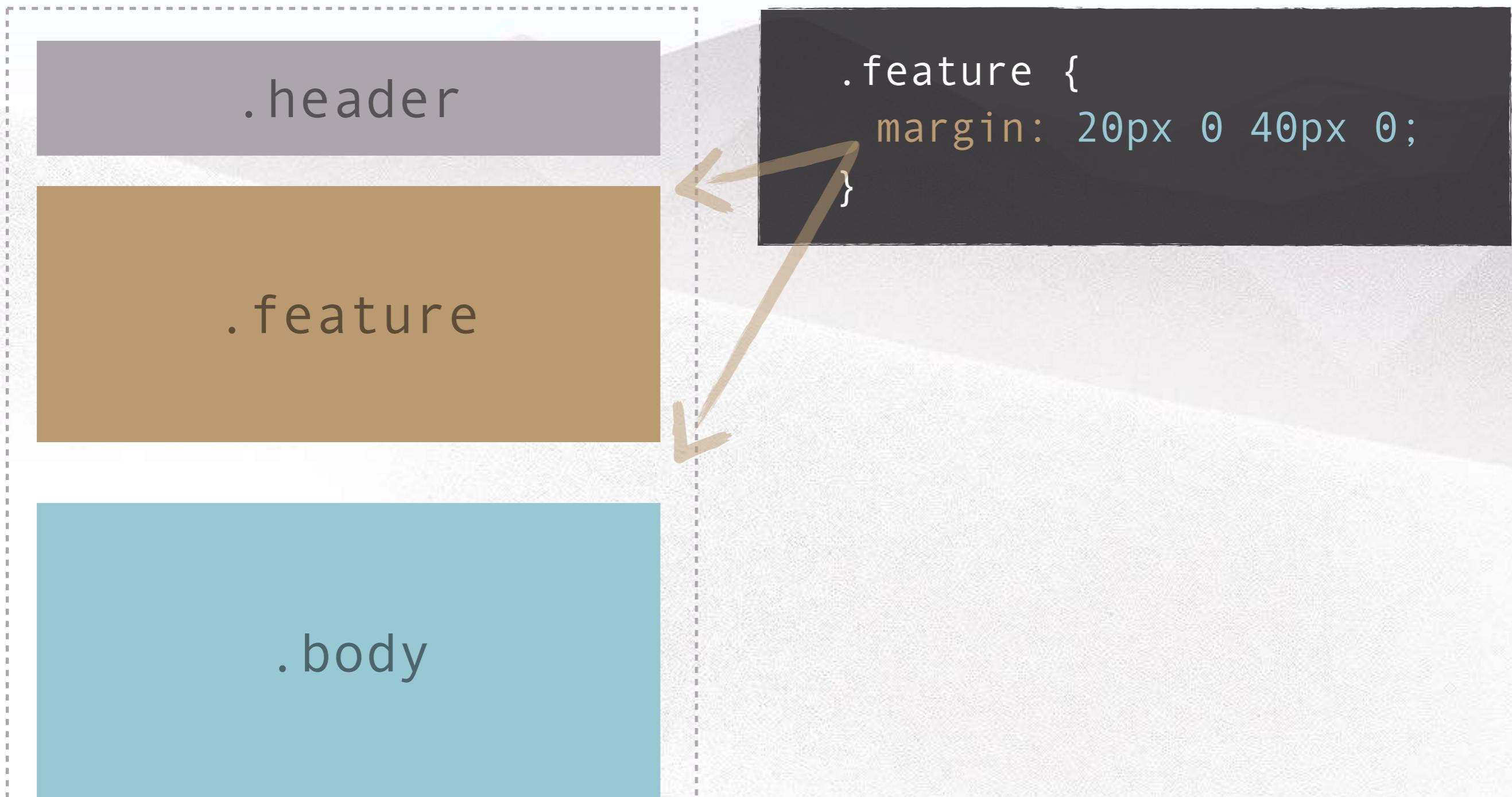
**CSS
SAFETY**

CSS SAFETY

Protecting Your Layout

Specificity Problems

