HTML, CSS, & Fundamentals of Development

School of Visual Concepts
Scott Thiessen, instructor

http://movingobjects.io/svc/
scott@movingobjects.io

Week 4

- Review
- Block vs Inline Elements
- Box Model
- CSS Positioning
- More CSS Tricks
- Coding from a Design file

Review

HTML Tags

```
<h1>    <strong>           <ul>
<h2>    <em>               <ol>
<h3>                       <li>
        <a href="">
<p>     <img src="">
```

CSS Review

```css
h1 {
    font-size: 32px;
}

body {
    background-color: black;
    font-family: serif;
    color: white;
}
```

CSS Review

Typography

`color`
`font-family`
`font-size`
`line-height`
`letter-spacing`

`font-style`
`font-weight`

`text-align`
`text-transform`
`text-decoration`

Sizing

`width`
`height`

Spacing

`margin`
`padding`

Backgrounds

`background-color`
`background-image`
`background-repeat`
`background-size`

Borders

`border`
`border-radius`

Special Selectors

`:hover`
`:active`

ID & Class

```
<p id="intro">
   The only intro paragraph.
</p>

<p class=“highlighted”>
    One of many possible
    highlighted paragraphs.
</p>
```

Select HTML Element

Use the element (tag) name

`<body>`	`body { }`
`<h1>`	`h1 { }`

Select by ID

Use the # sign

```
<div id="top">
```

```
div#top { }
#top { }
```

```
<ul id="nav">
```

```
ul#nav { }
#nav { }
```

Select by Class

Use a . (dot)

```
<a class="selected">    a.selected { }
                        .selected { }

<h1 class="special">    h1.special { }
                        .special { }
```

Select "descendent"

Use space between two selectors

```
<ul>
  <li>Item</li>
</ul>
```

```
ul li { }
```

```
<h1>
  <em>Hi</em> there!
</h1>
```

```
h1 em { }
```

CSS Selectors

Mix & Match

```
ul#nav li a { }
```

```
<ul id="nav">
  <li>
    <a href="#">Link</a>
  </li>
</ul>
```

Structural HTML Elements

New Tags

`<main>`

`<header>`

`<footer>`

`<section>`

`<article>`

`<div>`

Centering Your Website

HTML

```html
<body>
  <main>
    <p>Content</p>
    ...
  </main>
</body>
```

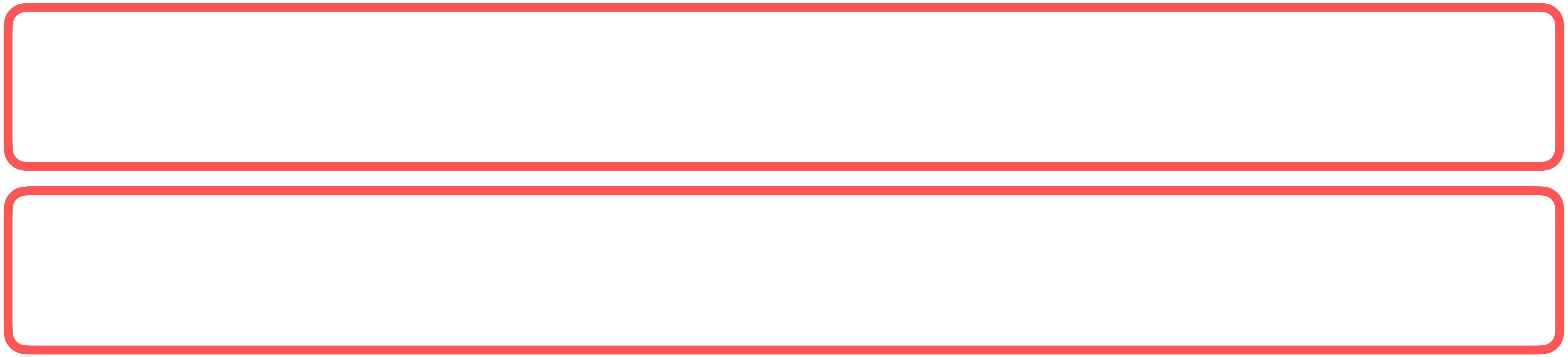
CSS

```css
main {
  width: 800px;
  margin-left: auto;
  margin-right: auto;
}
```

