Education SERVICES

Modern CSS Techniques

TEK systems education services

TEKsystems

Our people make IT possible.

TEKsystems Education Services

presents

Modern CSS Techniques

20750 Civic Center Drive
Suite 400, Oakland Commons II
Southfield, MI 48076
800.294.9360

COURSE CODE IN1403 / 01.20.2015

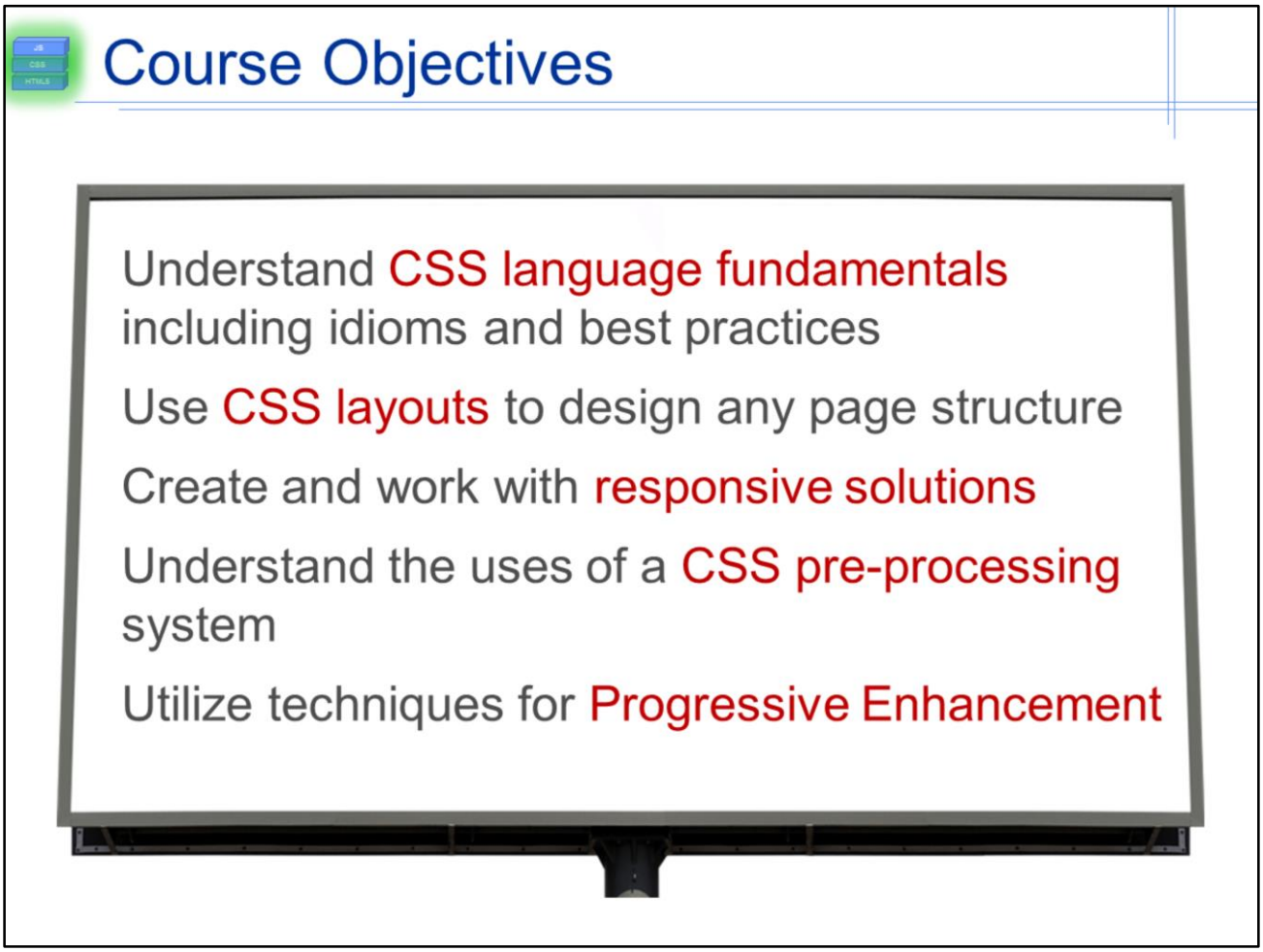
This course covers Modern CSS Techniques

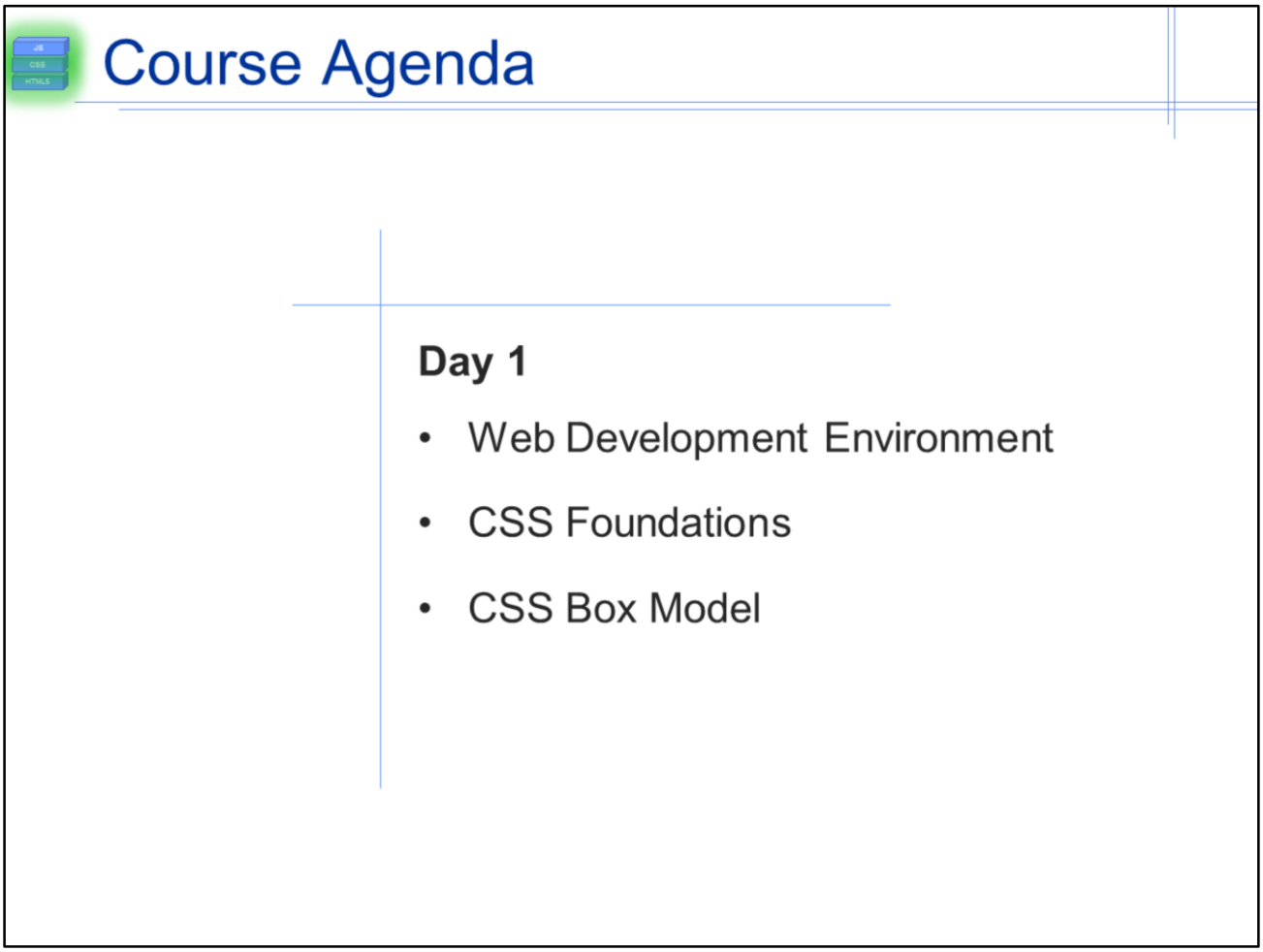
This version updated: 1/20/2015

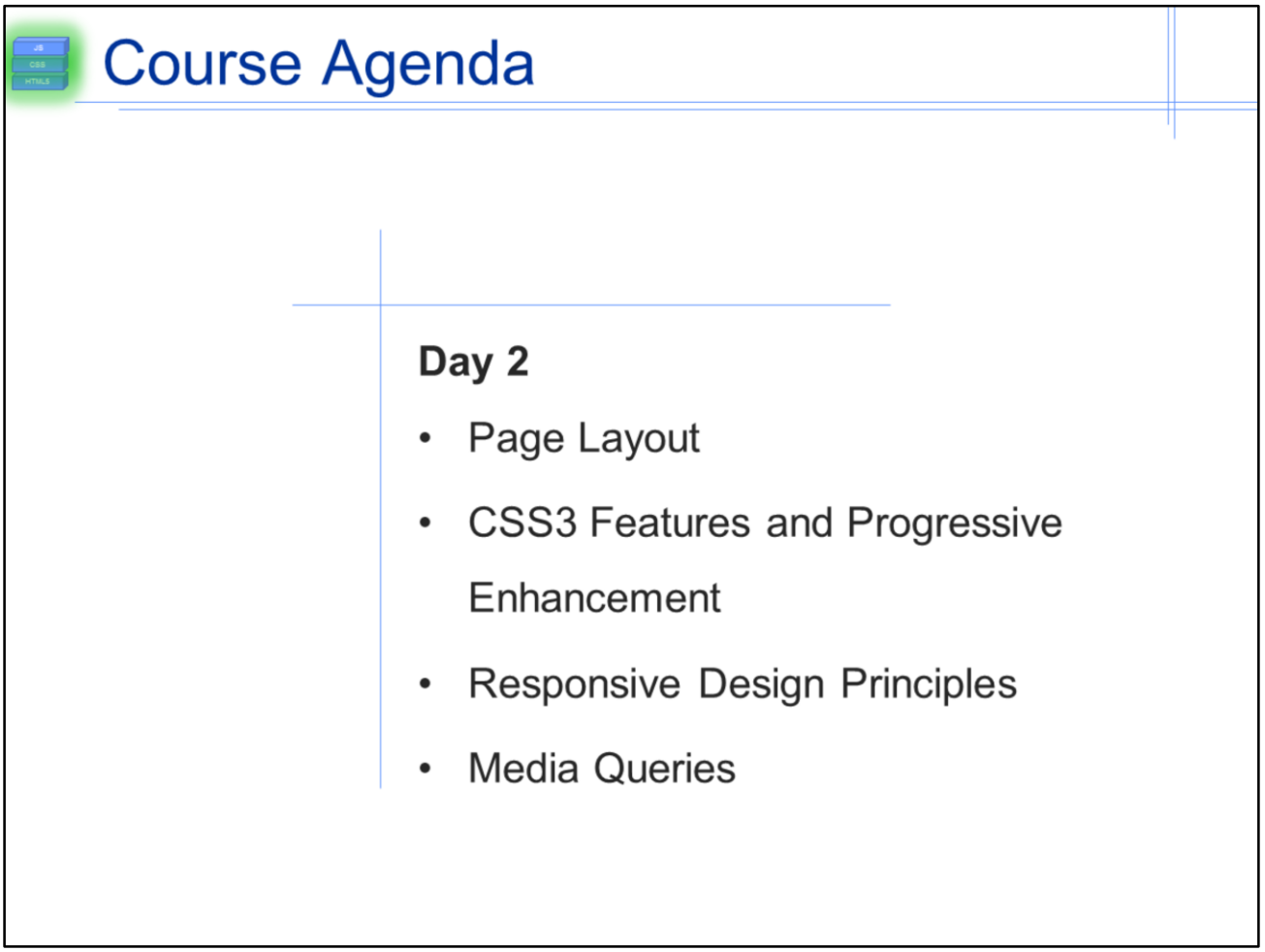
Notes

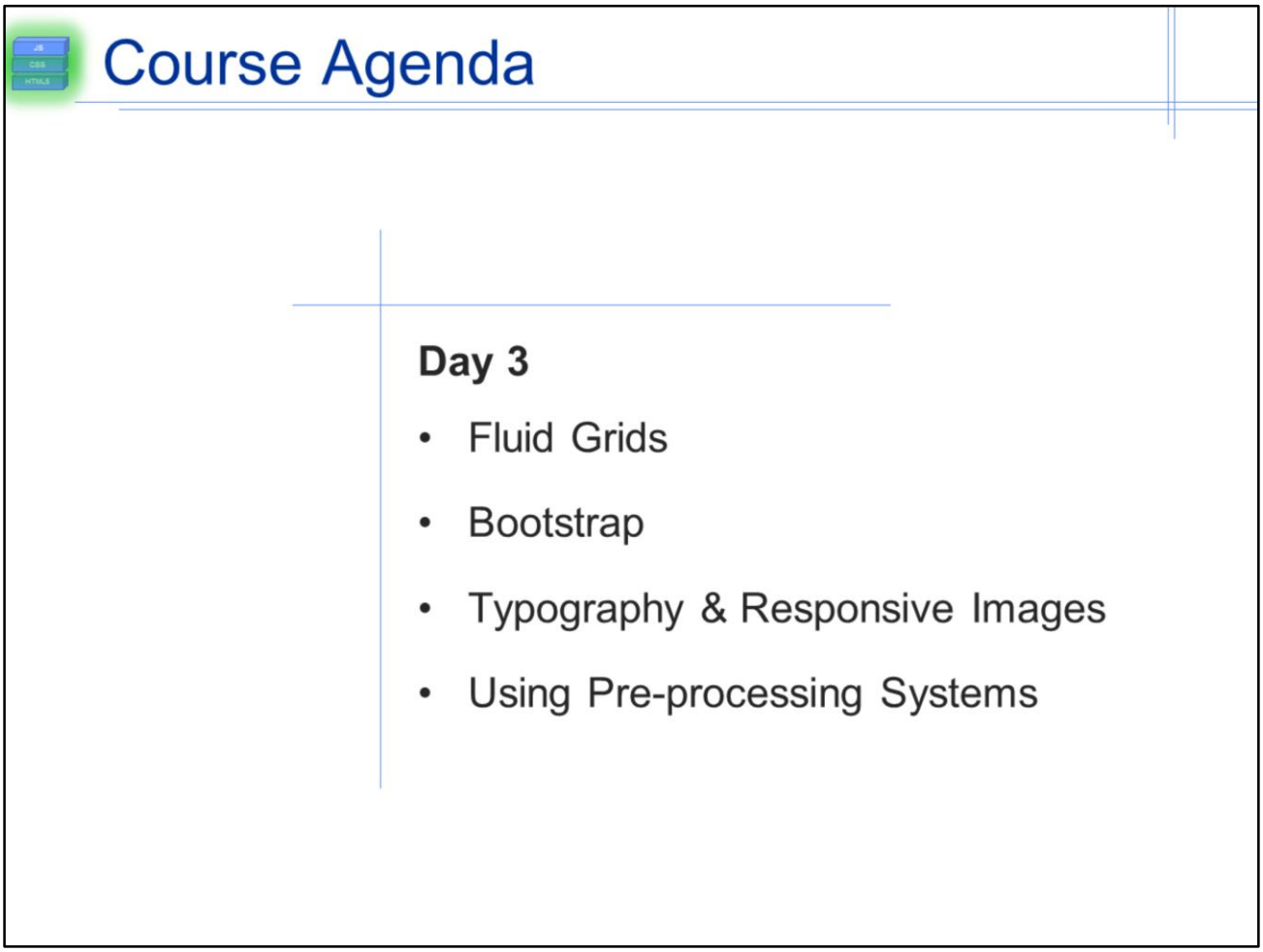
Chapters at a Glance

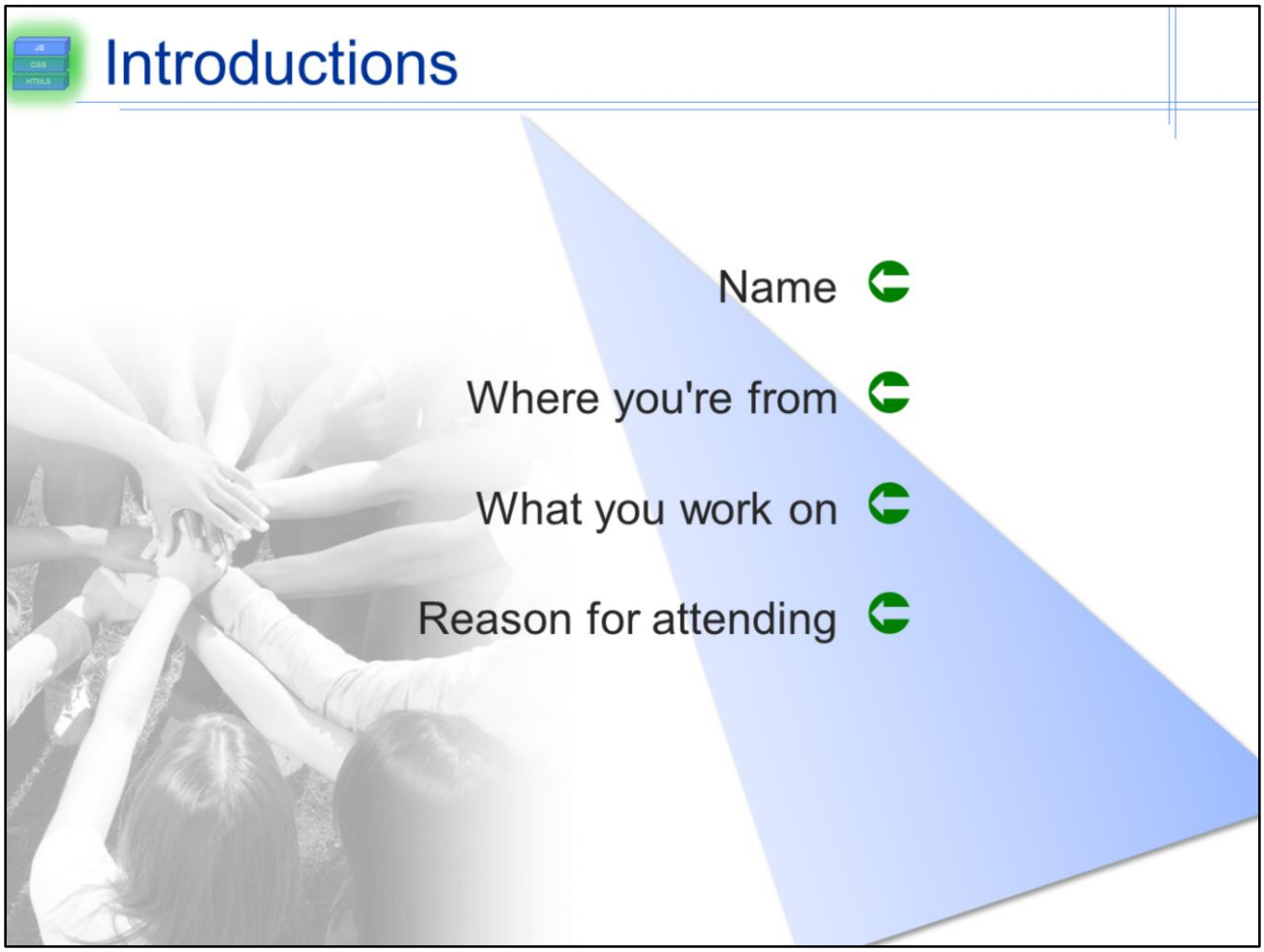
Notes











Logistics

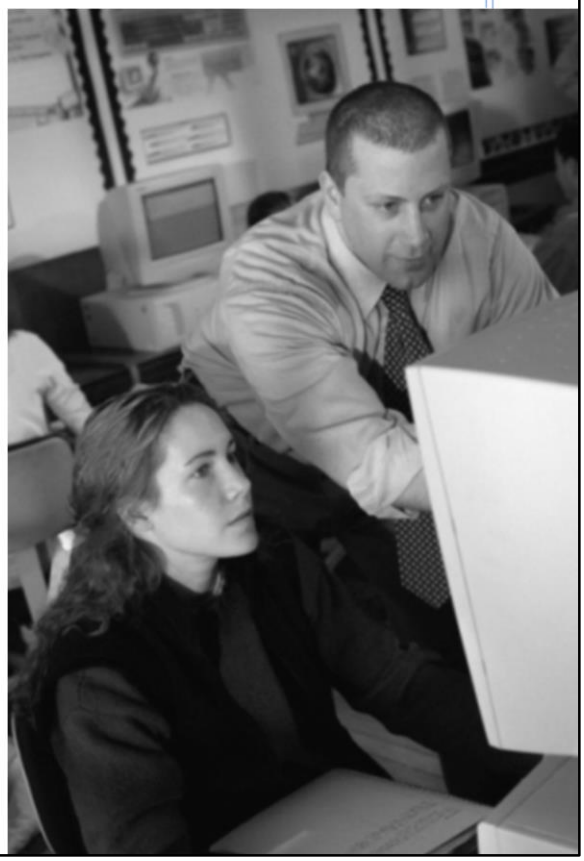
*Typical Daily Schedule**

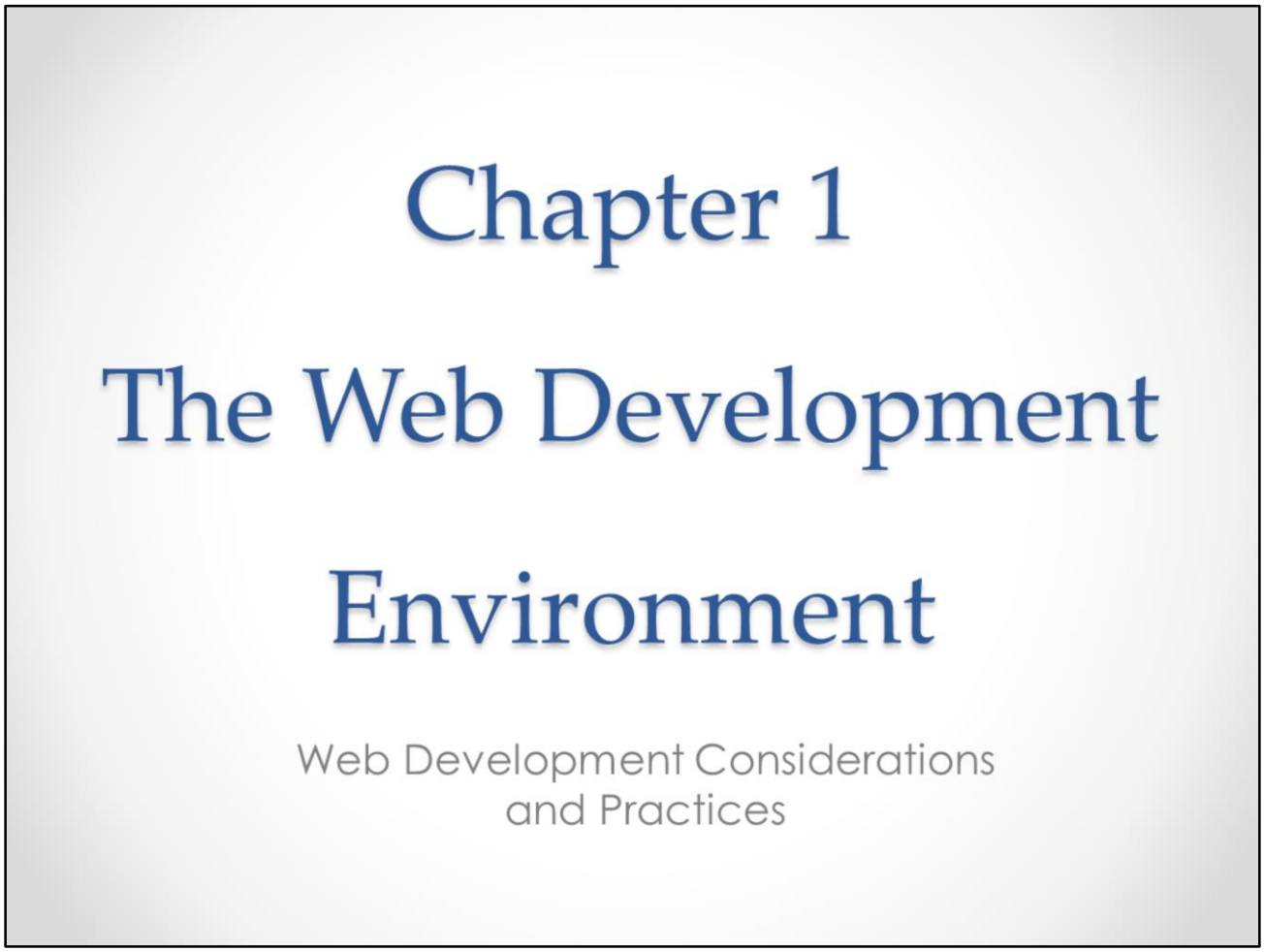
8:30	Start Day
9:45	Morning Break
11:30 – 12:30	Lunch
1:45	Afternoon Break 1
3:00	Afternoon Break 2
4:30	End of Day

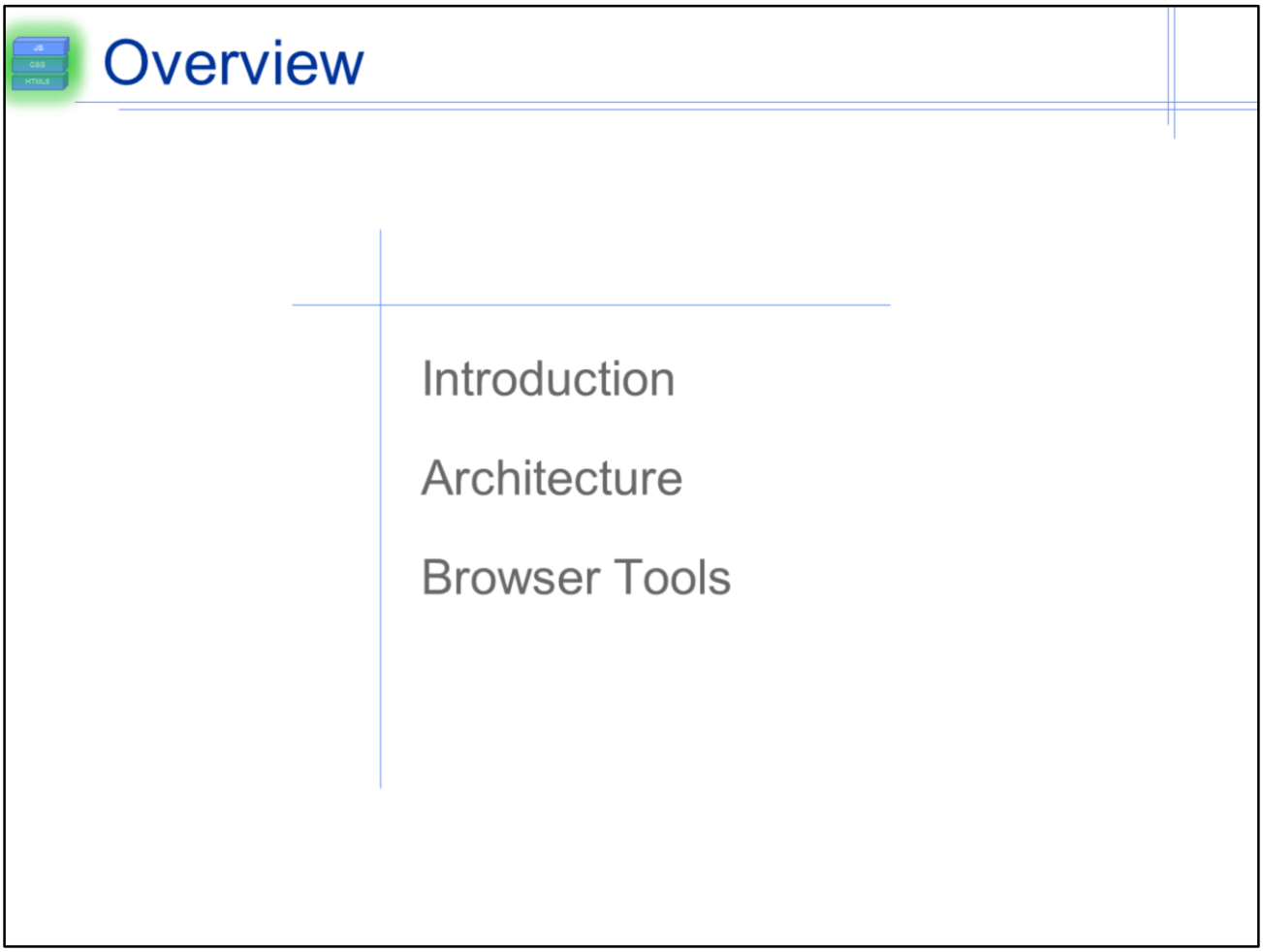
** Your schedule may vary*

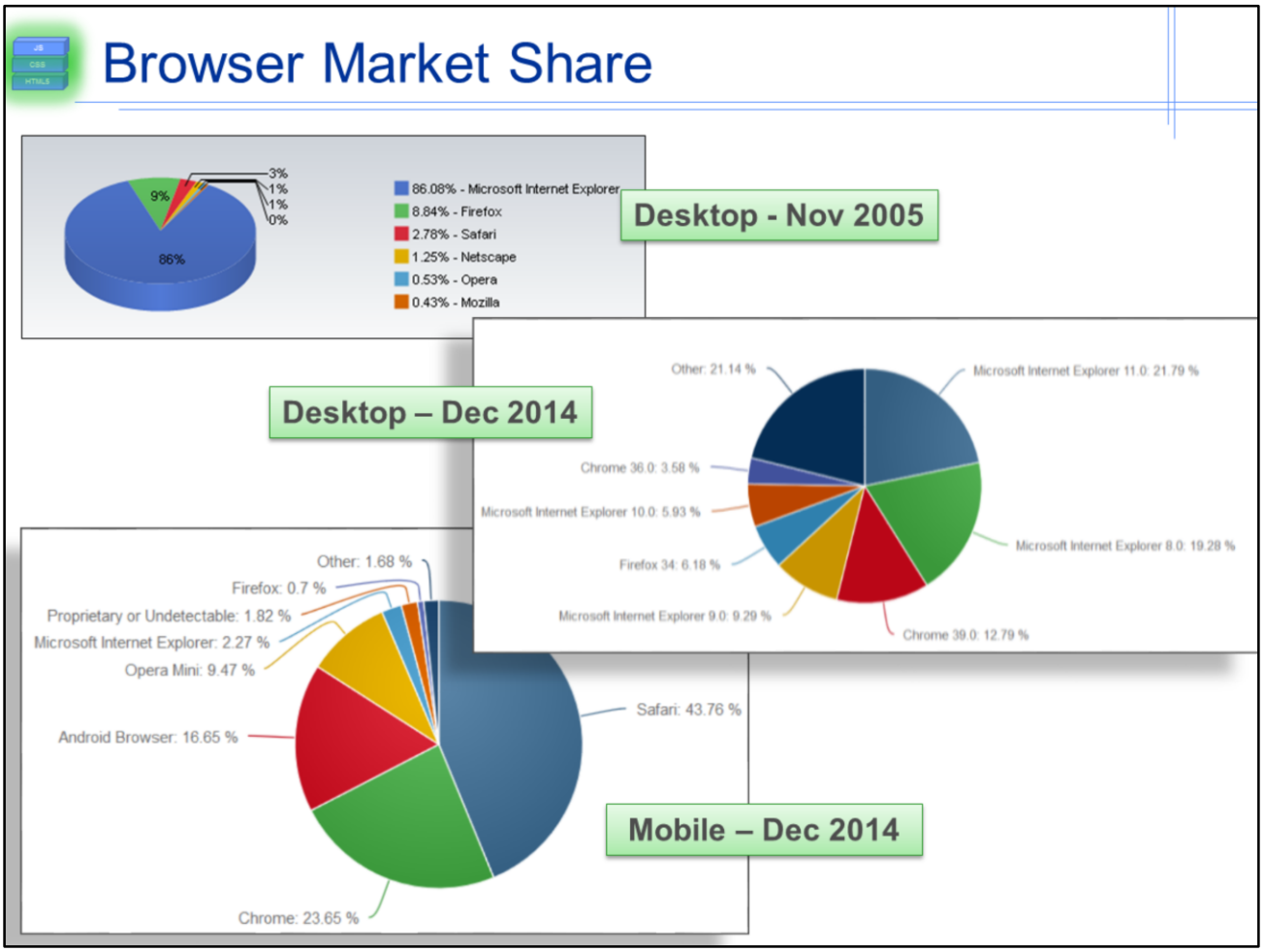
Get the Most from Your Experience

Ask Questions

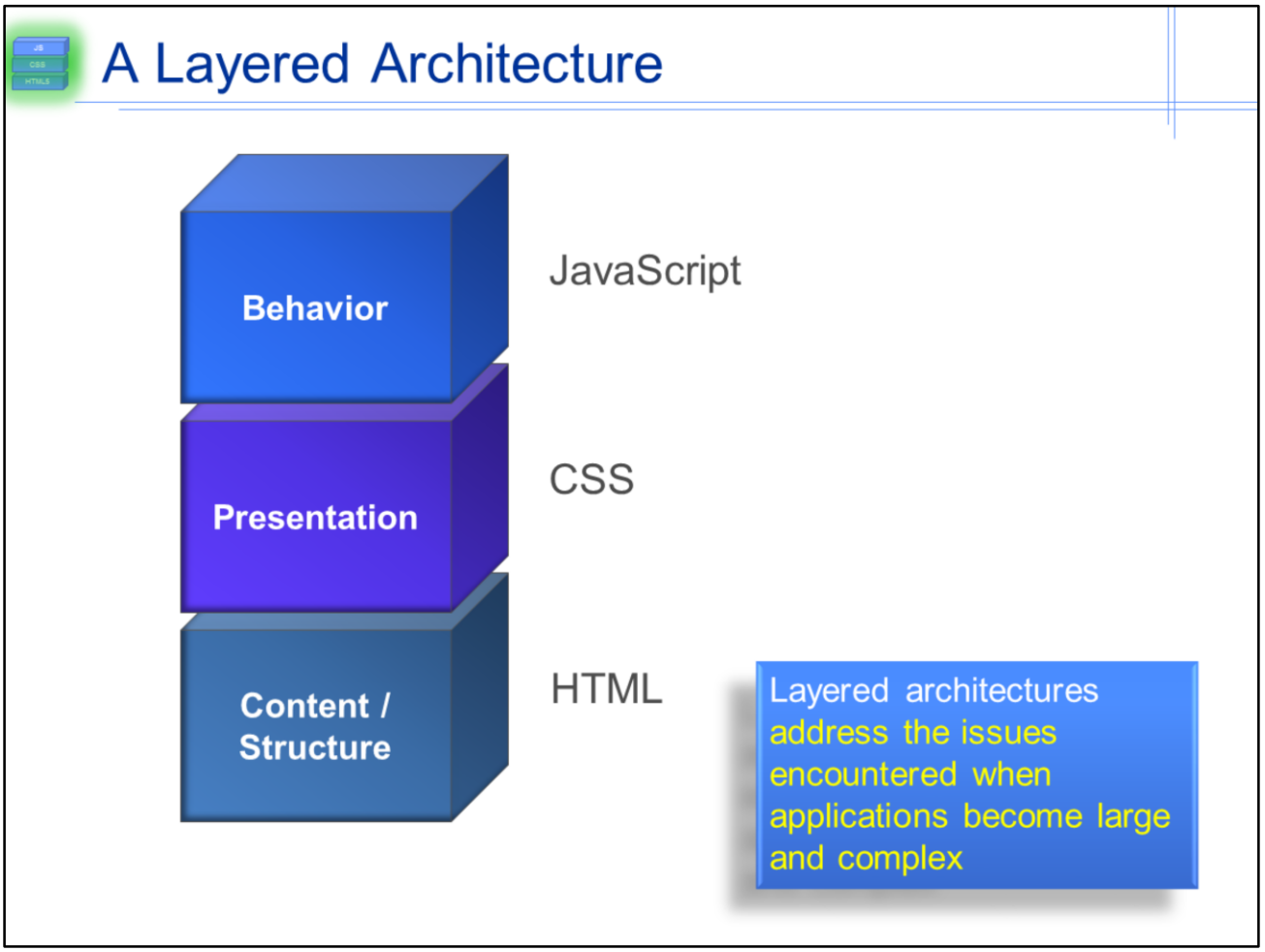




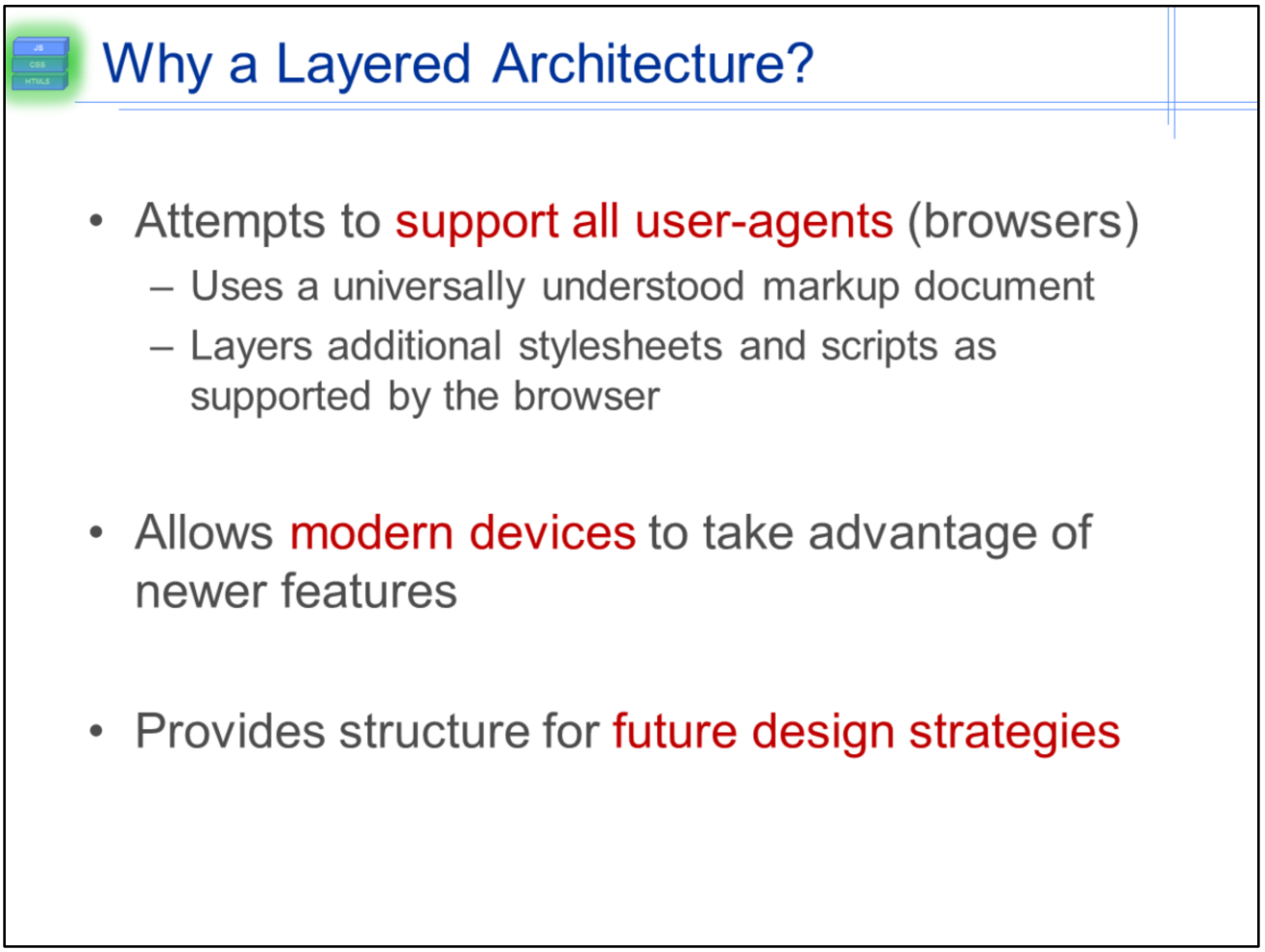




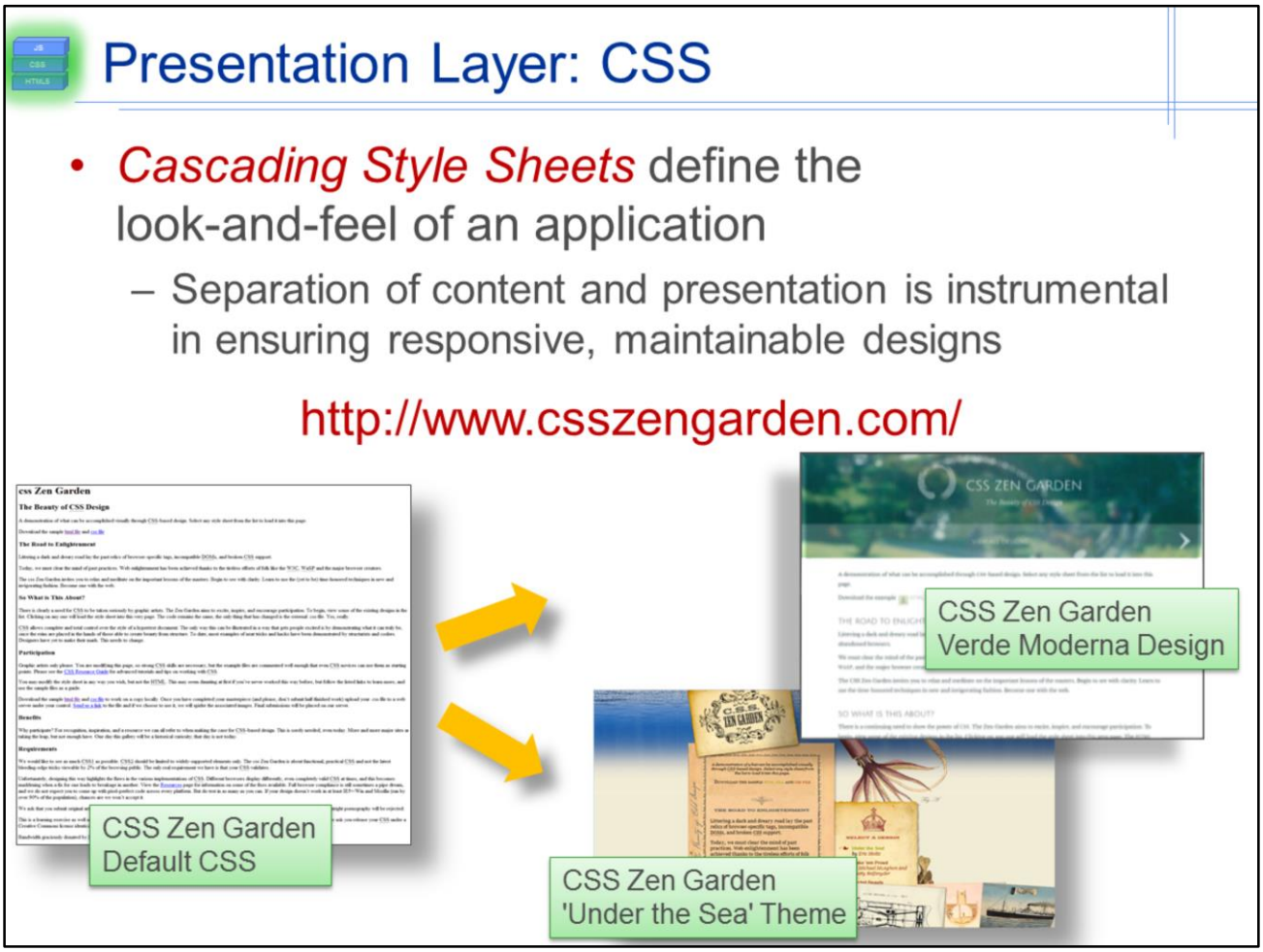
Source: marketshare.hitslink.com
Note the changes in desktop browser market share over the last 6 years. No longer should solutions be written for single-browsers. Even in situations where applications are targeted for internal clients whose browsers are strictly controlled, careful consideration should be given to targeting that single browser.



