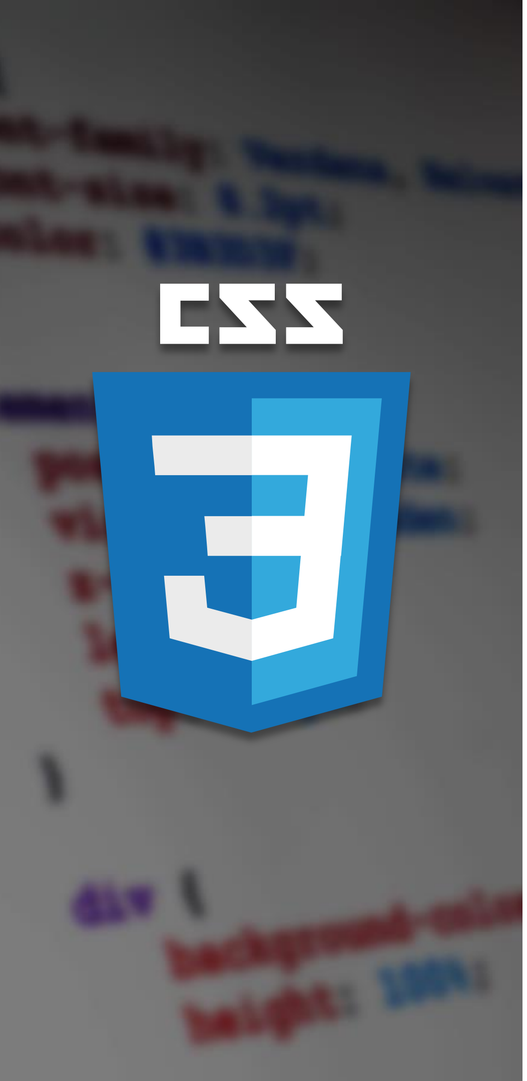
DataAccess
WORLDWIDE



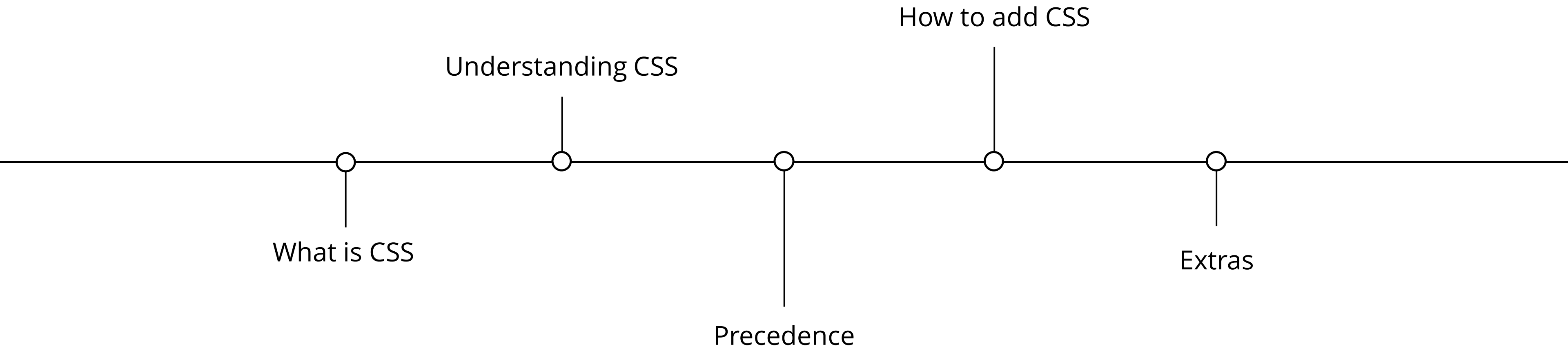
Unveiling the Basics of CSS and how it relates to the DataFlex Web Framework

Presented by **Roel Fermont**

Today more than ever, Cascading Style Sheets (CSS) have a dominant place in online business. CSS provides developers with complete control over the styling of websites and DataFlex web apps. CSS can make them look unique and attractive. And, it’s an easy technology to learn with high impact results. You should definitely try it - Roel will get you started! This 30 minute session will take you through the basics of CSS, making it easier to understand how and why it is used and, specifically how it’s related to the DataFlex Web Framework.

LOSE CONTROL
FIFTY SHADES
OF CSS
MARCH 27, 2015
SEATTLE
FOCUS FEATURES
FIFTYSHADESOFCSS.COM
/FIFTYSHADESOFCSS #FIFTYSHADES
UNIVERSAL

Today we're only scratching the surface...



Where it all started…

- 1989 - The dark ages of web design
- 1995 - Tables: the beginning
- 1995 - JavaScript comes to the rescue
- 1996 - Flash
- 1998 – CSS
- 2007 - Mobile grids and frameworks
- 2010 - Responsive web design
- 2010 - Flat design
- 2015 - The future of CSS

DataAccess WORLDWIDE

CSS

CASCADING

A set of rules resolving conflicts with multiple CSS rules applied to the same elements.

e.g. if there are two rules defining the color on your h1, the rule that comes last in the cascade will trump the other.

STYLE

How you want a certain part of your web application to look.

e.g. you can set things like colors, margins, fonts.

SHEETS

The ‘sheets’ are like templates, or a set of rules, for determining how your web application will look.

e.g. System.css, Application.css and Theme.css

CSS (all together) is a powerful **styling language** – **a set of rules** – used to control the look and feel of your web application.