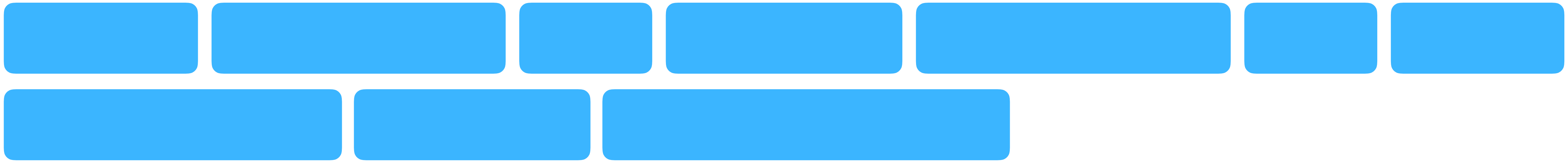
Block vs
Inline elements

Block vs Inline elements

Block



Inline



Block vs Inline elements

Block

- Create line-breaks
- Take up "space" on the page
- Fill the width of their container

Inline

- Don't create line-breaks
- Flow within other content on the page

Block vs Inline elements

Block

<h1><h2>

<p>

<ul><ol>
<li>

<main>
<section>

<div>

Inline

<strong> <em>

<a>

<img>

<span>

Text content

Block vs Inline elements

Protect Inline Elements

Wrap your inline elements inside block elements:

```
<img src="alone-and-sad.jpg">
```

```
<p><img src="happy.jpg"></p>
```

Block vs Inline elements

Never siblings

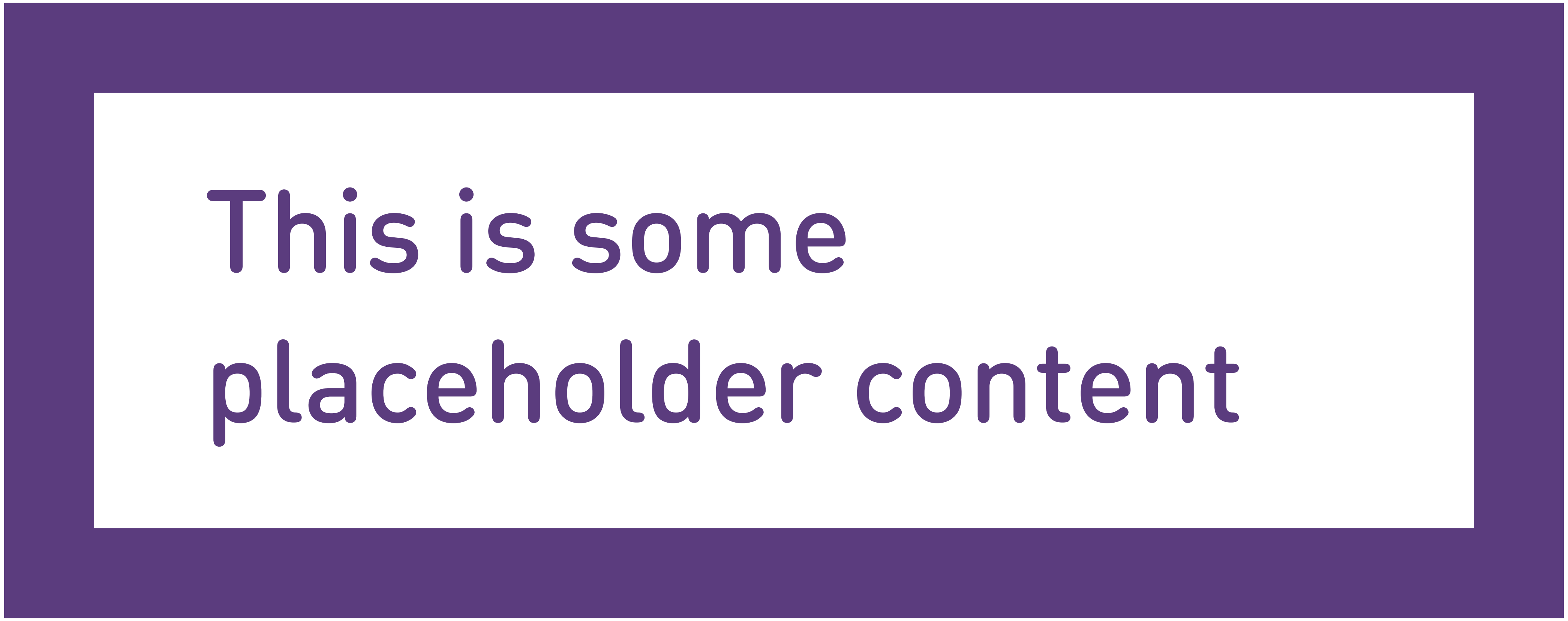
Inline and block elements are typically never siblings (adjacent in code). Block elements contain inline elements.

```
<p><img src="yes.jpg"></p>
```

```
<p class="no"></p>
<img src="so-awkward-rn.jpg">
```