An architecture makes solutions more manageable when developing complex applications - architecture on the client side is analogous to trying to provide structure on the server-side.

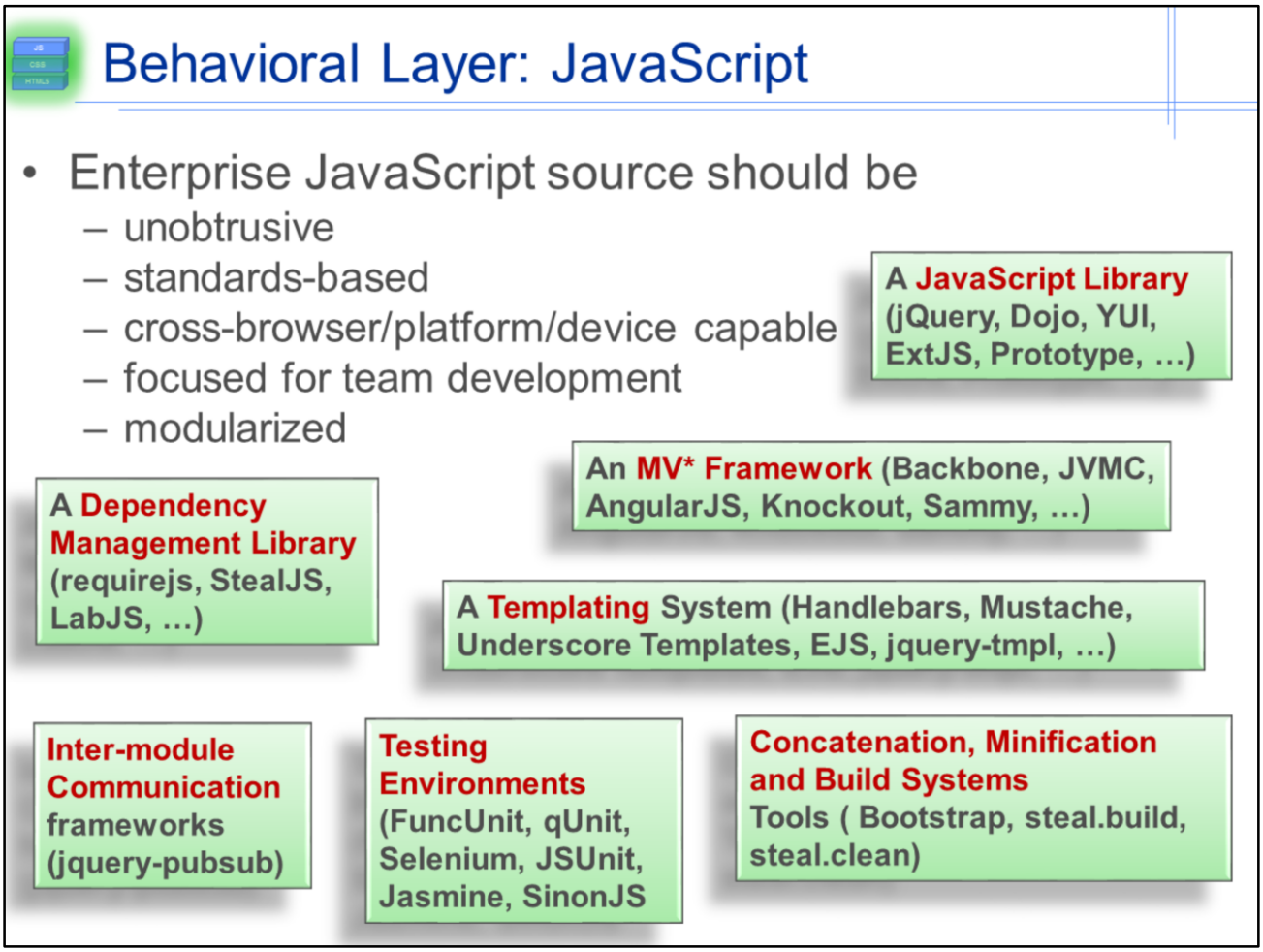


Layered architectures guard against changes to future end-user applications while at the same time attempt to work in the widest possible user-agents.

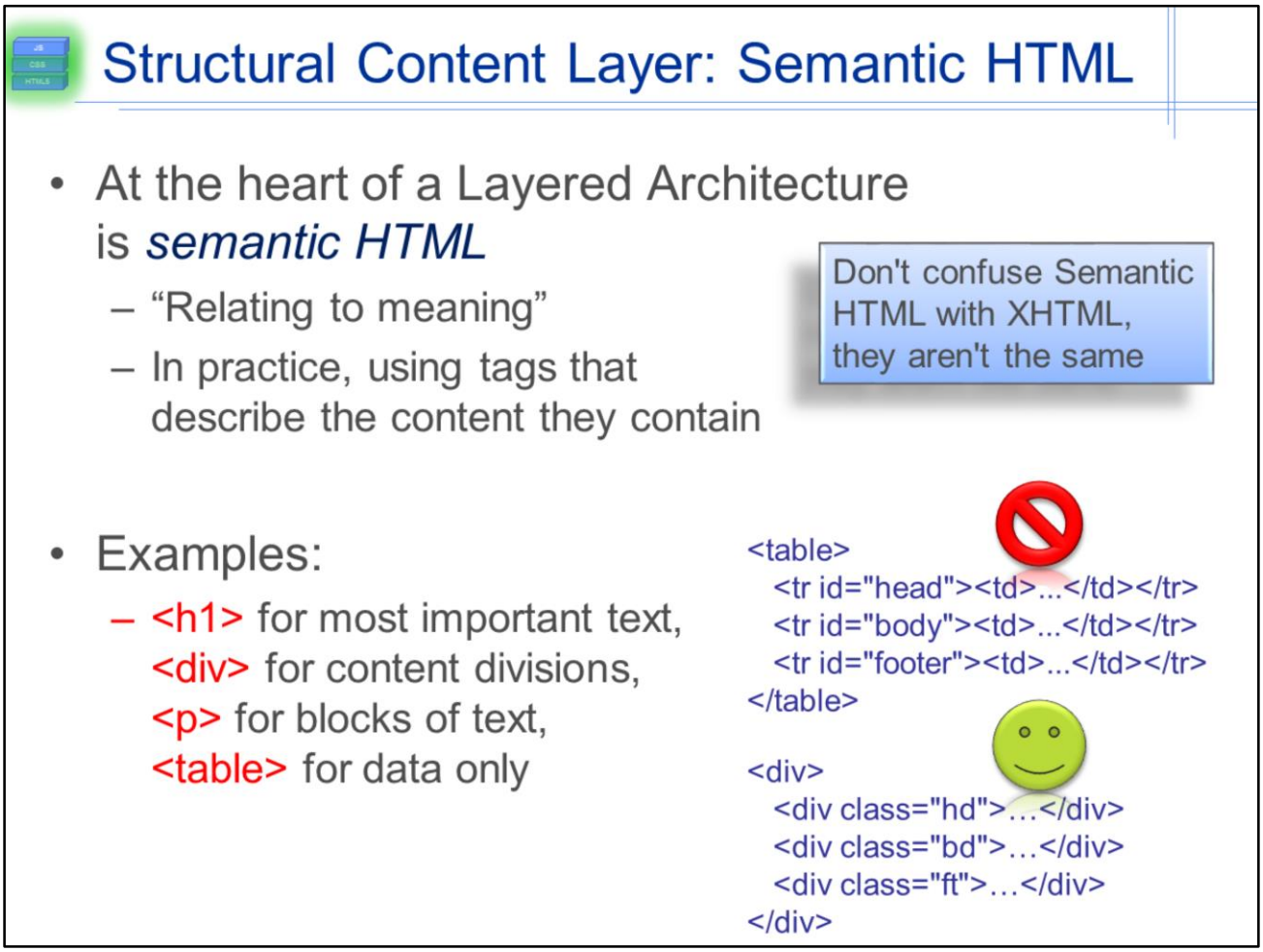


The presentation layer is controlled by CSS. By developing solutions where the presentation is entirely managed through CSS, a flexible, responsive layout can be created. In order to do his, it is vital to understand how to manipulate HTML elements (represented as DOM nodes within a browser) to generate a very specific, controlled (even dynamic) layout.

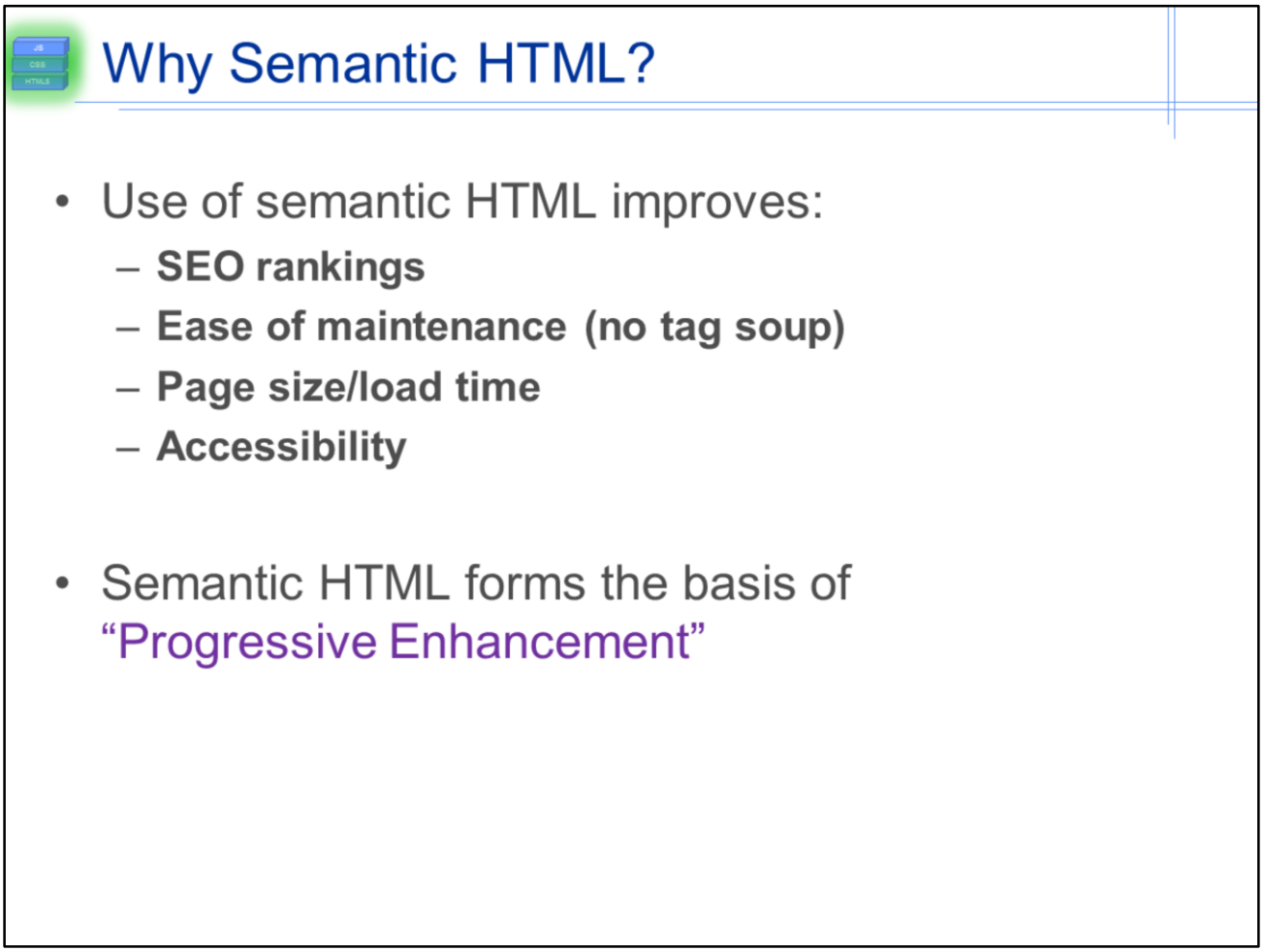
CSSZenGarden is a great example of a layered architecture.



Enterprise JavaScript is so named because it involves the use of frameworks that may not normally come up in smaller development environments. Larger team-based (enterprise) development typically have slightly more complex requirements for application development, testing, the build process, and the development environments.



Semantic HTML requires using HTML tags, not for presentation, but to provide as much semantic meaning as possible to the content. This is often difficult because HTML is a limited tag set, however, we can achieve some relative success in semantically marking up our documents. Don't confuse semantic HTML with XHTML. XHTML requires HTML to follow the syntactic rules of XML, it does not require semantic markup. It's possible to use XHTML and still have a poorly written document.
Using semantic HTML implies we will be using HTML elements to achieve maximum benefit for the content's purpose, not for the presentation's purpose.

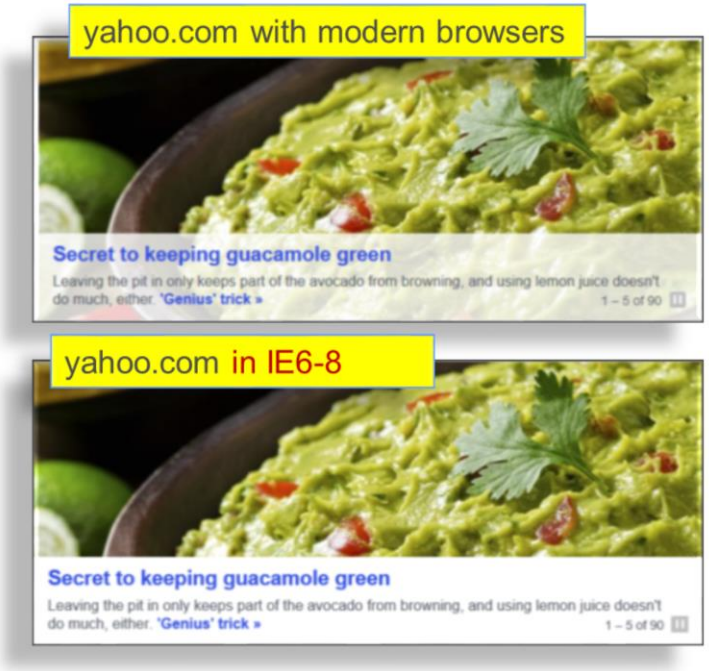


A semantically-driven approach to HTML increases useful life-span of that HTML.

"structural semantic markup provides lean, meaningful, accessible pages" – natekoechley

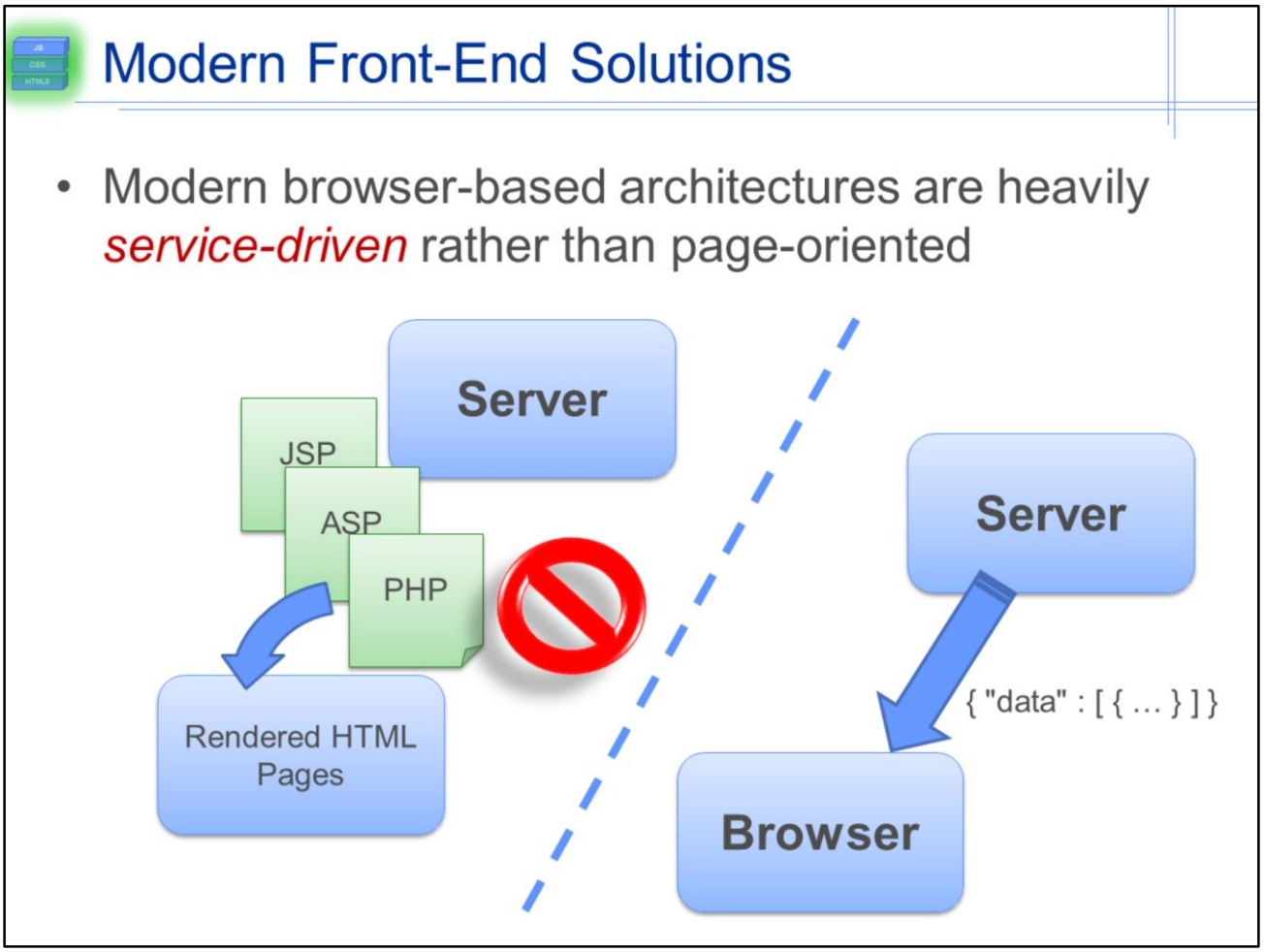
Progressive Enhancement

- *Creating the best possible solution for the widest number of users*
- Approach:
  - Develop a base level solution that puts content at the center of the design
  - Layer enhancements on top by adding presentational and behavioral features

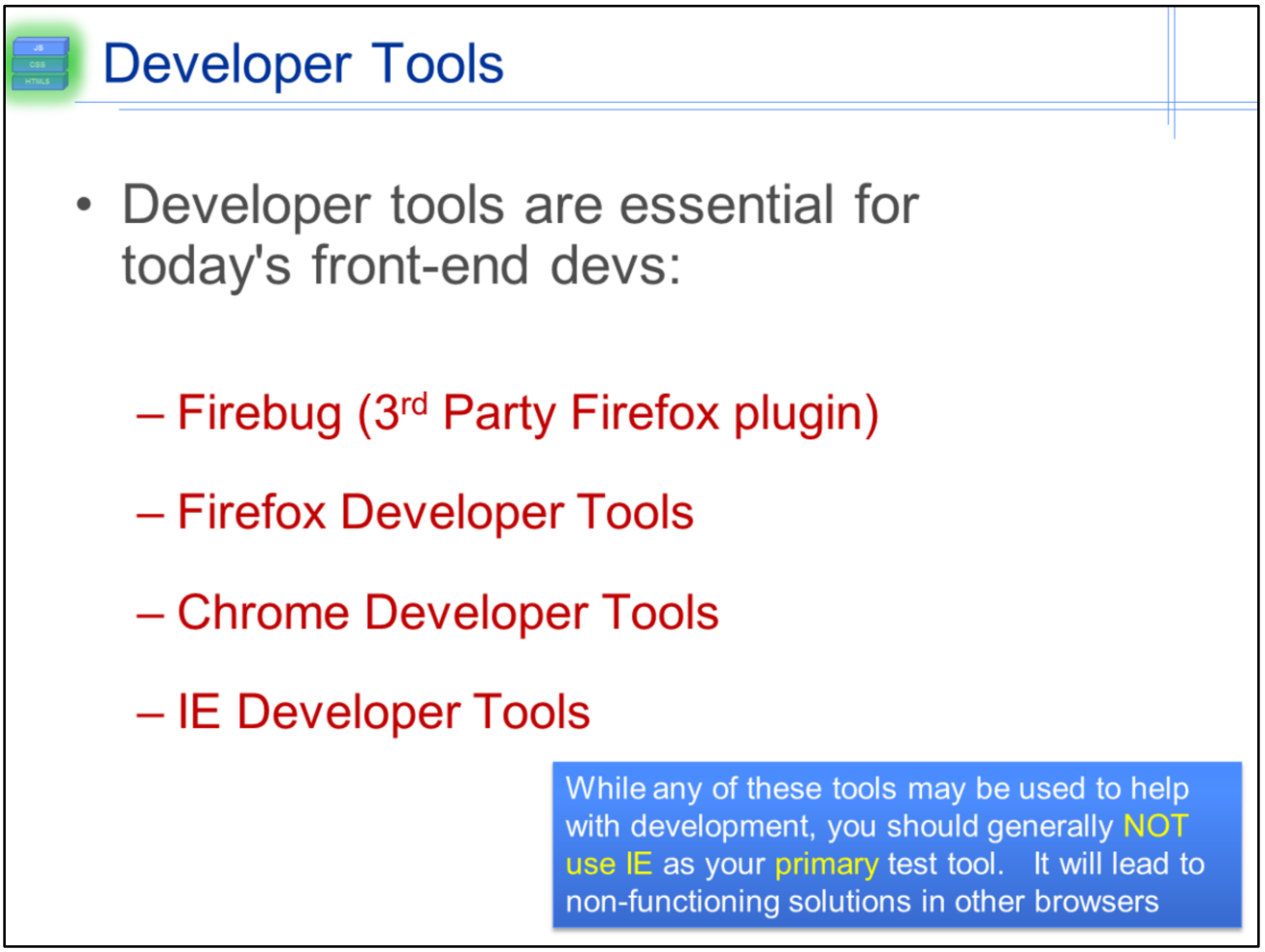


There are numerous examples of progressive enhancement in action. A YUI datatable widget is a good example because it transforms a basic HTML table (seen by all browsers) into a stylized, interactive one for browsers that support it.

Other examples of progressive enhancement include the use of CSS rounded corners, where IE browsers (8 and earlier) would see squared corner solutions while other browsers could see the more visually appealing rounded corner versions (assumes use of CSS rounded corner properties and not use of background image techniques).



Modern day thin-client solutions, especially those that exhibit the single-page, desktop-style application model, heavily emphasize service-oriented architectures. The data exchange format is often via JSON (JavaScript Object Notation) or perhaps JavaScript data or even proprietary data formats. While XML is not commonly used for most browser-based implementations, it can be selected as a data communications format.

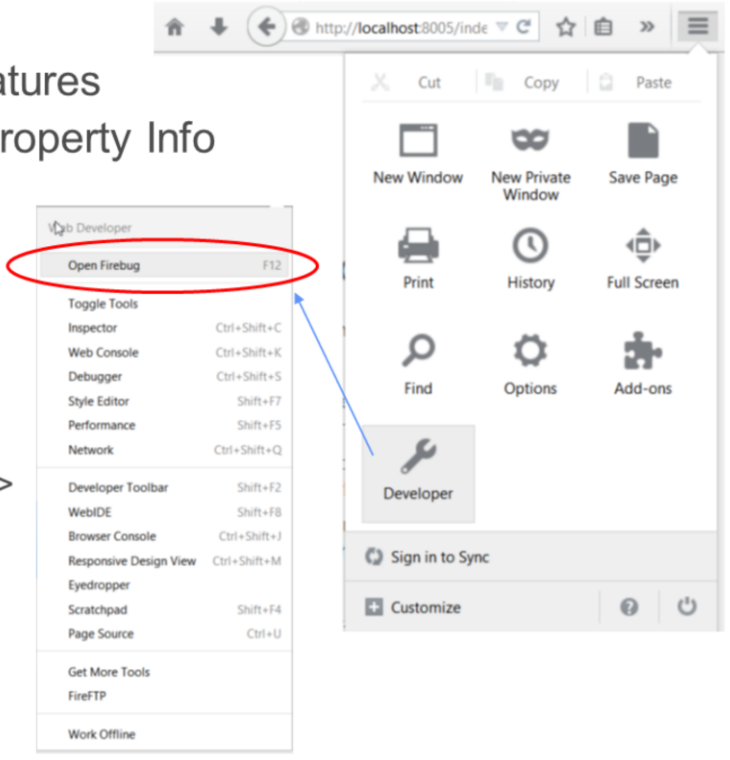


While IE11 has come a long way and features the best level of compatibility to date with the standards, most developers often prefer Chrome due to its speed and accuracy.

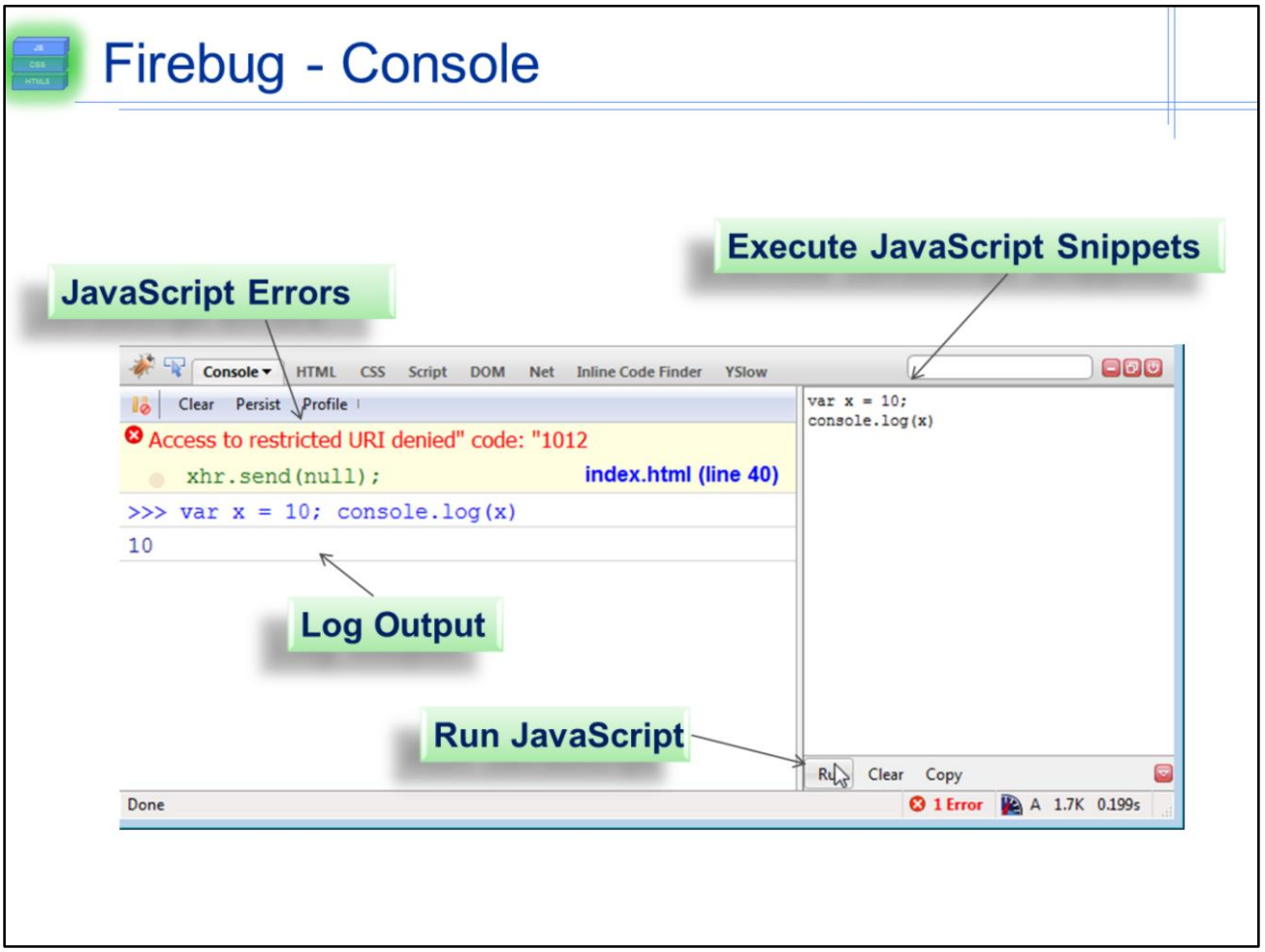
Firebug

Install from
http://getfirebug.com

- *Firefox plug-in that provides*
  - A Debugger
  - DOM Inspection Features
  - Box Model & CSS Property Info
  - Event Notifications
  - Profiling Info
  - XHR Monitoring
- F12 to launch, or
  Options Menu > Developer > Open Firebug...



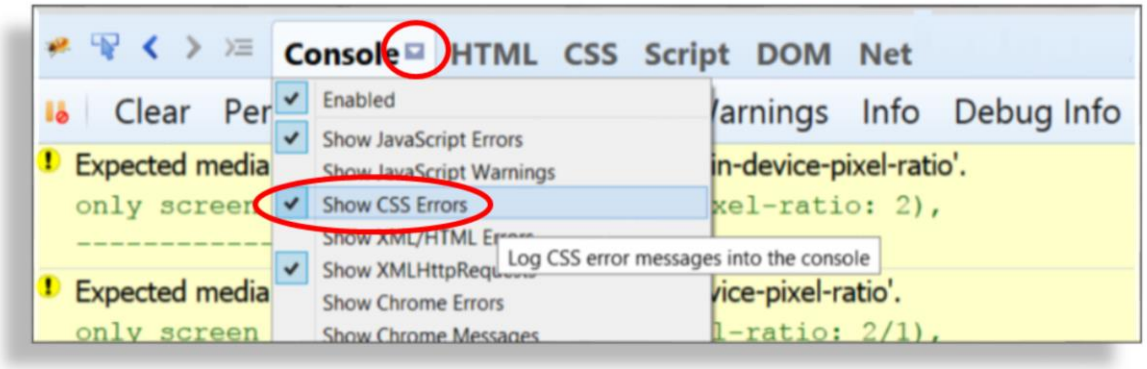
Test to see if Firebug is installed by pressing F12.



The Firebug console is useful for viewing errors, warnings, and log output. It can also be useful to test JavaScript Snippets.

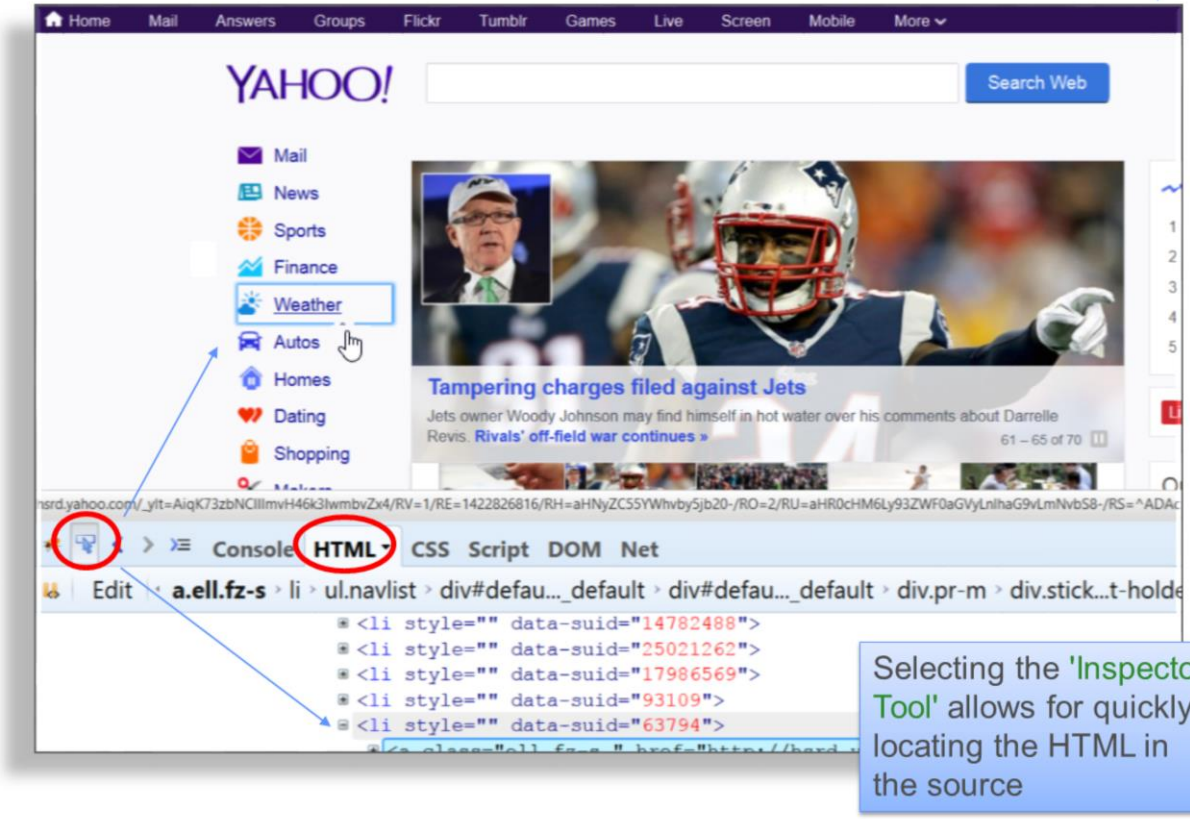
Firebug - Console (*for this class*)

- For this class, it might be useful to turn on CSS Errors (off by default) if you use Firebug:

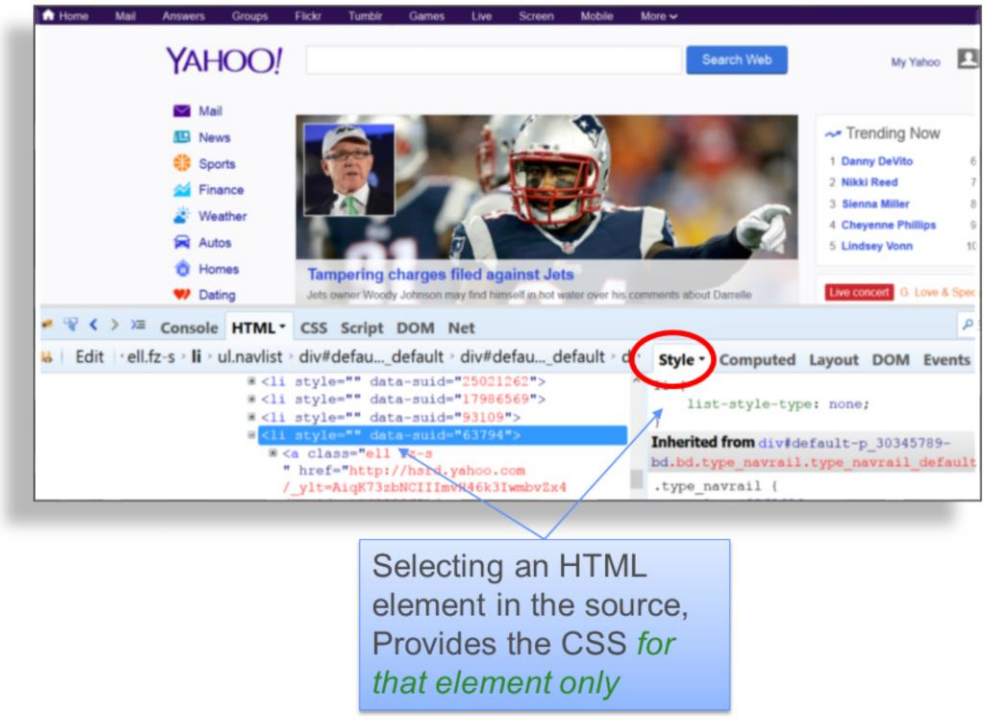


Though these items are referred to as CSS Errors, Firebug reports CSS items that the Mozilla rendering engine did not understand--it doesn't mean that you should remove the error as it may be a perfectly fine statement in other browsers.