PROTECTING YOUR LAYOUT



PROTECTING YOUR LAYOUT

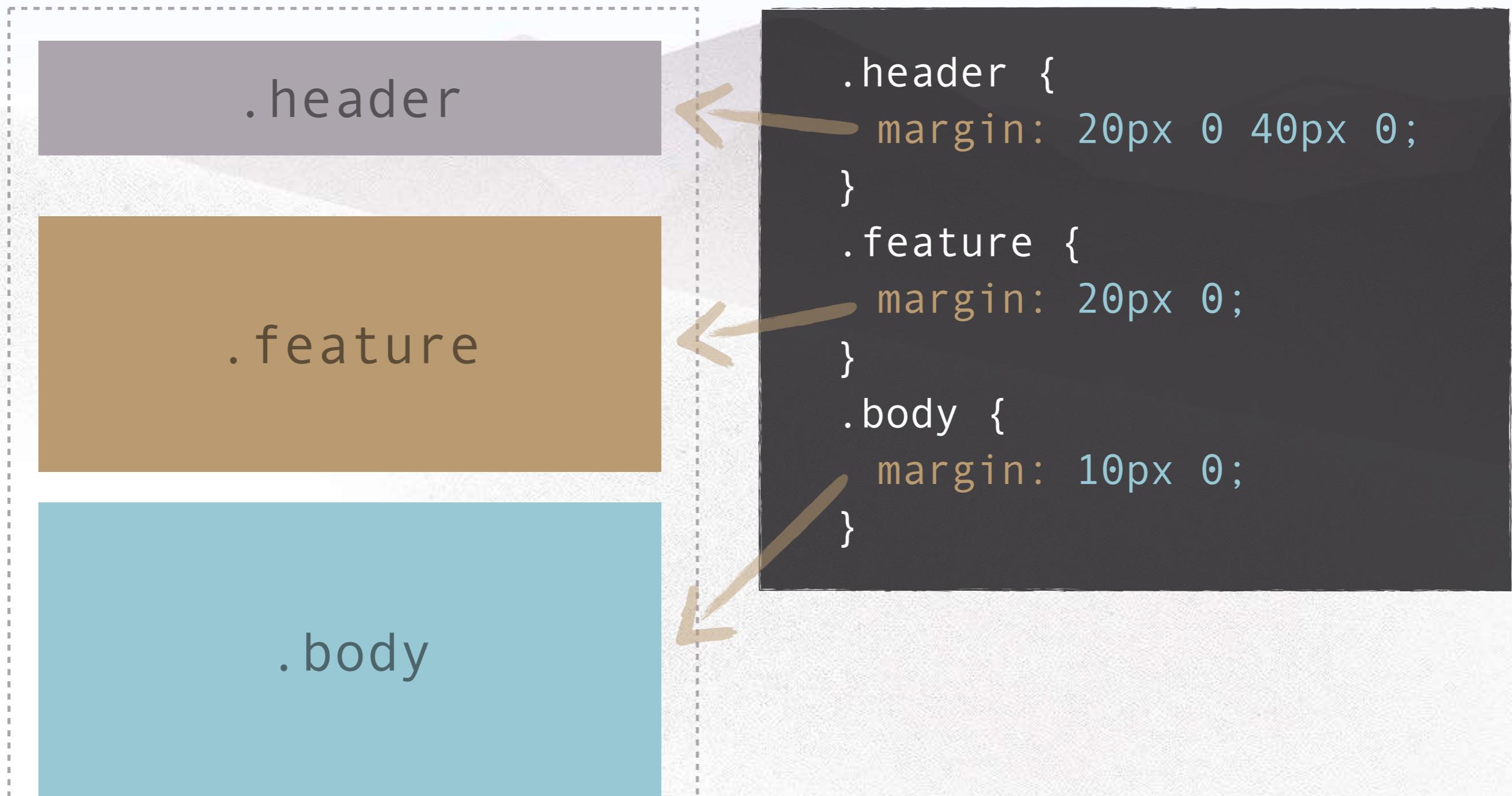
.header

!