Syntax

CSS Selector + CSS Declaration property / value = **CSS Rule**

```
Selector { property: value; }
```

CSS Selector

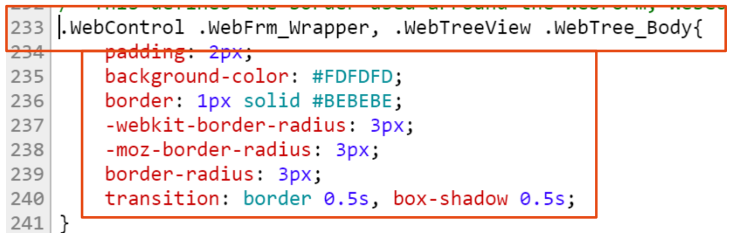
- Associates CSS rules with HTML elements

```
body { }        p { }          h1 { }
```

- Can have **multiple properties** and each property within can have **independent values**

- Apply styles to **multiple selectors** in the same rule
  by separating the selectors with commas.

```
h1, h2, h3 { }
```



```
.WebControl .WebFrm_Wrapper, .WebTreeView .WebTree_Body{
    padding: 2px;
    background-color: #FDFDFD;
    border: 1px solid #BEBEBE;
    -webkit-border-radius: 3px;
    -moz-border-radius: 3px;
    border-radius: 3px;
    transition: border 0.5s, box-shadow 0.5s;
}
```

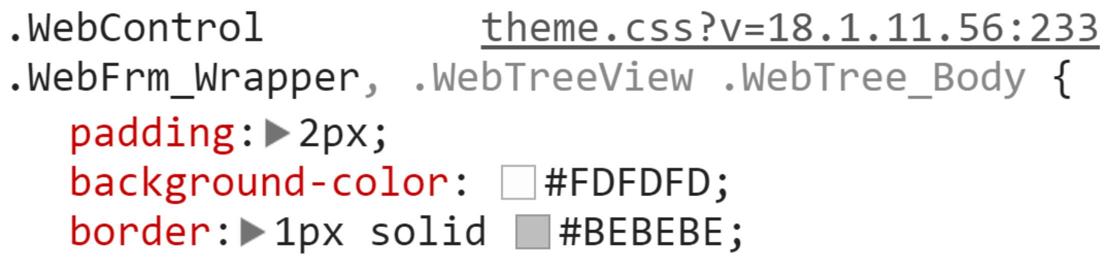
CSS Declaration

- Sits inside curly brackets **{ }** and is made up of two parts: a **property** and a **value**

```
background-color: #FDFDFD;
```

- Specify **multiple declarations** to a selector by separating the properties with a semi-colon (**;**)

- Specify **multiple properties**



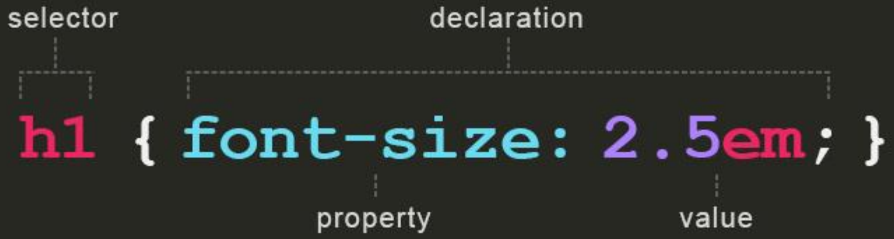
```
.WebControl                 theme.css?v=18.1.11.56:233
.WebFrm_Wrapper, .WebTreeView .WebTree_Body {
   padding:▶2px;
   background-color: ◻#FDFDFD;
   border:▶1px solid ▨#BEBEBE;
```

CSS Declaration: **Property**

- ✓ Large collection of property names
  also known as CSS identifiers
- ✓ Must be specified correctly
  Otherwise the declaration will be ignored
- ✓ Each property has its own **requirements and restrictions**
  e.g. color, font, width, height, border
- ✓ Separated by a colon (**:**)

CSS Declaration: **Value**

- ✓ Let you **specify the settings** for the chosen properties
- ✓ **Multiple values** within a property are separated by commas (**,**)
- ✓ If an individual value contains **more than one word** surround it with quotation marks (**' '**)



```css
h1 { font-size: 2.5em; }
```

DataAccess WORLDWIDE

Benefits (practical reasons)

CSS is much more than just a way of making cool looking web applications…

✓ Easily edit the formatting of multiple views (consistency)
Create rules and apply those rules to many elements within your web application.

✓ Performance
Rules are downloaded once by the browser, then are cached and used for each page load.

✓ Options
More options and control over the appearance because of a wide array of attributes.

✓ Efficiency
Quickly control all elements from one (or several) stylesheet(s)

Understanding CSS

1

CSS Box model

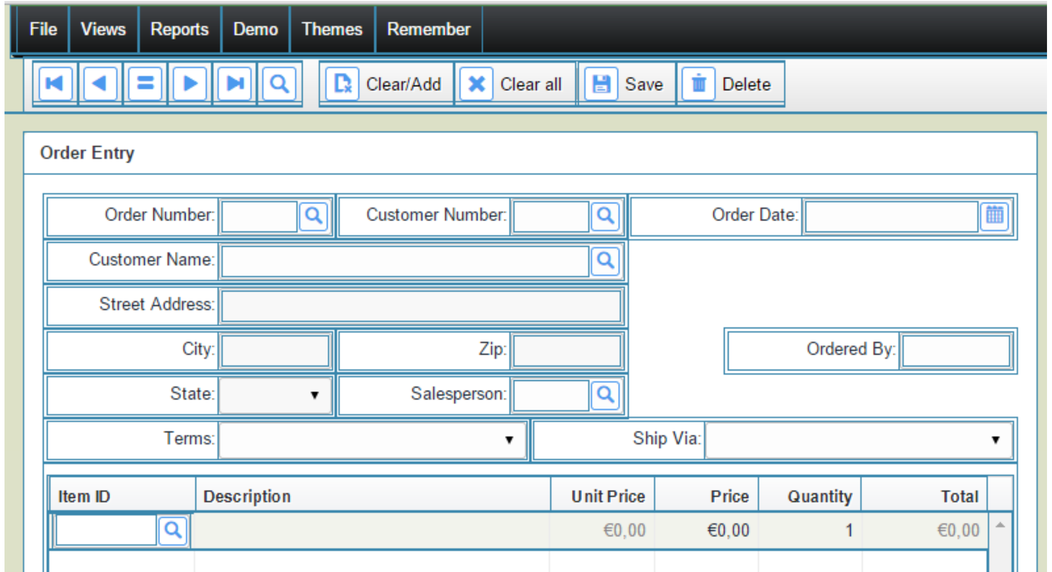
a box that wraps around HTML elements

The key to understand CSS is to **think inside the box!**

Imagine that there is an invisible box around every HTML element.

Manipulating block-level elements is in essence the way to lay out your web application.