Firebug - Using the *HTML* Tab

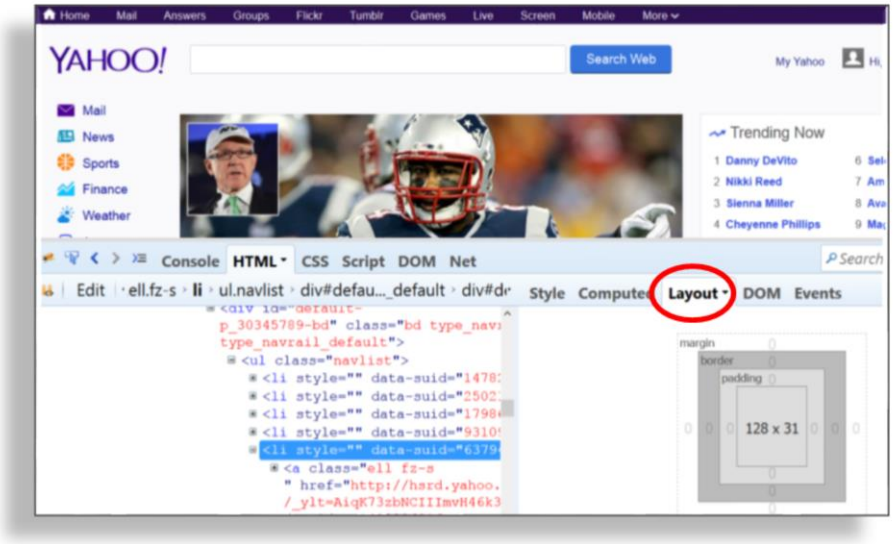


Firebug - *HTML* Tab - Style Information



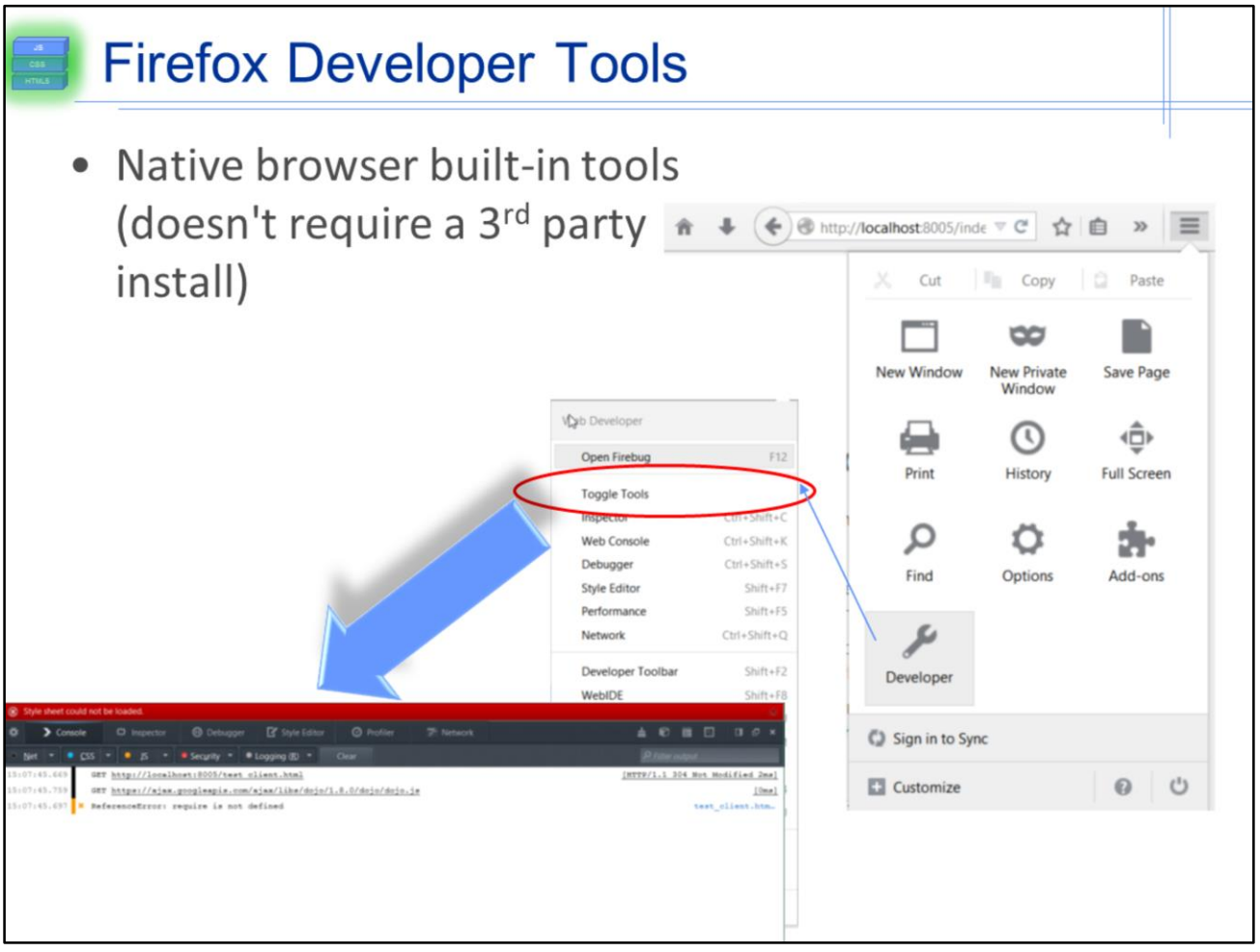
Once an element is selected, you may view the CSS related to this element in the Style tab (as shown).

Firebug - *HTML* Tab - Layout (Box Model)



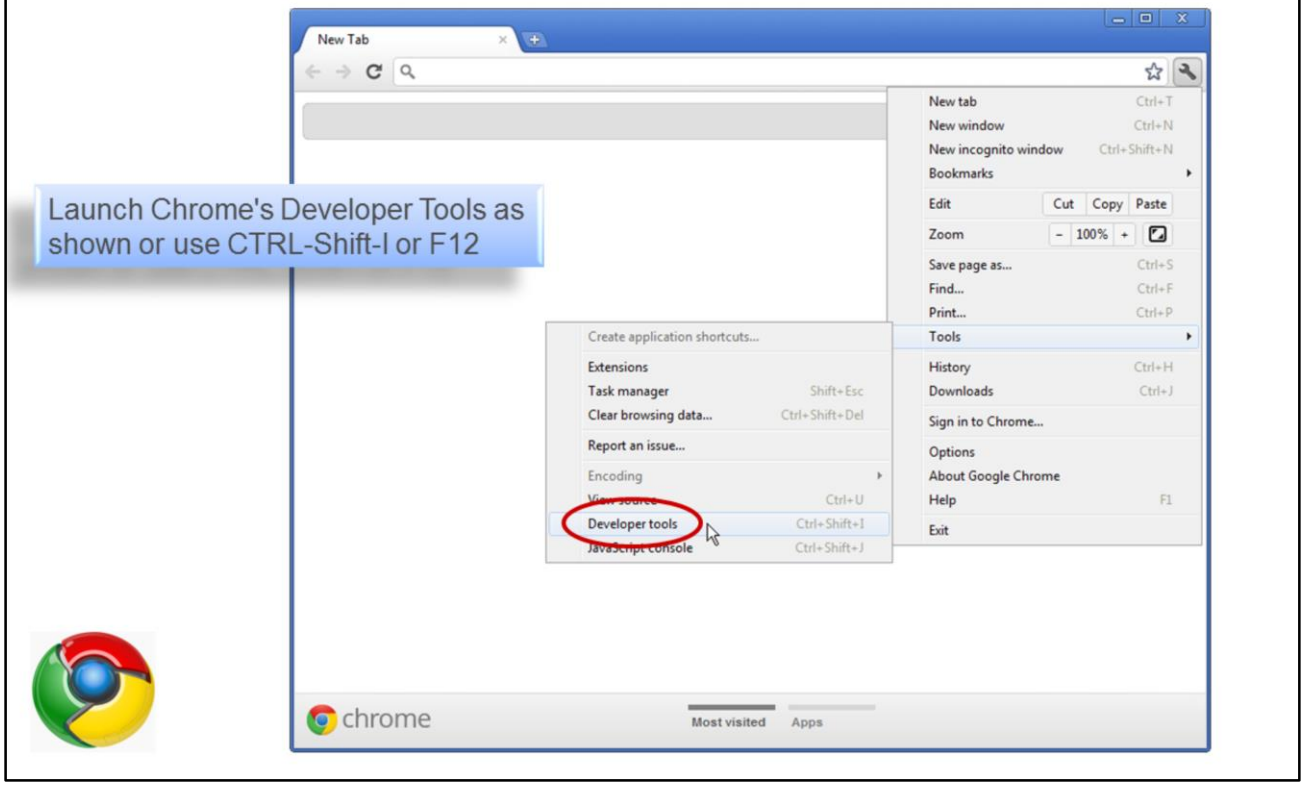
The Layout Tab provides a visual indication of the elements properties

Once an element is selected, you may also get a visual cue of the element's properties using the Layout Tab.

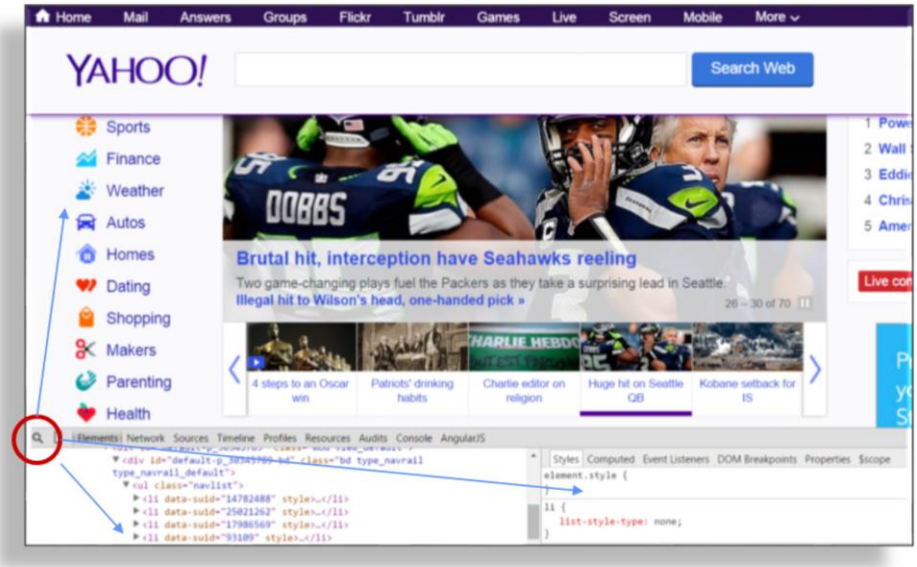


Firefox has a set of built in developer tools in lieu of Firebug if desired. These tools ship with Firefox and do not have to be installed. (CTRL-SHIFT-K).

Chrome Developer Tools



Chrome Developer Tools - *The Elements Tab*

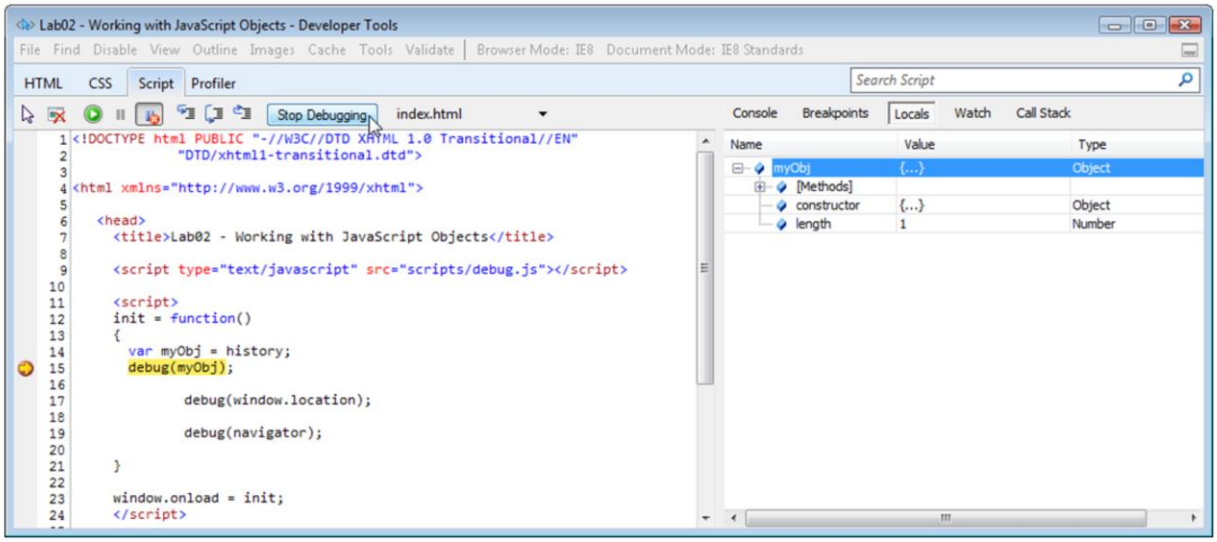


Use the 'Inspector Tool' to select elements on a page, then view the corresponding HTML source below

As with Firebug's HTML Tab, Chrome's Elements Tab serves a similar purpose

IE 8-10 Developer Tools

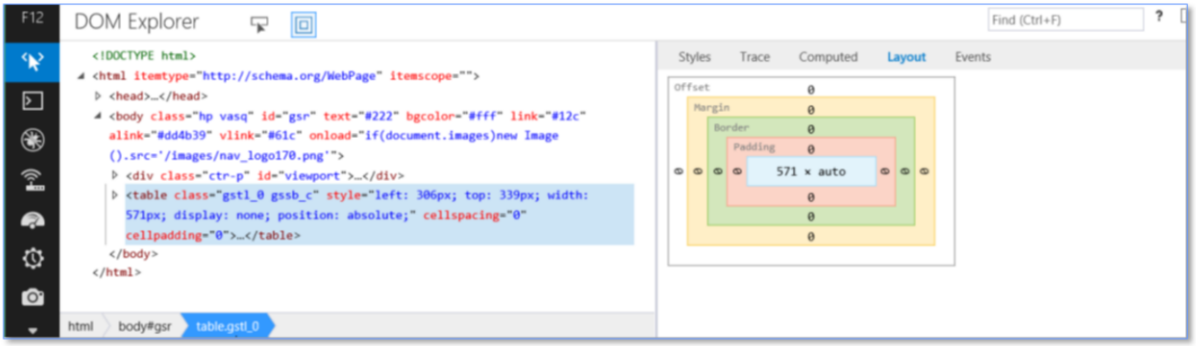
- IE 8+ provides a built-in debugger and DOM inspector also
  - Much more crude and limited in capabilities



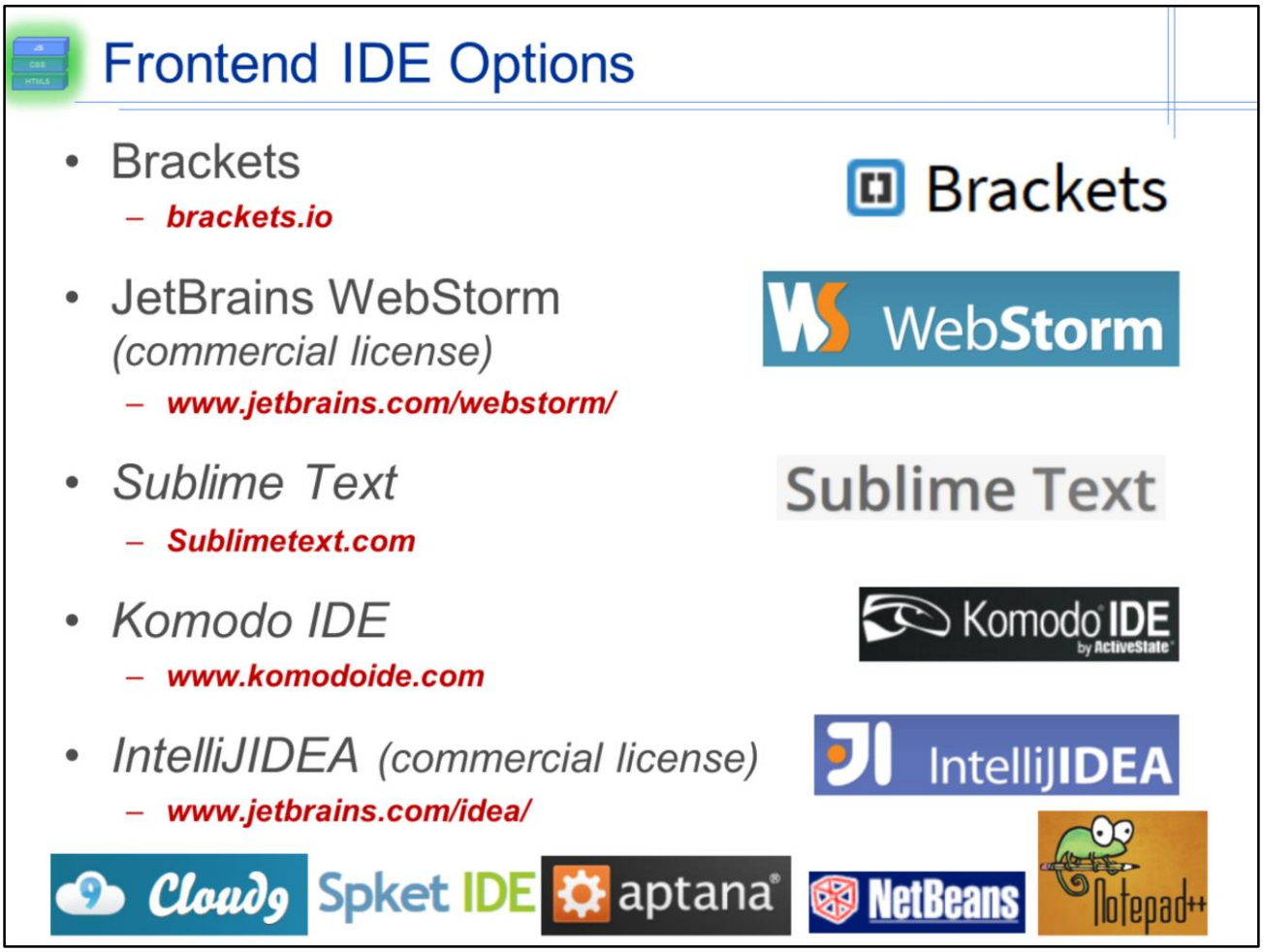
To open the developer tools, use an F12.

IE11 Developer Tools

- Use IE11 has improved its developer tools
  - Similar to Firebug in many ways



IE11 now provides developer tools on par with Firebug/Chrome Dev Tools. Selecting Browser/Document Modes can be done on the Emulation tab. (Bottom icon on the left).



Exercise 1 - It's Your Turn

Explore ESPN.com using Chrome Developer Tools or Firebug

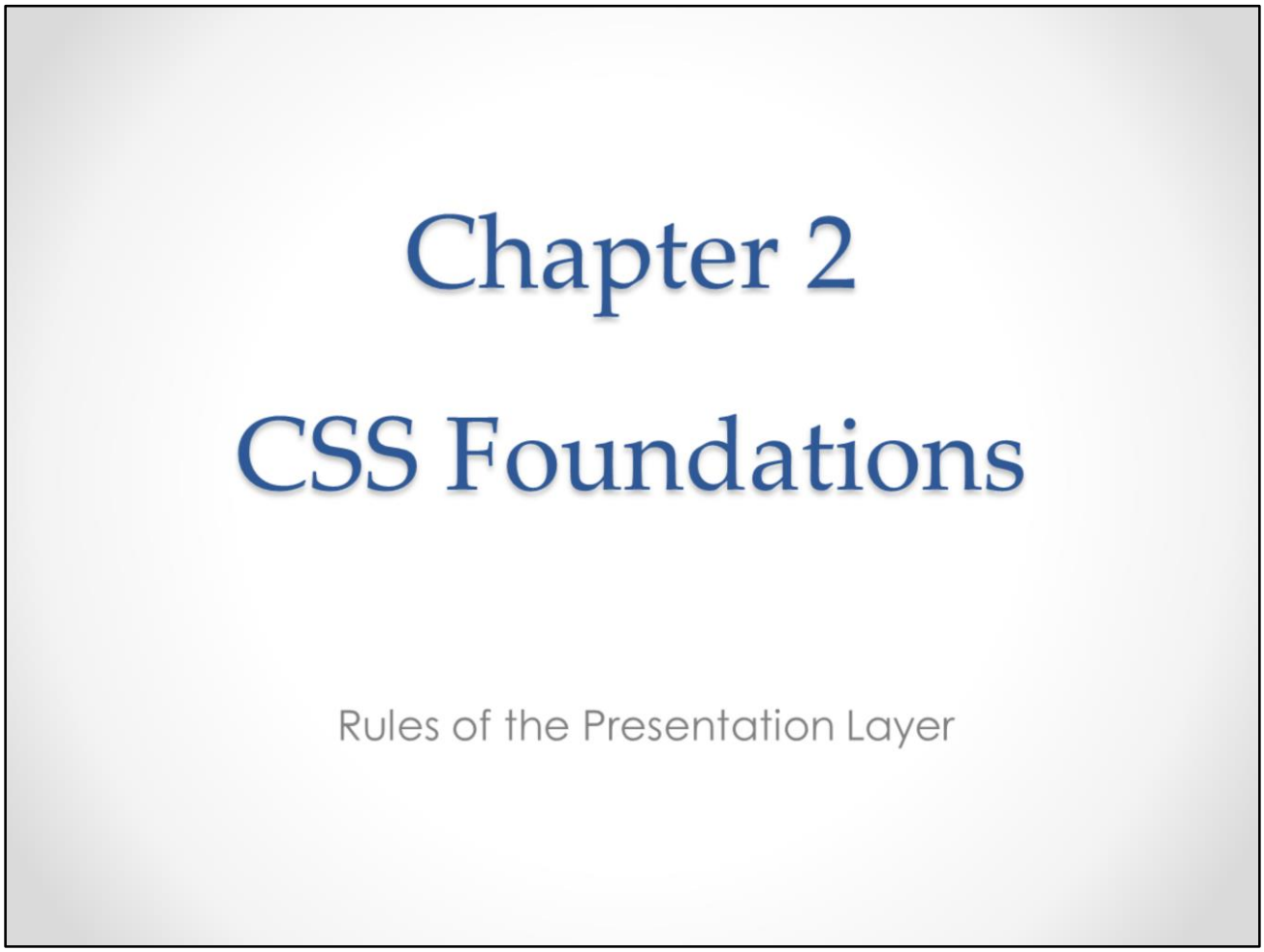
Perform Exercise 1

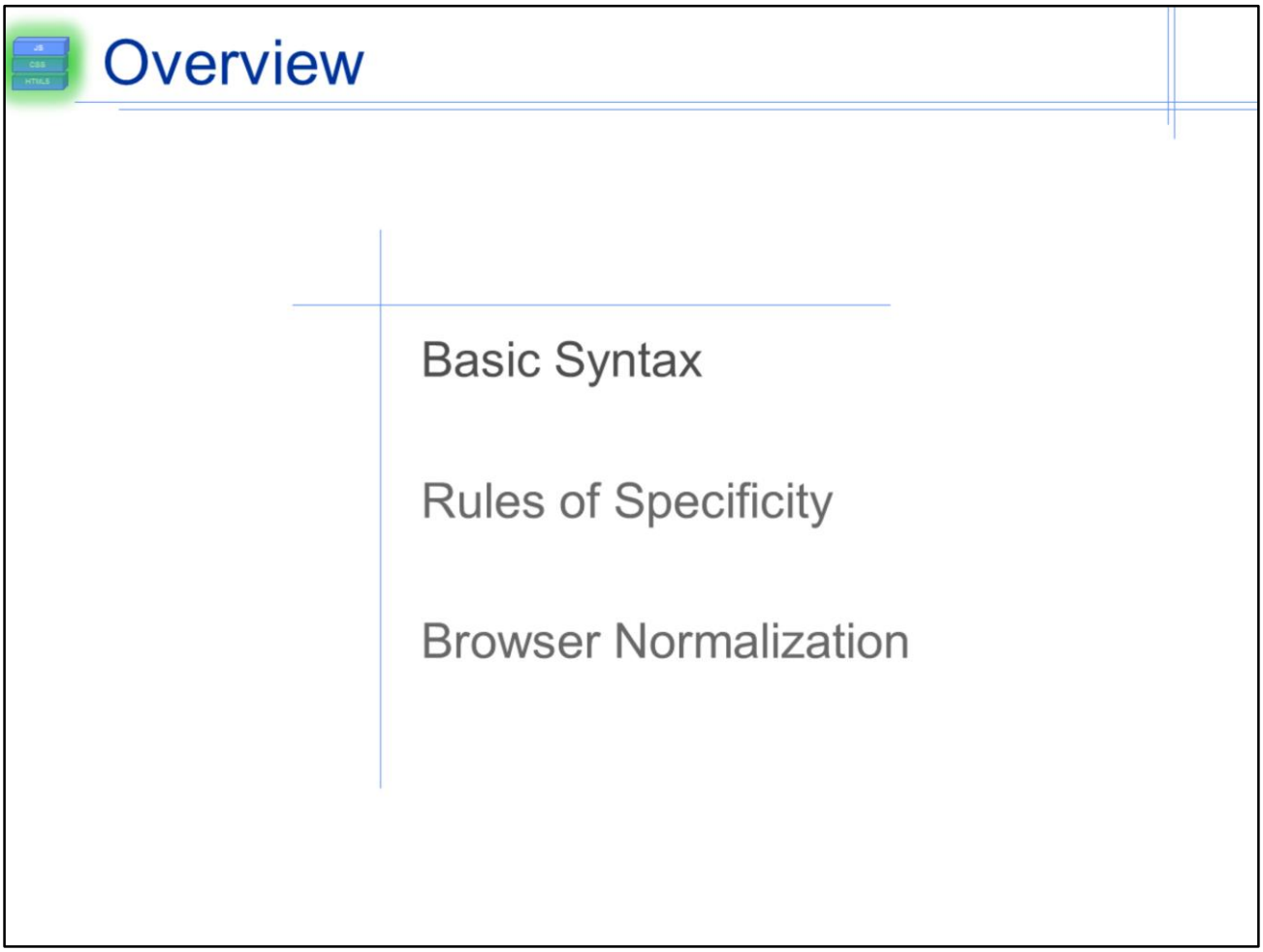
Using the Browser Developer Tools

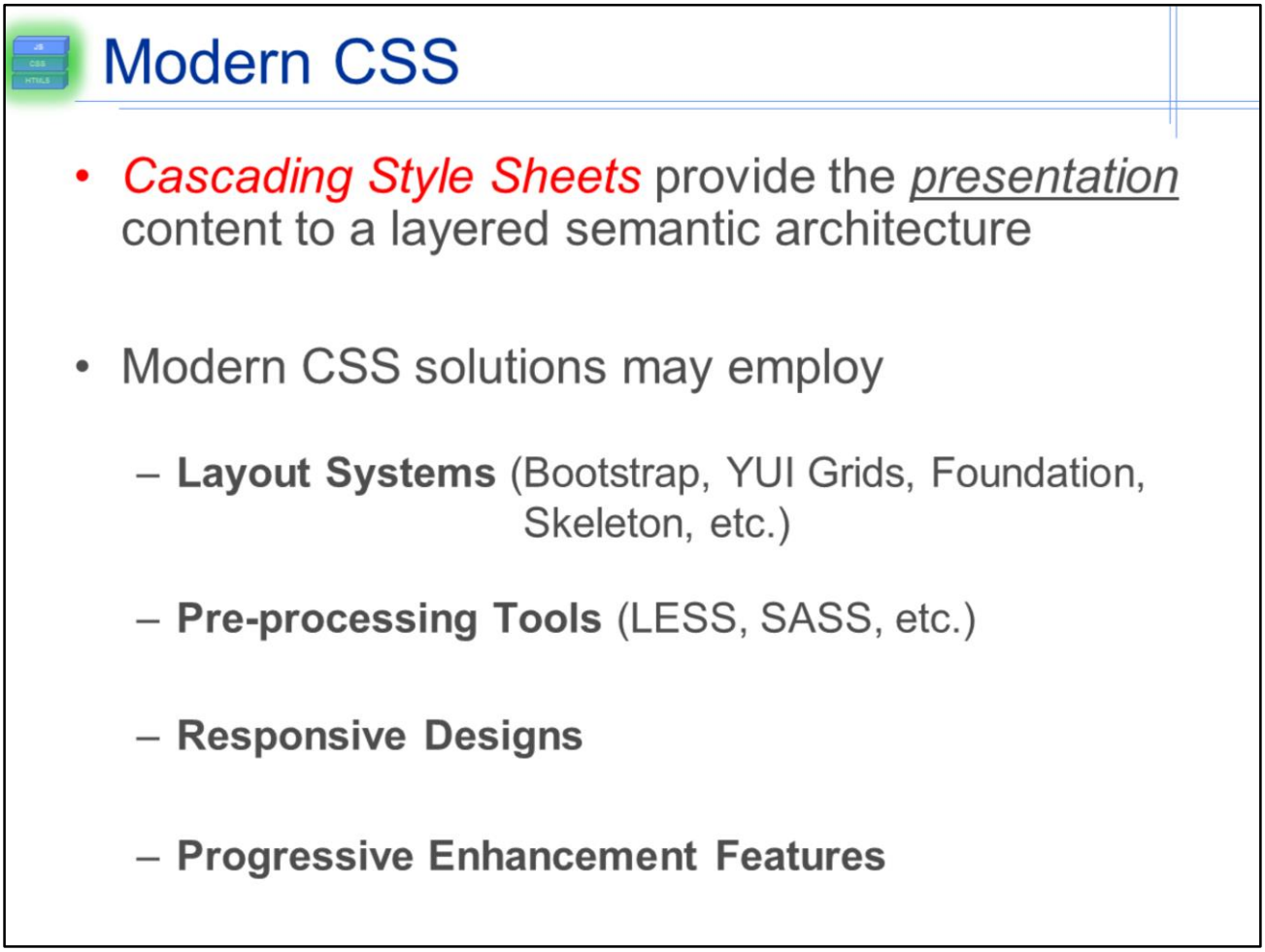
Locate step-by-step instructions for this lab in the back of your manual

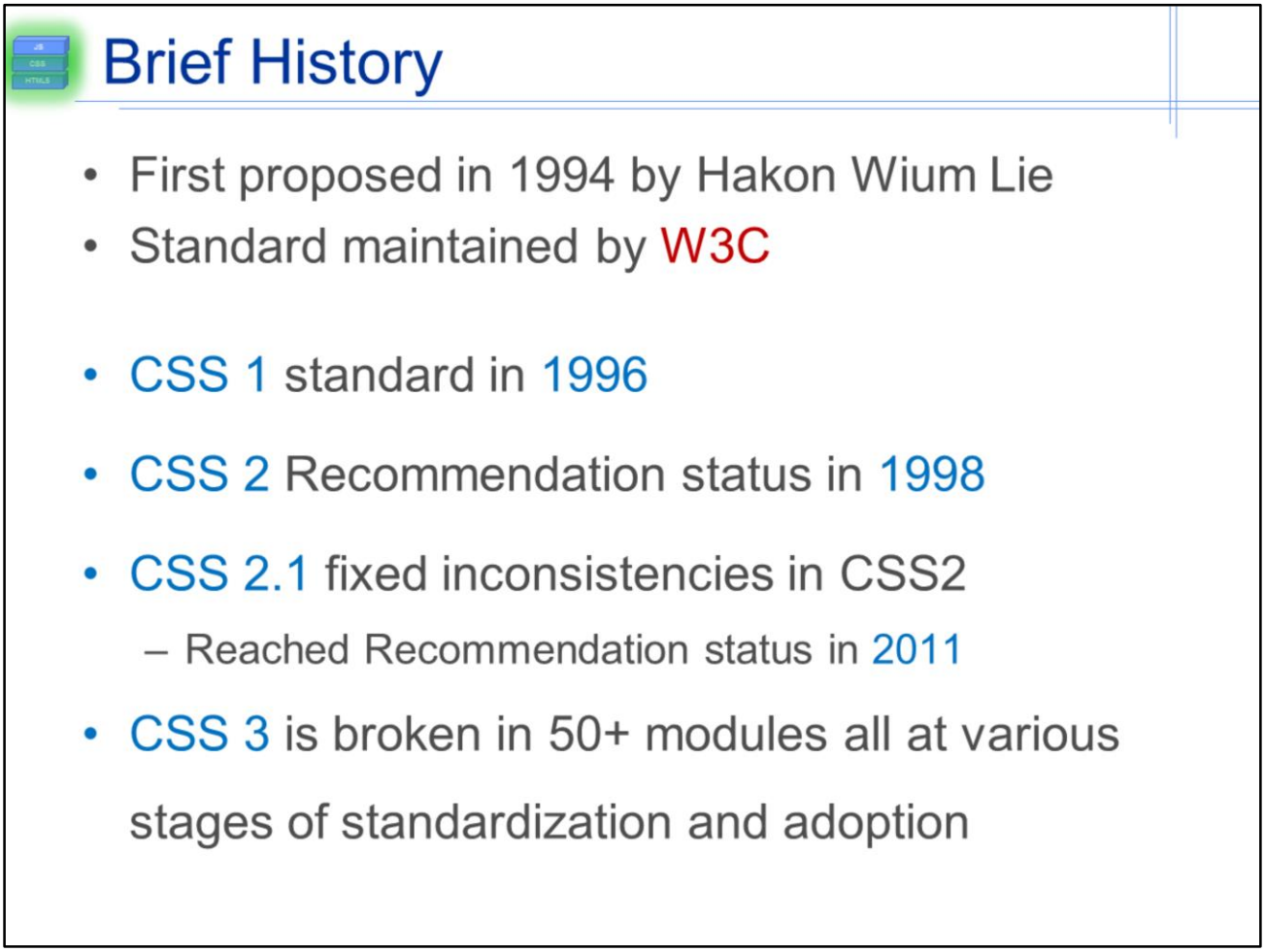
Summary

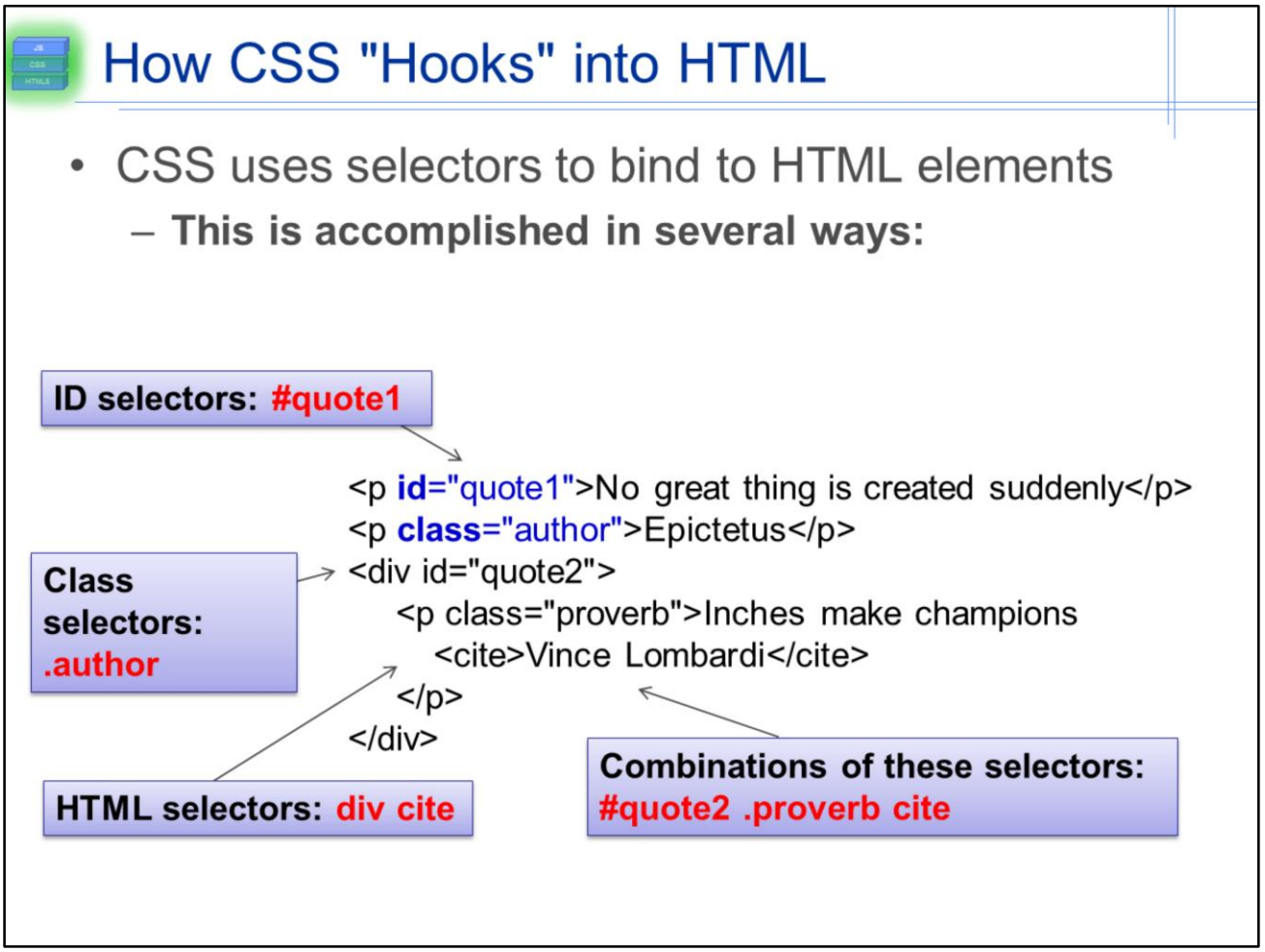
- Modern frontend development should combine several technologies (JavaScript, CSS, HTML) into a *layered architecture*
- Consideration for different browsing environments should be designed at the beginning of the development process
- Use newer tools and IDEs to facilitate the development process