.body

```
.feature {  
    margin: 20px 0 40px 0;  
}
```

PROTECTING YOUR LAYOUT



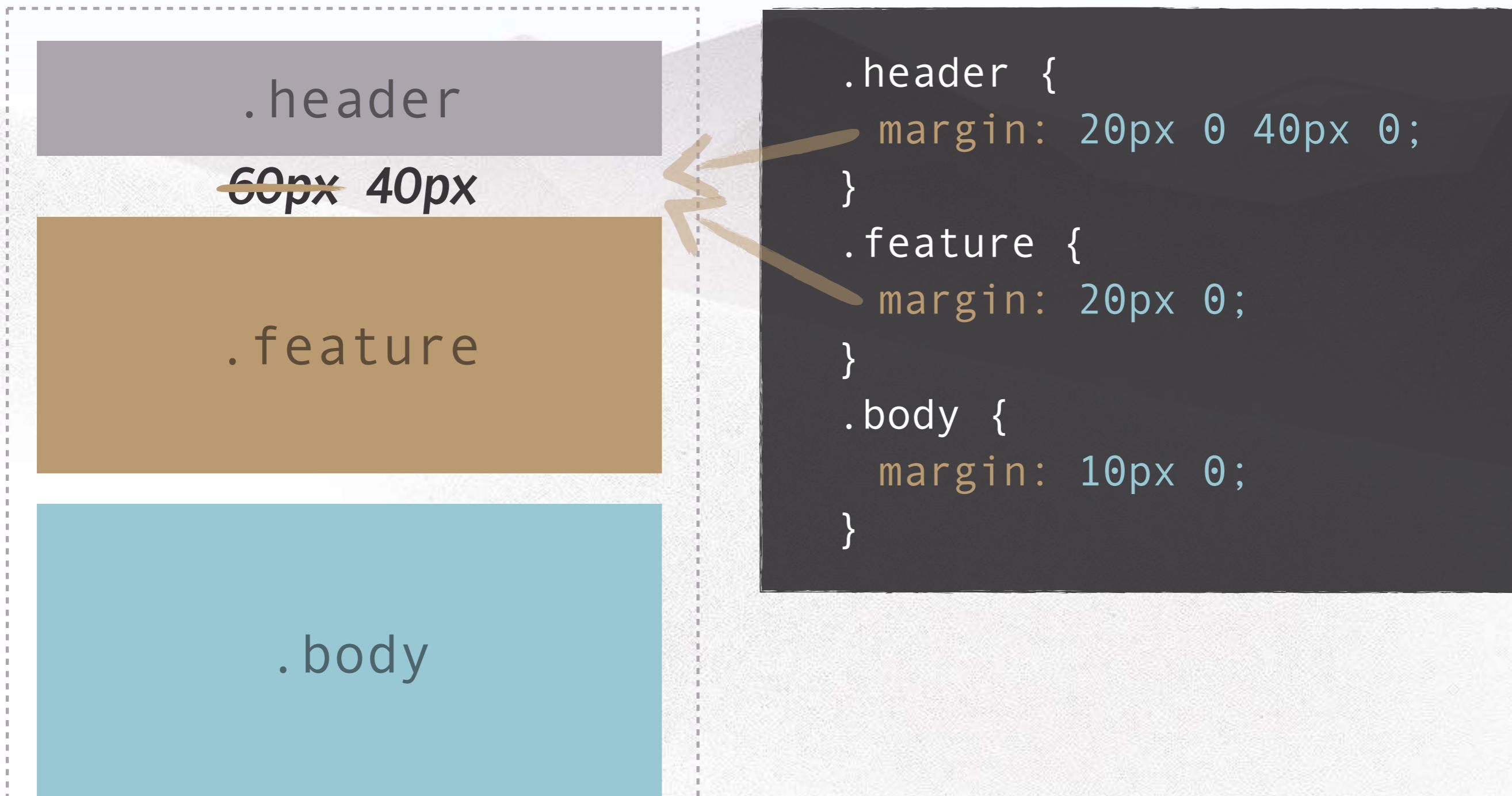
PROTECTING YOUR LAYOUT

.header

.body

```
.header {  
  margin: 20px 0 40px 0;  
}  
.feature {  
  margin: 20px 0;  
}  
.body {  
  margin: 10px 0;  
}
```

PROTECTING YOUR LAYOUT



PROTECTING YOUR LAYOUT



```
.header {  
  margin: 20px 0 40px 0;  
}  
.feature {  
  margin: 20px 0;  
}  
.body {  
  margin: 10px 0;  
}
```

PROTECTING YOUR LAYOUT

.header

.feature

.body

```
.header {  
  margin: 20px 0 40px 0;  
}  
.feature {  
  margin: 20px 0;  
}  
.body {  
  margin: 10px 0;  
}
```

PROTECTING YOUR LAYOUT

Collapsing margins will not occur when one or more block element has:

- ✗ Padding or border
- ✗ Relative or absolute positioning
- ✗ A float left or right

COLLAPSING MARGINS

See the spec for more info:

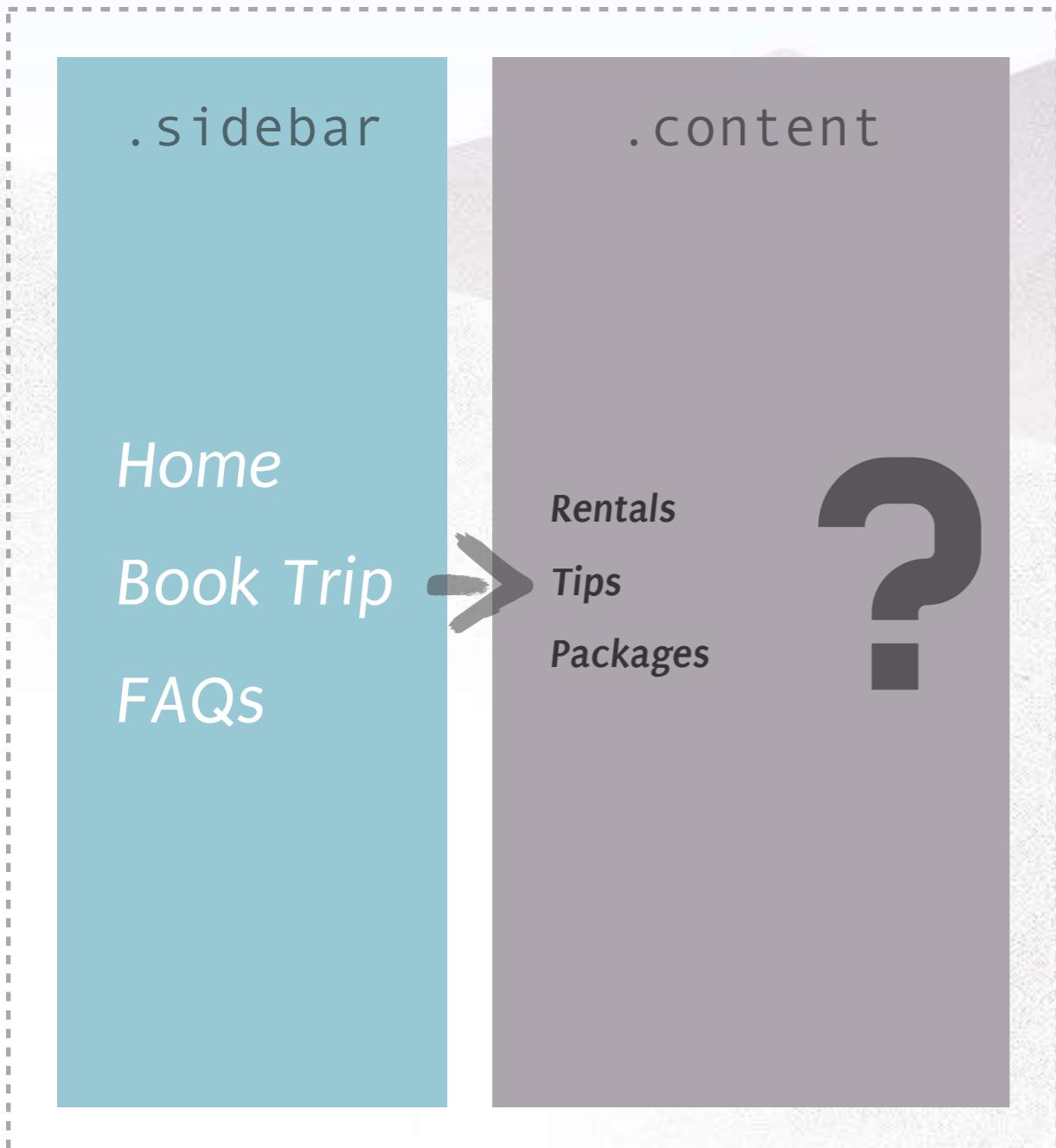
go.codeschool.com/collapsing-margins

CSS SAFETY

Protecting Your Layout

Specificity Problems

SPECIFICITY PROBLEMS



```
.sidebar ul li {  
    font-style: italic;  
    font-size: 14px;  
    color: #fff;  
    margin: 5px 0;  
}
```

SPECIFICITY PROBLEMS

.sidebar

Home

Book Trip ➔

FAQs

.content

Rentals

Tips

Packages

```
ul li {  
  font-style: italic;  
  font-size: 14px;  
  color: #fff;  
  margin: 5px 0;  
}
```

!