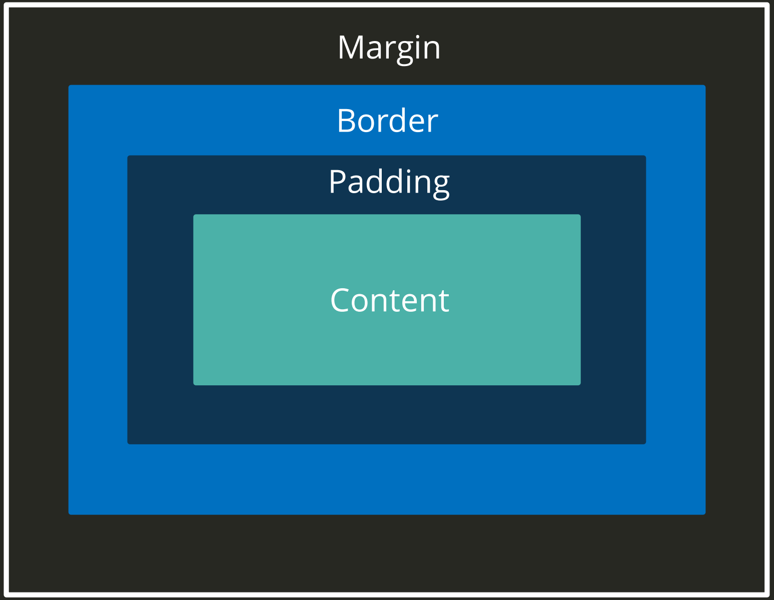
```
* {  border: 1px solid red !important;}
```



Order Entry

Order Number:
Customer Number:
Order Date:
Customer Name:
Street Address:
City:
Zip:
Ordered By:
State:
Salesperson:
Terms:
Ship Via:

Item ID	Description	Unit Price	Price	Quantity	Total
		€0,00	€0,00	1	€0,00



Content
The content of the box, where text and images appear

Padding
Clears an area around the content. The padding is transparent

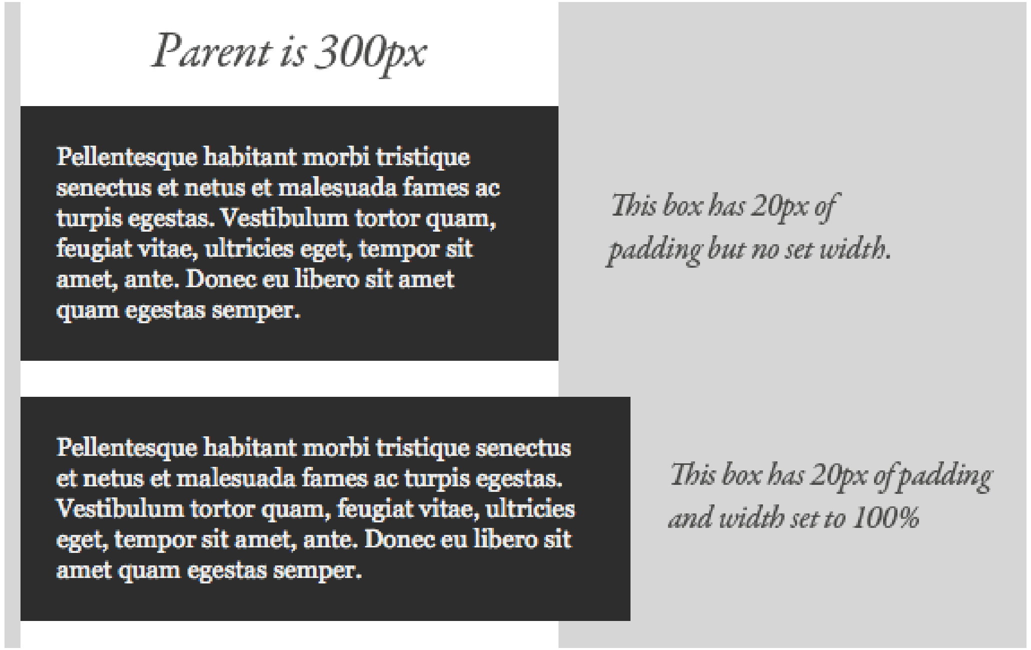
Border
A border that goes around the padding and content

Margin
Clears an area outside the border. The margin is transparent

The default width

If padding or borders are undeclared, they are either zero or the browser default value. The DataFlex Framework uses a **css reset**.

If you don't declare a width, the width will **remain 100%** in width and the padding and border will push inwards instead of outwards.



Parent is 300px
Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper.
This box has 20px of padding but no set width.
Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper.
This box has 20px of padding and width set to 100%

2

Block-level and Inline elements

elements can be divided into these two categories

DataAccess WORLDWIDE

Block-level elements

- If no width is set, it will expand to fill its parent container
- Can have margin and padding
- Generate a line break and can have dimensions

Examples:

`<p>, <div>, <ul>, <h1>`

Inline elements

- Flow along with text content, thus
- Will not clear previous content to drop to the next line
- Will ignore top and bottom margin, but will apply left and right margin, and any padding
- Will ignore width and height properties