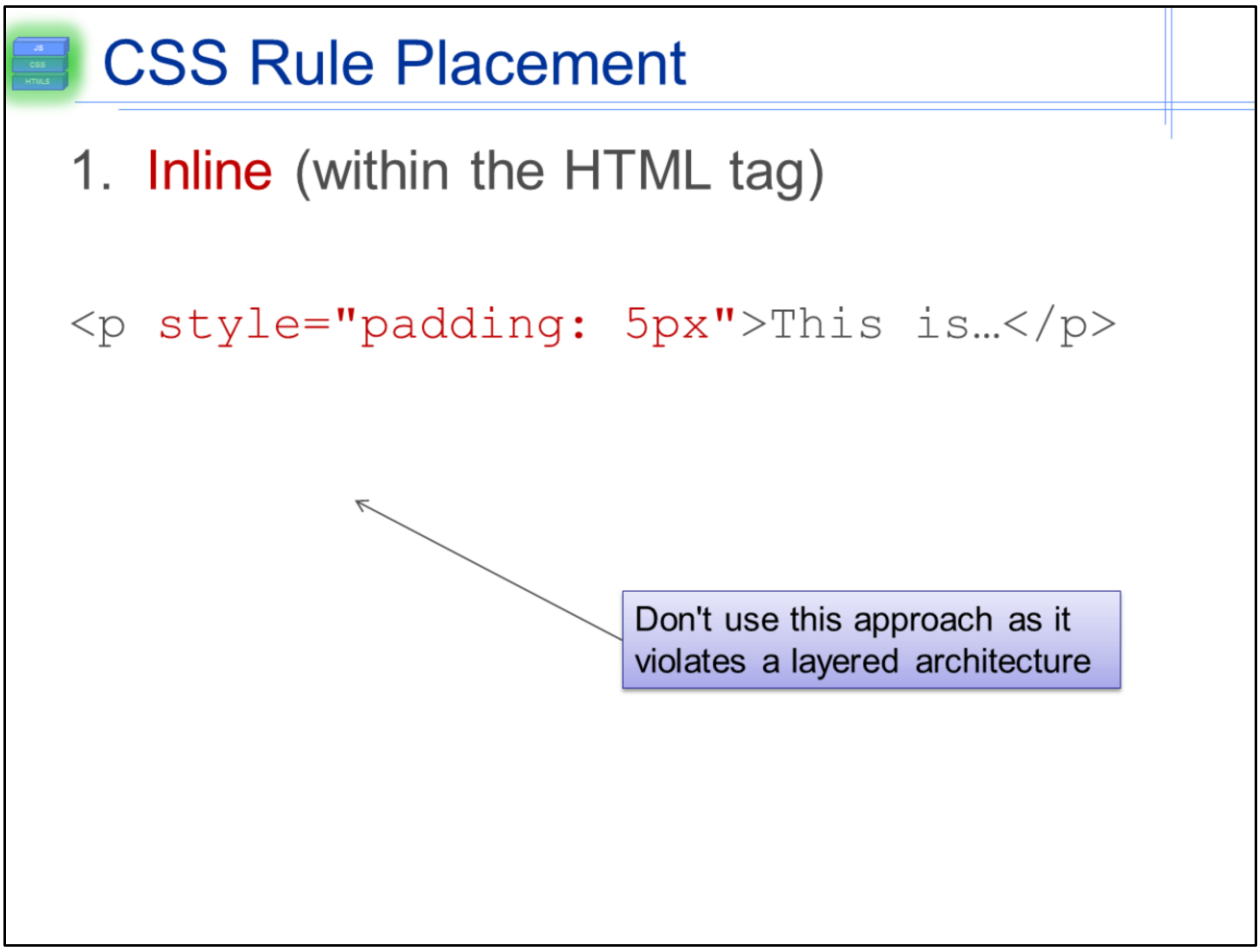


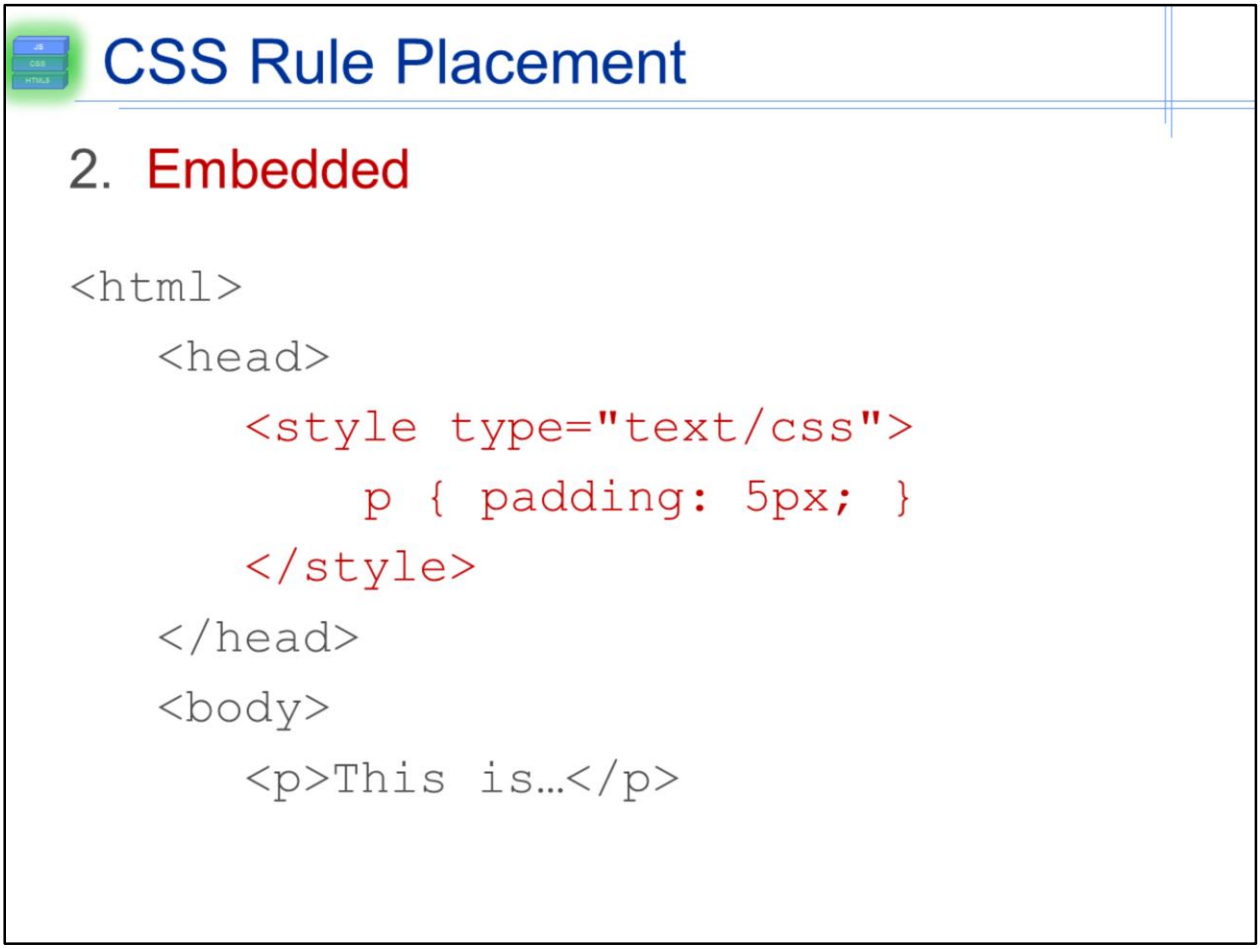


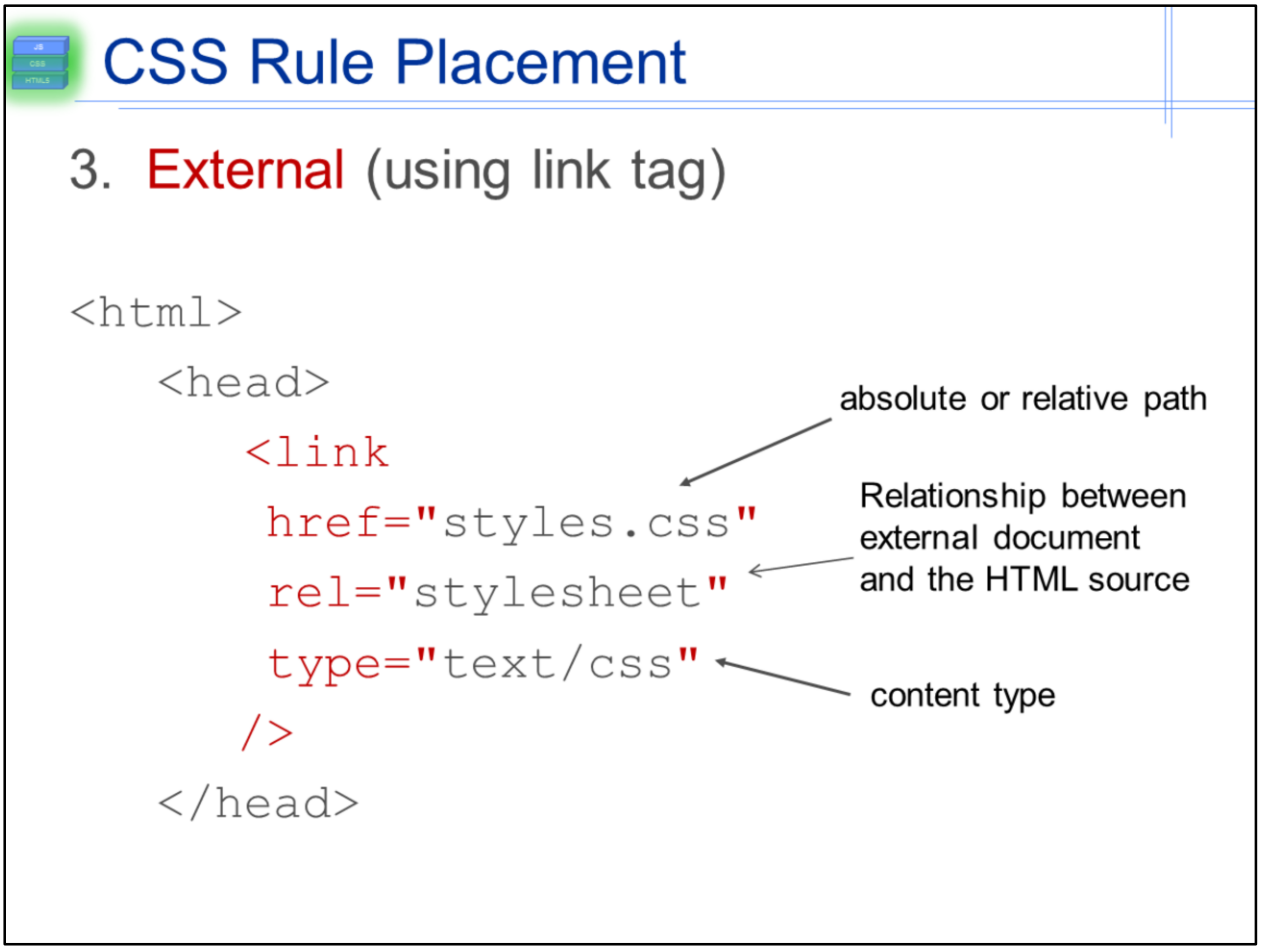




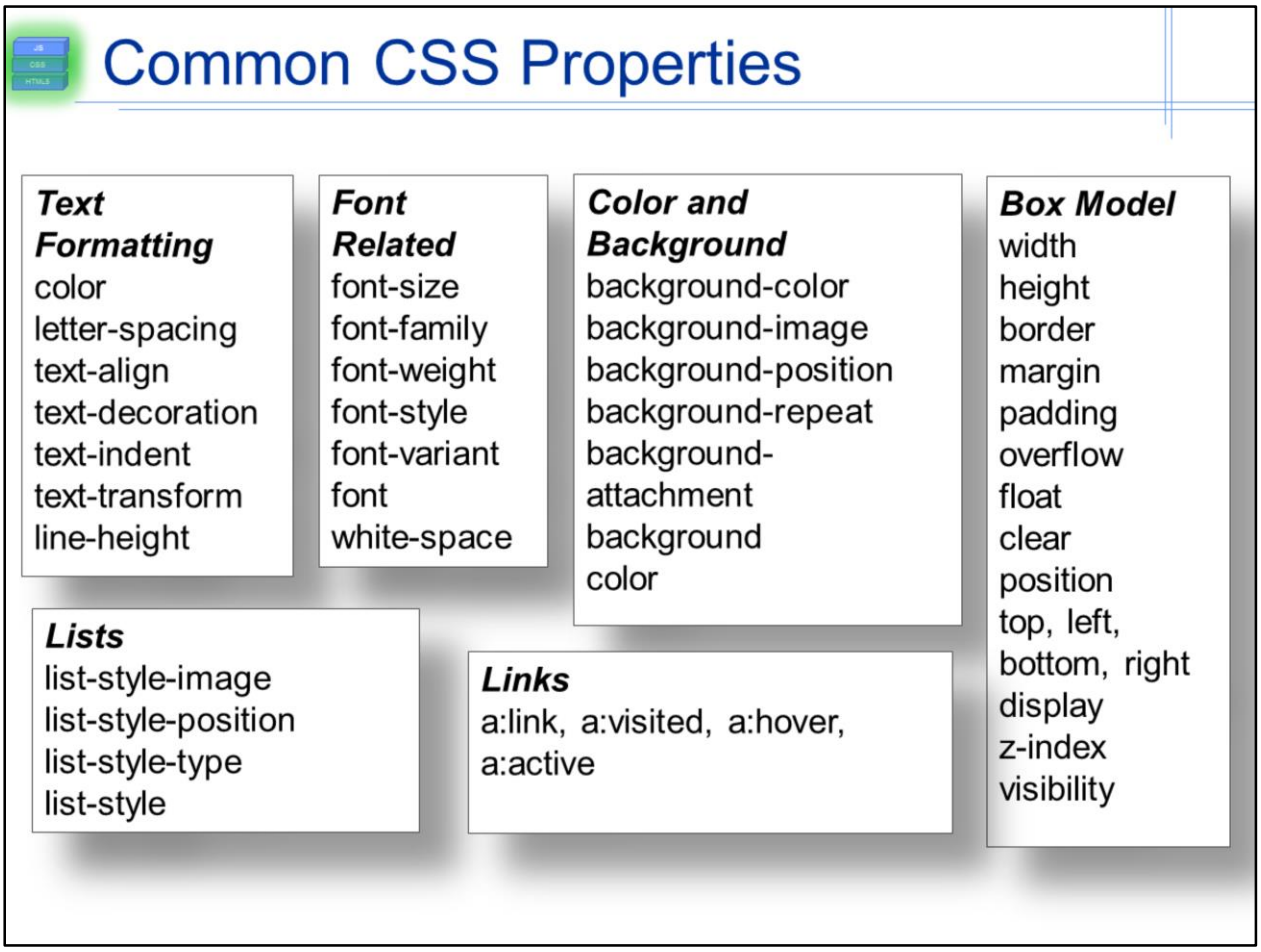


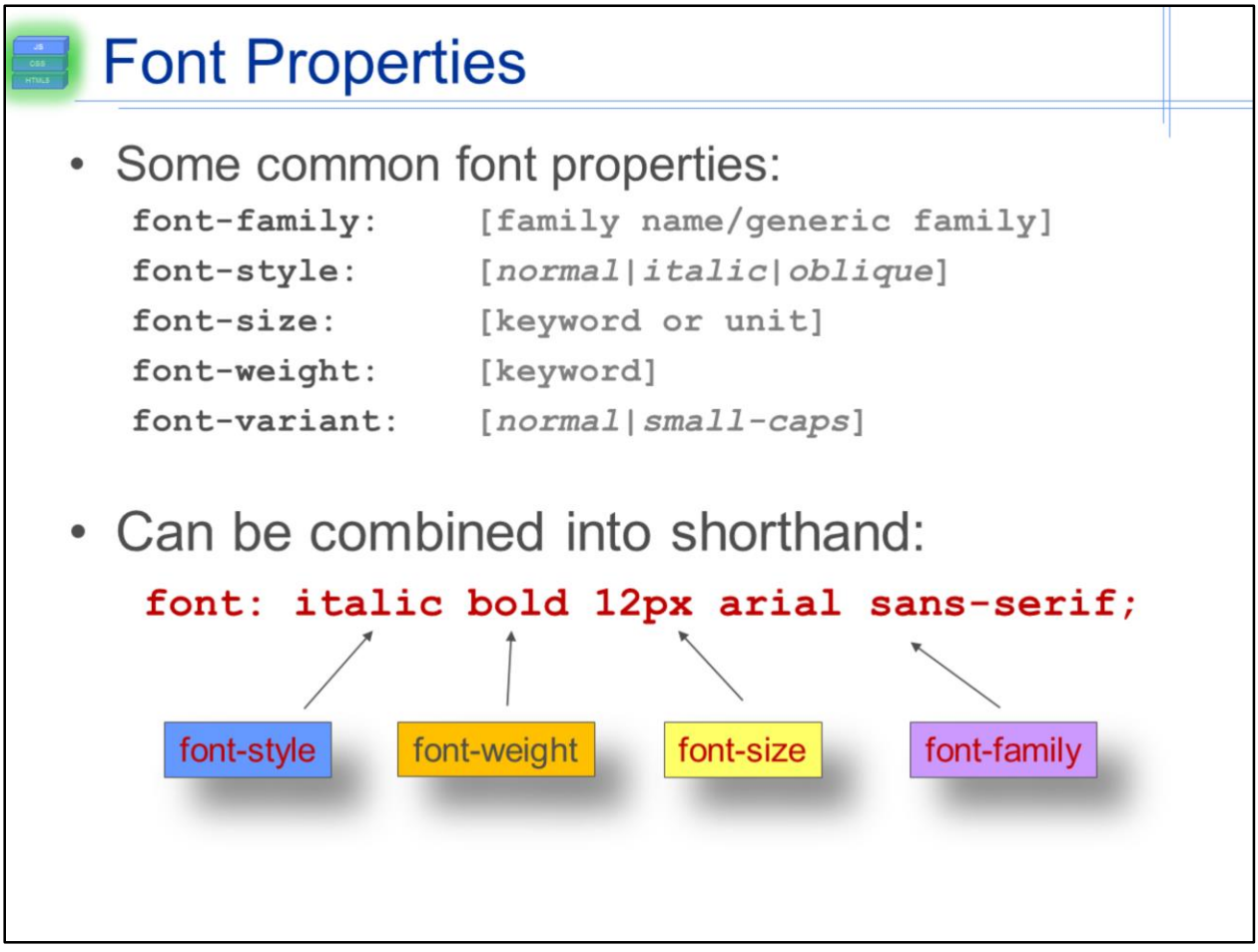




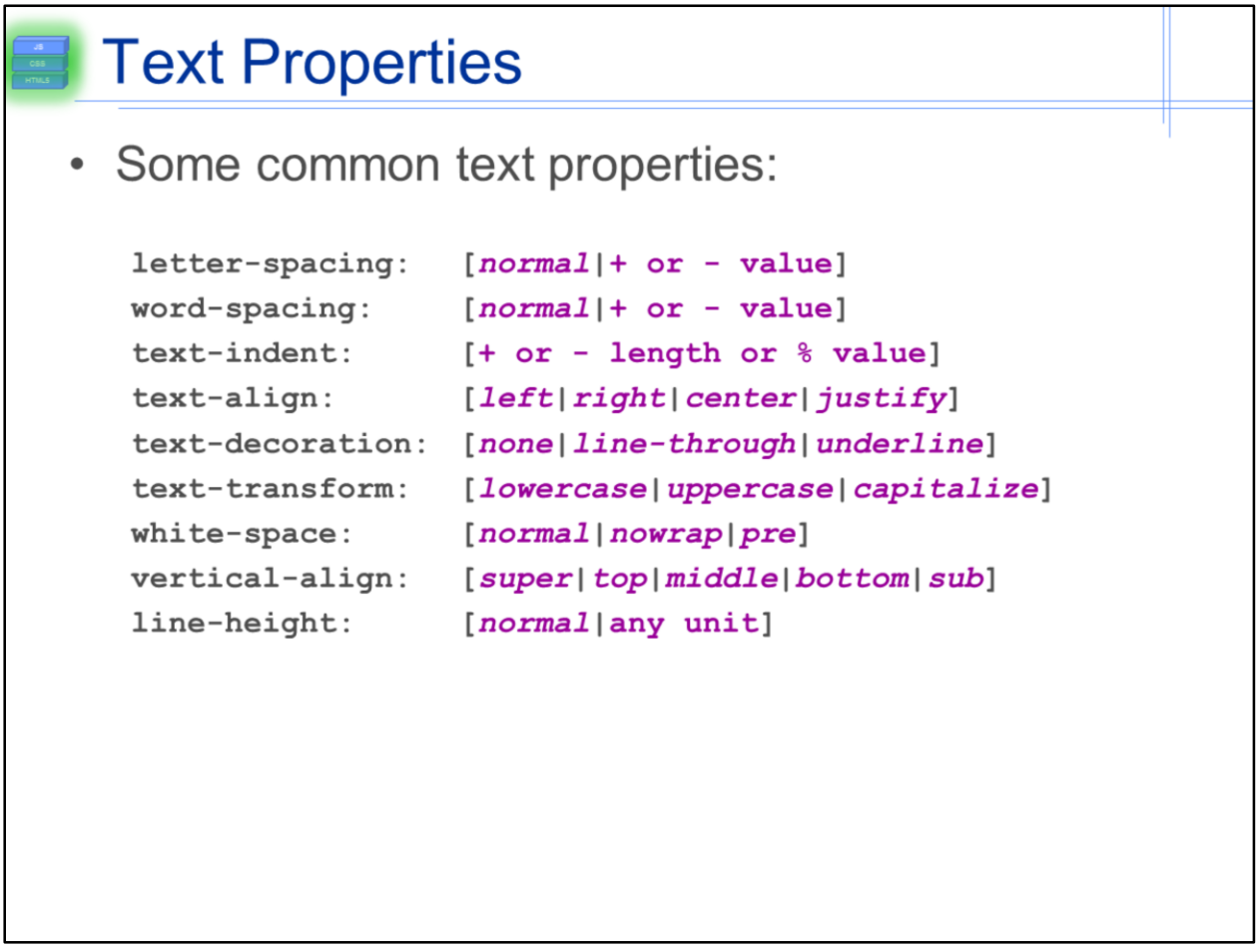


External CSS files are cached by browser – which can help with performance when you reuse the same CSS rules on multiple pages

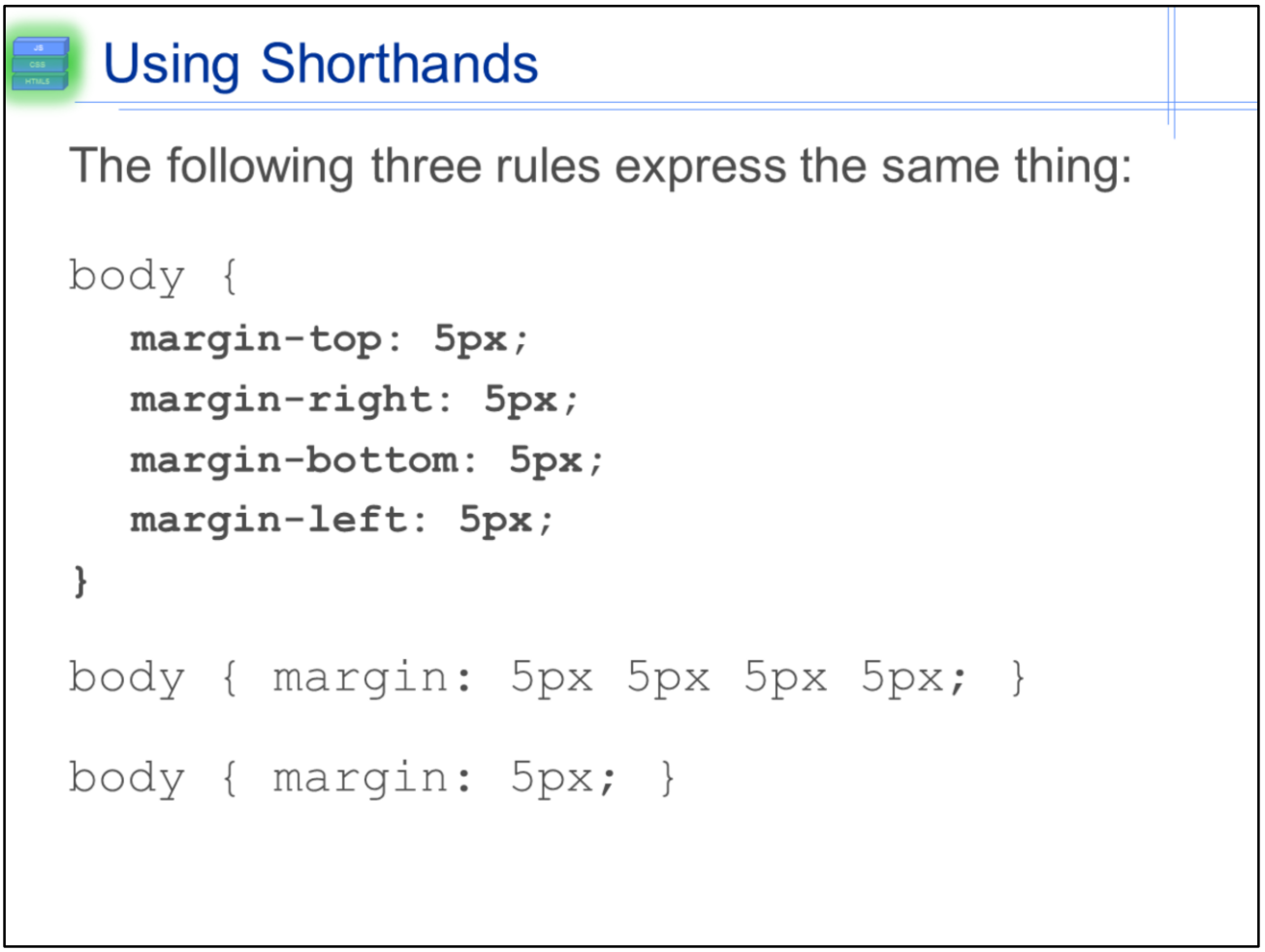




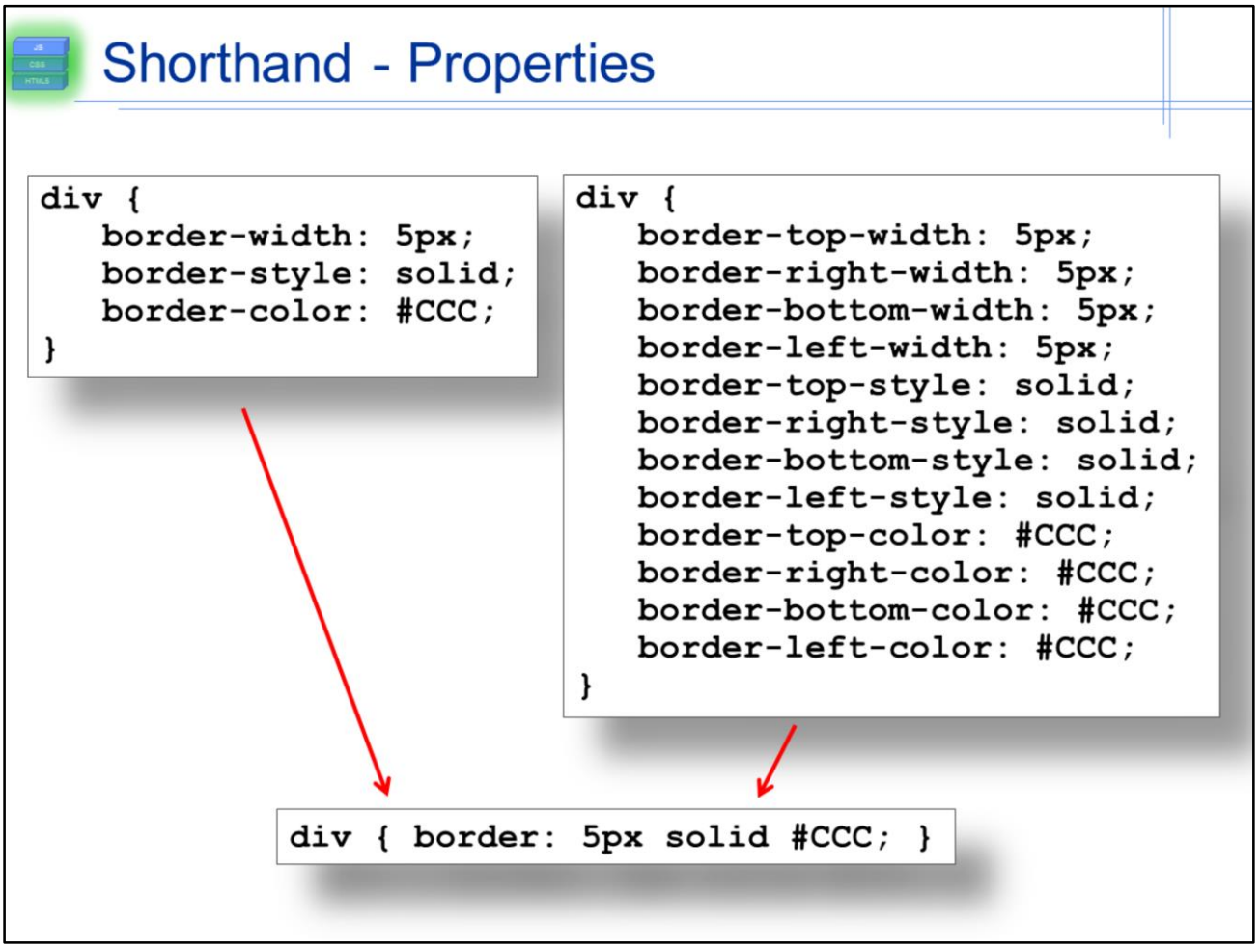
Not recommended to use font shorthand: if you don't specify the font-weight, font-style, or font-variant then these values will automatically default to a value of "normal" (which may override any normally inherited values!)



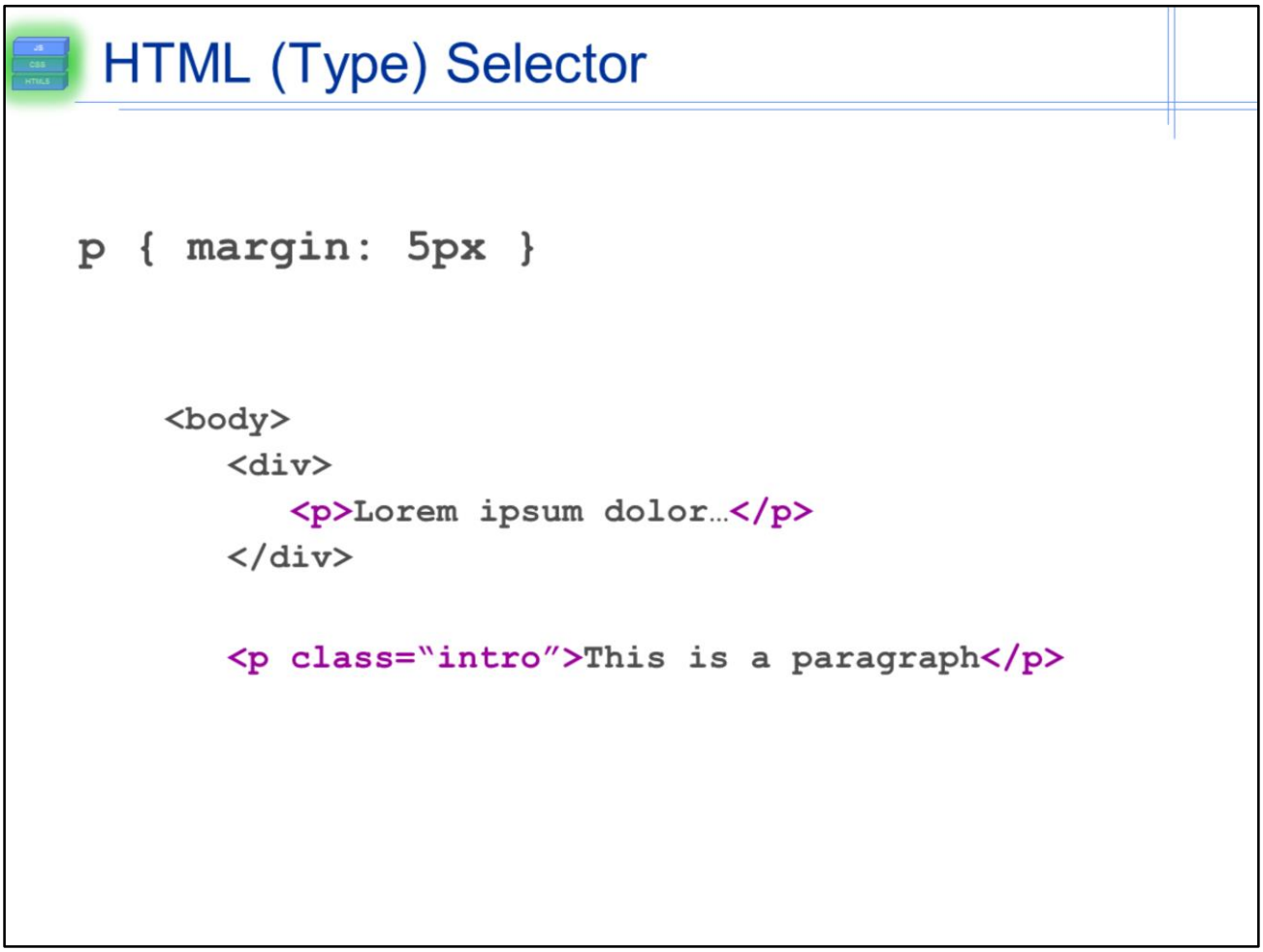
Text decoration can have multiple values at the same time
"pre" will retain preformatted text white-spacing
"line-height" defines the spacing between the lines (eg, single space, double space etc…)



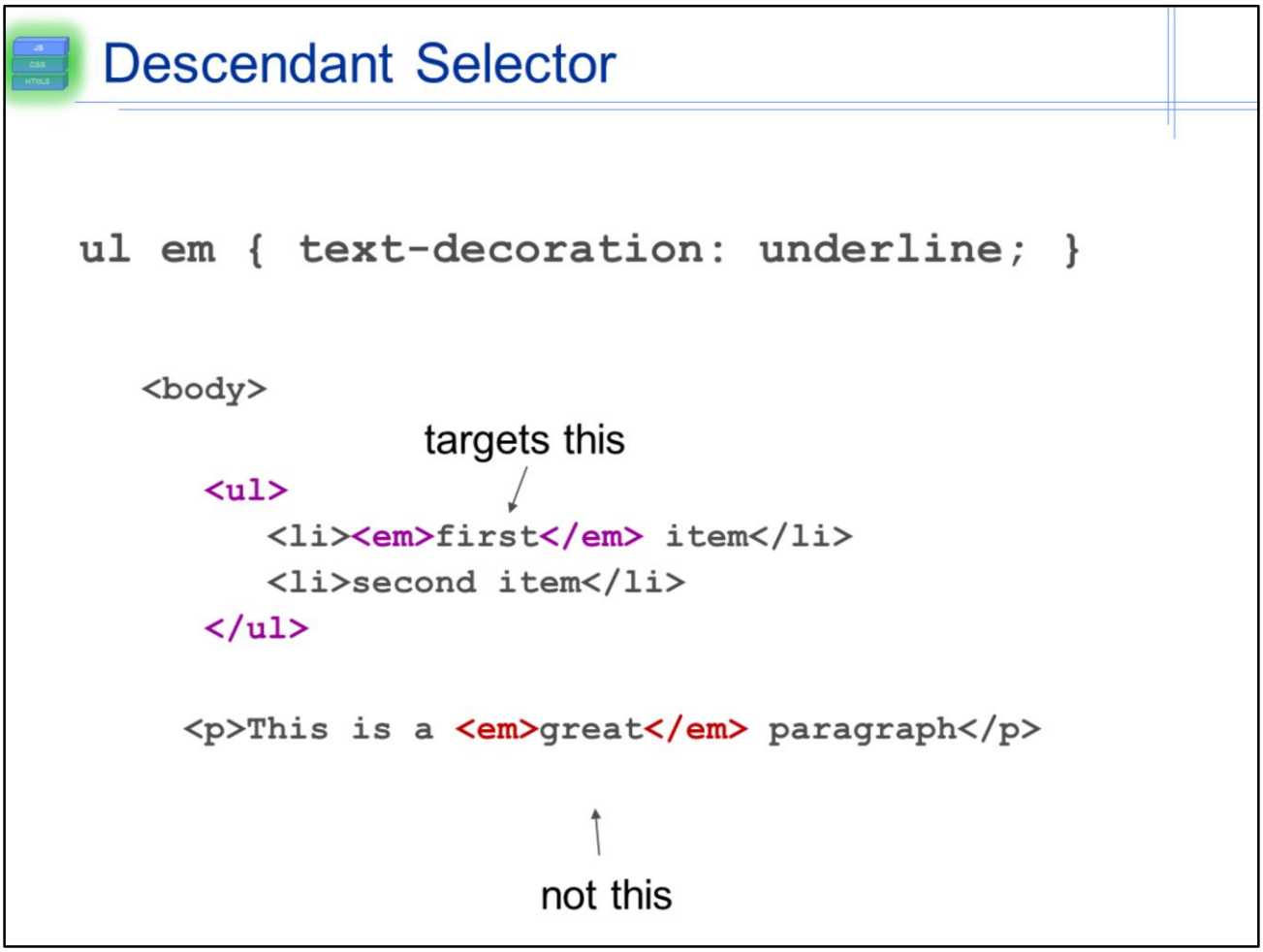
The order of the middle example is “top, right, bottom, left” (clockwise from top).