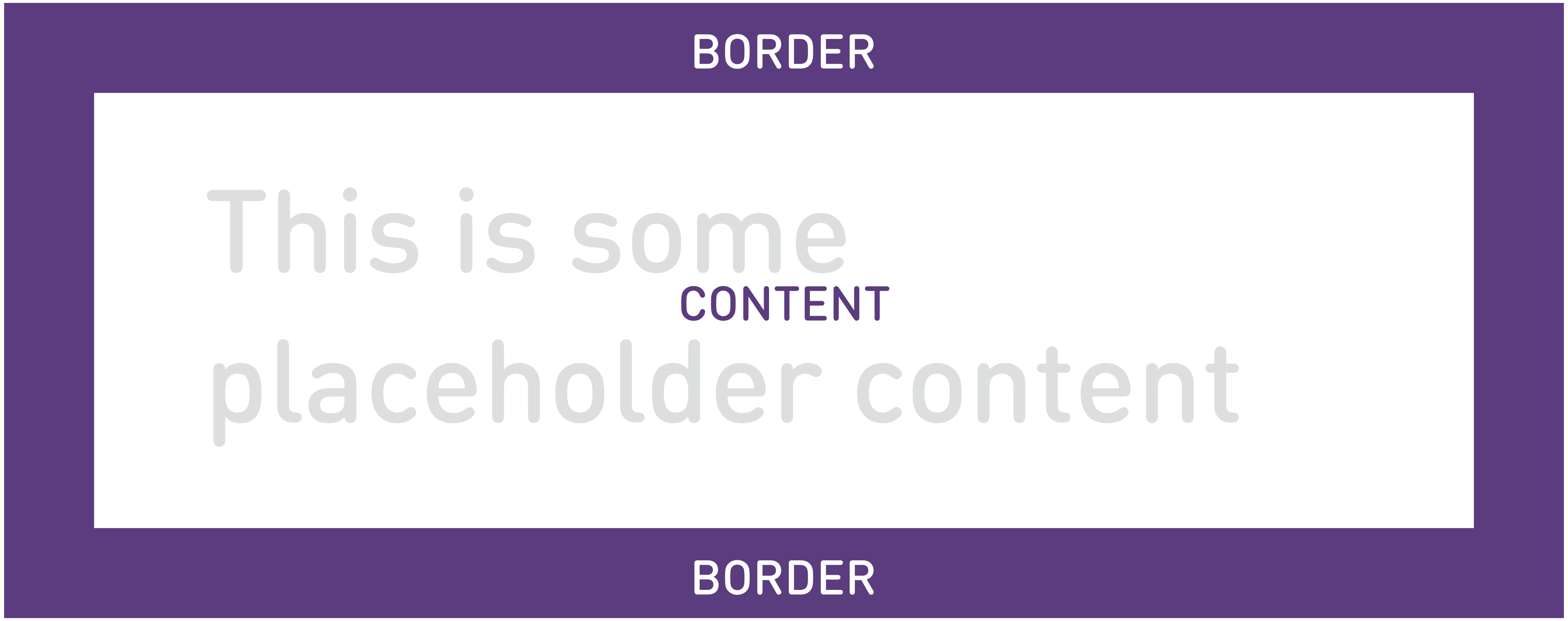
The
Box Model

The Box Model



This is some placeholder content

The Box Model



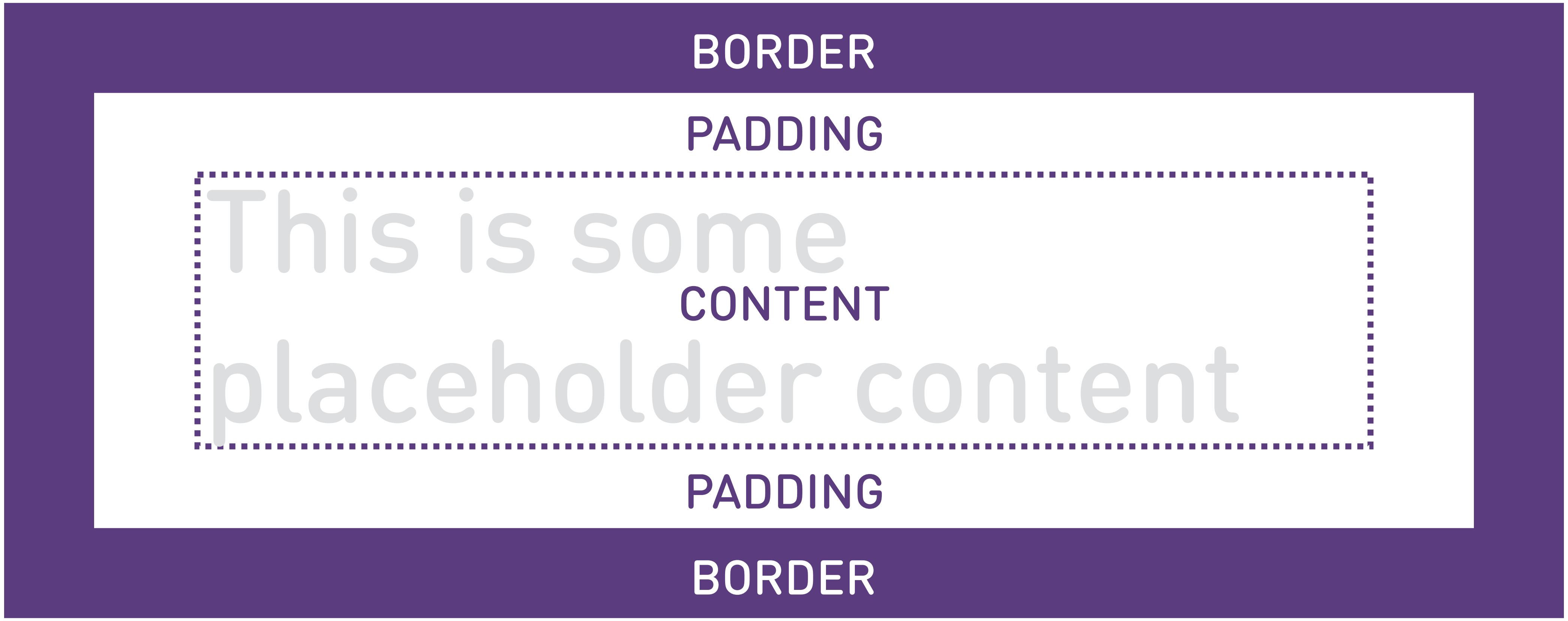
BORDER

This is some placeholder content

CONTENT