SPECIFICITY PROBLEMS

.sidebar

ul.featured

Home

Book Trip

FAQs

.content

ul.featured

Rentals

Tips

Packages

```
.featured li {  
    font-style: italic;  
    font-size: 14px;  
    color: #fff;  
    margin: 5px 0;  
}
```



SPECIFICITY PROBLEMS

.sidebar

ul.featured

Home

Book Trip

FAQs

.content

ul.featured

Rentals

Tips

Packages

```
.featured li {  
    font-style: italic;  
    font-size: 14px;  
    color: #fff;  
    margin: 5px 0;  
}  
ul li {  
    font-size: 12px;  
    margin: 3px 0;  
}
```



SPECIFICITY PROBLEMS

Resets & normalization:

- ✗ Eric Meyer's Reset CSS: go.codeschool.com/reset-css
- ✗ Normalize.css: go.codeschool.com/normalize-css



CROSS X COUNTRY

IMAGE ISSUES



IMAGE ISSUES

Image Use

Image Cropping

IMAGE USE

Layout or content?

```
<h4>Rental Products</h4>
<ul>
  <li class="snowmobile"></li>
</ul>
```



```
.snowmobile li {
  background: url(snowmobile.jpg);
  height: 300px;
  width: 400px;
}
```

IMAGE USE

Write it as an inline image instead:

```
<h4>Rental Products</h4>
<ul>
  <li></li>
</ul>
```



IMAGE USE

```
<h1>Feel the rhythm, feel the rhyme, get on up, it's  
bobsled time!</h1>  

```

!

IMAGE USE

Better as a background image:

```
<h1>Feel the rhythm, feel the rhyme, get on up, it's  
bobsled time!</h1>
```

```
h1 {  
background: url(divider.jpg);  
margin-bottom: 10px;  
padding-bottom: 10px;  
}
```



IMAGE USE

Review:

- ✗ Content should be marked up as inline images
- ✗ Layout elements should be defined as background images

IMAGE ISSUES

Image Use

Image Cropping

IMAGE CROPPING

IMAGE CROPPING



IMAGE CROPPING

```
.rental img {  
    height: 300px;  
    width: 400px;  
}
```



IMAGE CROPPING



IMAGE CROPPING

Overflow crop method:

```
<h4>Rental Products</h4>
<ul class="rental">
  <li class="crop">
    
  </li>
</ul>

.crop {
  height: 300px;
  width: 400px;
  overflow: hidden;
}
```

IMAGE CROPPING



IMAGE CROPPING

```
<h4>Rental Products</h4>
<ul class="rental">
  <li class="crop">
    
  </li>
```

```
.crop {
  height: 300px;
  width: 400px;
  overflow: hidden;
}
.crop img {
  height: auto;
  width: 400px;
}
```

6 |



IMAGE CROPPING



IMAGE CROPPING



IMAGE CROPPING

```
<h4>Rental Products</h4>
<ul class="rental">
  <li class="crop">
    
  </li>
```

```
.crop {
  height: 300px;
  width: 400px;
  overflow: hidden;
}
.crop img {
  height: 300px;
  width: auto;
}
```

6 |



IMAGE CROPPING

Alternatives:

- ✗ Resize images to a square < height and width of all of your images
- ✗ Resize them server-side
- ✗ Provide image-uploading instructions in your CMS

FLUID & RESPONSIVE LAYOUTS

Richard Rutter's technique:

clagnut.com/sandbox/imagetest



CROSS X COUNTRY

**SPRIGGY
SLALOMS**

SPRIGHTLY SLALOMS

Image Replacement

Sprites

IMAGE REPLACEMENT

Add descriptive text to image-replaced elements:

```
<a href="#" class="logo"></a>
```

```
.logo {  
background: url(logo.png);  
display: block;  
height: 100px;  
width: 200px;  
}
```