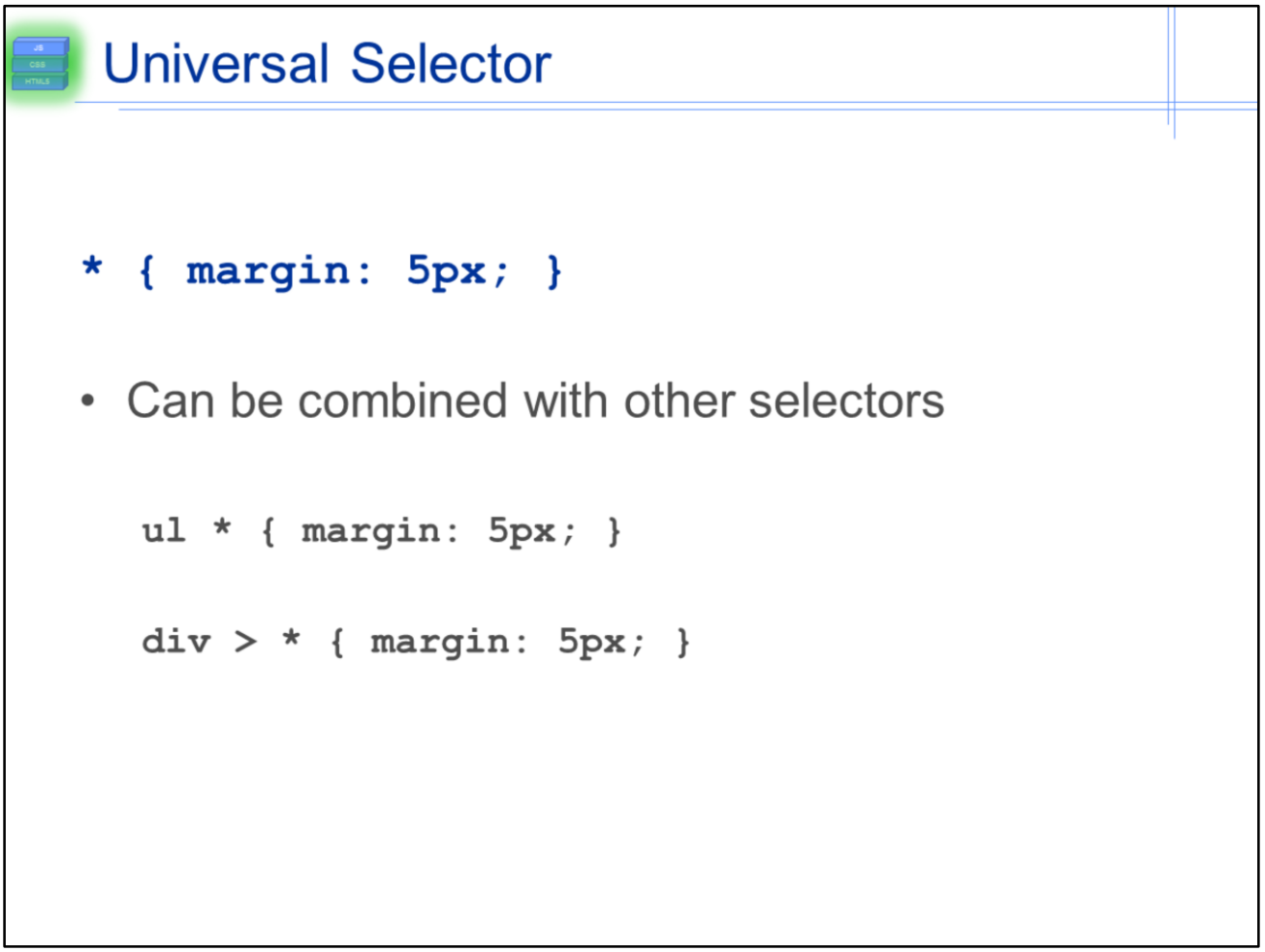
Order of values in the shorthand version can make a difference in some older browsers



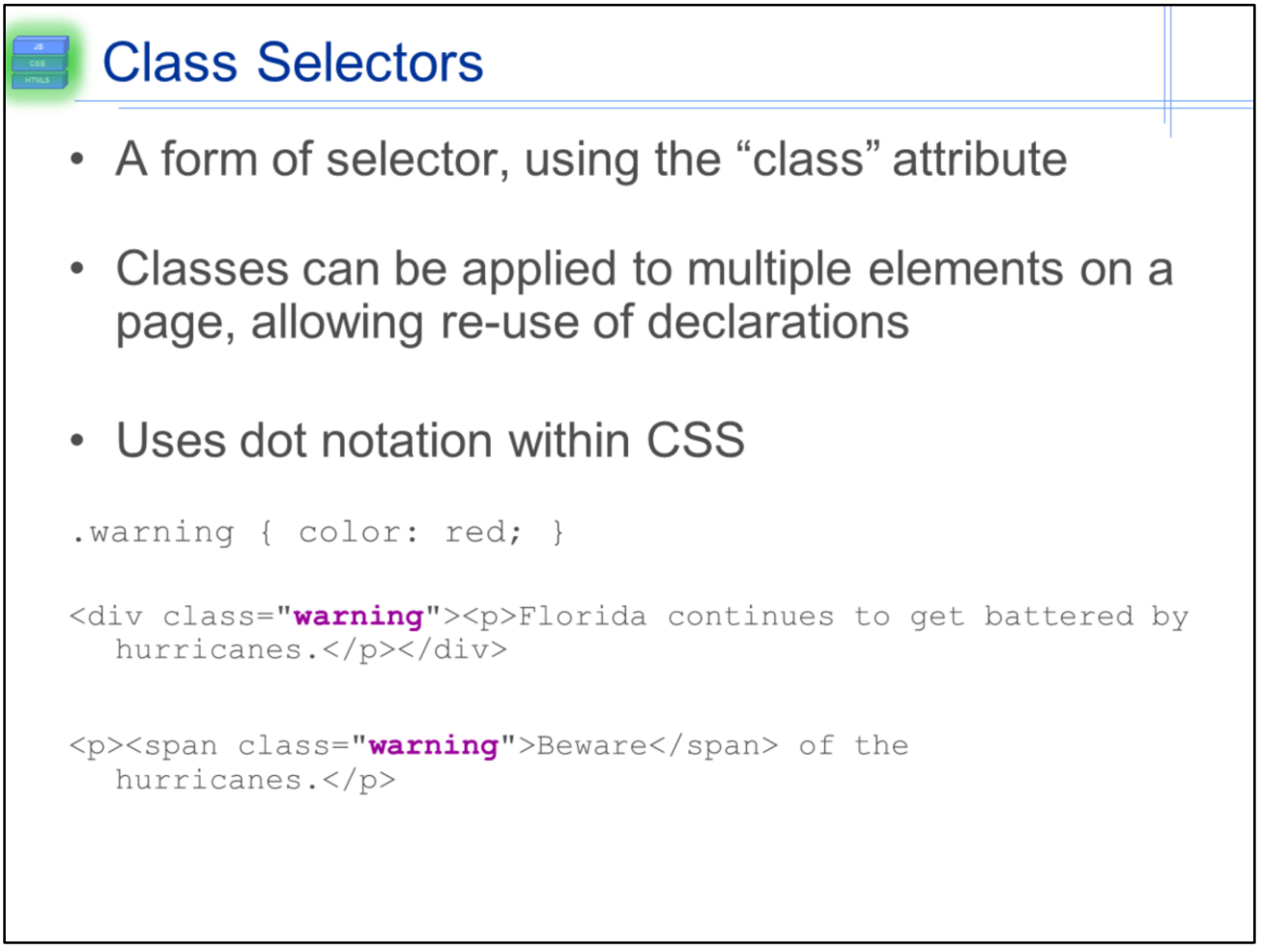
Targets *all* instances of a particular element



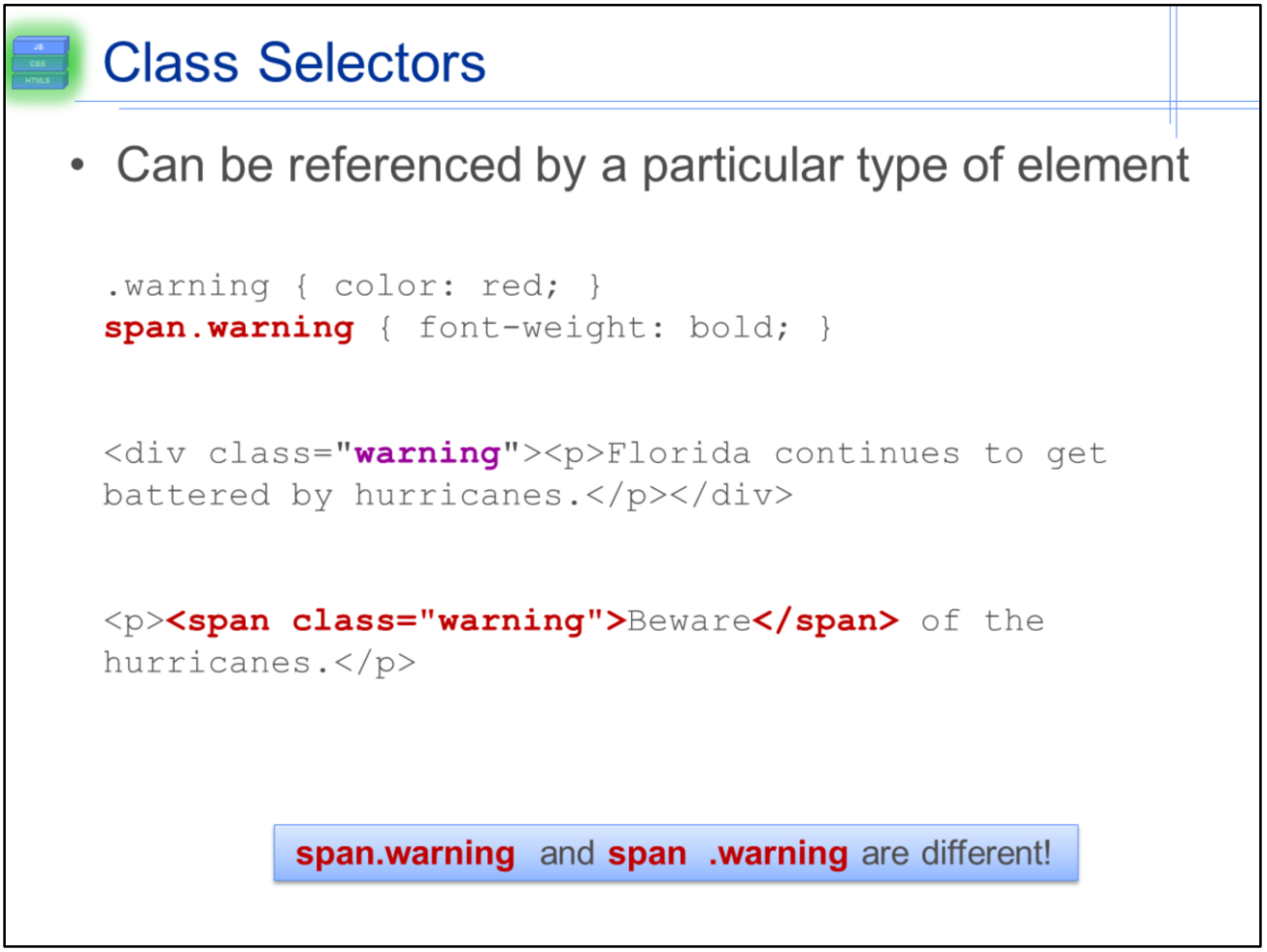
Targets any element that is a descendent

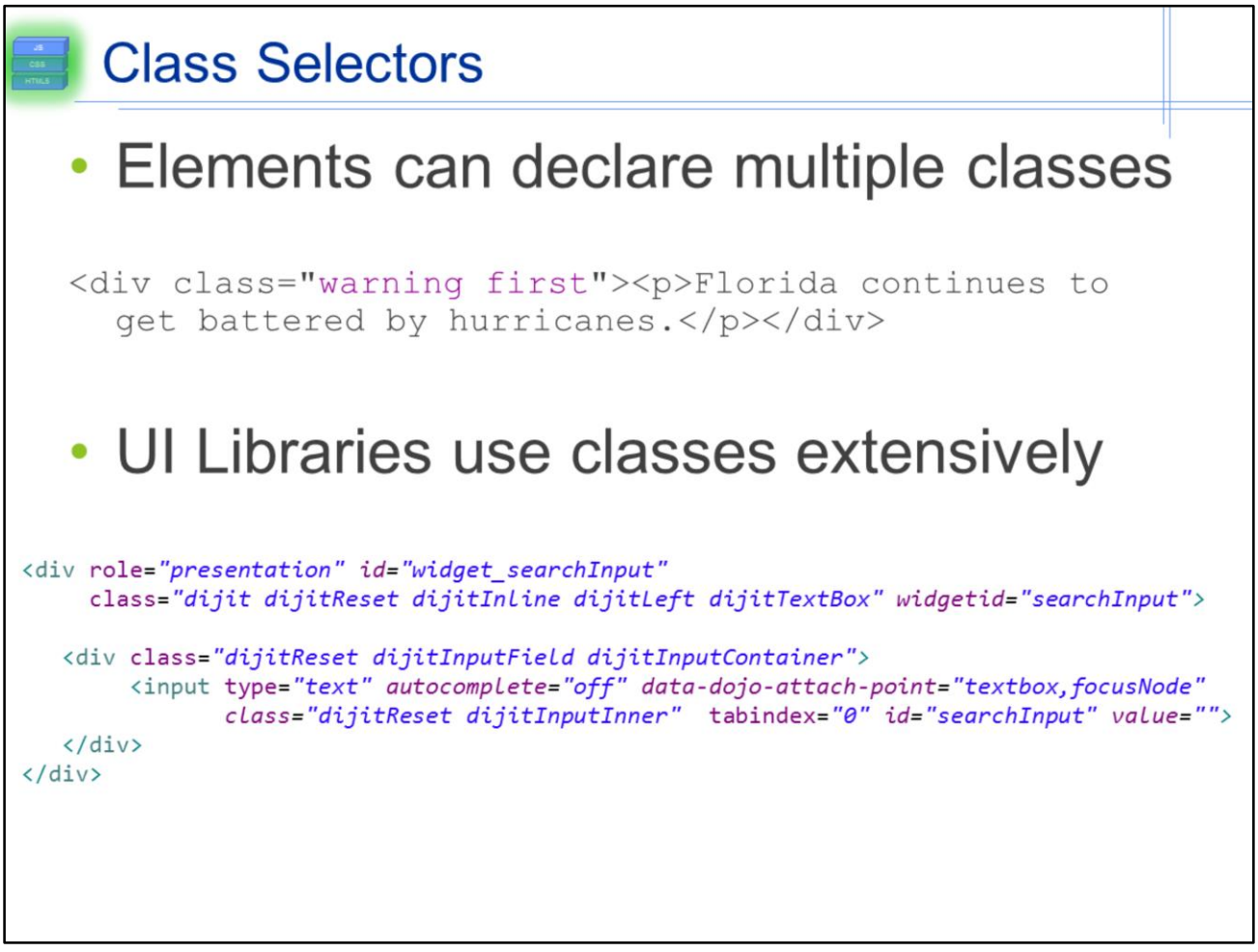


Targets *all* elements



Notice we’re using the same class “warning” on different elements





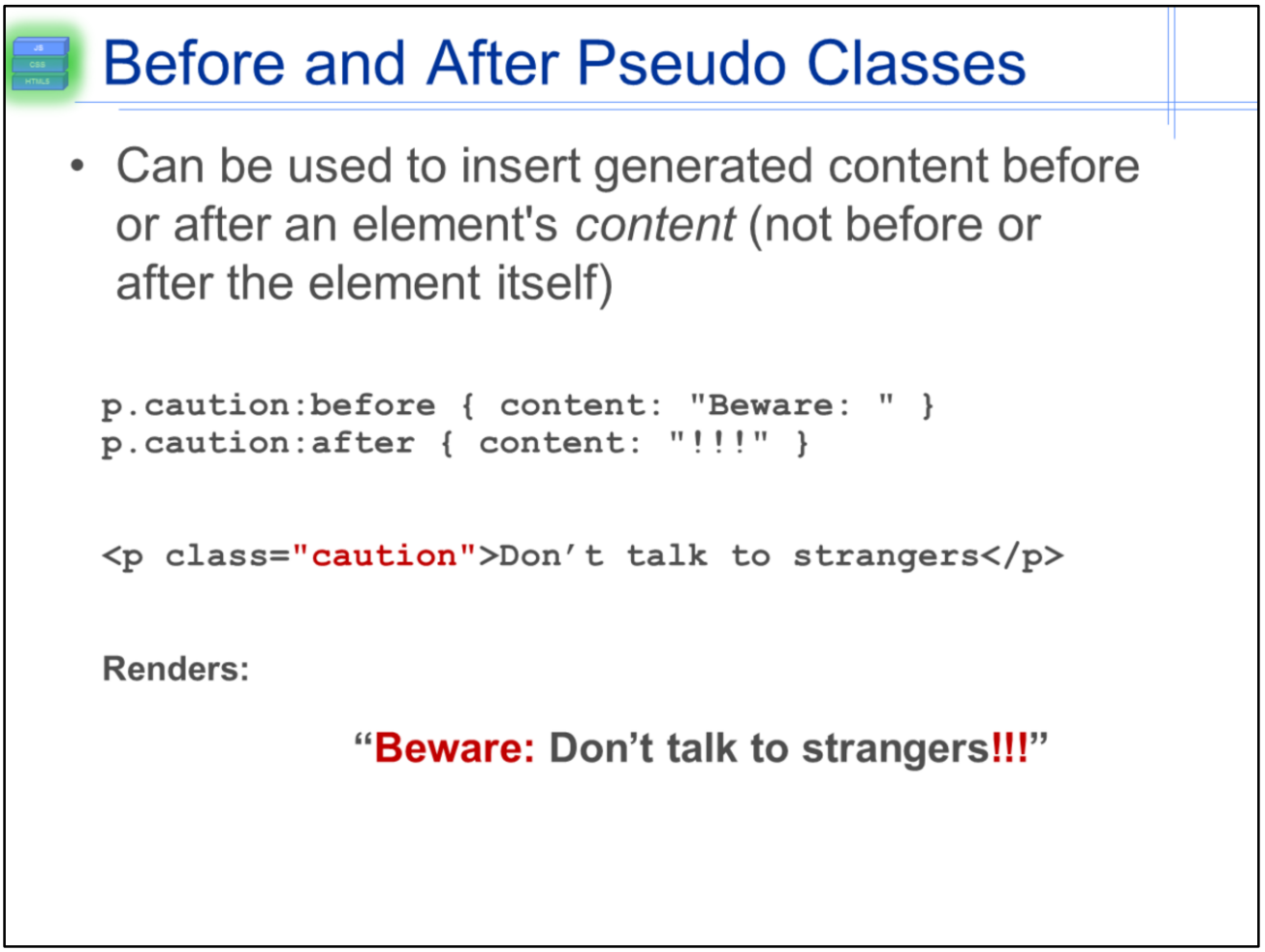
This example shows a Dojo implementation of a TextBox and Button.

ID Selectors

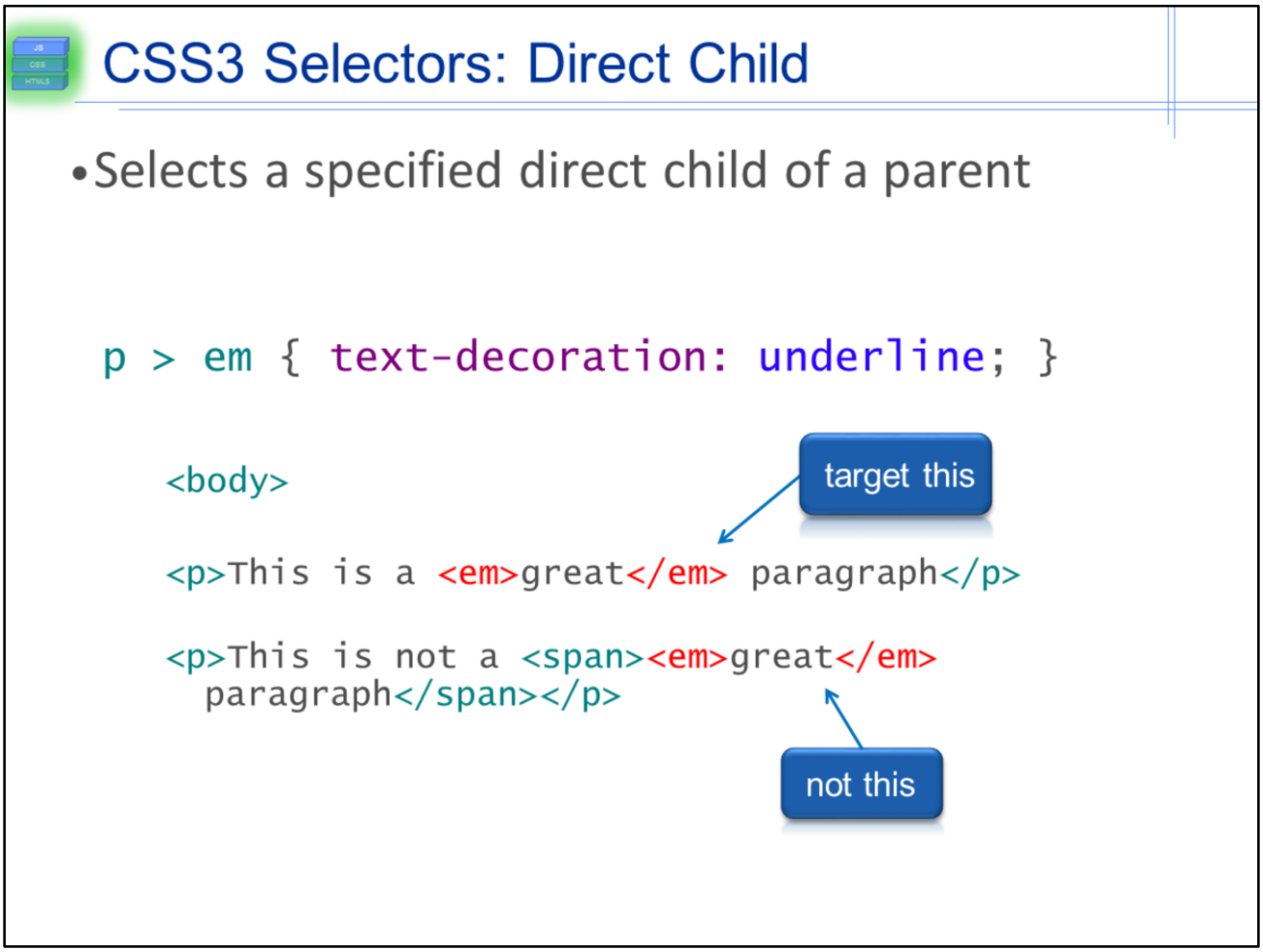
- Similar to class selectors, but each ID attribute must be unique on a page
- IDs are case sensitive
- Use hash (#) notation

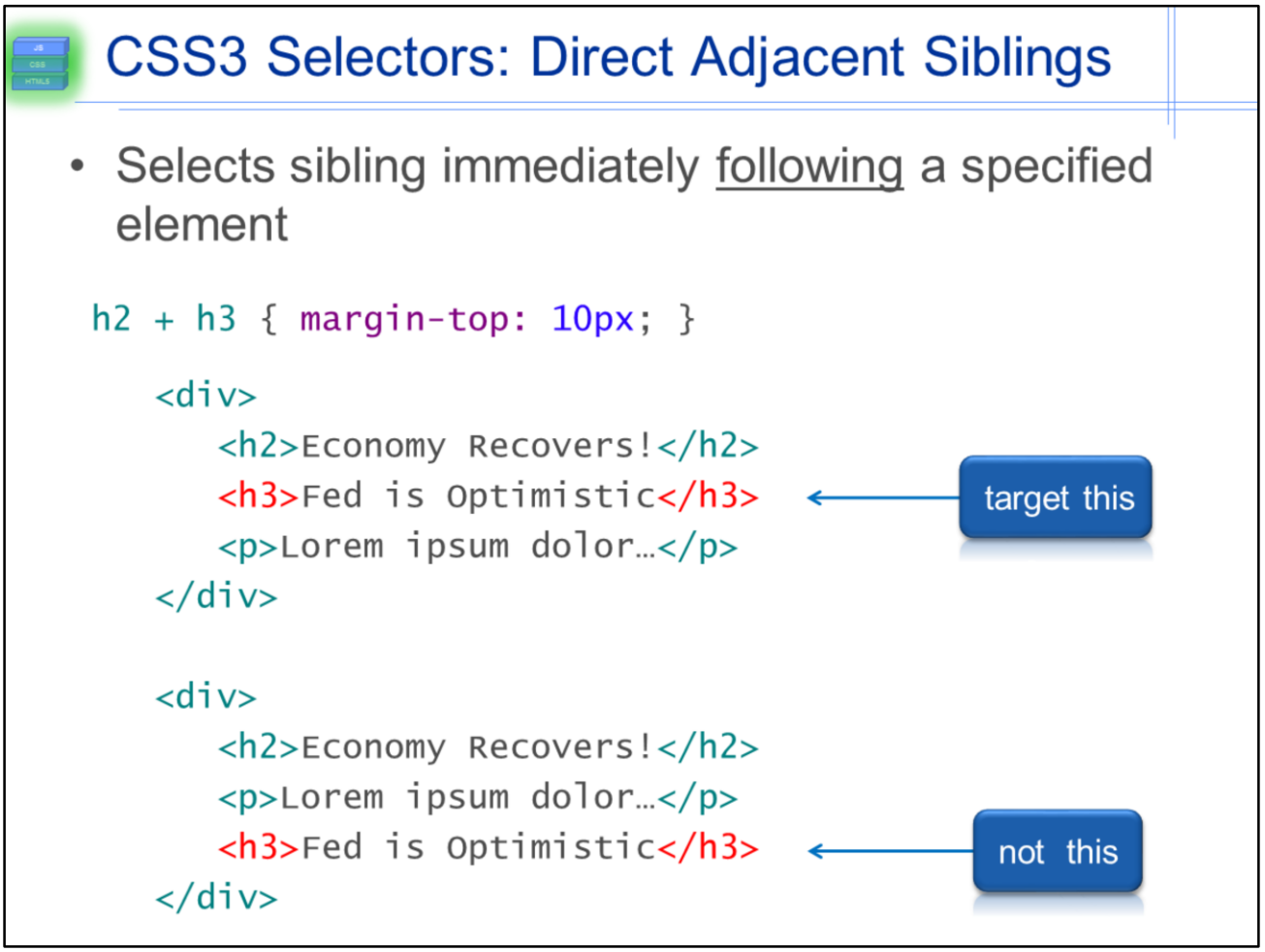
```
#mainNav { margin: 5px; }
#mainNav li { float: left; }
```

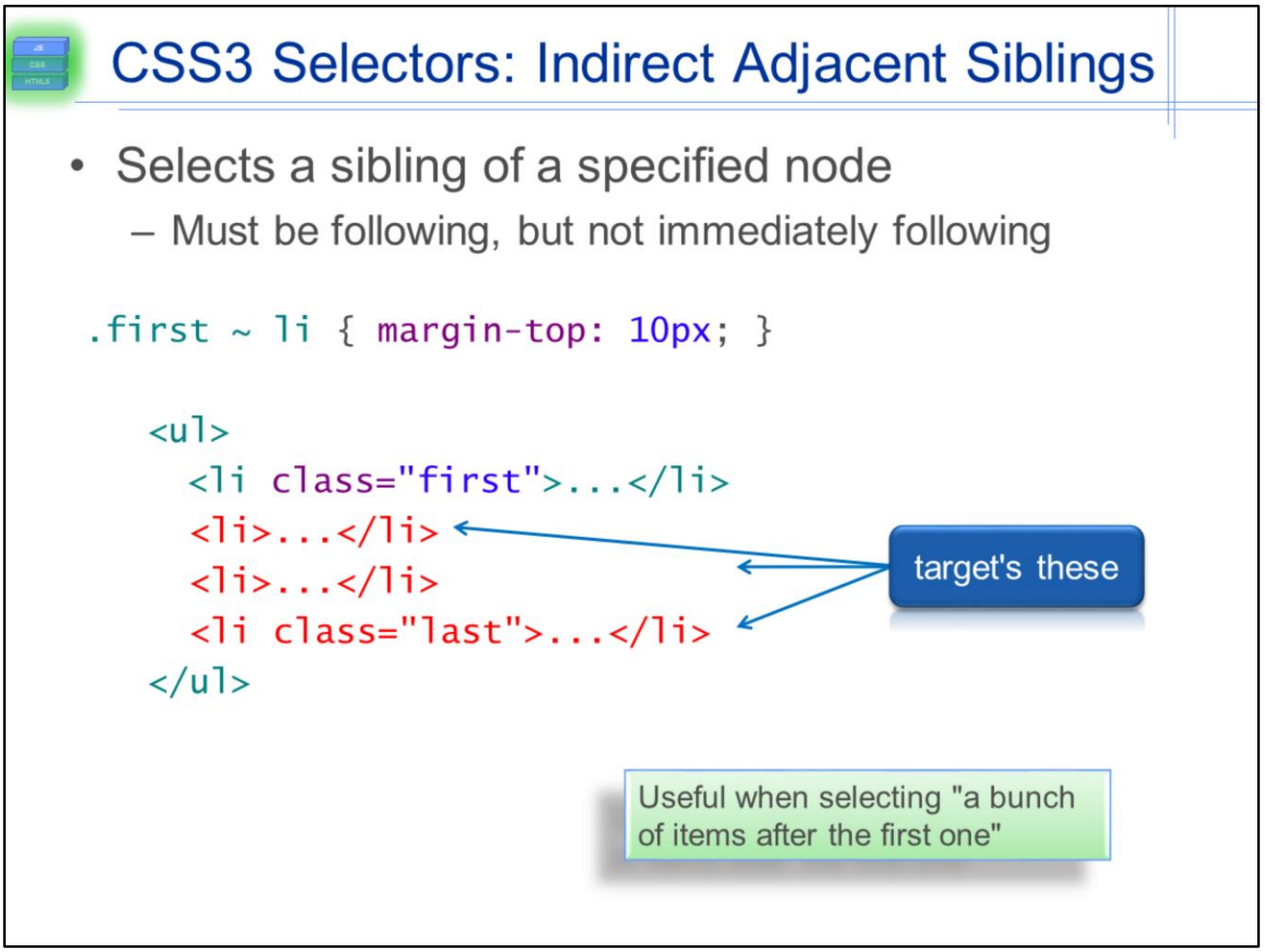
```
<ul id="mainNav">
  <li><a href="home.html">Home</a></li>
  <li><a href="about.html">About</a></li>
</ul>
```



You should not use :before and :after for general use because it mixes the layers in a layered architecture, but it can be very useful as a hack (presented in the clearfix discussion at the end of the next chapter).







CSS3 Selectors: Attribute Selectors

- Selects elements based on presence or values of attributes

```
img[title] {border: 2px solid #000;}
```

Styles any <img> with a title attribute.

```
img[title="Figure"] {border: 2px solid #000;}
```

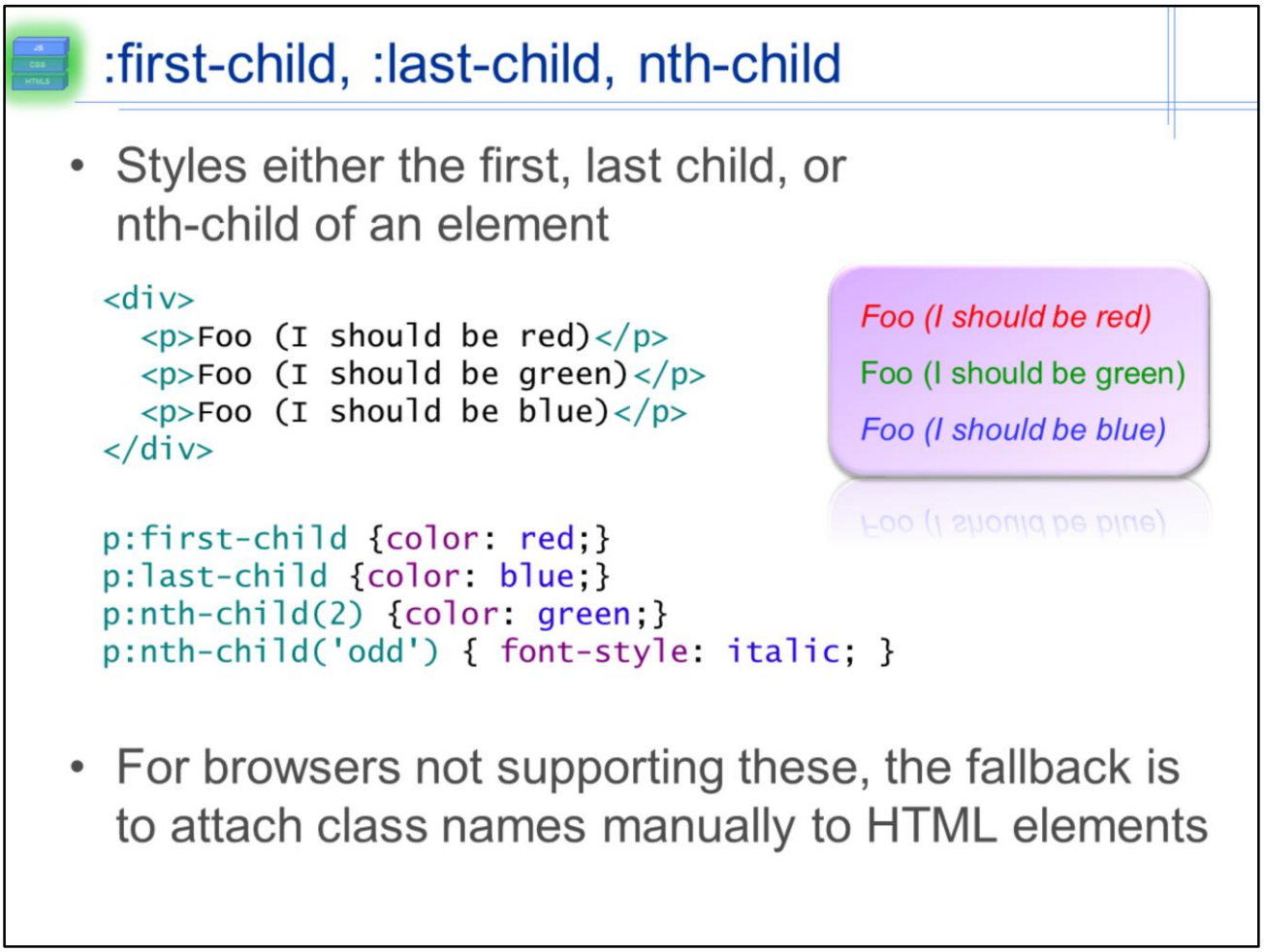
Styles any <img> with the title attribute "Figure", symbol = signifies an exact match.

```
img[title~="Figure"] {border: 2px solid #000;}
```

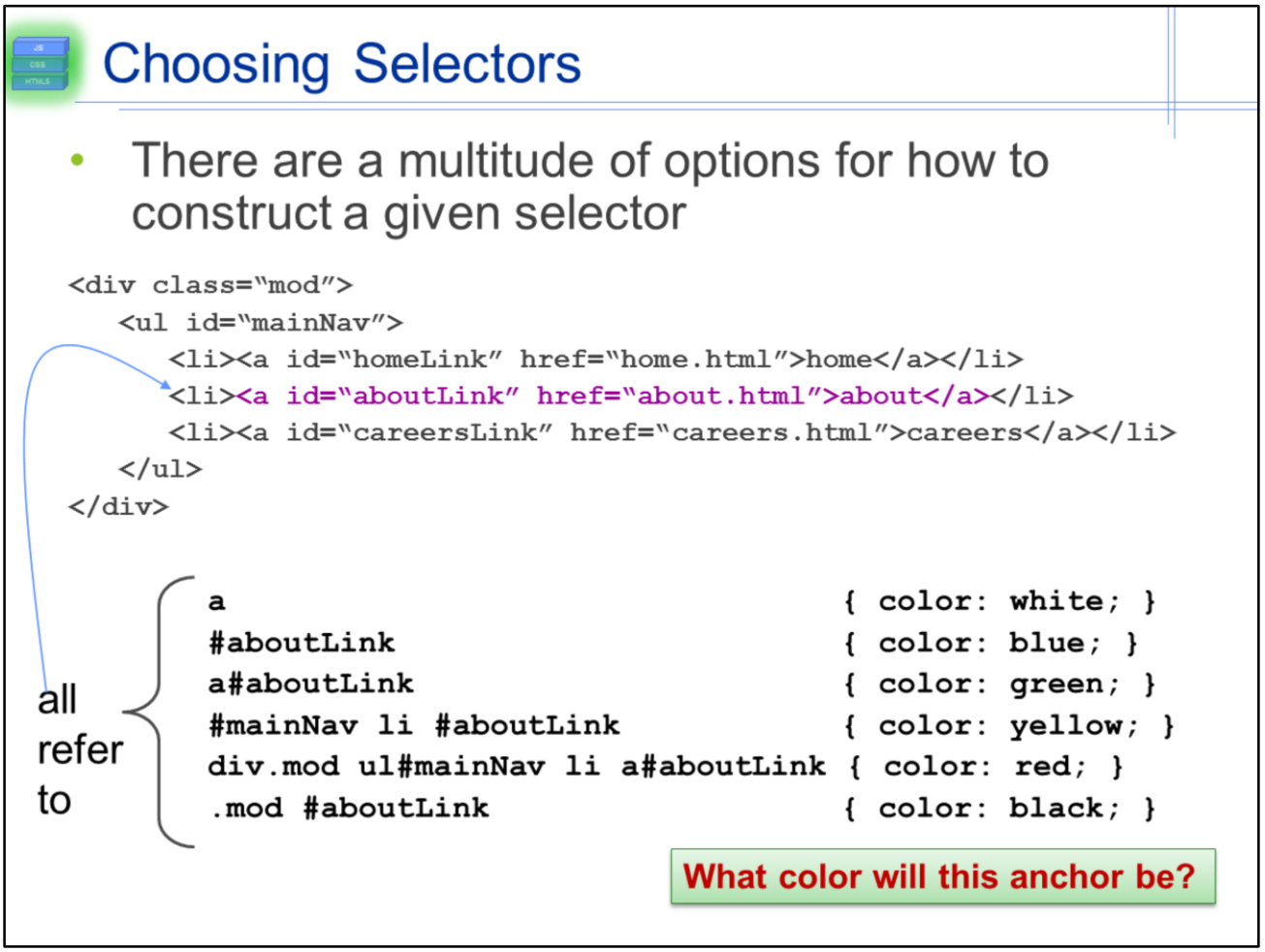
Note the additional "~". Styles any <img> with a title attribute that includes "Figure" in a space-separated list of words.

```
*[lang|="en"] {color: #CCC;}
```

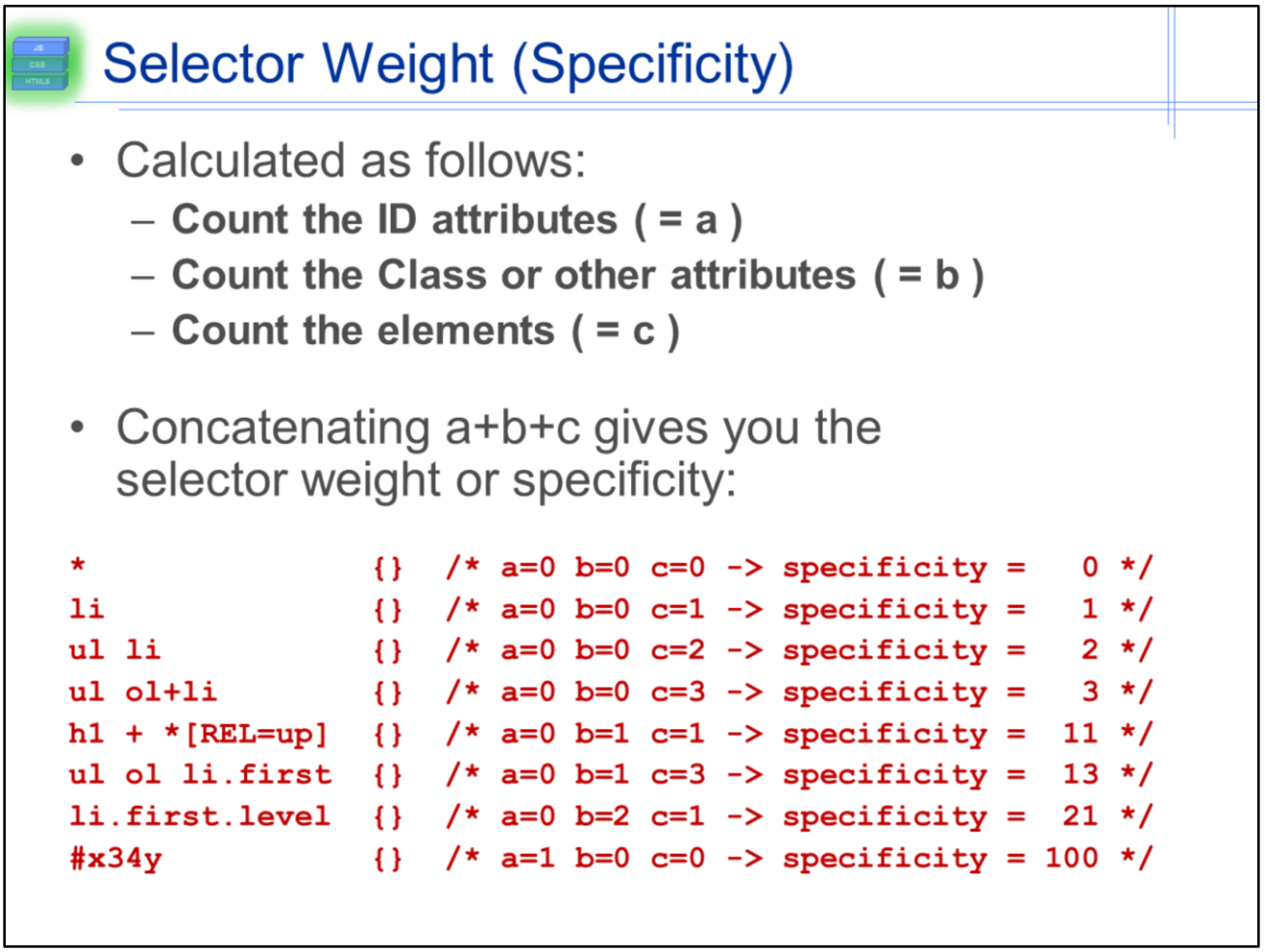
Styles any element with a lang attribute that begins with "en" in a hyphen-separated list.



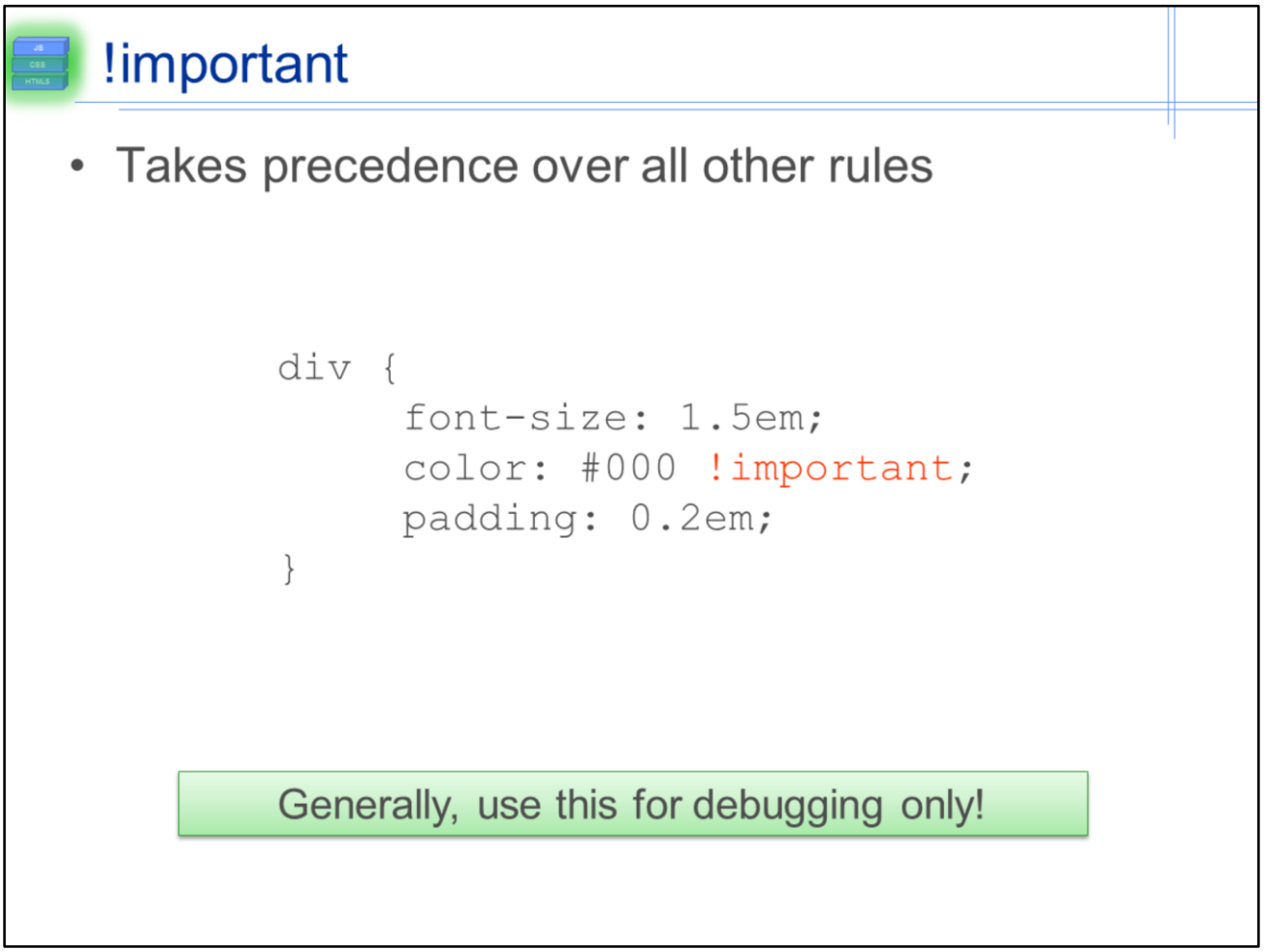
Some of the selectors presented here are actually from CSS2.1.