BORDER

The Box Model



BORDER

PADDING

This is some

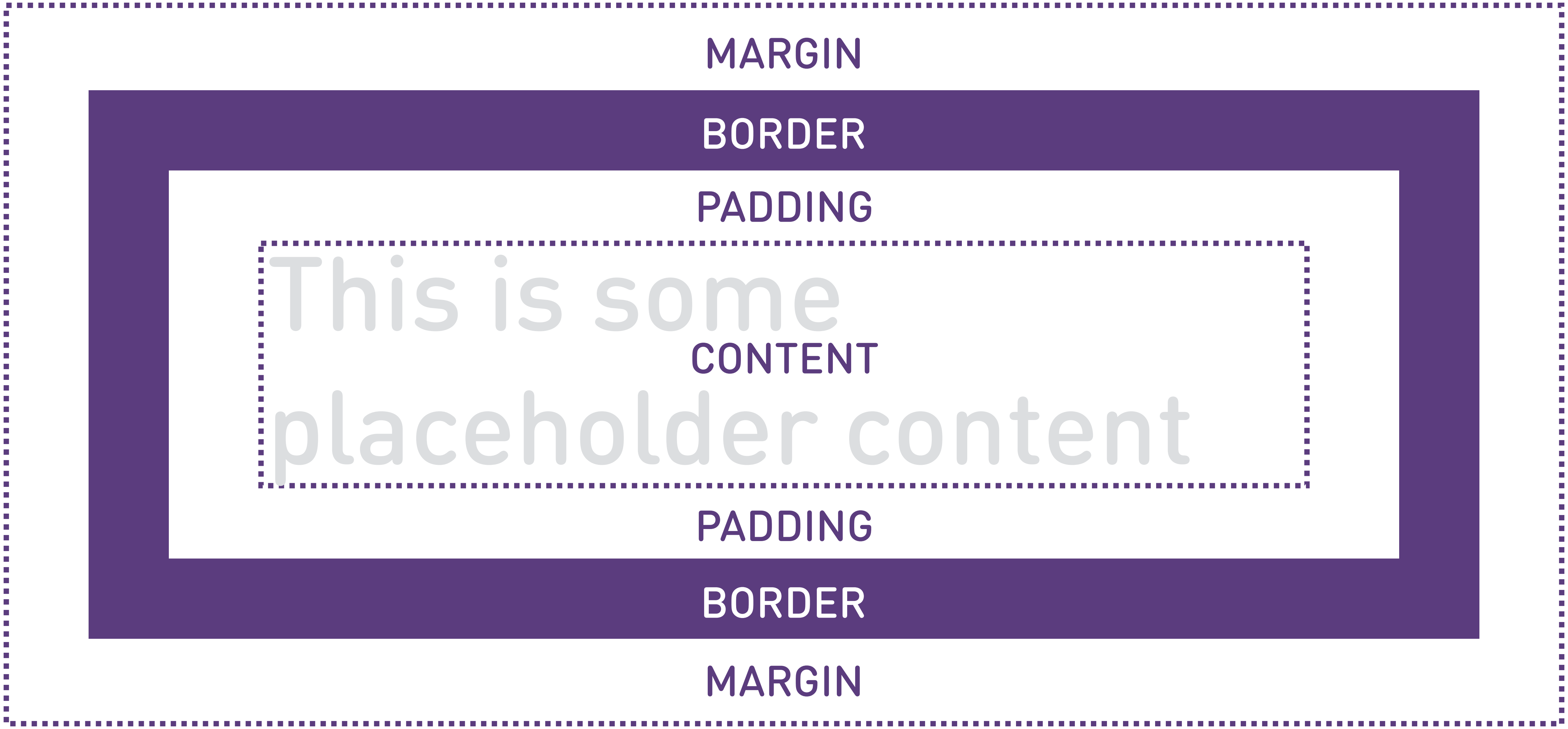
CONTENT

placeholder content

PADDING

BORDER

The Box Model



MARGIN

BORDER

PADDING

This is some placeholder content