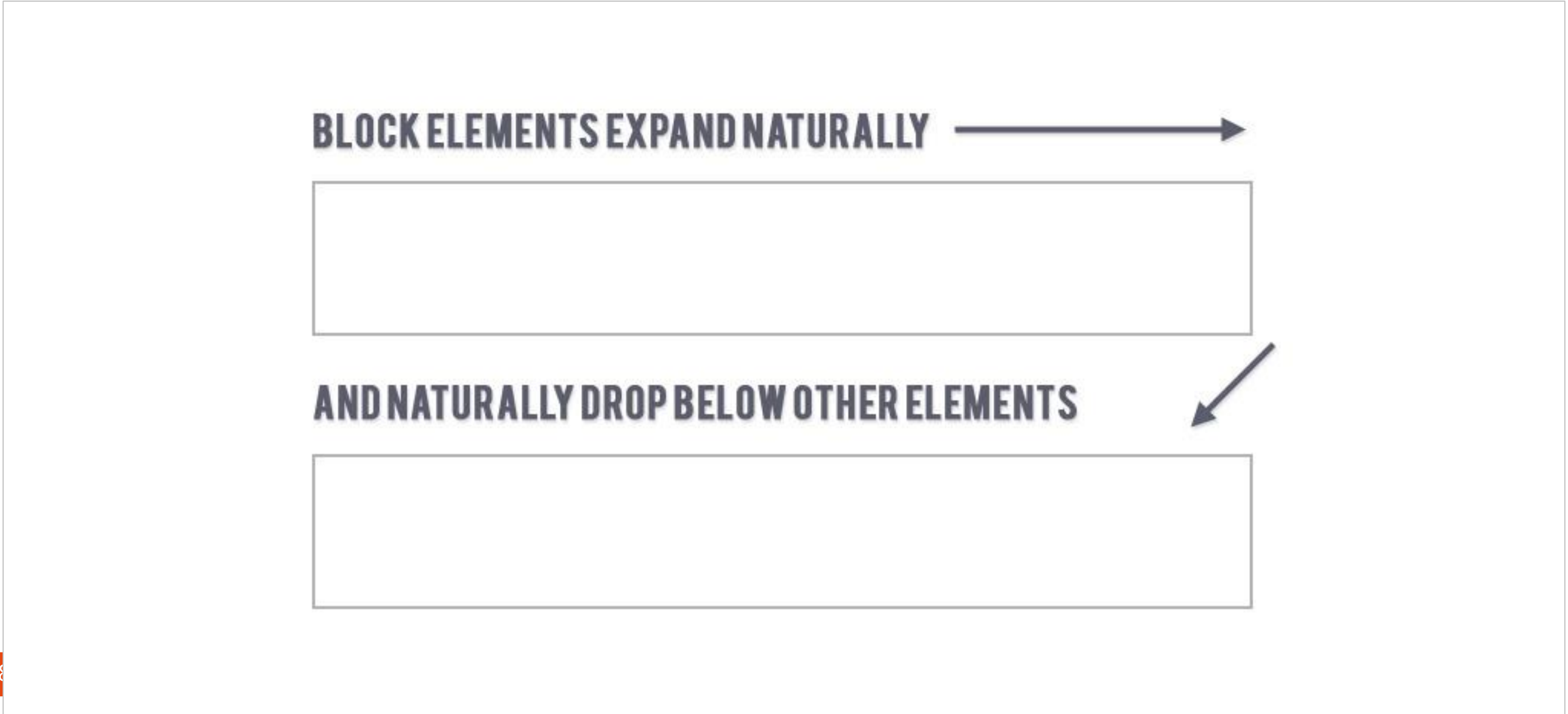
Examples:

`<a>, <span>, <strong>, <b>`

2

Block-level and Inline elements

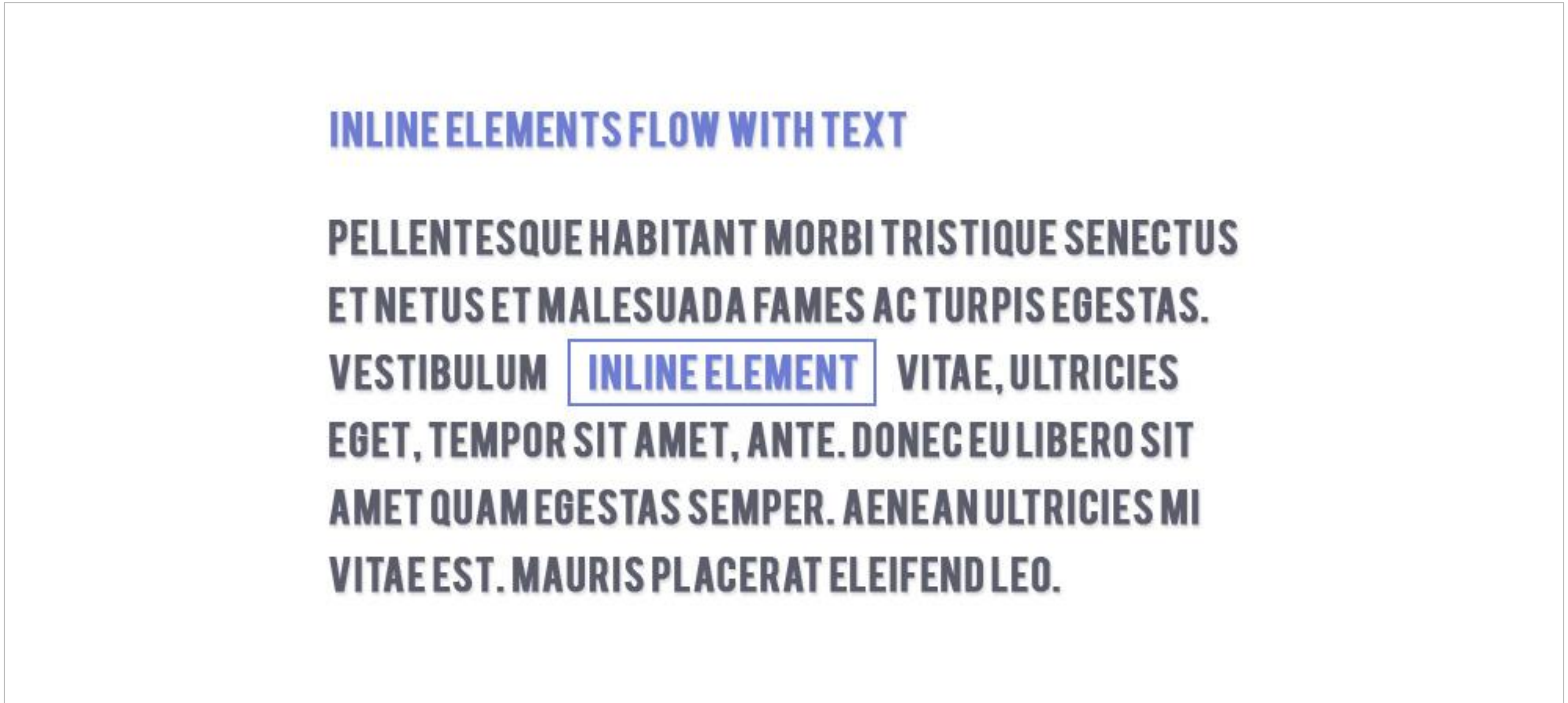
elements can be divided into these two categories