The answer is red. The reason is explained on the next slide.

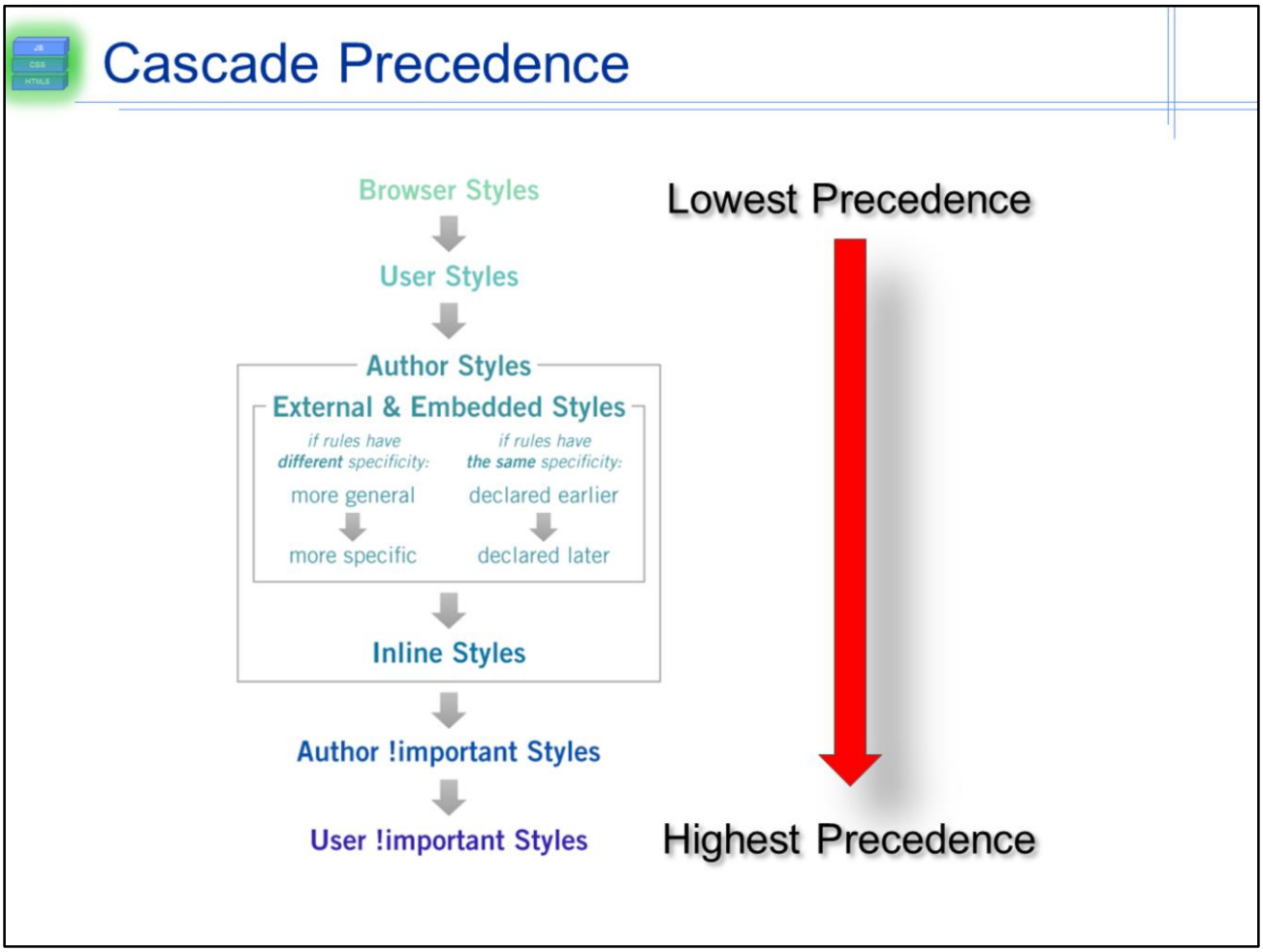


Specificity is a misleading term: longer selectors can appear to be more specific, but can actually have lower weight (last, and next to last examples above)

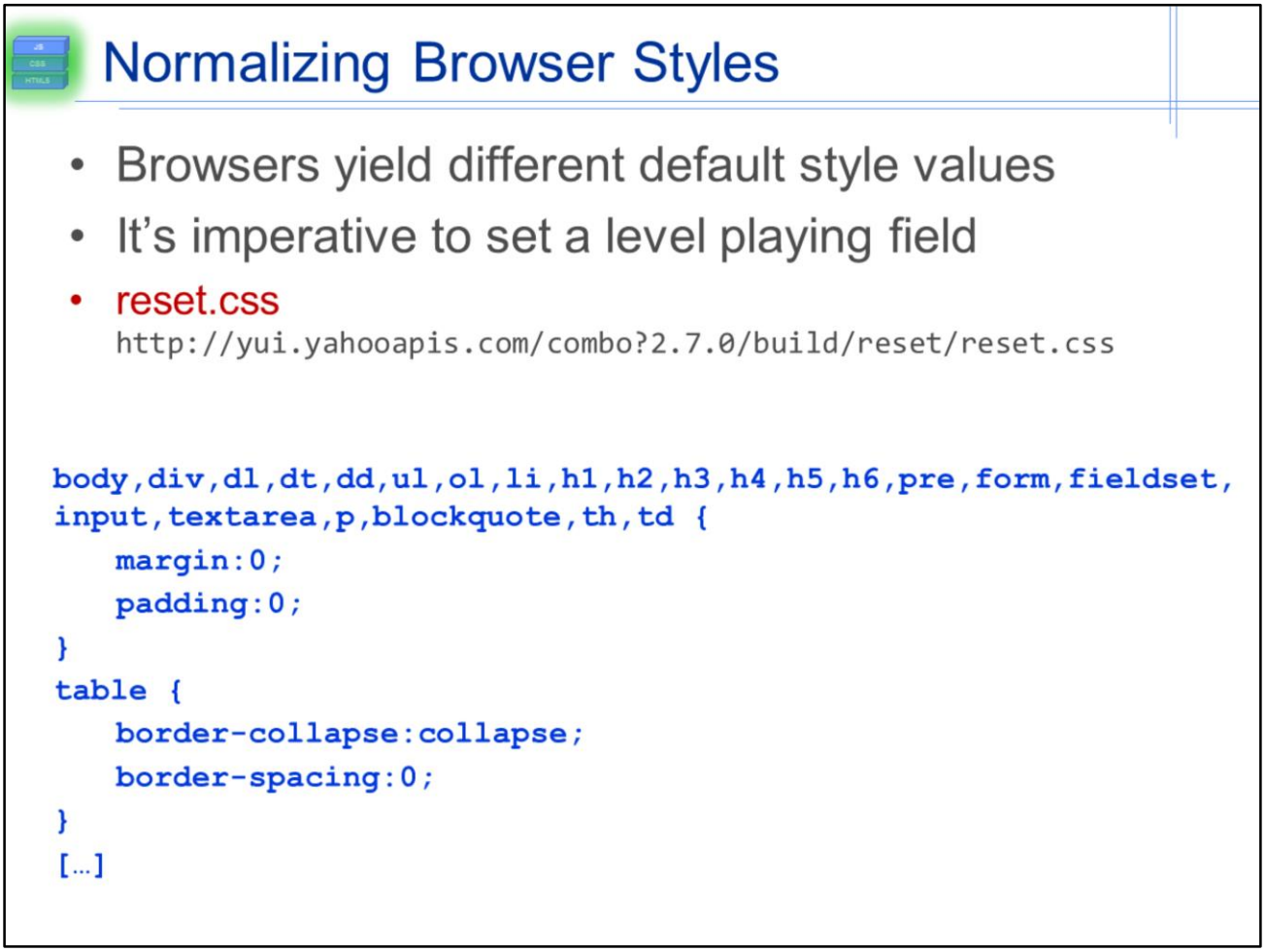


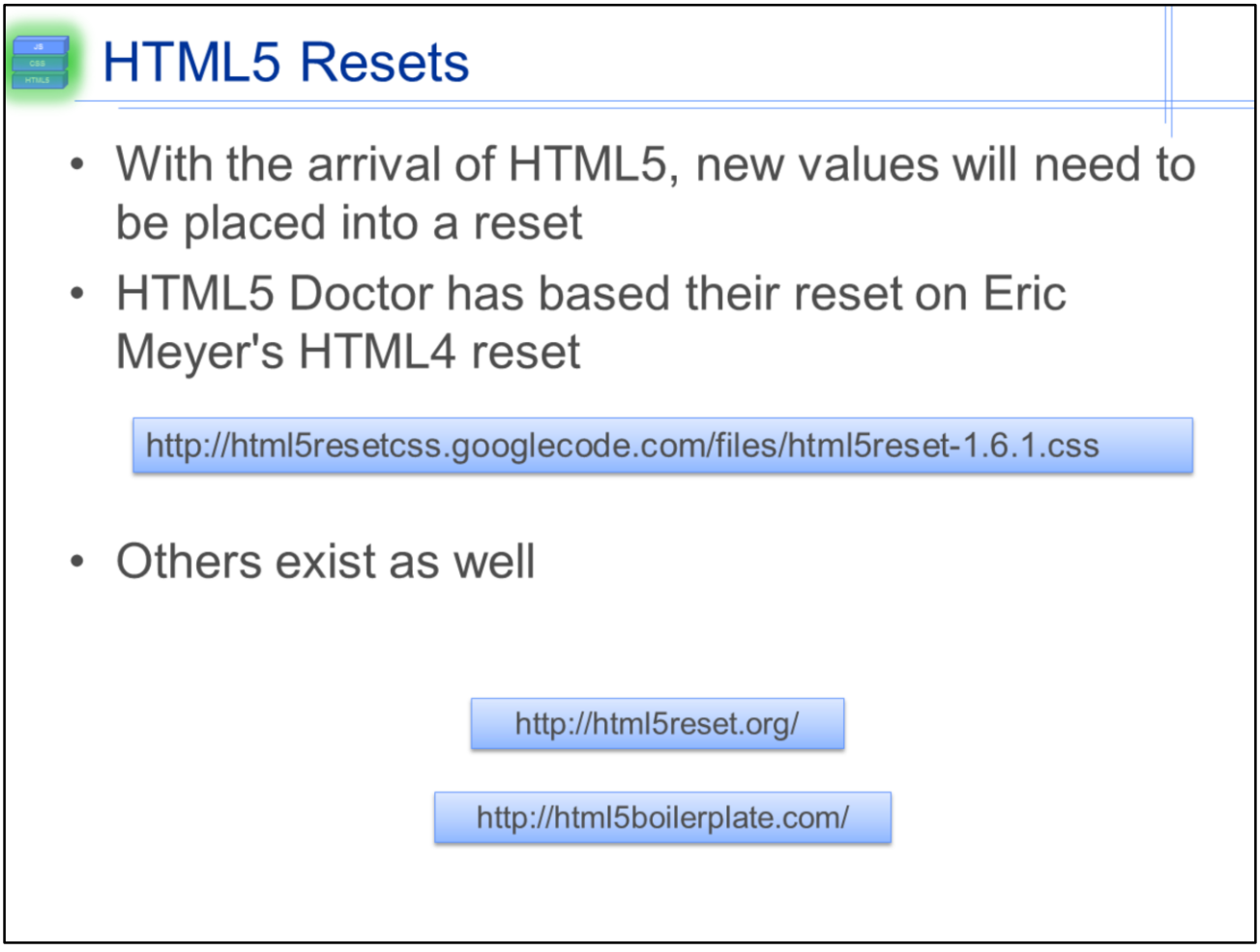
The use of !important is a bad practice to begin taking advantage of. Yes, the argument could be made that when using certain frameworks, in order for your selectors to "beat" their selectors, you might use !important. However, once you use it, it has to be used again if you ever want to beat *that* selector.

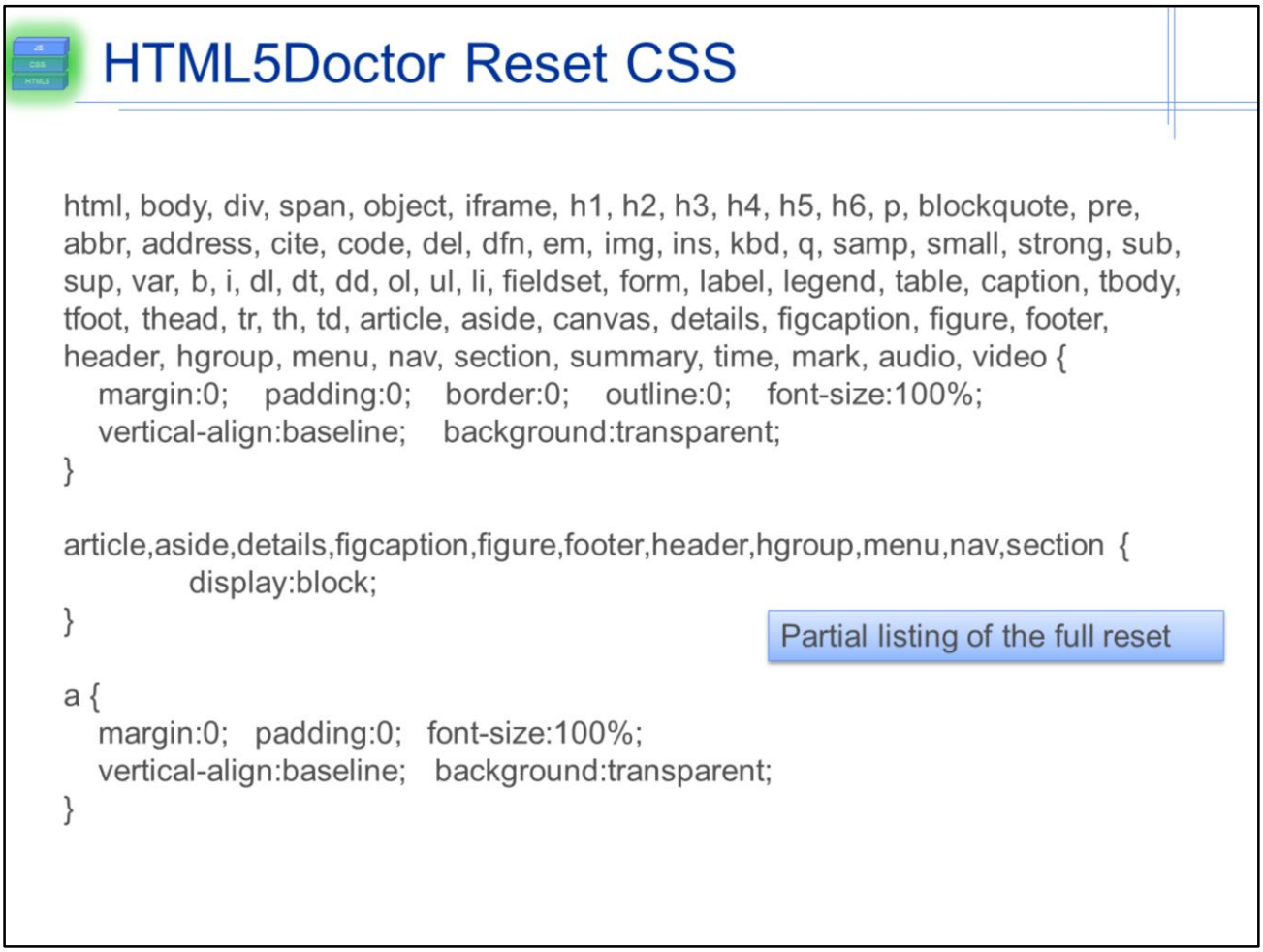
It leads to a slippery slope that finds !important being used all over the place before long.



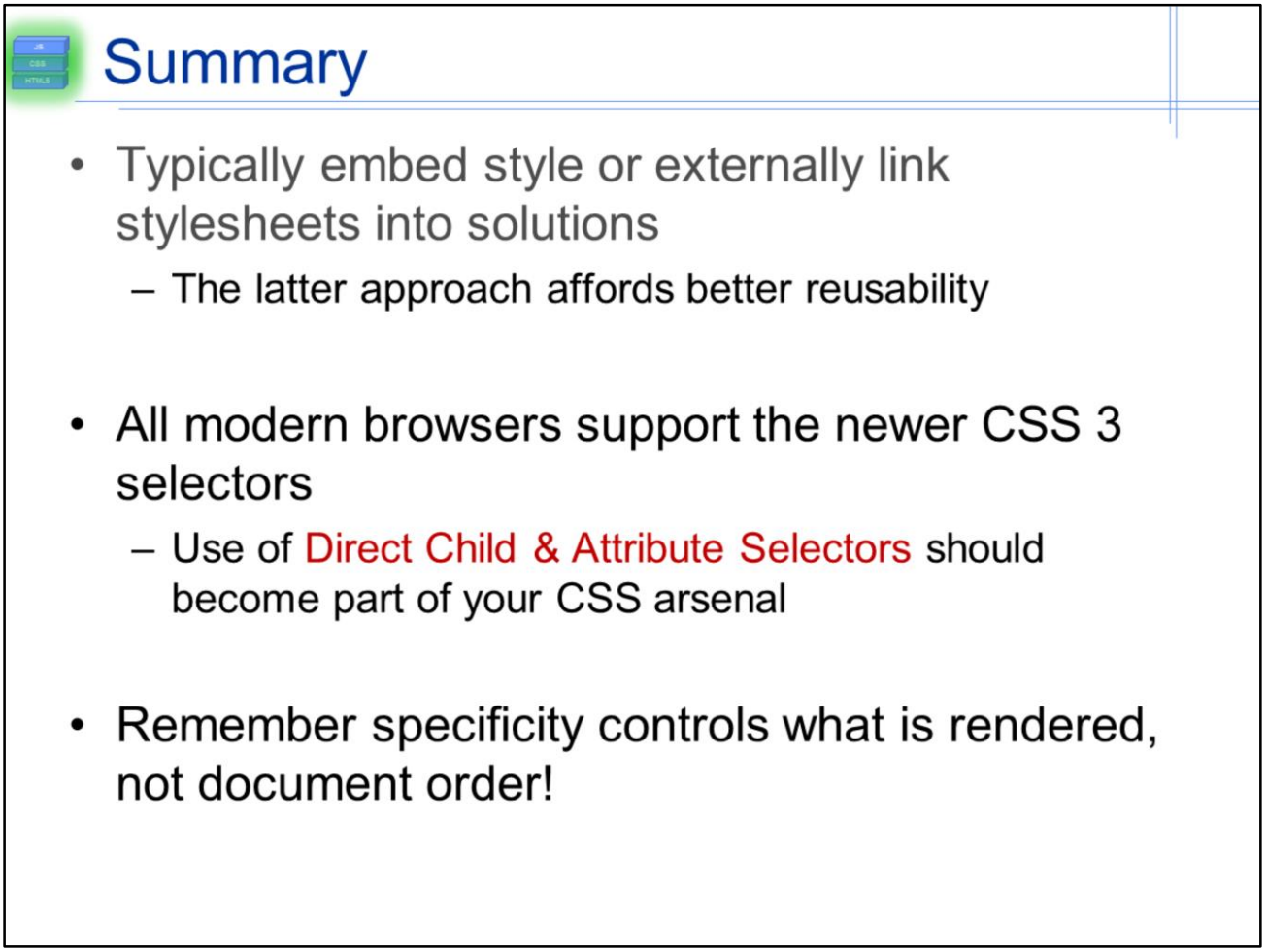
http://www.communitymx.com/content/article.cfm?page=2&cid=2795D

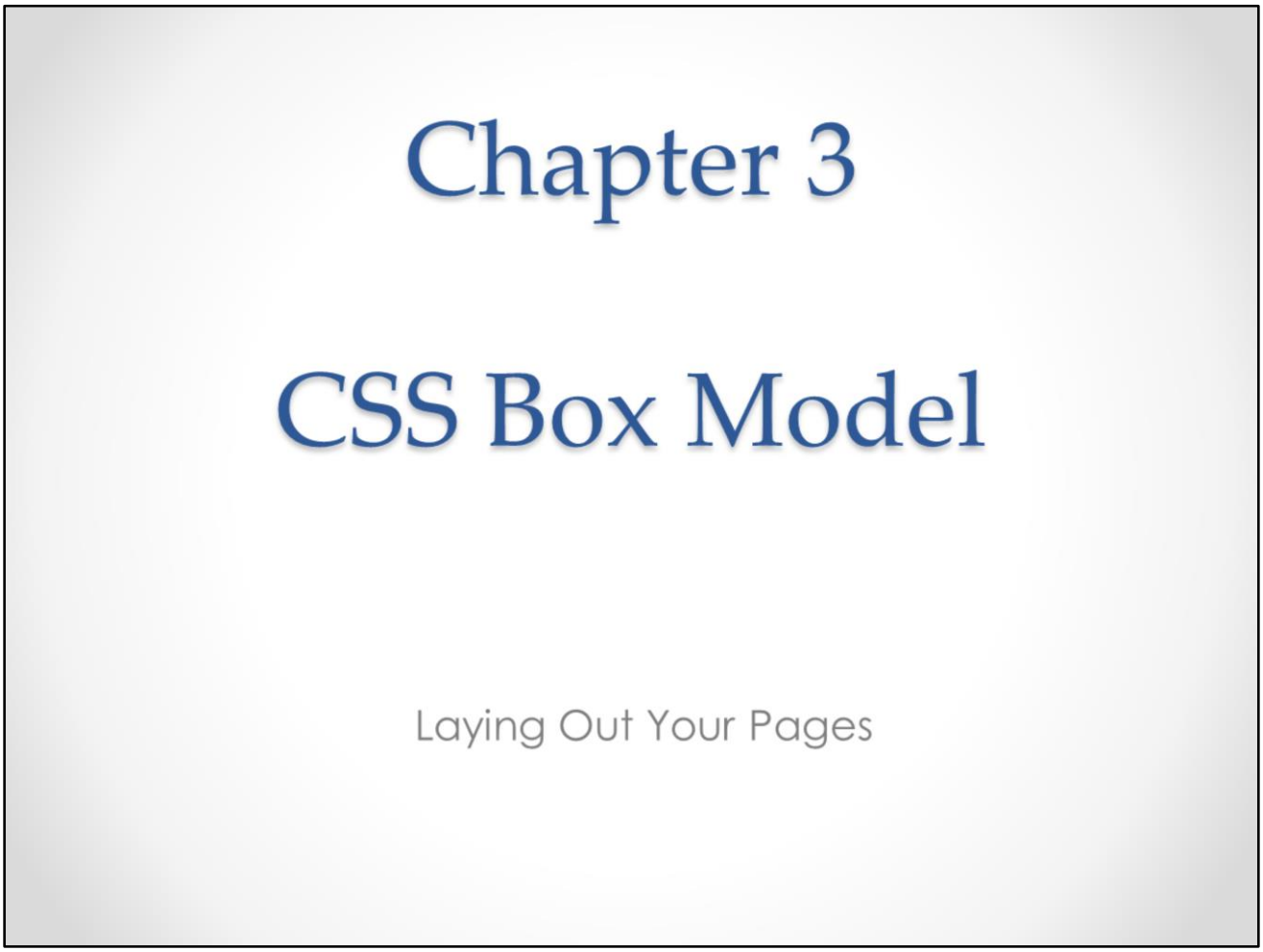


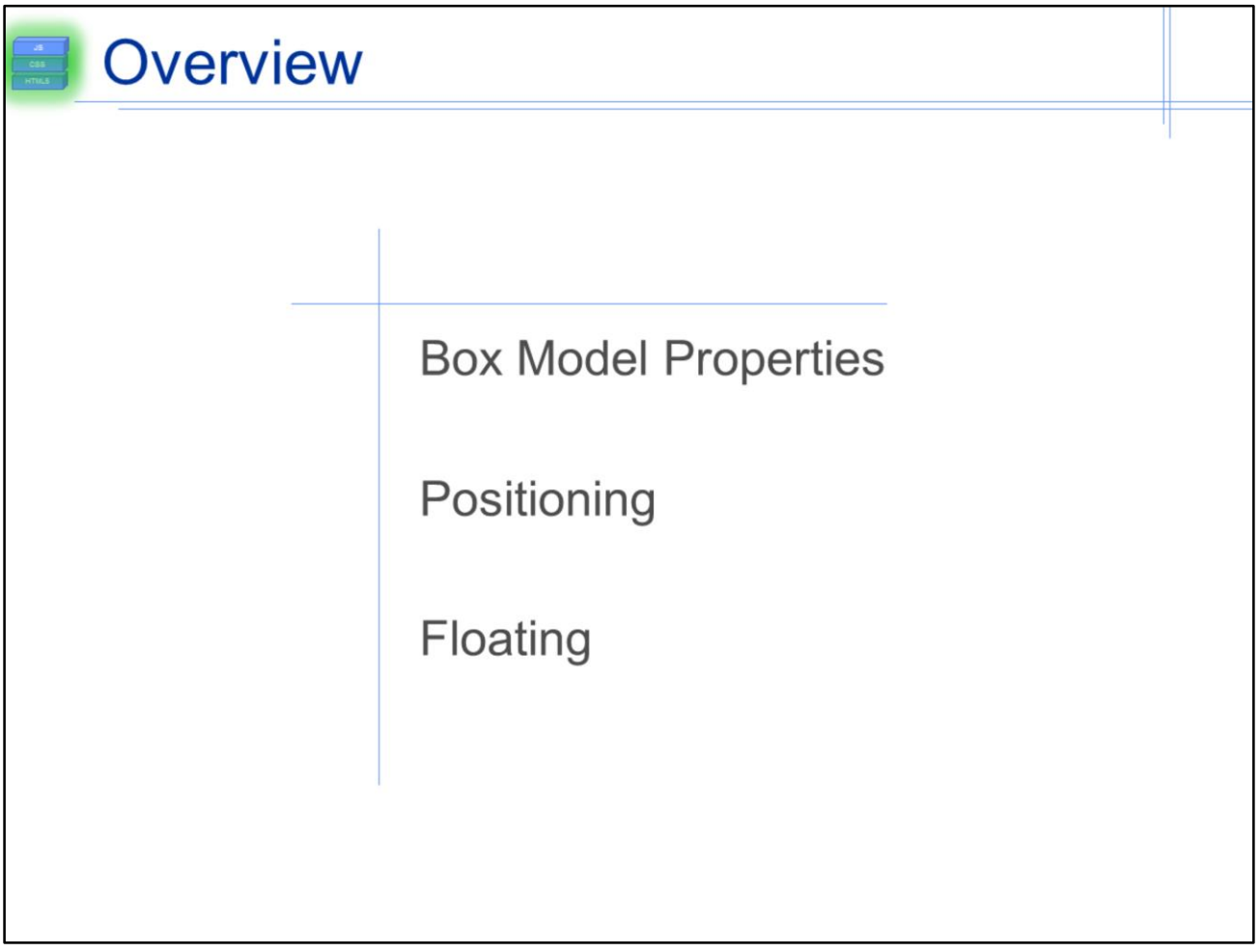


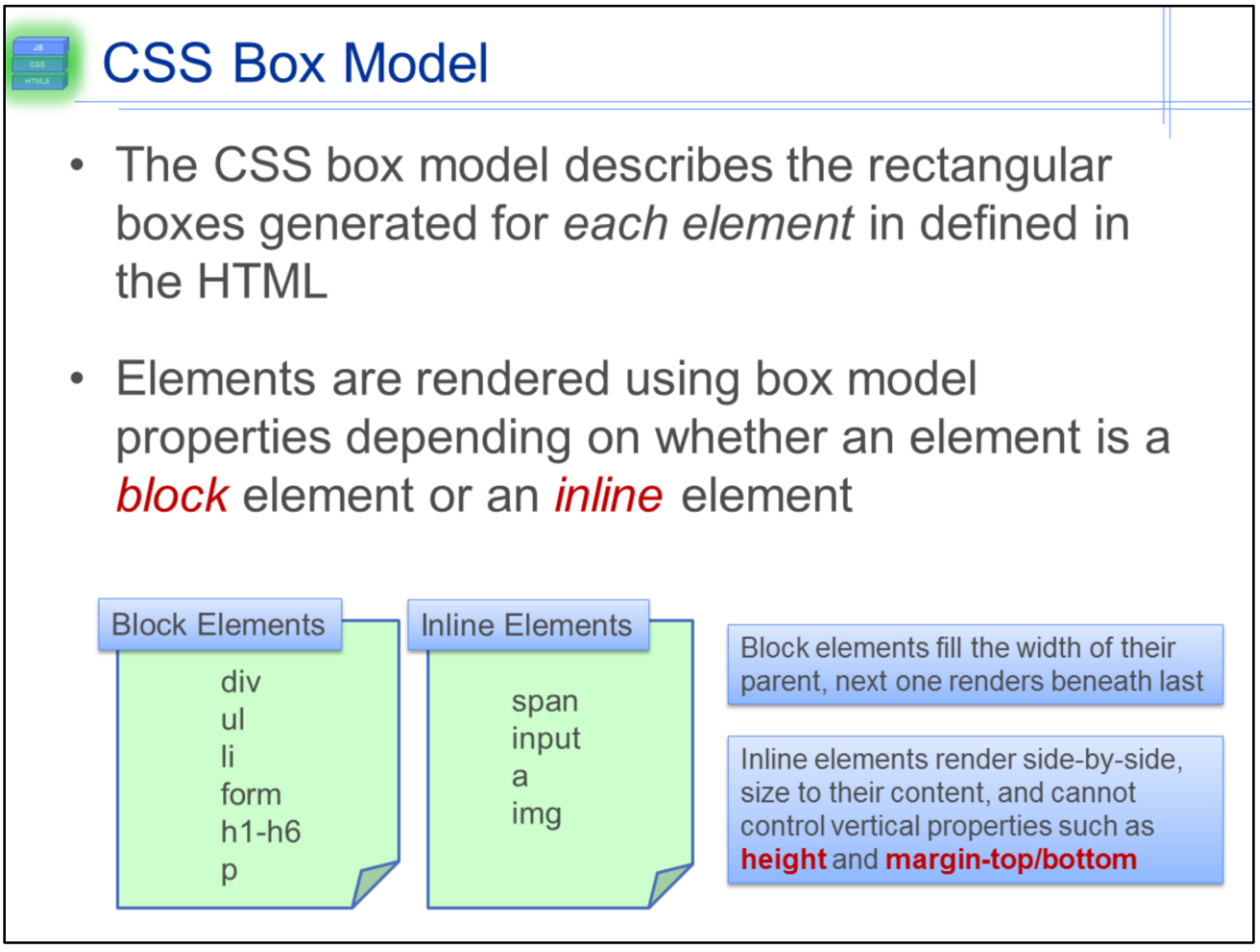


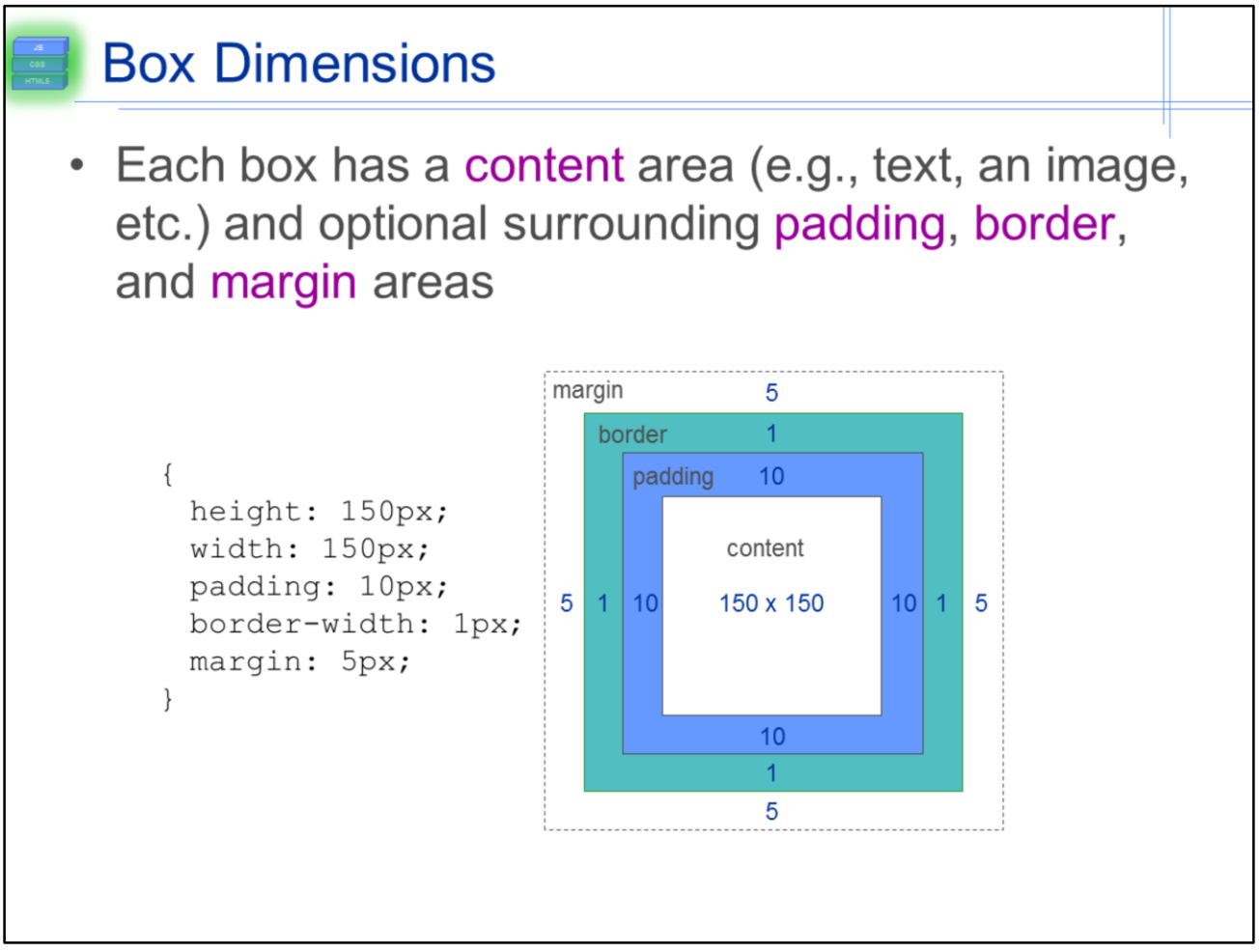
html5doctor.com Reset Stylesheet



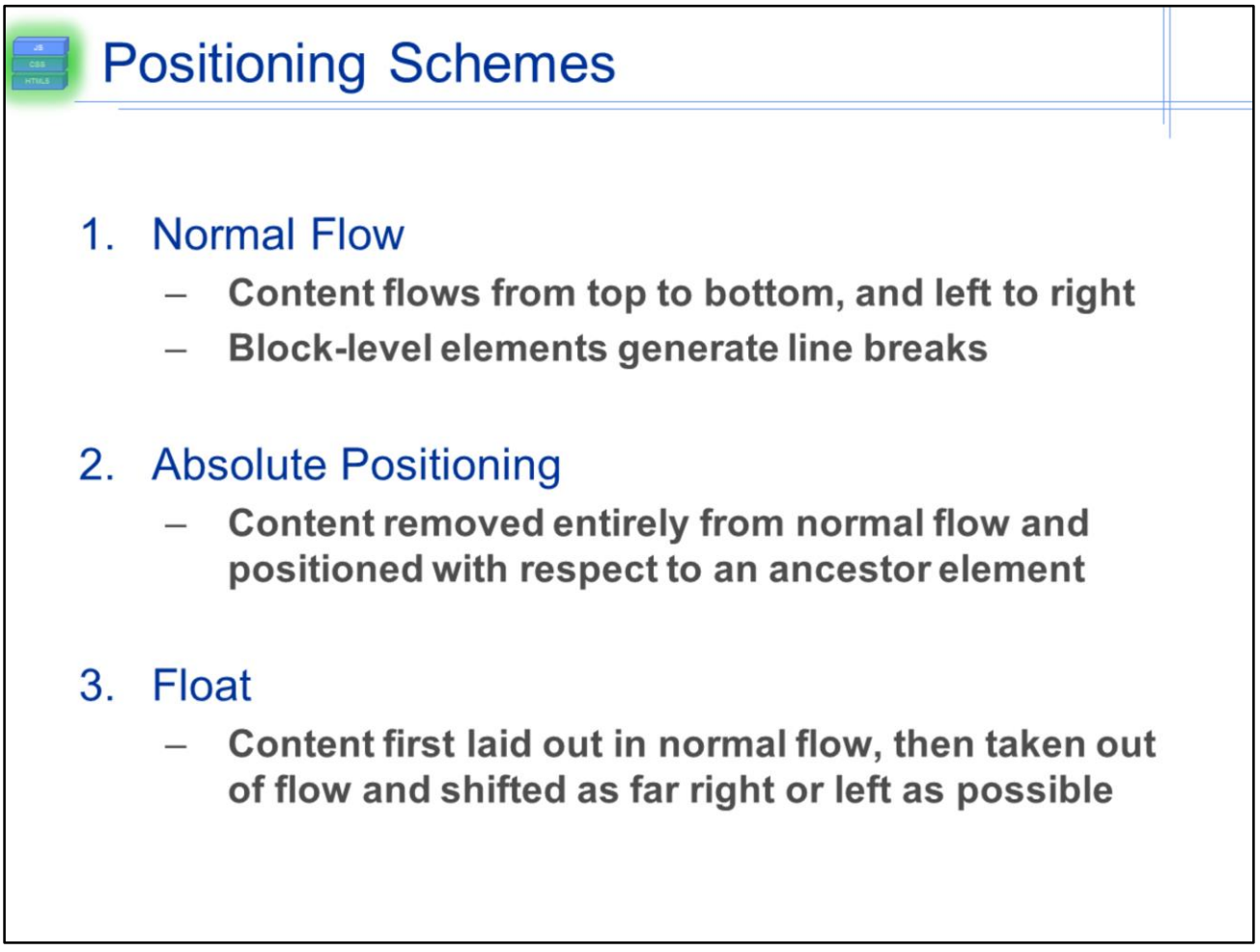








Margin is transparent: background colors or images will not show in that area.