IMAGE REPLACEMENT

Add descriptive text to image-replaced elements:

```
<a href="#" class="logo">Sven's Snowshoe Emporium</a>
```

```
.logo {  
background: url(logo.png);  
display: block;  
height: 100px;  
width: 200px;  
}
```



IMAGE REPLACEMENT

Text-indent hides the placeholder text:

```
<a href="#" class="logo">Sven's Snowshoe Emporium</a>
```

```
.logo {  
background: url(logo.png);  
display: block;  
height: 100px;  
width: 200px;  
text-indent: -9999px;  
}
```



SPRIGHTLY SLALOMS

Image Replacement

Sprites

SPRITES

```
<a href="#" class="logo">Sven's Snowshoe Emporium</a>
```

```
.logo {  
background: url(logo.png);  
display: block;  
height: 100px;  
width: 200px;  
text-indent: -9999px;  
}  
.logo:hover, .logo:focus {  
background: url(hover.png);  
}
```



SPRITES

Issues with swapping background images:

- ✗ Adds an extra HTTP request
- ✗ Image is not preloaded



SPRITES

Let's combine the images into one file:



SPRITES

defaults to: 0 0

Using background-position:

```
.logo {  
background: url(logo.png);  
display: block;  
height: 100px;  
width: 200px;  
text-indent: -9999px;  
}  
.logo:hover, .logo:focus {  
background-position: 0 -100px;  
}
```

x-axis change
y-axis change



SPRITES

Using background-position:

```
.logo {  
background: url(logo.png);  
display: block;  
height: 100px;  
width: 200px;  
text-indent: -9999px;  
}  
.logo:hover, .logo:focus {  
background-position: 0 -100px;  
}
```

defaults to: 0 0



x-axis change
y-axis change



SPRITES

Advantages to the sprite approach:

- ✗ Reduces number of HTTP image requests
- ✗ Removes loading flash / need for preload



Gmail



GitHub

SPRITES

Multiple images & states:



SPRITES

Multiple images & states:

```
<ul class="group">  
  <li><a href="#" class="twitter">Twitter</a></li>  
  <li><a href="#" class="github">Github</a></li>  
</ul>
```

```
li {  
  float: left;  
  margin-right: 10px;  
}
```



Twitter Github !

assumes the default list styles are reset

SPRITES

Multiple images & states:

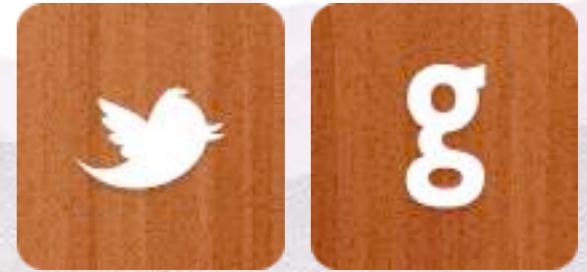
```
.twitter, .github {  
background: url(social.png);  
display: block;  
height: 100px;  
width: 100px;  
text-indent: -9999px;  
}
```



SPRITES

Multiple images & states:

```
.twitter, .github {  
background: url(social.png);  
display: block;  
height: 100px;  
width: 100px;  
text-indent: -9999px;  
}  
.github {  
background-position: -100px 0;  
}
```



SPRITES

Multiple images & states:

```
.twitter:hover, .twitter:focus {  
    background-position: 0 -100px;  
}  
.github:hover, .github:focus {  
    background-position: -100px -100px;  
}
```



SPRITES

Multiple images & states:

```
.twitter:hover, .twitter:focus {  
    background-position: 0 -100px;  
}  
.github:hover, .github:focus {  
    background-position: -100px -100px;  
}
```



EASIER SPRITES

Tools for simplifying workflow:

spritecow.com

SPRITES

base64 encoding:

- ✗ Directly embed images into your CSS
- ✗ IE8+

Link #2

```
background-image: url(data:image/png;base64,iVB0...);
```



CROSS X COUNTRY

**PSEUDO
STIZMARK**



PSEUDO SITZMARK

Pseudo Classes

Pseudo Elements

PSEUDO CLASSES

Altering the last item:

```
<ul>
  <li><a href="#lodging">Lodging</a></li>
  <li><a href="#rentals">Rentals</a></li>
  <li class="last"><a href="#lessons">Lessons</a></li>
</ul>
```

```
li {
  border-bottom: 1px solid #aaa;
}
.last {
  border-bottom: 0;
```