BLOCK ELEMENTS EXPAND NATURALLY
AND NATURALLY DROP BELOW OTHER ELEMENTS

2

Block-level and Inline elements

elements can be divided into these two categories



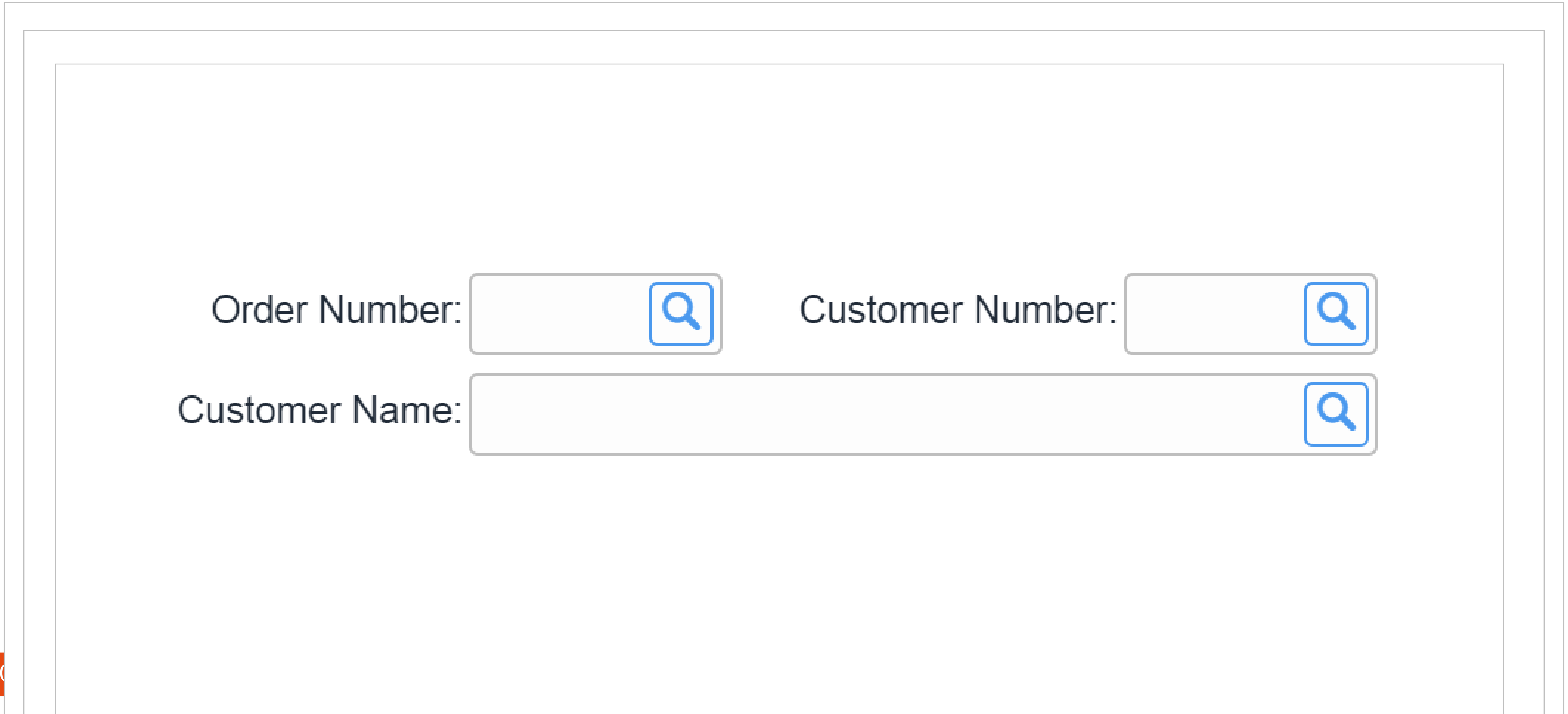
INLINE ELEMENTS FLOW WITH TEXT

PELLENTESQUE HABITANT MORBI TRISTIQUE SENECTUS ET NETUS ET MALESUADA FAMES AC TURPIS EGESTAS. VESTIBULUM INLINE ELEMENT VITAE, ULTRICIES EGET, TEMPOR SIT AMET, ANTE. DONEC EU LIBERO SIT AMET QUAM EGESTAS SEMPER. AENEAN ULTRICIES MI VITAE EST. MAURIS PLACERAT ELEIFEND LEO.

2

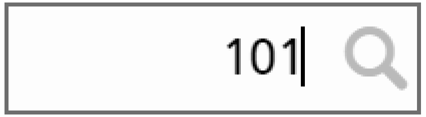
Block-level and Inline elements

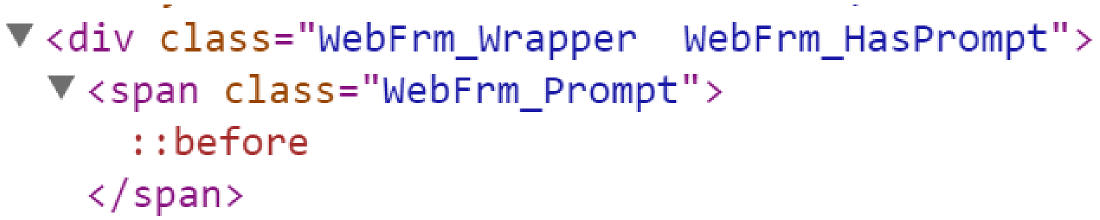
elements can be divided into these two categories



DIV vs Span

- ✓ The **`<div>`** element allows you to group a set of elements together in one block-level box. Is a **block-level element**.
- ✓ The **`<span>`** element acts like an inline equivalent of the `<div>` element. Is an **inline element**.





```
▼<div class="WebFrm_Wrapper  WebFrm_HasPrompt">
  ▼<span class="WebFrm_Prompt">
      ::before
    </span>
```

3

Selectors

selectors allow you to target rules to specific elements in an HTML document

- ✓ Are case sensitive
  so they must match element names an attribute values exactly.
- ✓ There are many different types of CSS selectors

3

Selectors

selectors allow you to target rules to specific elements in an HTML document

Universal Selector

Targets all elements

```
div * { }
```

Type Selector

Targets element names

```
h1, p, span { }
```

Class Selector

Targets class names

```
.WebFrm_Wrapper { }
```

Id Selector

Targets id names

```
#OWEBAPP{ }
```

Child Selector

Targets a direct child of another

```
li > a { }
```

Descendant Selector

Targets a decendent of another

```
ul li { }
```

Attribute Selector

Targets also any other attribute

```
Input [type="text"] { }
```

Class selectors VS ID selectors

- ✓ An ID is more specific than a class
- ✓ An element can have both an ID and multiple classes
- ✓ IDs or Classes can use UPPER and lower case
- ✓ IDs are prefixed with a hash (#) symbol and Classes are prefixed with a period (.)