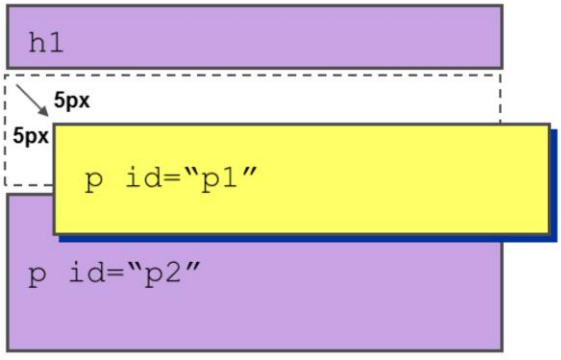


We use the “position” and “float” properties to assign a scheme for each element

Relative Positioning

- Part of the normal flow scheme
- Should really be called “offset positioning”
- Content is laid out according to normal flow, and then offset from its original position

```
<h1></h1>
<p id="p1"></p>
<p id="p2"></p>
[…]

p#p1 {
   position: relative;
   top: 5px;
   left: 5px;
}
```



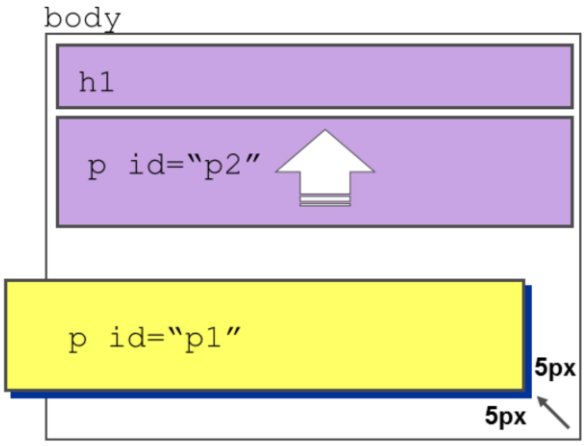
The space that the element would have occupied in normal flow is maintained. Element now layers over neighboring elements.

Properties can be top|right|bottom|left
Values can be positive or negative

Absolute Positioning

- Removed from normal flow

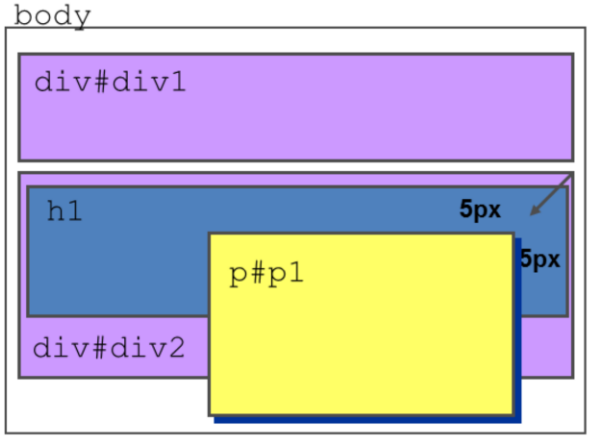
```
<body>
   <h1></h1>
   <p id="p1"></p>
   <p id="p2"></p>
</body>
[…]

p#p1 {
   position: absolute;
   bottom: 5px;
   right: 5px;
}
```



Absolute Positioning

- Positioned with respect to the body *or* the next highest *positioned* ancestor
  - Travel up the document tree until encountering an element with `position: relative|absolute`

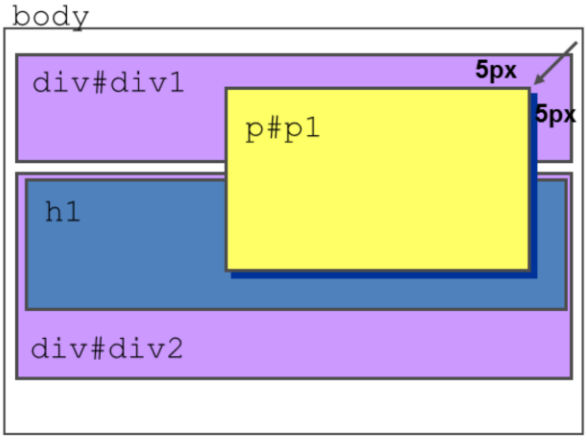
```
<div id="div1"></div>
<div id="div2">
   <h1></h1>
   <p id="p1"></p>
</div>
[…]

p#p1 {
   position: absolute;
   top: 5px;
   right: 5px;
   width: 50%;
}
div#div2 {
   position: relative;
}
```



Absolute Positioning

- Positioned with respect to the body *or* the next highest *positioned* ancestor
  - Travel up the document tree until encountering an element with `position: relative|absolute`

```
<div id="div1"></div>
<div id="div2">
   <h1></h1>
   <p id="p1"></p>
</div>
[…]

p#p1 {
   position: absolute;
   top: 5px;
   right: 5px;
   width: 50%;
}
```

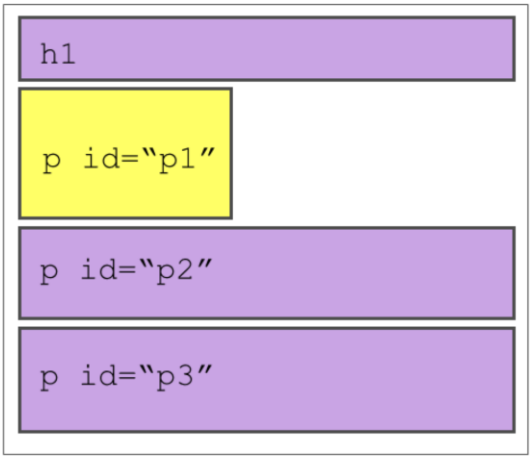


Showcase what happens when positioning is removed from ancestor div (now in relation to the body)

Floated Elements

- First, content is laid out according to normal flow

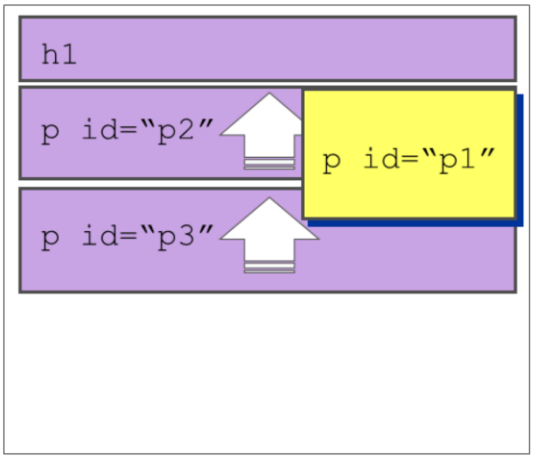
```
<h1></h1>
<p id="p1"></p>
<p id="p2"></p>
<p id="p3"></p>
[…]

p#p1 {
   width: 40%;
}
```



Rules of Floating *(1 of 5)*

- Floats shift up then left or right as far as possible
  - **Subsequent block elements move up behind floated element**

```
<h1></h1>
<p id="p1"></p>
<p id="p2"></p>
<p id="p3"></p>
[…]

p#p1 {
   width: 40%;
   float: right;
}
```

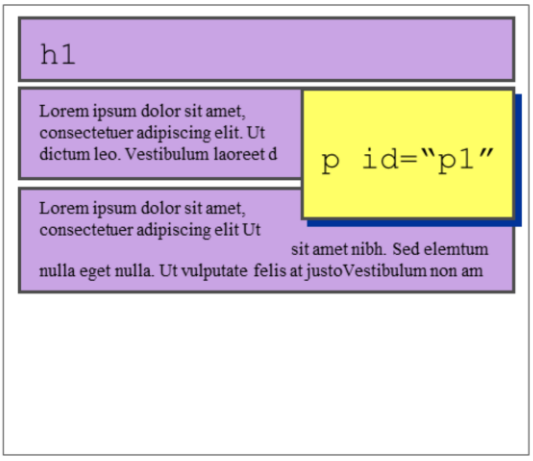


Floats are, in relation to block level elements, removed from the flow. They will lift up to the first sibling element that precedes it that is not floated.

Rules of Floating *(2 of 5)*

- Inline content (i.e. text) flows *around* floated elements

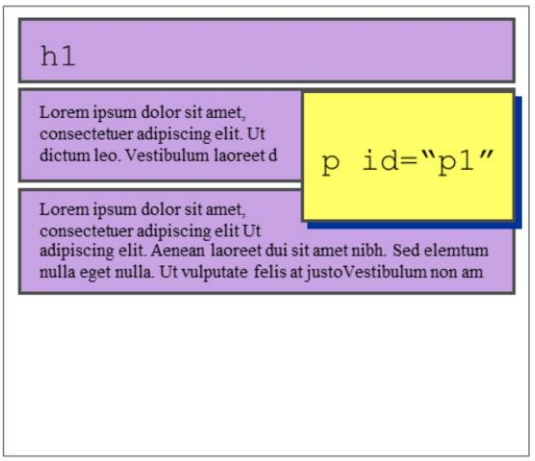
```
<h1></h1>
<p id="p1">Lorem...</p>
<p id="p2">Lorem...</p>
<p id="p3"></p>
[...]

p#p1 {
   width: 40%;
   float: right;
}
```



Rules of Floating *(3 of 5)*

- Floats are block-level elements
  - Will override value of "display" property
  - However, must have an explicit width set, else shrink-to-fit

```
p#p1 {
   width: 40%;
   float: right;
}
```

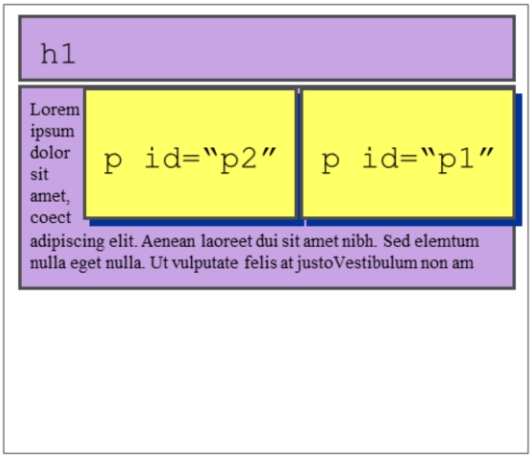


(So, it's not entirely removed from the normal flow) Key difference from absolutely positioned elements.

Rules of Floating *(4 of 5)*

- Floats are shifted left or right until their outer edge touches the containing block edge *or the outer edge of another float*

```
<h1></h1>
<p id="p1"></p>
<p id="p2"></p>
<p id="p3">Lorem…</p>
[…]

p#p1,
p#p2 {
   width: 40%;
   float: right;
}
```

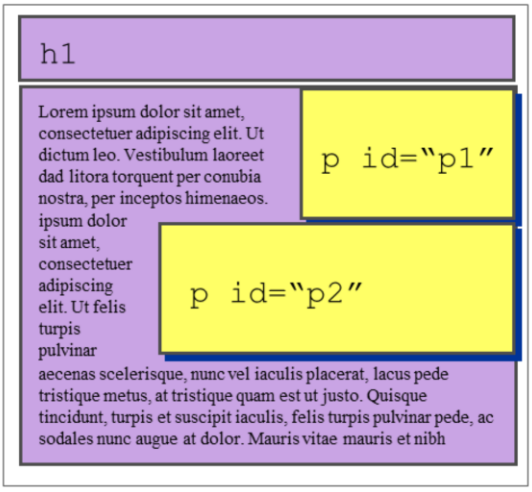


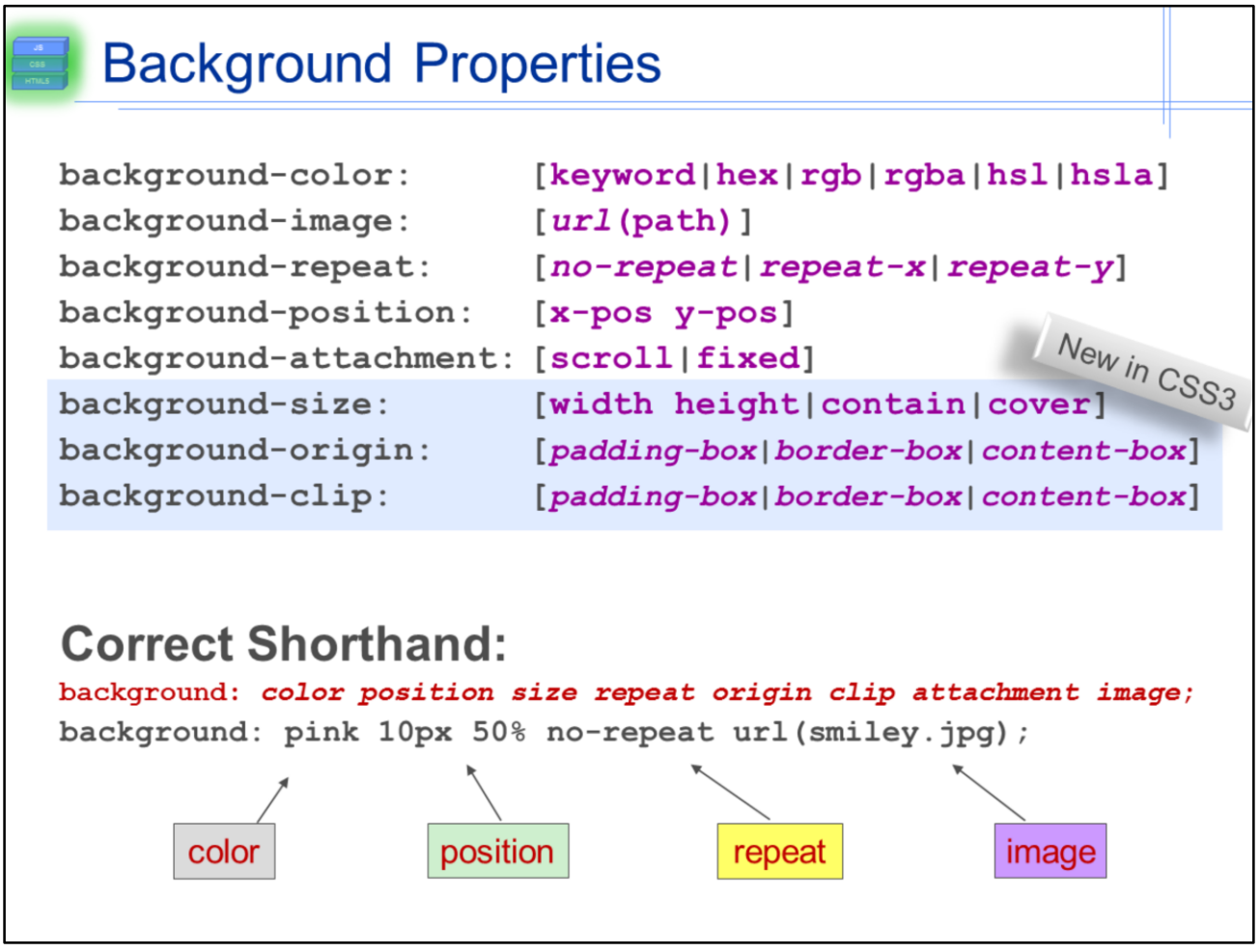
All floated objects interact with each other on the same “pane” or “context”. (vs. Absolutely Pos. elements will stack, each on their own layer.)

Rules of Floating *(5 of 5)*

- If there isn't enough horizontal room on the current line for the float, it is shifted downward, line by line, until a line has room for it.

```
<h1></h1>
<p id="p1"></p>
<p id="p2"></p>
<p id="p3">Lorem…</p>
[…]

p#p1 {
   width: 40%;
   float: right;
}
p#p2 {
   width: 70%;
   float: right;
}
```





examples/03_css/bgnd_prop.html
IE8 and earlier do not support multiple background images. Newer browsers do.

background-size can be one of several values including a width and height or
contain - image will fit within background
cover - image will completely cover the background, possibly clipping it

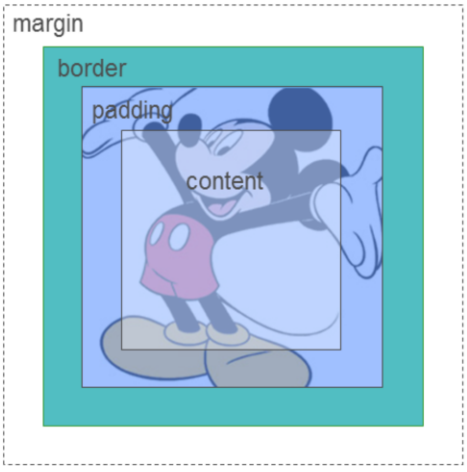
background-origin: specifies where background-position will be relative to

background-clip: specifies the area to be painted by the background-image

Using the *background* Property

- Any background properties (image, color etc..) will apply to the content and padding areas.

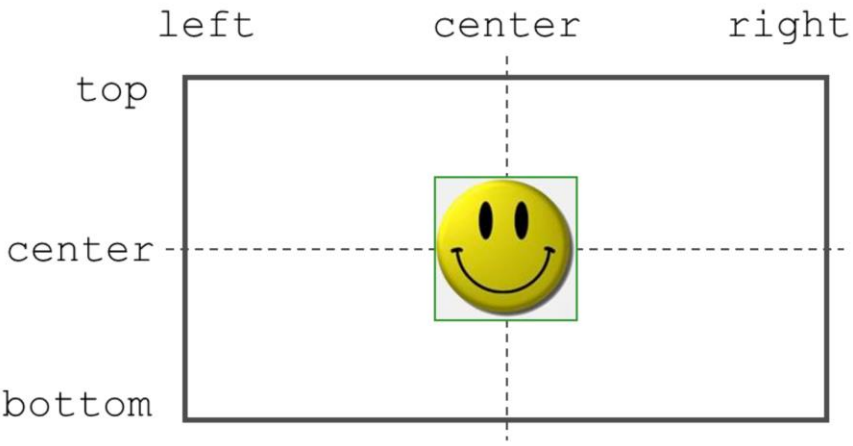
```
{
 height: 150px;
 width: 150px;
 padding: 10px;
 border-width: 1px;
 margin: 5px;
 background: url(mickey.jpg)
   no-repeat;
}
```



Margin is transparent: background colors or images will not show in that area.

Background Positioning

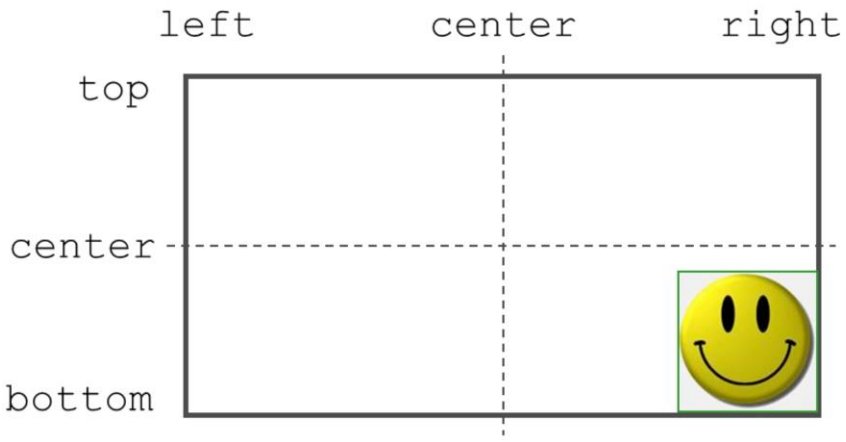
- By keyword




```
background: url(smiley.jpg)no-repeat center center;
```

Background Positioning

- By keyword

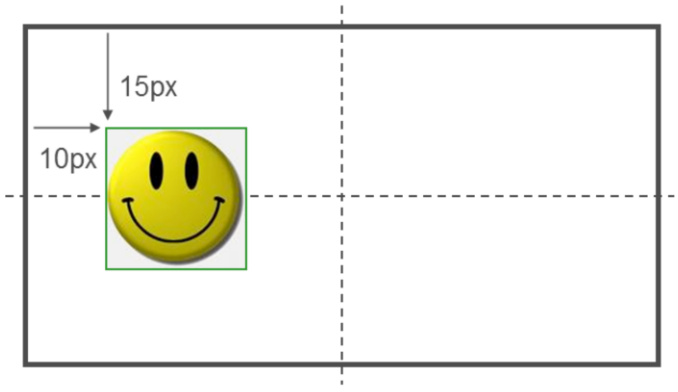



```
background: url(smiley.jpg)no-repeat right bottom;
```

Keyword here sets the right of the image with the right of the container, and the bottom with the bottom

Background Positioning

- By fixed unit

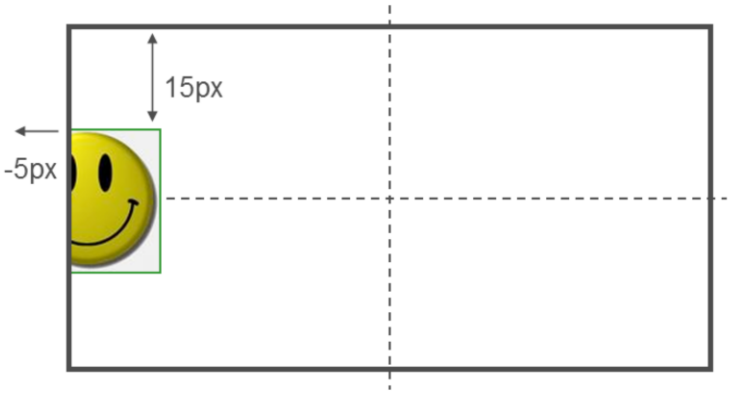



```
background: url(smiley.jpg)no-repeat 10px 15px;
```

Sets the start position (top/left) of the background image within the container element

Background Positioning

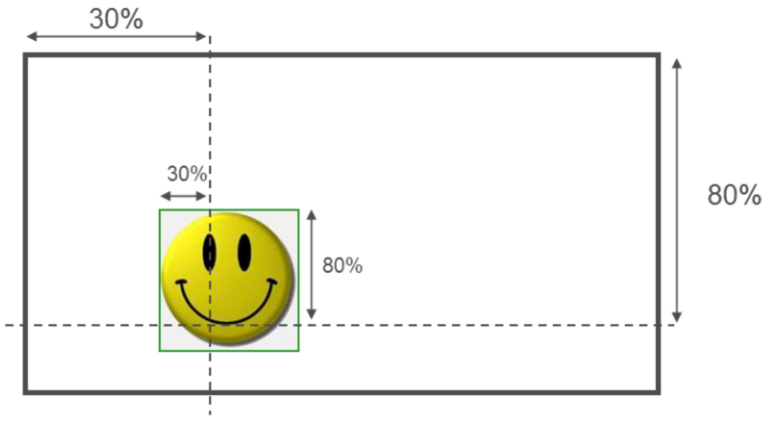
- By fixed unit




```
background: url(smiley.jpg)no-repeat -5px 15px;
```

Background Positioning

- By percentage



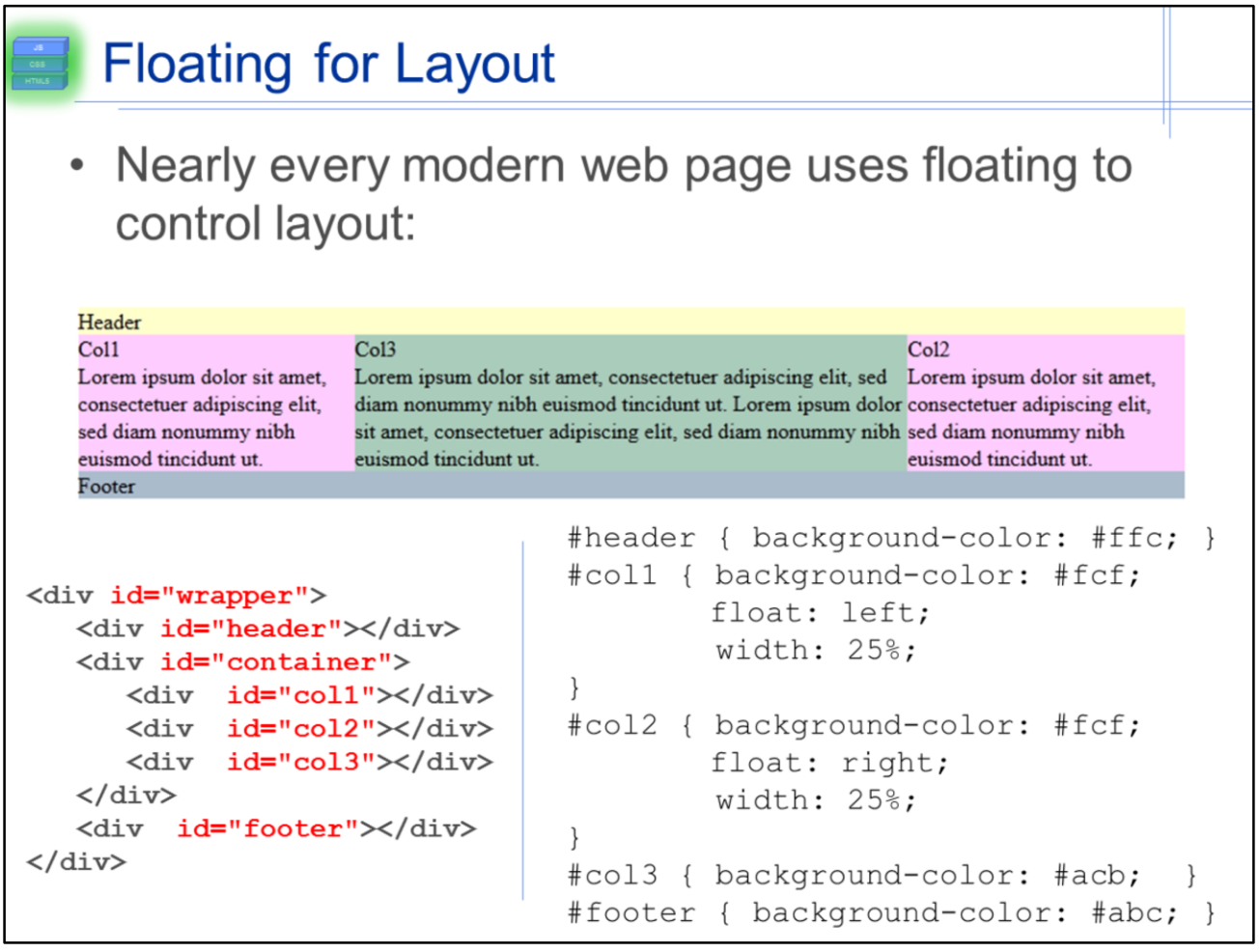

```
background: url(smiley.jpg)no-repeat 30% 80%;
```

Matches the x-axis percentage of the container with the same x-axis percentage of the background image
Same with y-axis
Very hard to predict results – usually avoided.

Exercise 2 - It's Your Turn

- Create the following by using CSS (HTML is already provided). Work from the labs/lab02/starter folder.

- Hints:
  - **You should only have to complete lab02.css**
  - **Use floating to make list items stack horizontally**
  - **Add the background image to each anchor**
  - **The image is approx. 100 x 75 pixels**
  - **Hints: The image is on the anchor, to give inline elements a height and width you must make them block elements**
  - **For this exercise, give the <ul> a height**

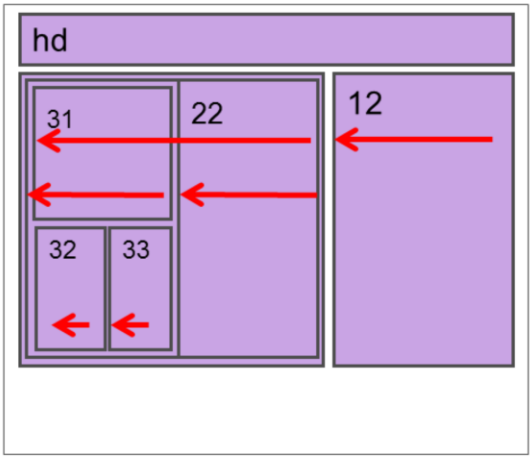


See examples in examples/03_css/floating0.html

Two-Columns

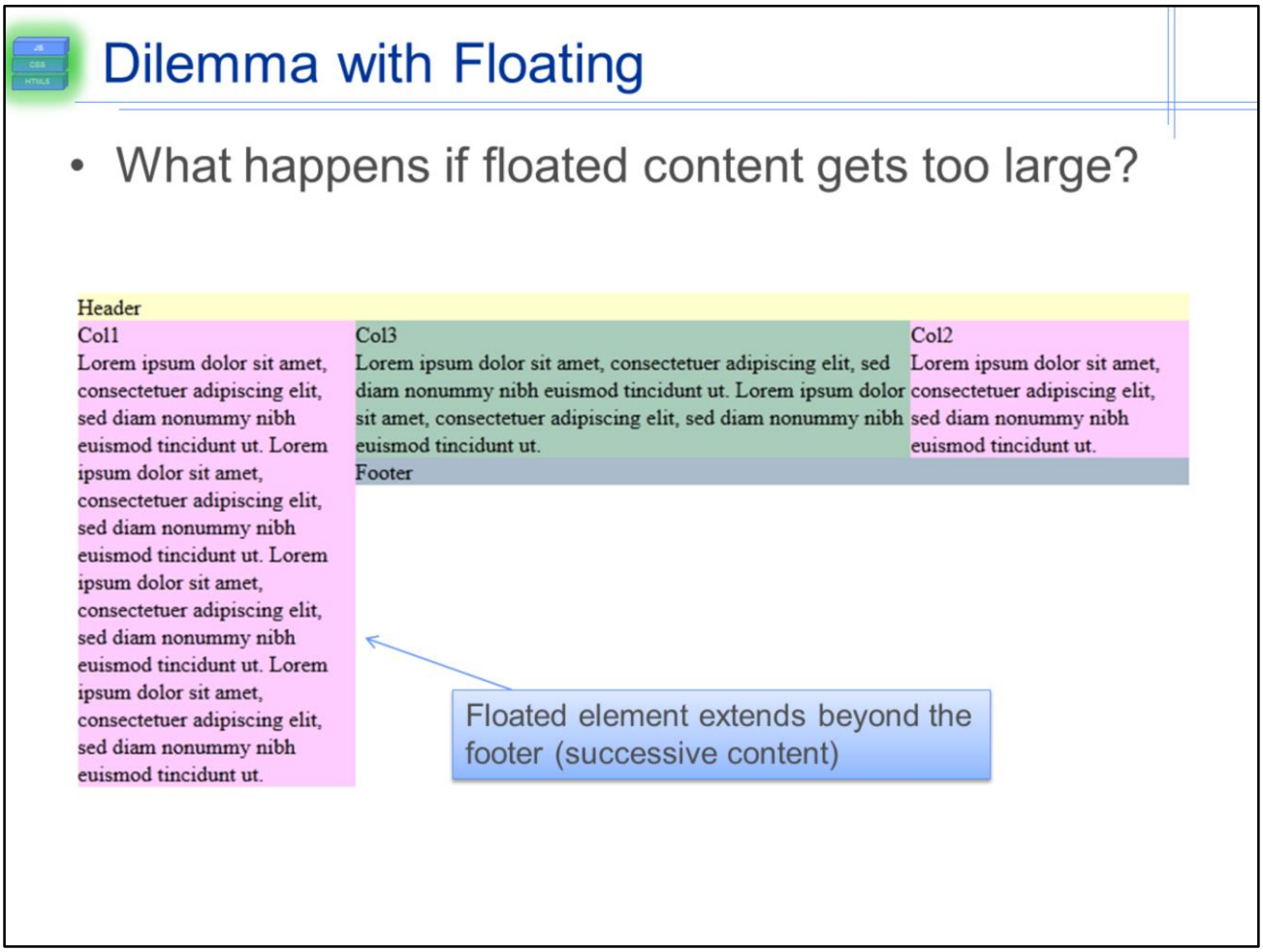
- Many layouts are decomposed into two columns

```
<div id="wrapper">
   <div id="hd"></div>
   <div id="col11">
      <div  id="col21">
         <div  id="col31"></div>
         <div  id="col32"></div>
         <div  id="col33"></div>
      </div>
      <div  id="col22"></div>
   </div>
   <div id="col12"></div>
</div>
```




```
#col11, #col21, #col32 { width: 55%; float: left; }
#col12, #col22, #col33 { width: 45%; float: left; }
```

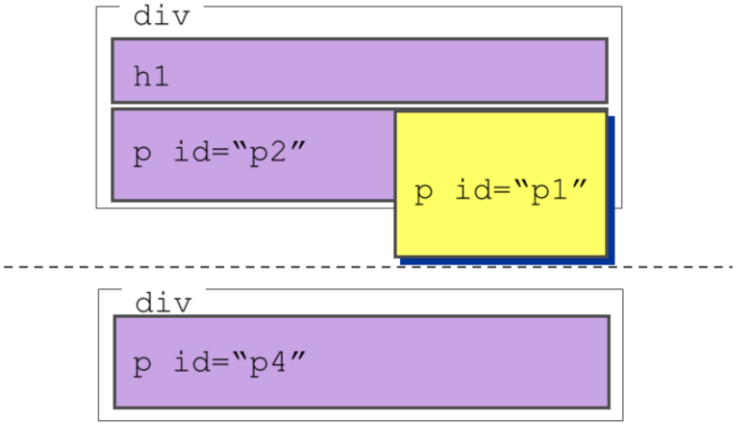
See examples/03_css/floating1.html



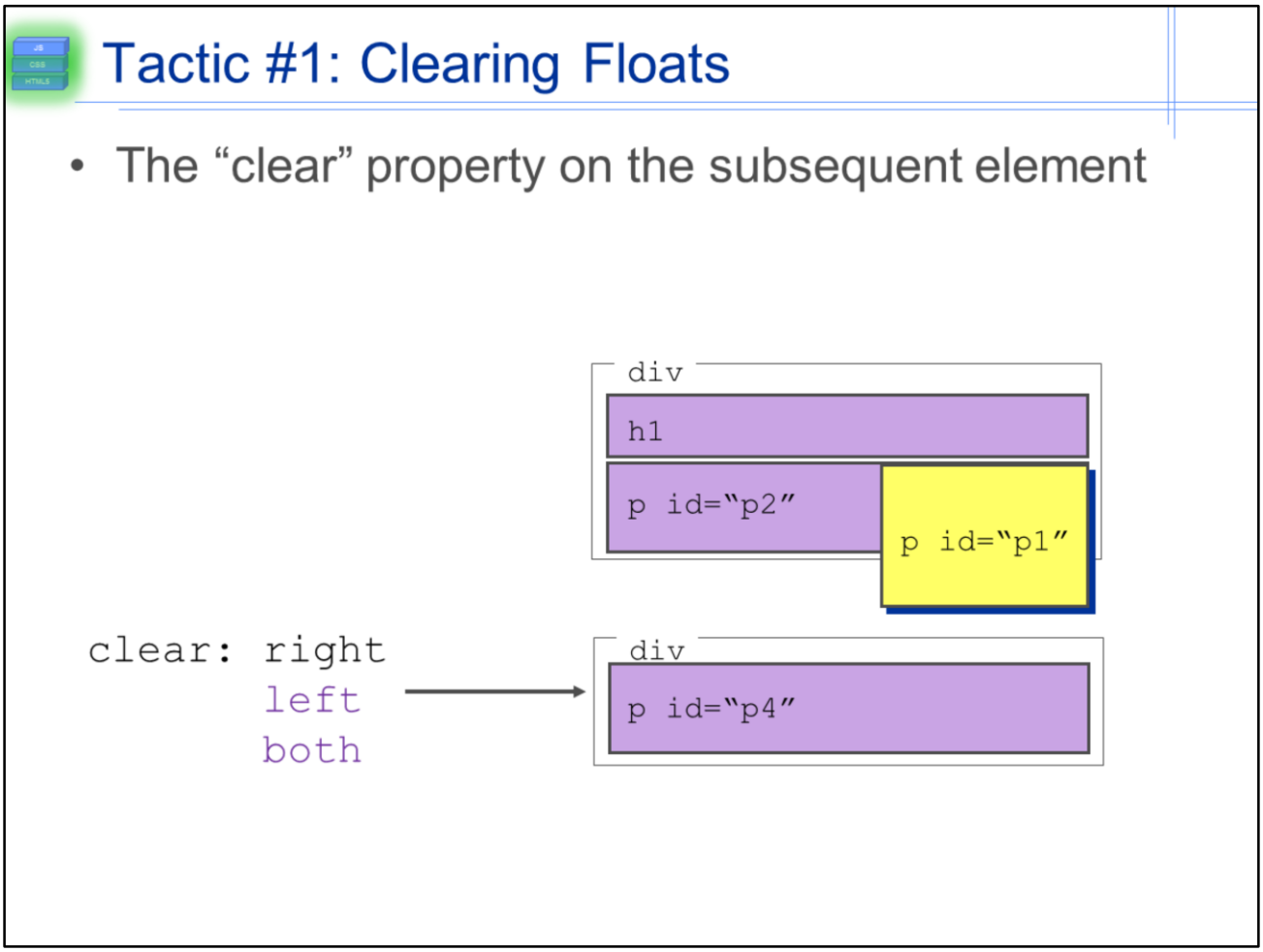
See examples in examples/03_css/floating2.html

Solving this Dilemma: Clearing

- Often, you want to define a break point where content stops going up behind a float



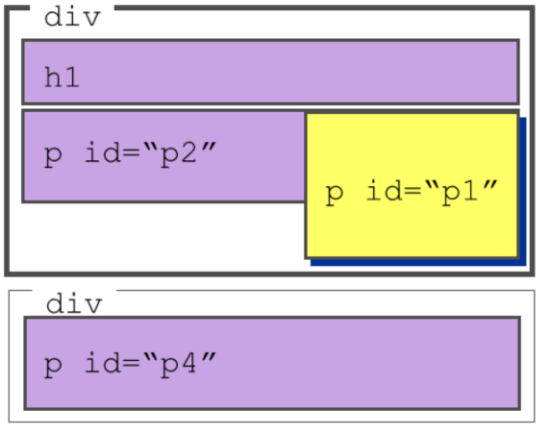
4 ways to do this…



The disadvantage to this approach is that the subsequent div must "know" to clear the content above it. There is a coupling or dependency created.

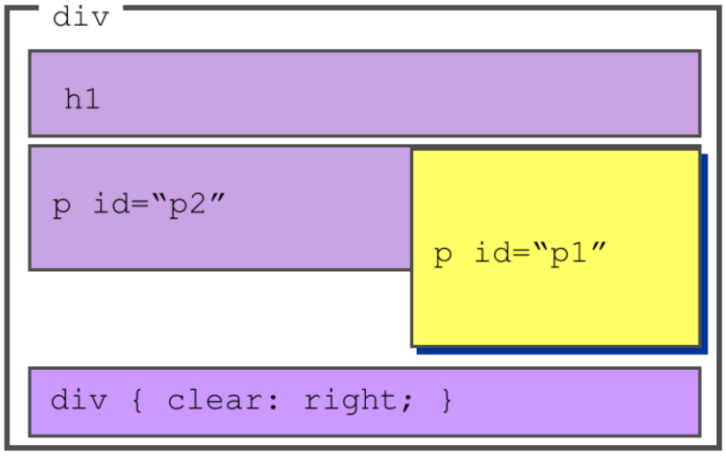
Tactic #2: Clearing Floats

- Force the parent to enclose the float;
  "Float nearly everything"



This approach leads to the task of needing to float everything.

Tactic #3: Self-clearing DIV



By placing a inside the parent of a floated element, but after the floated element, we can give it the clear property.

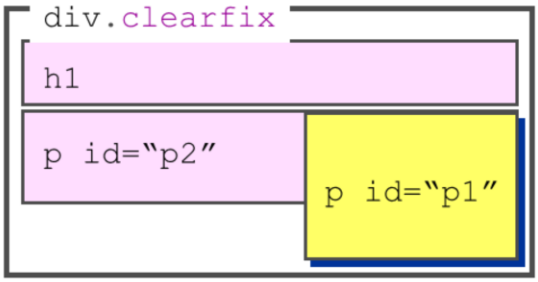
This has no height, cannot be seen, but ensures the parent always encapsulates the floated element.

This approach introduces a non-semantic element, usually a div placed after the floated content which then clears. This works nicely, but the div serves no purpose but to aid in presentation.