- **LODGING**

- **RENTALS**

- **LESSONS**



PSEUDO CLASSES

Simplifying with last-child:

```
<ul>
  <li><a href="#lodging">Lodging</a></li>
  <li><a href="#rentals">Rentals</a></li>
  <li><a href="#lessons">Lessons</a></li>
</ul>
```

```
li {
  border-bottom: 1px solid #aaa;
}
li:last-child {
  border-bottom: 0;
```

- **LODGING**

- **RENTALS**

- **LESSONS**



PSEUDO CLASSES

Allow you to conditionally select an element based on state or position:

- × Start with a colon (:)

```
.twitter:hover, .twitter:focus {  
    background-position: 0 -100px;  
}  
.github:hover, .github:focus {  
    background-position: -100px -100px;  
}
```

PSEUDO CLASSES

Zebra striping:

```
<ul>
  <li><a href="#lodging">Lodging</a></li>
  <li class="even"><a href="#rentals">Rentals</a></li>
  <li><a href="#lessons">Lessons</a></li>
  <li class="even"><a href="#menu">Menu</a></li>
</ul>
```

```
.even {
  background-color: #444245;
}
```

LODGING

RENTALS

LESSONS

MENU



PSEUDO CLASSES

Simplifying with nth-child:

```
<ul>
  <li><a href="#lodging">Lodging</a></li>
  <li><a href="#rentals">Rentals</a></li>
  <li><a href="#lessons">Lessons</a></li>
  <li><a href="#menu">Menu</a></li>
</ul>
```

```
li:nth-child(even) {
  background-color: #444245;
```



LODGING

RENTALS

LESSONS

MENU



PSEUDO CLASSES

Simplifying with nth-child:

```
li:nth-child(odd) {  
    background-color: #444245;  
}
```

LODGING

RENTALS

LESSONS

MENU

PSEUDO CLASSES

Simplifying with nth-child:

```
li:nth-child(an+b) {
```

- × Matches items in intervals of a , starting with the element at position b (or 0 , if b isn't set)

```
li:nth-child(2n) { /* Even */
```

```
li:nth-child(2n+1) { /* Odd */
```

PSEUDO CLASSES

A *selection of useful selectors:*

`:hover / :focus / :active / :visited`

`:first-child / :last-child / :only-child`

`:nth-child() / :nth-of-type()`

PSEUDO-ER

For a full list and support:

go.codeschool.com/css-pseudo-classes and

go.codeschool.com/css-reference

PSEUDO SITZMARK

Pseudo Classes

Pseudo Elements

PSEUDO ELEMENTS

Ending the last paragraph with a decorative element:

```
<article>
  <p>Coffee? Hah! Our cocoa is far better.</p>
  <p>Visit from 4-5 for cocoa happy hour!*</p>
</article>
```

Coffee? Hah! Our cocoa is far better.
Visit from 4-5 for cocoa happy hour!*



PSEUDO ELEMENTS

Ending the last paragraph with a decorative element:

```
<article>
  <p>Coffee? Hah! Our cocoa is far better.</p>
  <p>Visit from 4-5 for cocoa happy hour!</p>
</article>
```

```
article p:last-child:after {
  content: '\2744';
}
```

Coffee? Hah! Our cocoa is far better.

Visit from 4-5 for cocoa happy hour! 