Do **NOT start an ID or Class name with a number**! You can however use **numeric and non-alpha numeric** values in your ID and Class

e.g.

```
#WebDP_BtnNext-2
or
.WebDP_BtnPrevious-2
```

Class selectors VS ID selectors

ID selectors

- Are unique
- Can be used only once

Use an id if:

- The style is used only only once
- The style is specific to a certain area

```
#myID
```

Class selectors

- Are not unique
- An element can have multiple classes and multiple elements can have the same class

Use a class if:

- The style is used in various places
- The style is general

```
.myClass
```

Pseudo class selectors

- ✓ Is a CSS selector with a colon (**:**) preceding them.

```
a:hover{ }
```

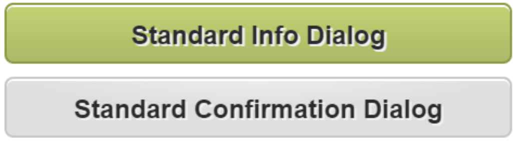
- ✓ They allow you to style content dynamically

*The difference between **Pseudo Elements** and **Pseudo Selectors** is that Pseudo Elements don’t select any ‘real’ element.*

Pseudo class selectors

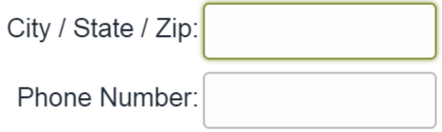
Link-related selectors

```
:link { }
:visited { }
:hover { }
:active { }
```



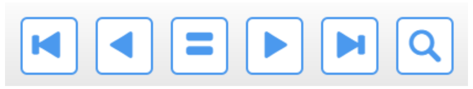
Input-related selectors

```
:focus { }
:disabled { }
```



Content-related elements

```
::before { }
::after { }
```



*The difference between **Pseudo Elements** and **Pseudo Selectors** is that Pseudo Elements don't select any 'real' element.*

Using multiple classes

- ✓ They can make it easier to add unique styles to elements without having to create a whole new style

```
<p class="item roundedCorners left"> … </p>
```

- ✓ They can give you more control. You can target elements that have combinations of classes and Ids

- ✗ Using multiple classes can be very complicated

- ✗ You run the risk of having the styles override the style for another

Selects the element which has an ID header and also a class name item.

Selects all elements with the class item that are decendents of the element with an ID header.

DataAccess WORLDWIDE

Using multiple classes

Can you spot the difference between these two selectors?

```
#header.item { }


<div id="header" class="item">

</div>
```

Selects the element which has an ID header and also a class name item.

```
#header .item { }

<div id="header" class="item">
  <div class="item">

  </div>
</div>
```

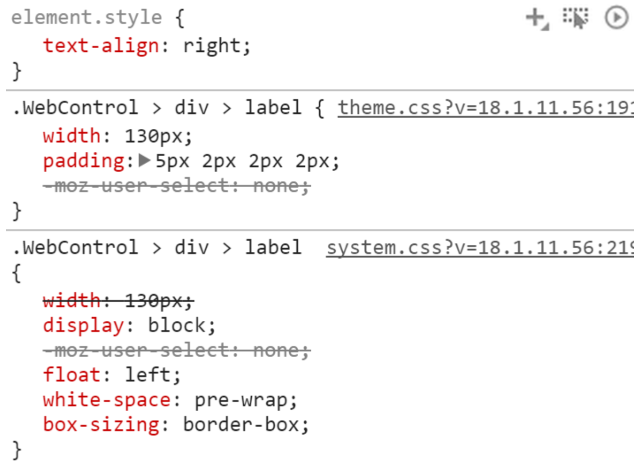
Select all elements with the class item that are decendents of the element with an ID header.

4

Precedence

The Cascade, Specificity, Inheritance

- Understanding how CSS rules cascade means you can write simpler style sheets.
- If there a two or more rules that apply to the same element, it is important to understand which will take precedence.



```
element.style {
    text-align: right;
}
.WebControl > div > label { theme.css?v=18.1.11.56:19
    width: 130px;
    padding:▶5px 2px 2px 2px;
    ~~-moz-user-select: none;~~
}
.WebControl > div > label  system.css?v=18.1.11.56:21
{
    ~~width: 130px;~~
    display: block;
    ~~-moz-user-select: none;~~
    float: left;
    white-space: pre-wrap;
    box-sizing: border-box;
}
```

Three things that control which CSS rule applies to a given html element:

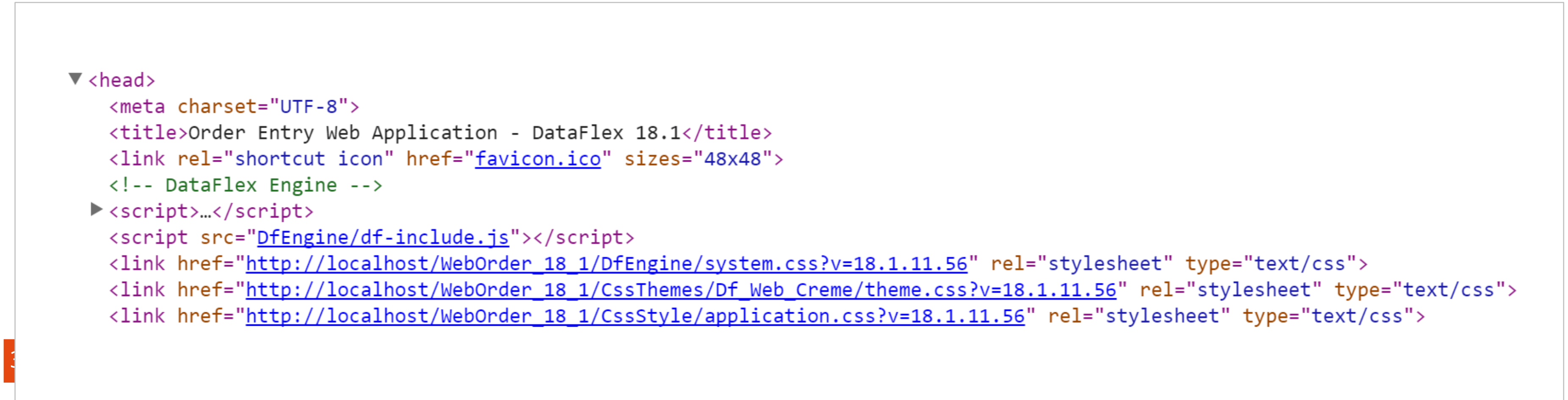
The Cascade **Specificity** **Inheritance**