CONTENT

PADDING

BORDER

MARGIN

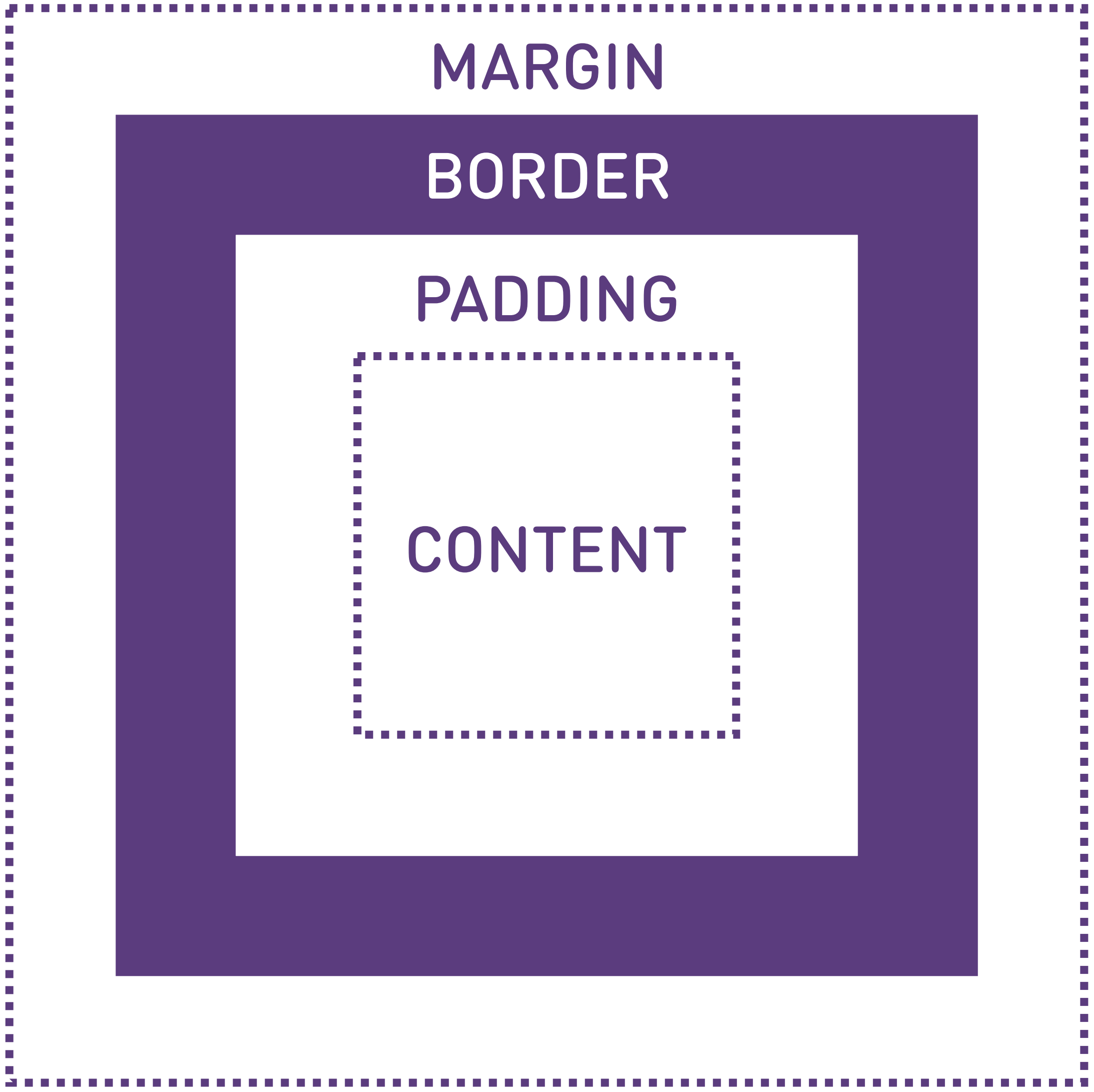
The Box Model

```
p {

    width: 100px;
    height: 100px;

    margin: 10px;
    border: 10px;
    padding: 10px;

}
```