PSEUDO ELEMENTS

A *selection of useful pseudo elements:*

:before / :after

IE8+

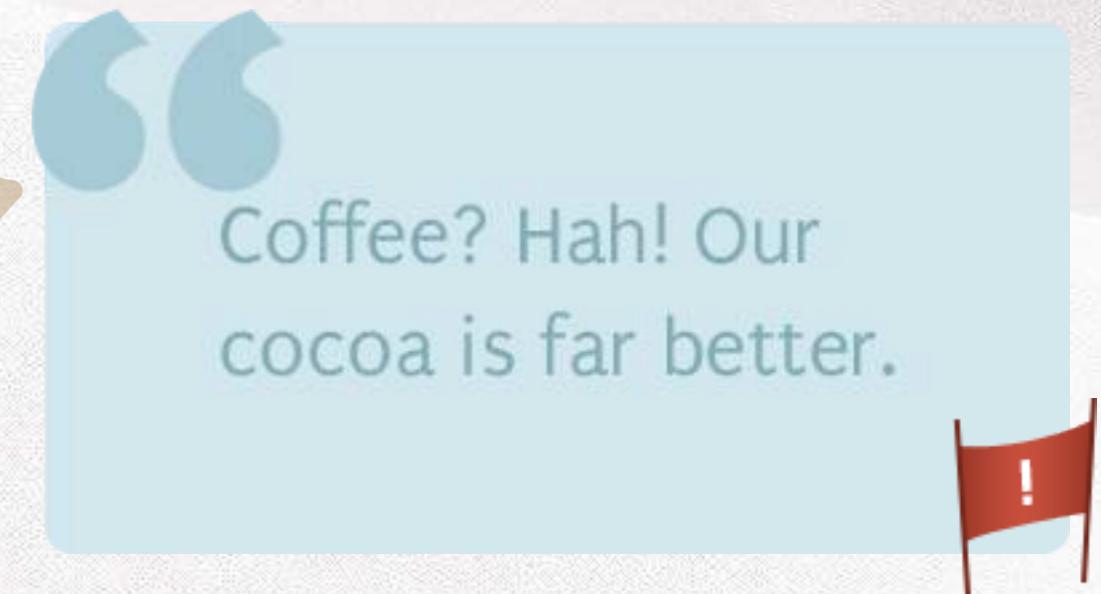
:first-letter / :first-line

PSEUDO ELEMENTS

positioned element

```
<blockquote> ←  
  Coffee? Hah! Our cocoa is far better.  
  <span></span>  
</blockquote>
```

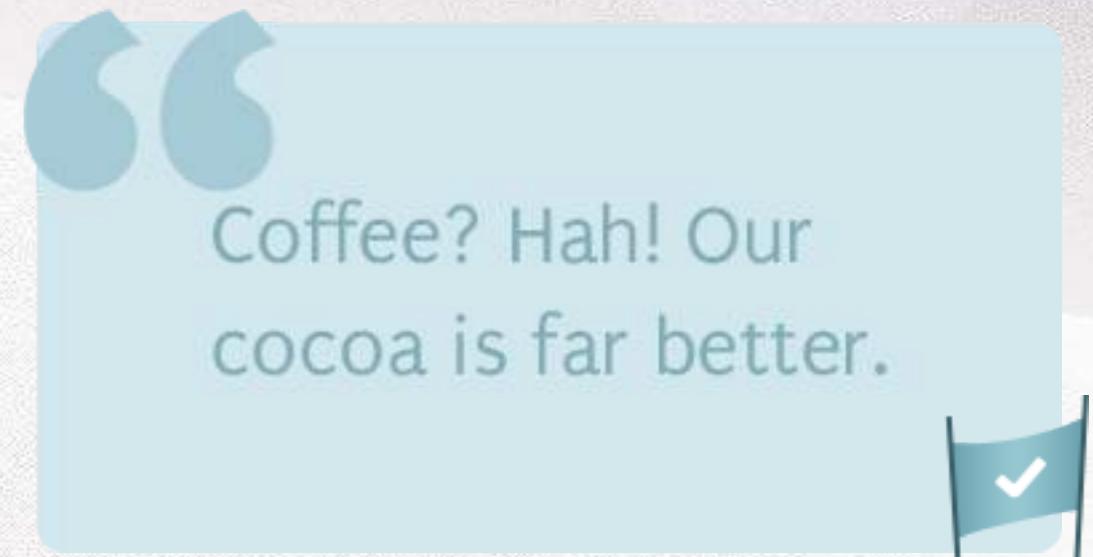
```
blockquote span {  
  background: url(quote.png);  
  display: block;  
  height: 30px;  
  width: 30px;  
  position: absolute;  
  top: -5px;  
  left: -5px;  
}
```



PSEUDO ELEMENTS

```
<blockquote>  
  Coffee? Hah! Our cocoa is far better.  
</blockquote>
```

```
blockquote:before {  
  content: ''; ←  
  background: url(quote.png);  
  display: block;  
  height: 30px;  
  width: 30px;  
  position: absolute;  
  top: -5px;  
  left: -5px;  
}
```



PSEUDO ELEMENTS

:before and :after effectively add two additional places per element on the page for styling

PSEUDO FUN

See what you can pull off using
pseudo elements:

go.codeschool.com/css-pseudo-elements



CROSS X COUNTRY