The Cascade

- ✓ Is a mechanism for determining which styles should be applied to a given element, based on the rules that have cascaded down from various sources.
- ✓ It takes importance, origin, specificity, and source order of style rules into account.
- ✓ It assigns a weight to each rule and this weight determines which rule takes precedence, when more than one applies

The Cascade

- Is a mechanism for determining which styles should be applied to a given element, based on the rules that have cascaded down from various sources.
- It takes importance, origin, specificity, and source order of style rules into account.



```
▼ <head>
    <meta charset="UTF-8">
    <title>Order Entry Web Application - DataFlex 18.1</title>
    <link rel="shortcut icon" href="favicon.ico" sizes="48x48">
    <!-- DataFlex Engine -->
  ▶ <script>…</script>
    <script src="DfEngine/df-include.js"></script>
    <link href="http://localhost/WebOrder_18_1/DfEngine/system.css?v=18.1.11.56" rel="stylesheet" type="text/css">
    <link href="http://localhost/WebOrder_18_1/CssThemes/Df_Web_Creme/theme.css?v=18.1.11.56" rel="stylesheet" type="text/css">
    <link href="http://localhost/WebOrder_18_1/CssStyle/application.css?v=18.1.11.56" rel="stylesheet" type="text/css">
```

DataAccess WORLDWIDE

System CSS
(framework)

- CSS required for functioning of controls
- ***AppHtml\DfEngine\System.css*** *(external style sheet)*

Theme CSS
(designer)

- Global styles for the controls defined in CSS file
- Switch using psTheme property on oWebApp
- ***AppHtml\CssThemes\<psTheme>\Theme.css*** *(external style sheet)*

Custom CSS Classes
(developer & designer)

- Application specific CSS declarations
- Assigned using psCSSClass or psHtmlId
- ***AppHtml\CssStyle\Application.css*** *(external style sheet)*

DataFlex Properties
(developer)

- Properties on DataFlex classes
- ***Inline styles*** *set by JavaScript engine (inline styles)*

Specificity

- It determines which CSS rule is applied
- Every selector has its specificity
- If two rules share the same weight, source and specificity, the later one is applied.

```
#content h1 {
  padding: 5px;
}

#content h1 {
  padding: 10px;
}
```

1	* { }	0
2	li { }	1 (one element)
3	li:first-line { }	2 (one element, one pseudo-element)
4	ul li { }	2 (two elements)
5	ul ol+li { }	3 (three elements)
6	h1 + *[rel=up] { }	11 (one attribute, one element)
7	ul ol li.red { }	13 (one class, three elements)
8	li.red.level { }	21 (two classes, one element)
9	style=""	1000 (one inline styling)
10	p { }	1 (one HTML selector)
11	div p { }	2 (two HTML selectors)

`h1` is more specific than `*`

`p b` is more specific than `p`

`p#OWEBAPP` is more specific than `p`

CSS Selector	Description
Inherited styles	Lowest specificity of all selectors - since an inherited style targets the element's parent, and not the HTML element itself.
*	Lowest specificity of all directly targeted selectors
element	Higher specificity than universal selector and inherited styles.
attribute	Higher specificity than element selector
class	Higher specificity than attribute, element and universal selectors.
ID	Higher specificity than class selector.
Combined selectors	Gets the specificity of the selectors combined.
CSS properties set directly on element, insidestyle attribute.	Stronger specificity than ID selector.

DataAccess WORLDWIDE

Inheritance

- ✓ Elements inherit styles from their parent container
- ✓ Not all CSS properties are inherited
  e.g. margin and padding are non-inherited

HTML	Relationship
`<body>`	Parent of the HTML document
`  <div>`	Parent of ul and li, child of body
`    <ul>`	Parent of li, child of div and body
`      <li></li>`	Child of ul, div, and body
`    </ul>`	
`  </div>`	
`</body>`	

5

Adding CSS

there are various ways to adding CSS to your web application

- ✓ Internal
- ✓ Inline
- ✓ External

```
<head>
<style type="text/css">
  body {
    color: #000;
    font-family: Helvetica, sans-serif;
  }
  p {
    line-height: 1,5;
    font-size:14px;
  }
</style>
</head>
```

5

Adding CSS

there are various ways to adding CSS to your web application

- ✓ Internal
- ✓ Inline
- ✓ External

```
<p style="font-size:14px;">DataFlex</p>
```

5

Adding CSS

there are various ways to adding CSS to your web application

- ✓ Internal
- ✓ Inline
- ✓ External

```
<head>
  <link rel="stylesheet" type="text/css"
  href="CssThemes/theme.css">
</head>
```

Extras: **CSS Reset**

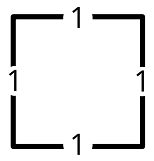
/* Eric Meyer's Reset CSS v2.0 - http://cssreset.com

```
*/html,body,div,span,applet,object,iframe,h1,h2,h3,h4,h5,h6,p,blockquo
te,pre,a,abbr,acronym,address,big,cite,code,del,dfn,em,img,ins,kbd,q,s
,samp,small,strike,strong,sub,sup,tt,var,b,u,i,center,dl,dt,dd,ol,ul,l
i,fieldset,form,label,legend,table,caption,tbody,tfoot,thead,tr,th,td,
article,aside,canvas,details,embed,figure,figcaption,footer,header,hgr
oup,menu,nav,output,ruby,section,summary,time,mark,audio,video{border:
0;font-size:100%;font:inherit;vertical-
align:baseline;margin:0;padding:0}article,aside,details,figcaption,fig
ure,footer,header,hgroup,menu,nav,section{display:block}body{line-
height:1}ol,ul{list-
style:none}blockquote,q{quotes:none}blockquote:before,blockquote:after
,q:before,q:after{content:none}table{border-collapse:collapse;border-
spacing:0}
```