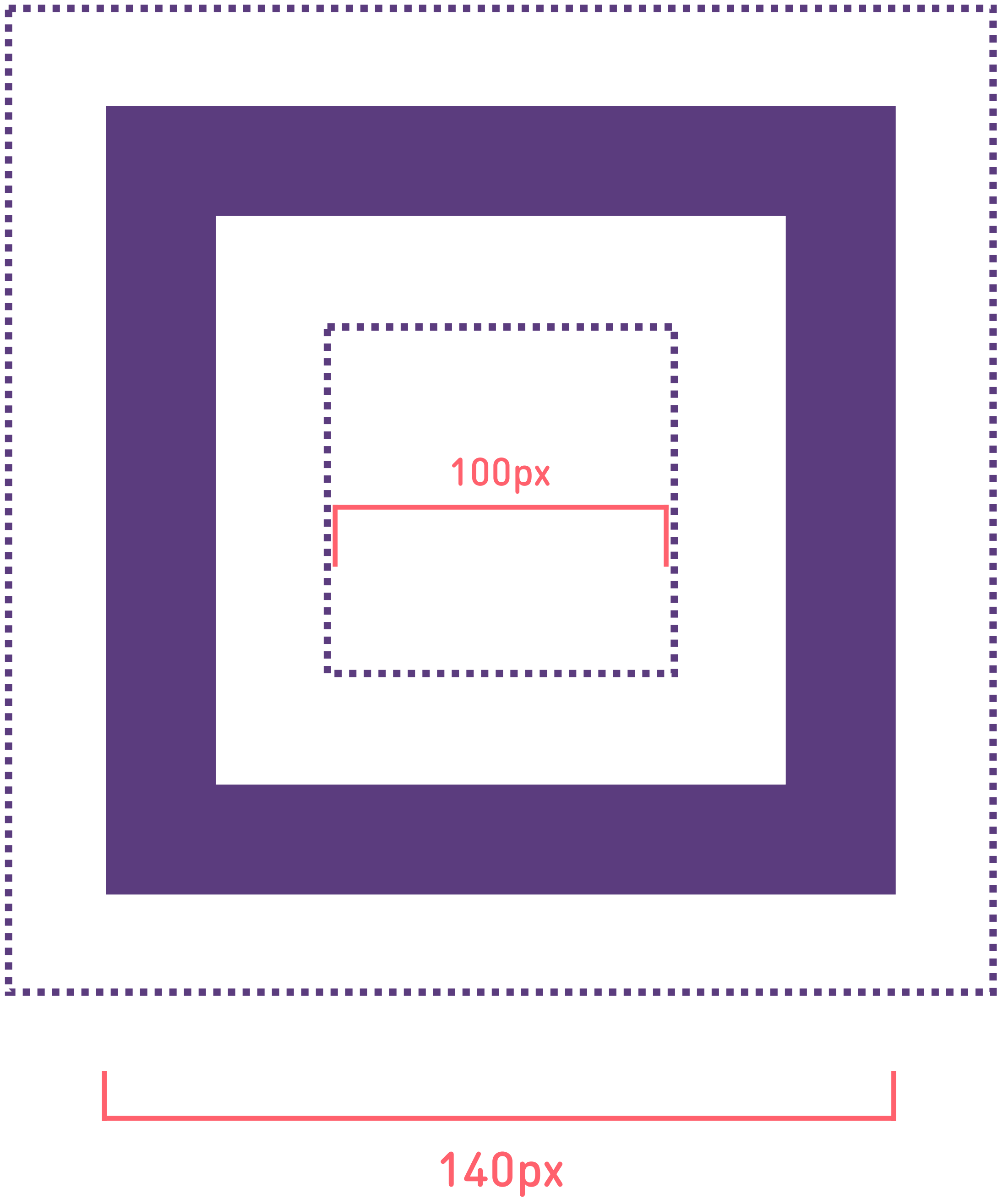
The Box Model

```
p {

    width: 100px;
    height: 100px;

    margin: 10px;
    border: 10px;
    padding: 10px;

}
```