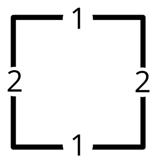
www.cssreset.com

Extras: **CSS Shorthand**

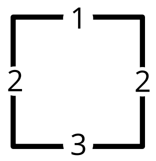
Shorthands handling properties **related to edges of a box**



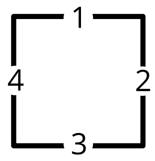
The 1-value syntax



The 2-value syntax

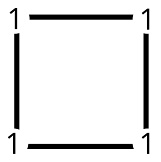


The 3-value syntax

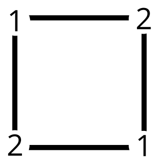


The 4-value syntax

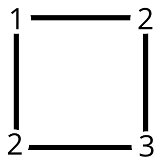
Shorthands handling properties **related to corners of a box**



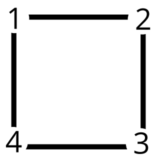
The 1-value syntax



The 2-value syntax



The 3-value syntax



The 4-value syntax

Background properties

```css
background-color: #000;
background-image: url (images/bg.gif);
background-repeat: no-repeat;
background-position: top right;
```

```css
background: #000 url (images/bg.gif) no-repeat top right;
```

Margin and Padding properties

```css
margin-top: 10px;
margin-right: 5px;
margin-bottom: 10px;
margin-left: 5px;
```

```css
margin: 10px 5px 10px 5px;
```

```css
margin: 10px 5px;
```

the values are in clockwise order from top: top, right, bottom, left

Got lost along the way?

CSS Box-model
Content / Padding / Border / Margin

Block-level and Inline elements
div / span

Selectors and Pseudo selectors
Class and ID selectors

Precedence
The Cascade / Specificity / Inheritance

Adding CSS
Inline / Internal / External

Resources

www.w3schools.com/css

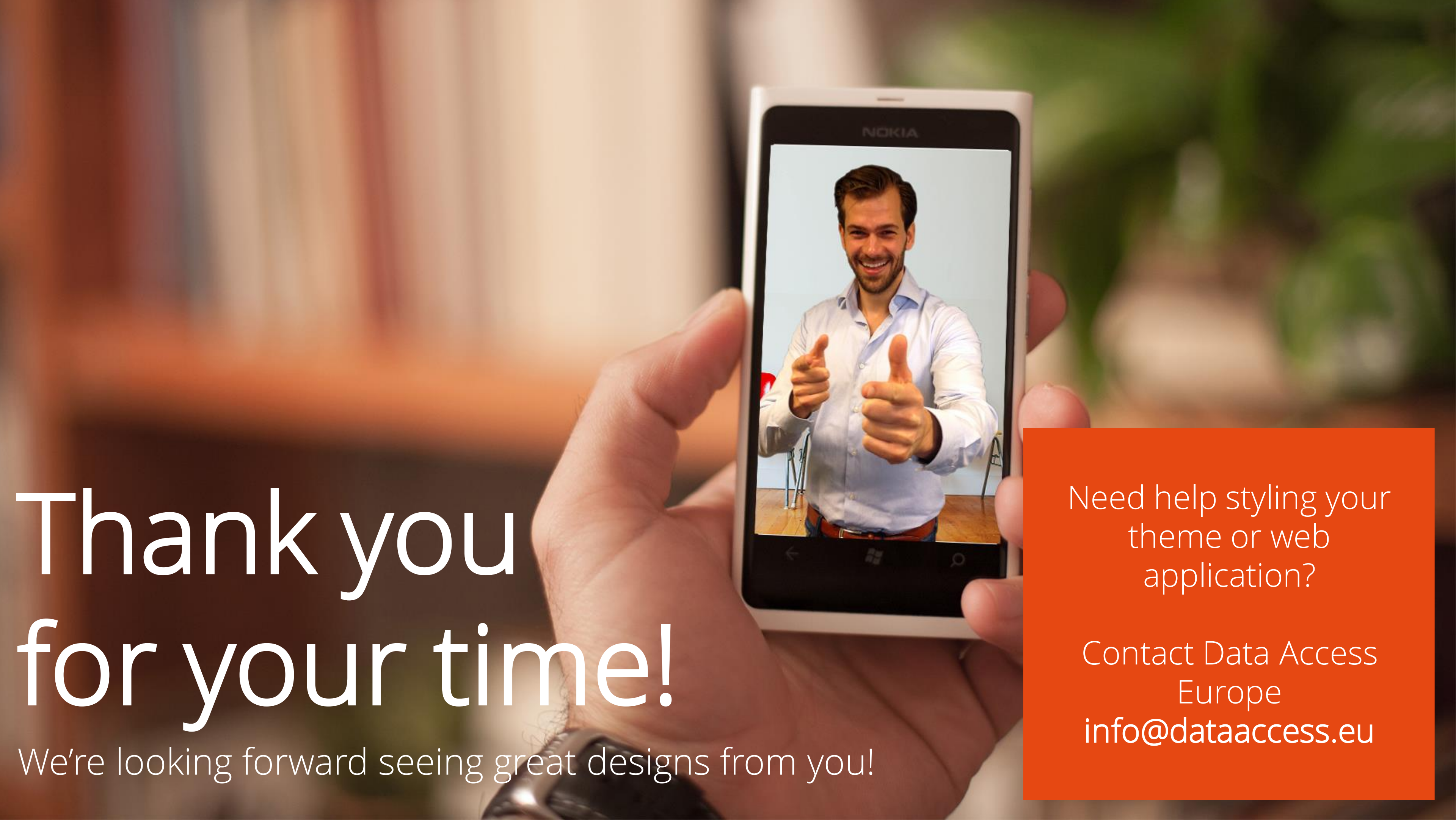
www.codecademy.com/en/tracks/web

www.codeschool.com/paths/html-css

… and there are many more resources

The **possibilities with CSS are endless**. Today was just scratching the surface.

CSS is the way to go, so why not get there now!



Thank you for your time!

We're looking forward seeing great designs from you!

Need help styling your theme or web application?

Contact Data Access Europe
info@dataaccess.eu