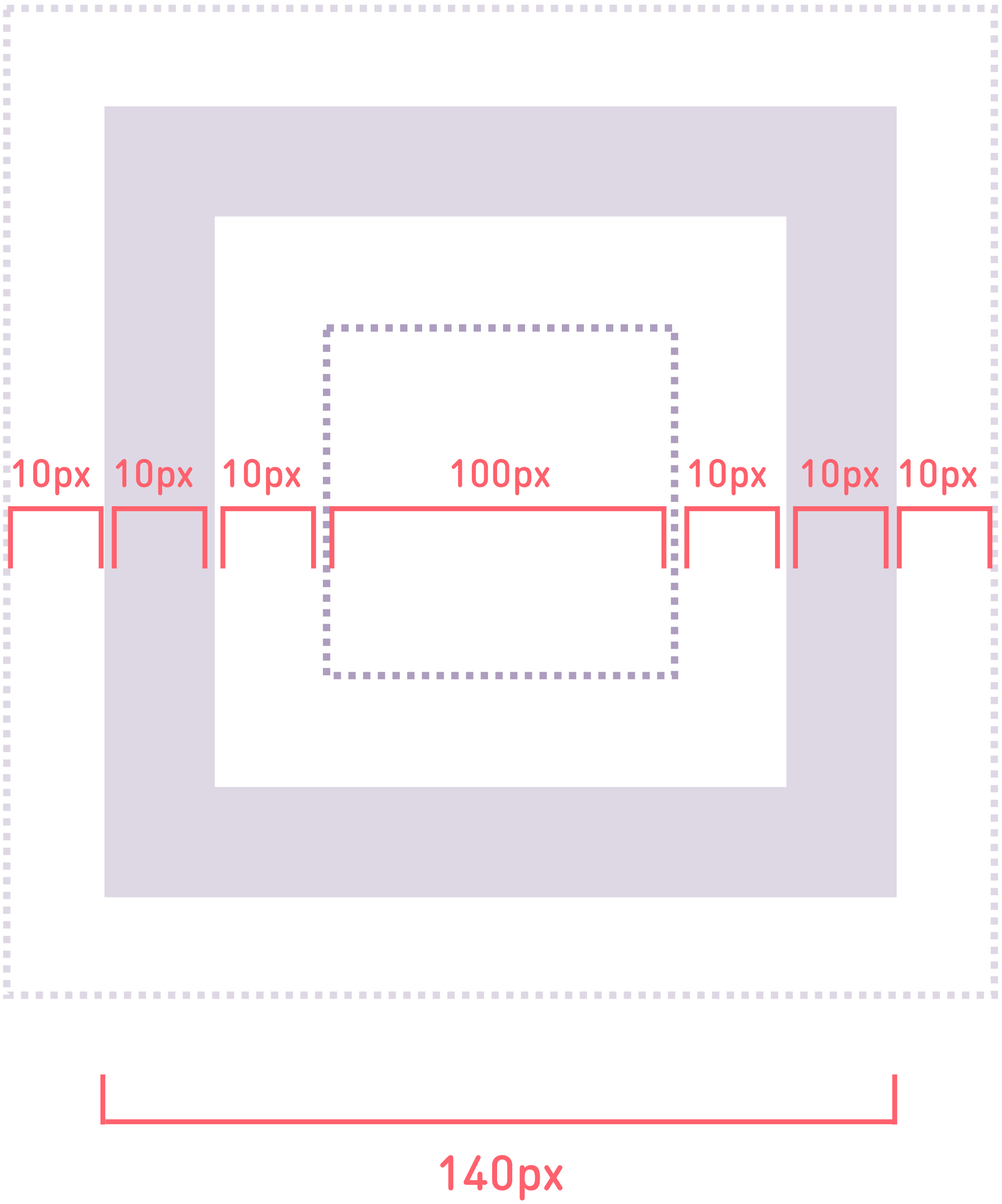
The Box Model

```
p {

    width: 100px;
    height: 100px;

    margin: 10px;
    border: 10px;
    padding: 10px;

}
```



The Box Model

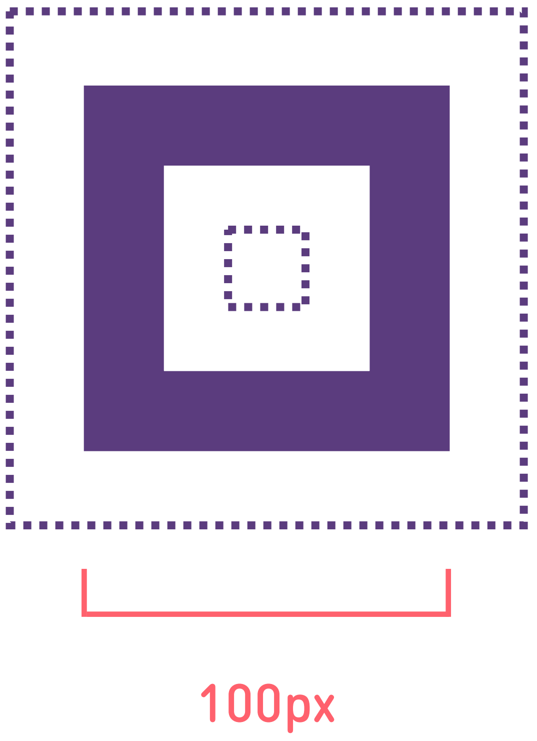
```
p {

    box-sizing: border-box;

    width: 100px;
    height: 100px;

    margin: 10px;
    border: 10px;
    padding: 10px;

}
```



Make boxes easier to size

Add this code to your CSS file:

```css
* {
    box-sizing: border-box;
}
```

Positioning

What does it do?

Allows us to place elements in specific locations on the page.

Positioning methods

- **`relative`**: place in relation to the flow of the page
- **`absolute`**: place in a specific place outside of the flow of the page
- **`fixed`**: place relative to the browser window, unaffected by scrolling

How it works

You set (1) the positioning method, and (2) the distance from the left/right and top/bottom in CSS.

```
div {
    position: absolute;
    left: 25px;
    top: 25px;
}
```

Positioning

```
div {
    position: relative;
    left: 25px;
    top: 25px;
}
```

Repositions `div` 25px to the right, and 25px down from where it was in the flow of the page.

Positioning

```
div {
    position: absolute;
    left: 25px;
    top: 25px;
}
```

Pulls `div` out of the flow of the page and positions it 25px from the left side, and 25px from the top of the page.

Positioning

```
div {
   position: fixed;
   left: 25px;
   top: 25px;
}
```

Anchors `div` to 25px from the left side, and 25px from the top of the browser window, unaffected by scrolling.

Relative to container

If an element positioned with `absolute` is contained inside another positioned element, it will be positioned relative to that container instead of the page.

Positioning

Arrange in front/behind

Set z-index to a whole number to move elements in front or behind each other. The higher the z-index, the more in front.

```
div {
    z-index: 5;
}
```

More
CSS Tricks

Visibility properties

```
display: none;
```

Removes elements entirely from the page

Visibility properties

```
visibility: hidden;
```

Hides elements, but they will still take up space on the page

Visibility properties

```
opacity: 0.5;
```

Makes elements transparent. (The value is a decimal number from 0 to 1)

More Selectors

```
:first-child { }
:last-child { }
```

Selects the first (or last) element that matches the selector

More Selectors

```
:nth-child(1) { }
:nth-child(even) { }
```

Selects specifically numbered matching elements (in these cases, the first and every even numbered element)

Coding

based on a design

Process Overview

1. Put content in HTML
2. Add structural HTML elements
3. Style elements in CSS, one chunk at a time, from top to bottom of the page; adjust HTML as needed

Coding based on a design

1. Put Content in HTML

- Use standard base HTML code to start
- Add text and images using appropriate HTML elements: headings, paragraphs, images, lists

2. Add Structure to HTML

- Group elements that go together in structural HTML elements (`main`, `header`, `footer`, `section`, `div`)
- Add `id`s and `class`es where it seems appropriate.

3. Style Elements in CSS

- Select one element at a time. Make sure selector works (e.g., add obvious background-color) before adding rules.
- Apply one CSS property at a time to bring it closer to the design
- Compare to design, edit, repeat

Week 4
Project

Week 4 Project

Master Your Layout

Take the layout & positioning strategies we've learned this week and implement them into your website

Ideas

- Create a multi-column layout
- Find an interesting way to position your navigation
- Move text over the top of images
- Add a `fixed` position header

Bonus

- Look into the CSS `transform` property to try rotating, skewing, or scaling different elements on your page.

Questions? Comments?

- Visit http://movingobjects.io/svc for class slides, code samples, resources

- Email me: scott@movingobjects.io