Tactic #4: The Best Technique

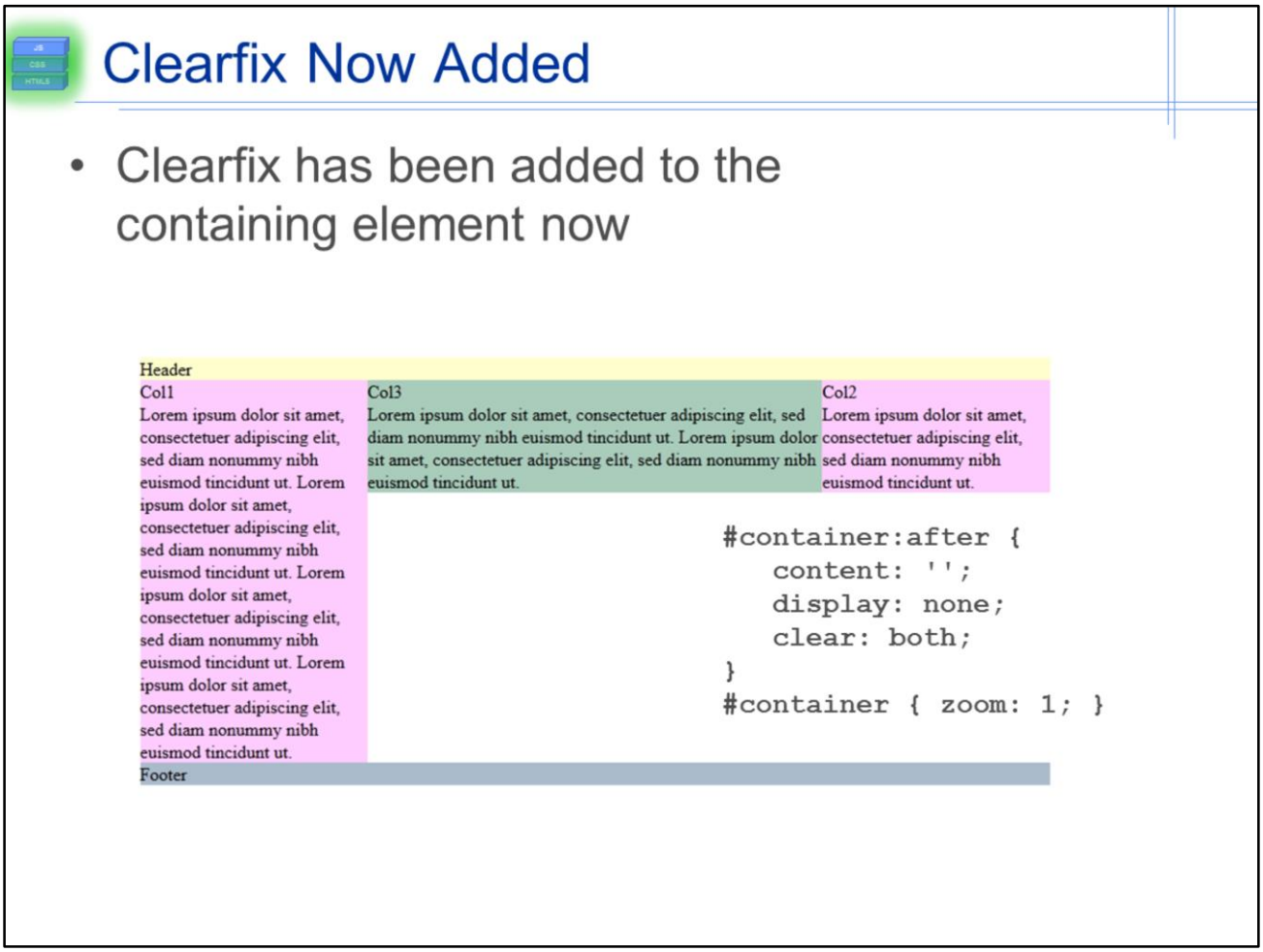
- The clearfix technique

```
.clearfix:after {
   content: "";
   display: block;
   clear: both;
}

.clearfix {
   zoom: 1; /* IE hack */
}
```



What is this hack with IE?

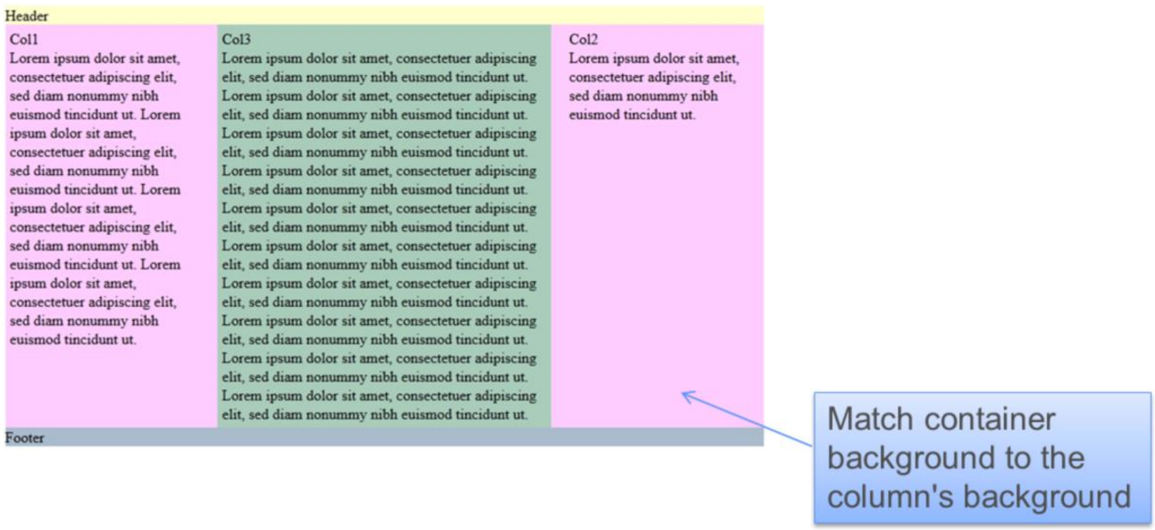
Read on…



See examples in examples/03_css/floating3.html

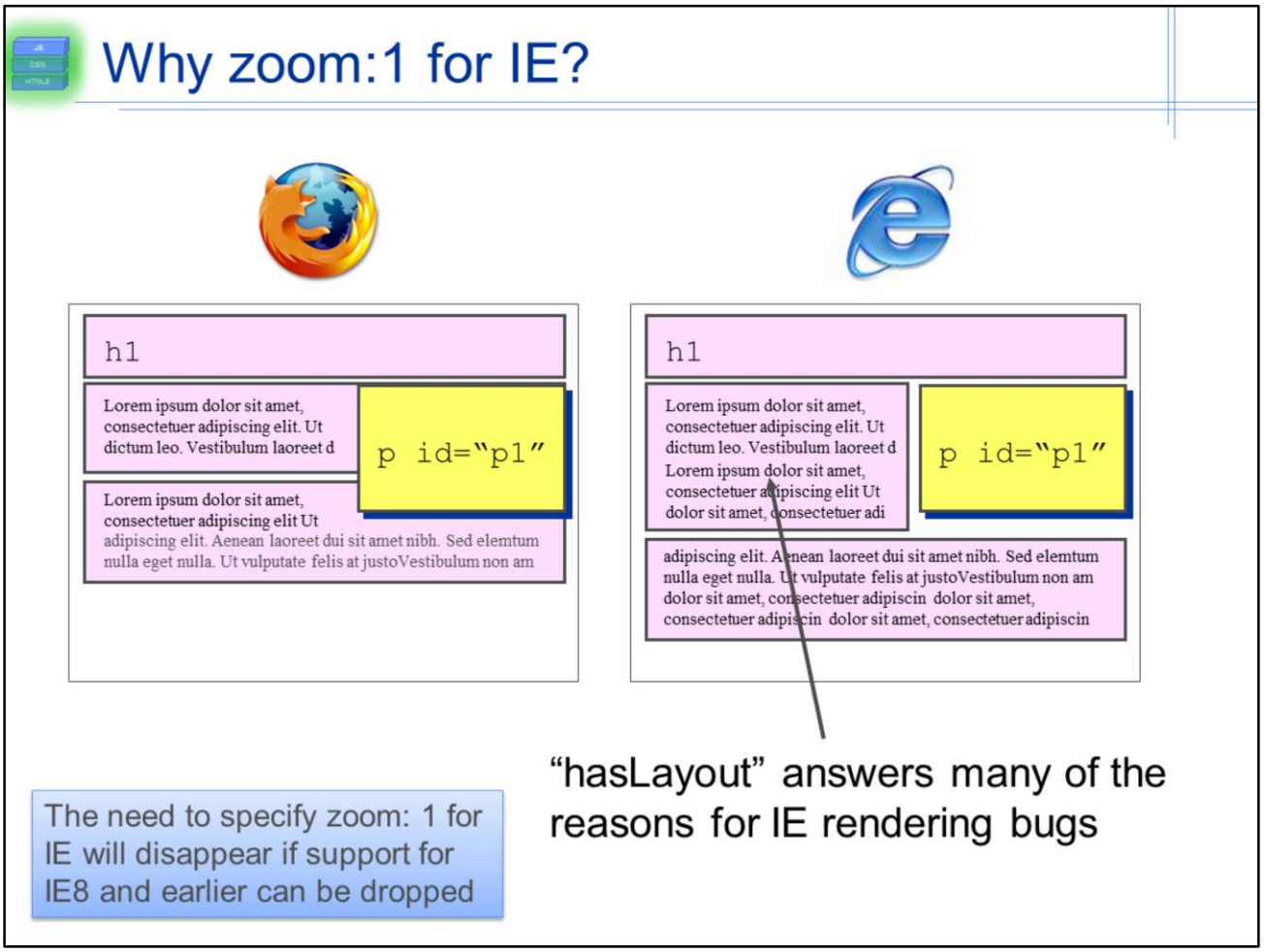
What About My Uneven Column Heights?

- Often CSS will require a little crafty thinking or some trickery
  - Our columns are given the appearance of being the same height



See examples in examples/03_css/floating4.html

For this trick to work, the middle column should be the longest one. But generally this requirement is not a problem.



In older IE's, a block element with "hasLayout" will not run behind the floated element as the spec dictates. Instead, the entire block will run alongside the float.

hasLayout is an important feature to understand how IE behaves when content renders. For more on this hidden property, read this very well written article: http://www.satzansatz.de/cssd/onhavinglayout.html

hasLayout is used to either "trigger" layout if set or NOT trigger layout if NOT set. hasLayout, however, is a readonly property and therefore it cannot be directly set. Instead, a number of properties can be used to "trigger" hasLayout in IE. (refer to the above cited article for these). This also explains why zoom:1 on the previous page happens to work for the clearfix on IE.

Exercise 3 - It's Your Turn

- Return to lab 02 and modify the solution by removing the 'height' on the <ul> or the depending on your solution
- Use 'clearfix' technique to force the purple background to remain in tact. Ensure the solution renders as before, this time without the <ul> or height

Exercise 4 - It's Your Turn

Multi-column Layout

Locate step-by-step instructions for this lab in the back of your manual

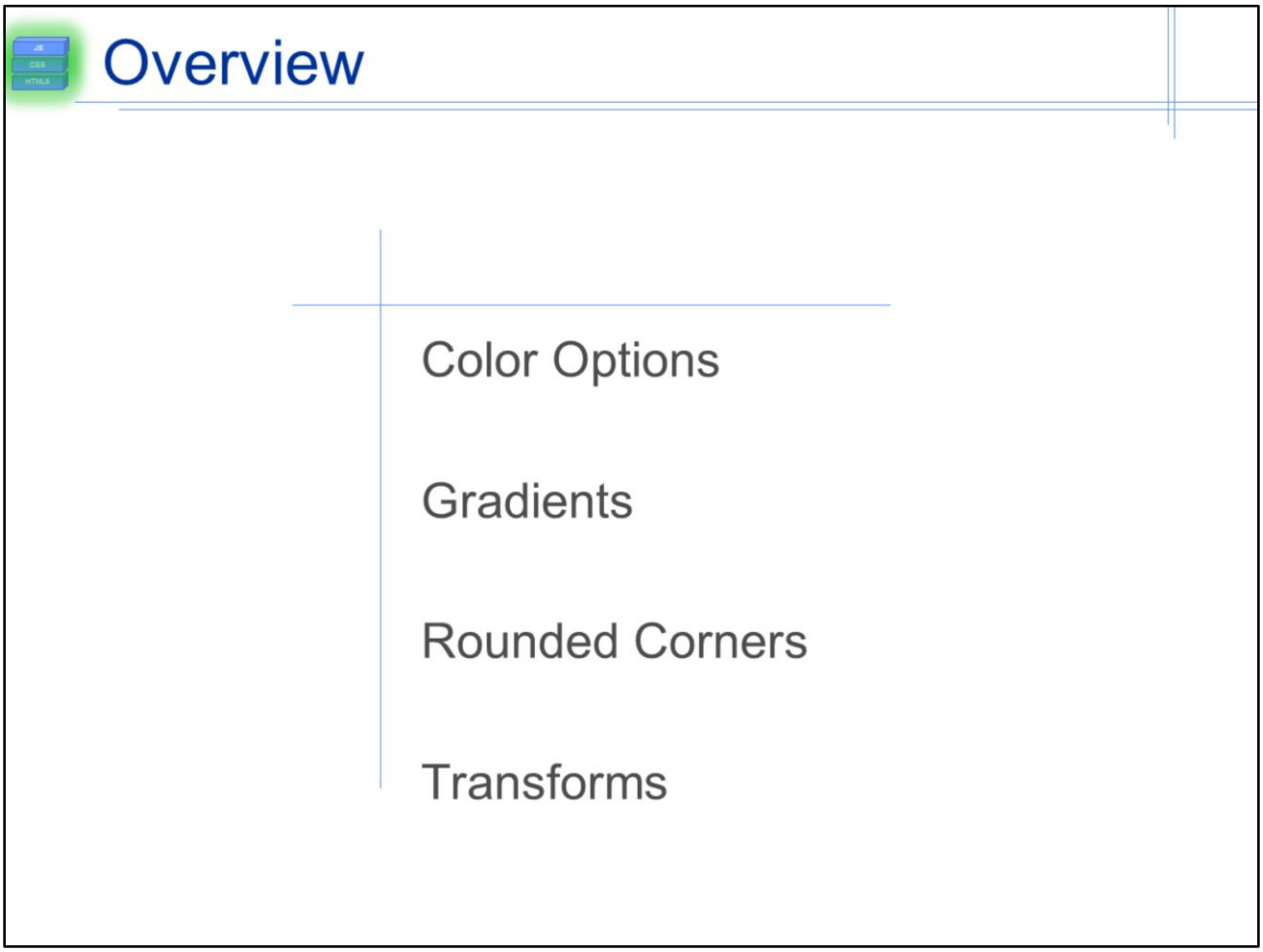
Summary

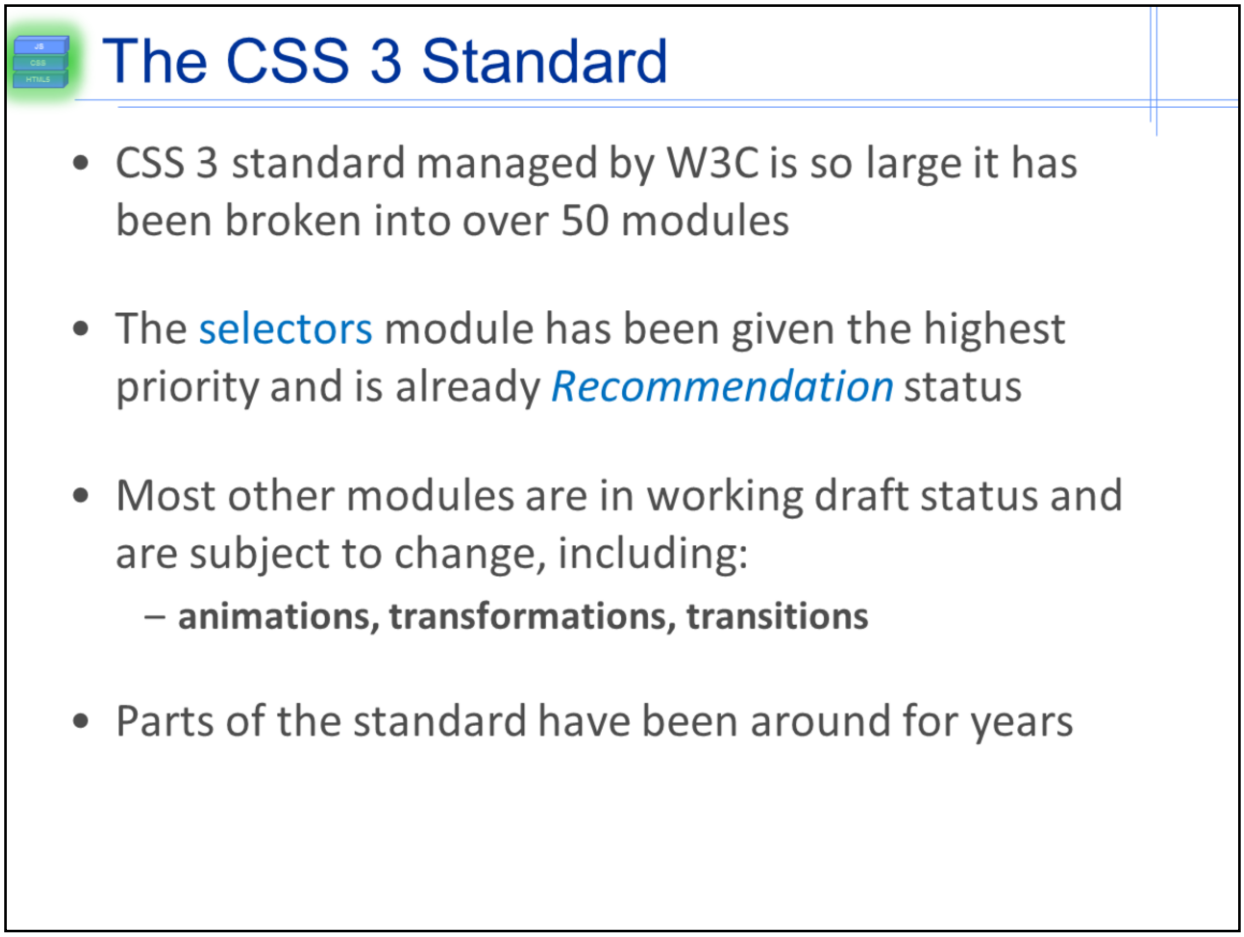
- The CSS Box Model defines numerous properties that help control an element's look
- Lay boxes out using floating and/or absolute positioning
- Some favor the use of display:inline-block instead of floating
  - While this alternative approach is okay to use, be aware that display:inline-block can cause an undesired space to occur between elements
  - Extra work must then be taken to remove that space if desired

Chapter 4

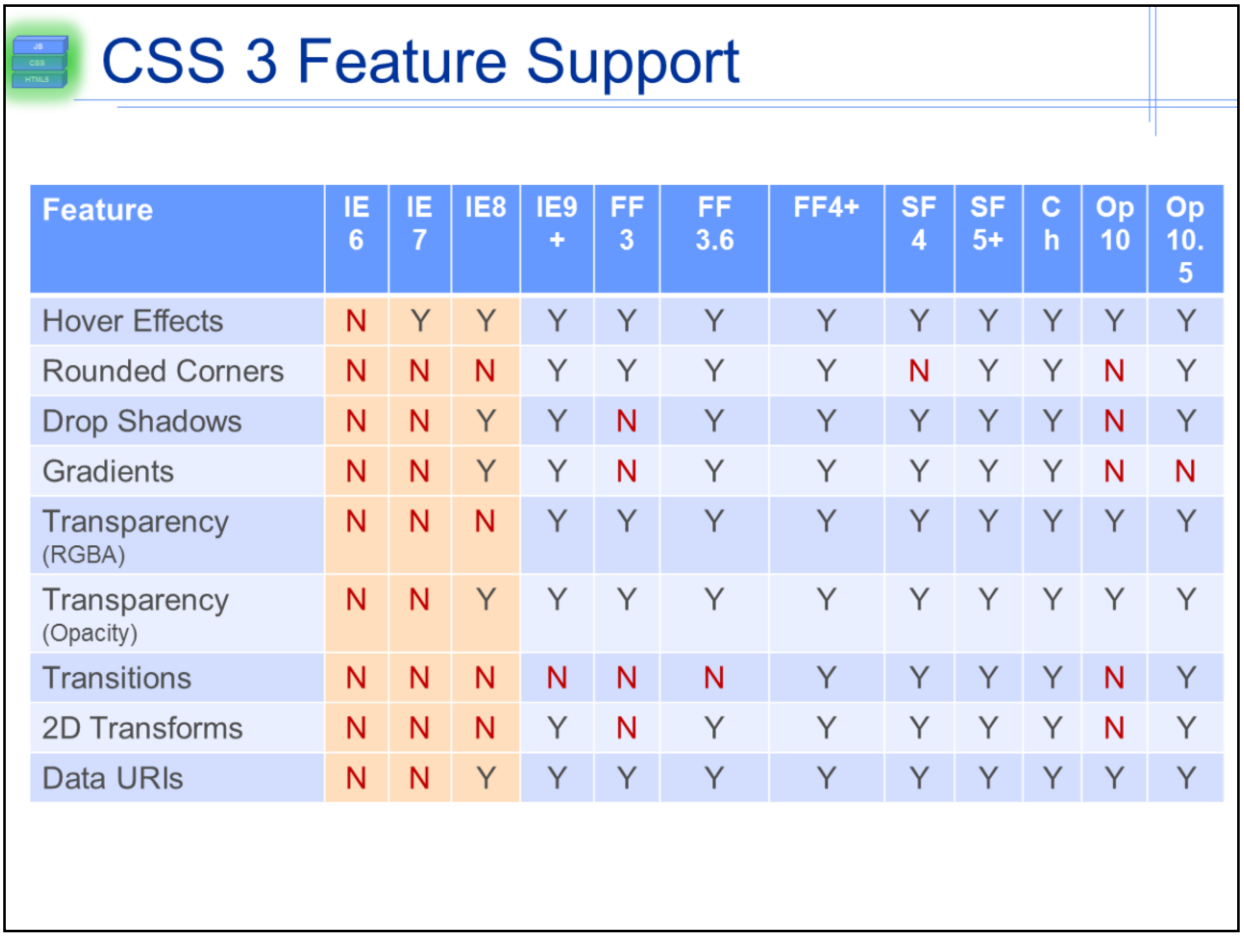
CSS 3 Features

Enhancing Solutions with Style





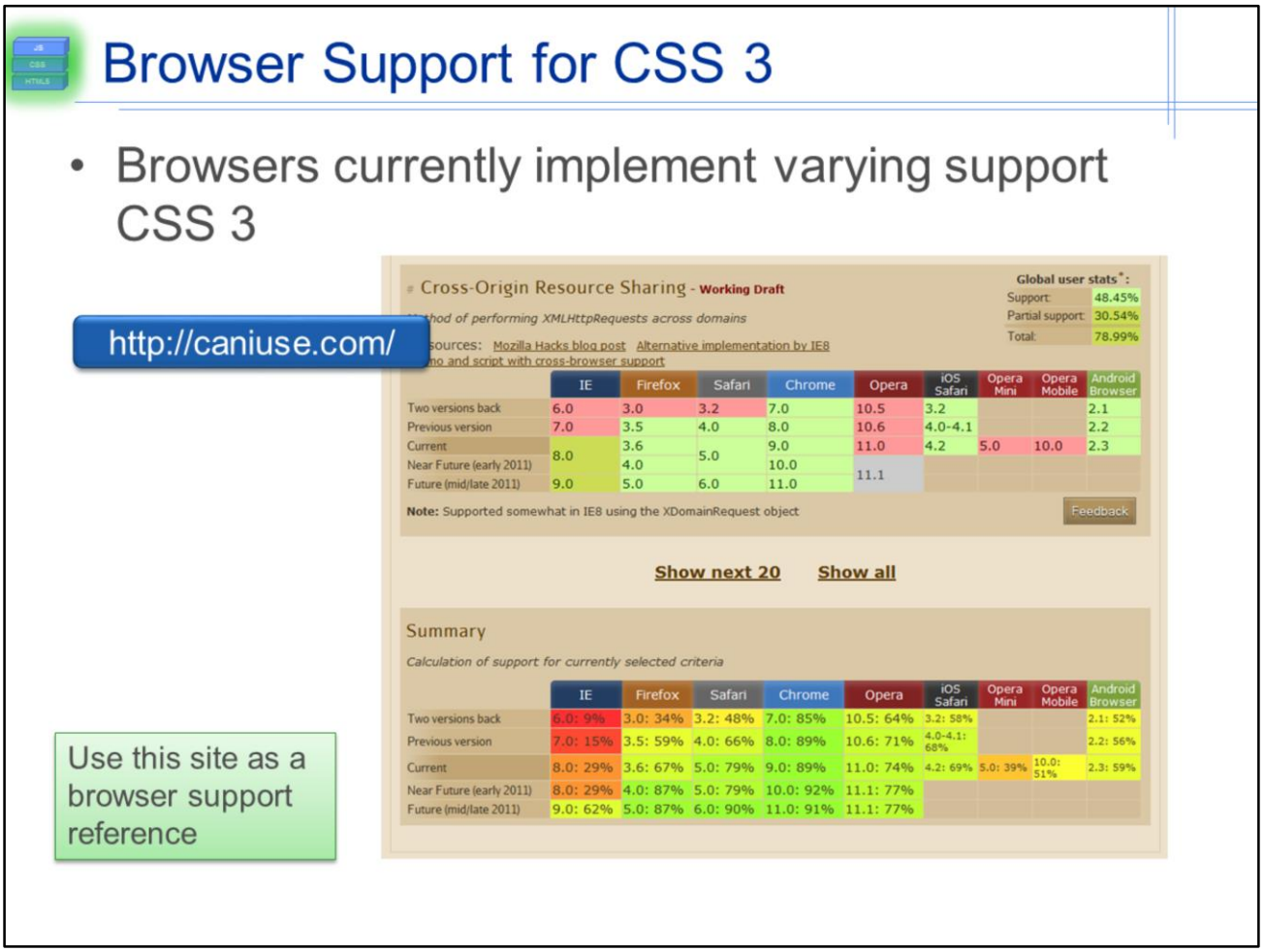
Good reference on CSS3 standards status: http://www.css3.info/modules/



CSS 3 Feature Support

Feature	IE 6	IE 7	IE8	IE9 +	FF 3	FF 3.6	FF4+	SF 4	SF 5+	C h	Op 10	Op 10. 5
Hover Effects	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Rounded Corners	N	N	N	Y	Y	Y	Y	N	Y	Y	N	Y
Drop Shadows	N	N	Y	Y	N	Y	Y	Y	Y	Y	N	Y
Gradients	N	N	Y	Y	N	Y	Y	Y	Y	Y	N	N
Transparency (RGBA)	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
Transparency (Opacity)	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Transitions	N	N	N	N	N	N	Y	Y	Y	Y	N	Y
2D Transforms	N	N	N	Y	N	Y	Y	Y	Y	Y	N	Y
Data URIs	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

IE9 and earlier do not support 3D transforms, border-images, animations, transitions, multiple column layouts, and text-shadow.



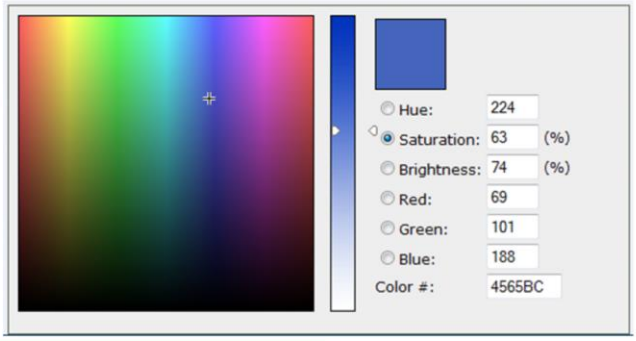
There are a number of sites that provide matrices, summaries, or charts of browser support. Some of these sites include:

html5demos.com
html5readiness.com
findmebyip.com/litmus
en.wikipedia.org/wiki/Comparison_of_web_browsers

The one shown here provides information current through the latest (even next) release of each browser.

New CSS Color Options

- CSS3 provides additional ways of specifying color selections
  - Classic approach: hex values, RGB percentage/value
  - New approaches:
    - RGBA
    - HSL (Hue/Saturation/ Lightness)
    - HSLA



RGBA & HSLA offer a fourth value: alpha transparency where:
0 = fully transparent
1 = completely opaque

http://www.w3.org/TR/css3-color/

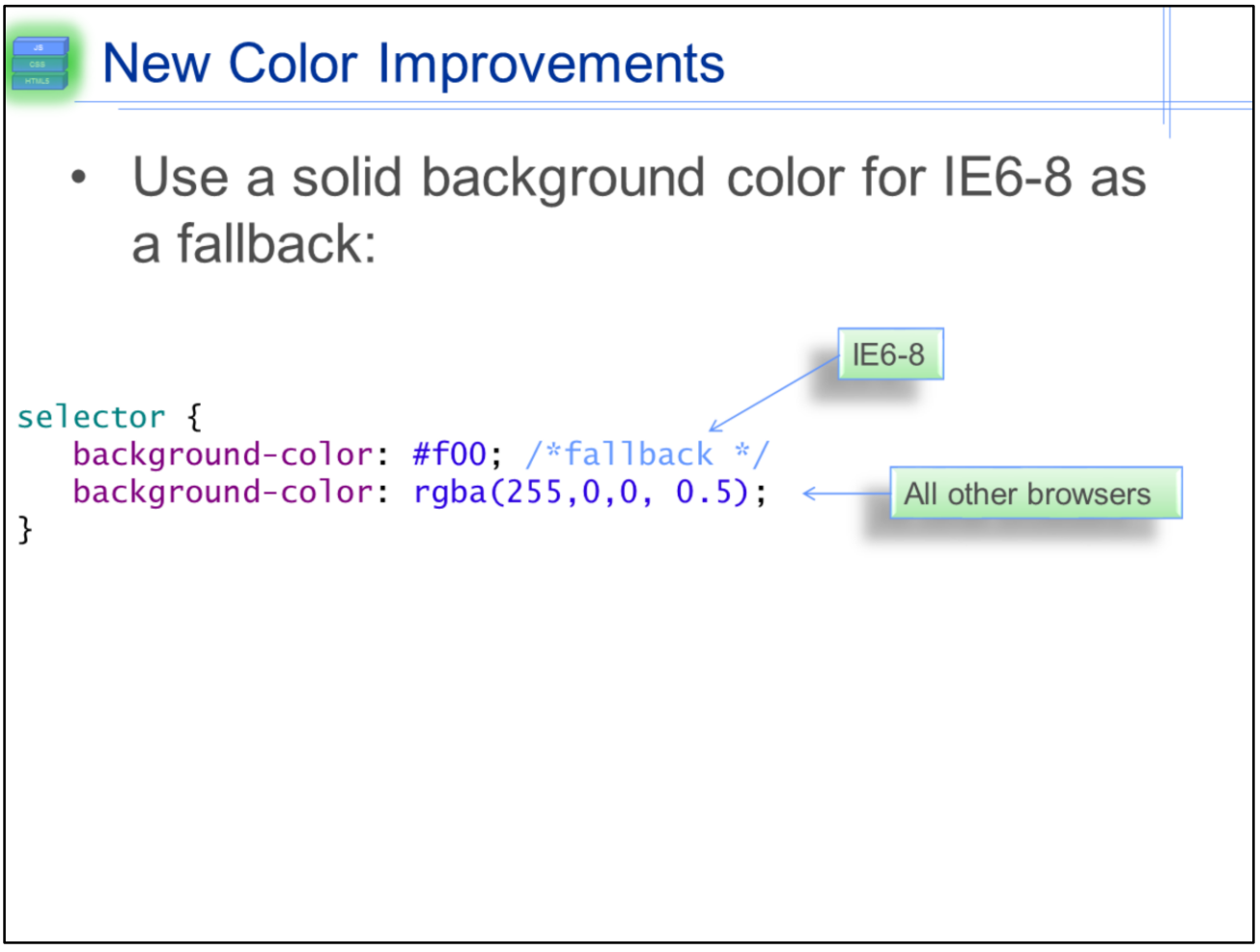
Hexadecimal notation does not have an equivalent way to express alpha-transparency values.

How HSL Works

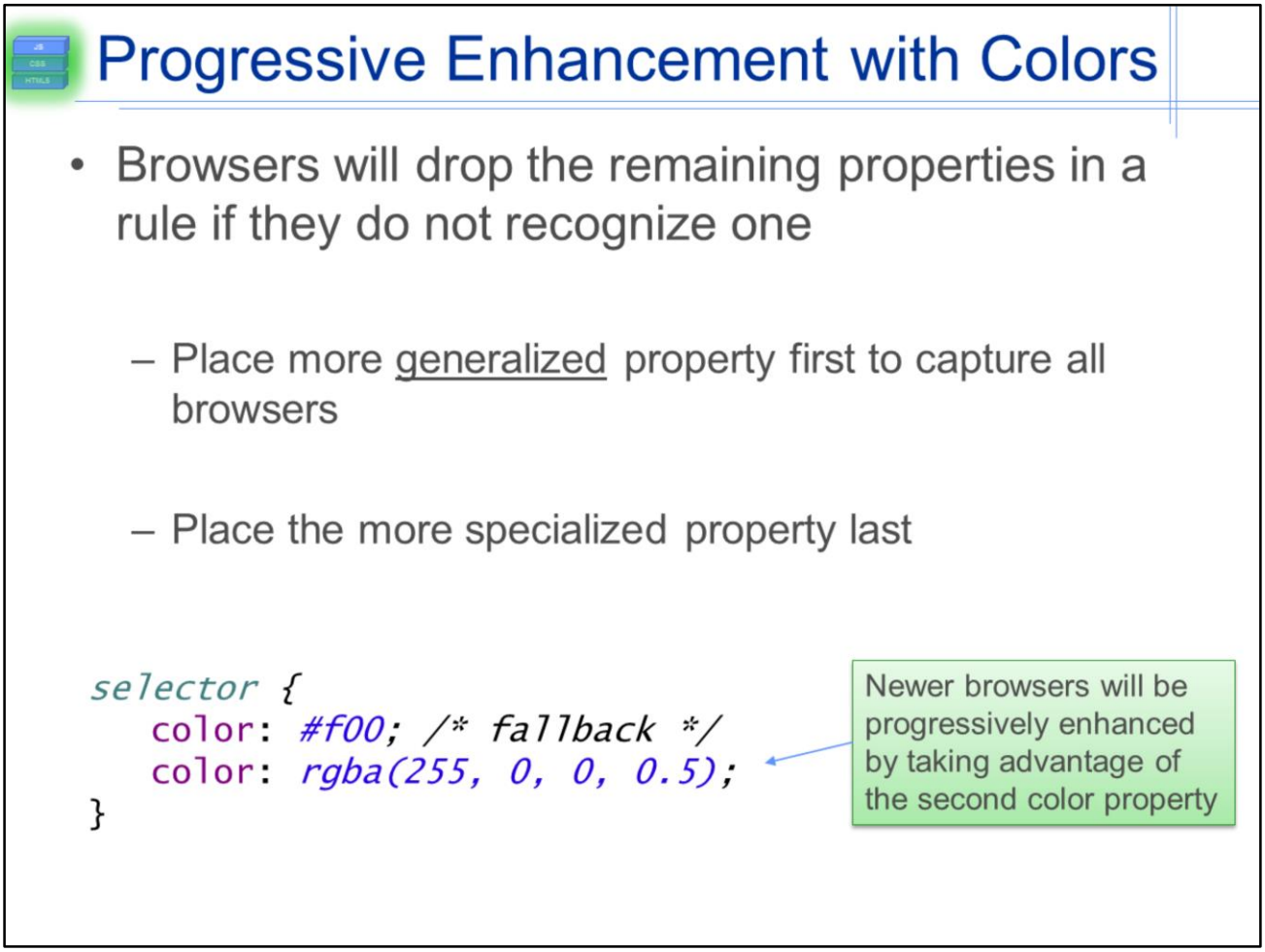
- HSL (Hue / Saturation / Lightness) works as follows:
  - **Hue is defined as a degree on a color wheel**
    - 0 (360) is red
    - 240 is blue
    - 120 is green
  - **Saturation is a percentage of the full color (0% – 100%)**
  - **Lightness (0% dark – 100% white)**

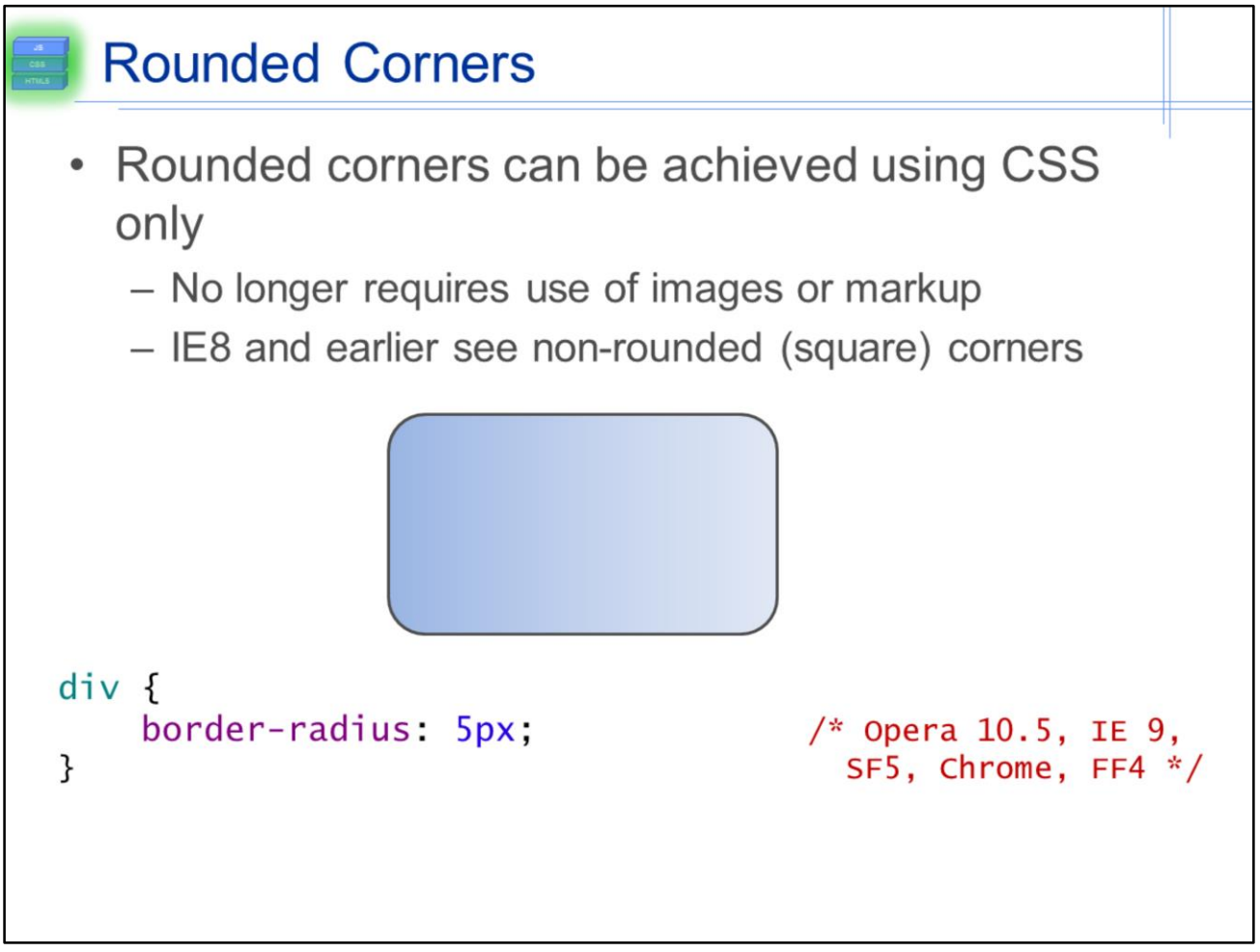
New Color Examples

```
div {
  color: rgb(0,0,0);
  background-color: rgb(255,255,255);
}

div {
  color: rgba(0,0,0,0.6);              /* semitransparent black */
  background-color: rgba(255,255,255,0.6);  /* semitrans. wht */
}


div {
  color: hsl(0,0%,0%);                    /* black */
  background-color: hsl(0,0%,100%);       /* white */
}

div {
  color: hsla(0,0%,0%,0.6);            /* semitransparent white */
  background-color: hsla(0,0%,100%,0.6);   /* semitrans black */
}
```



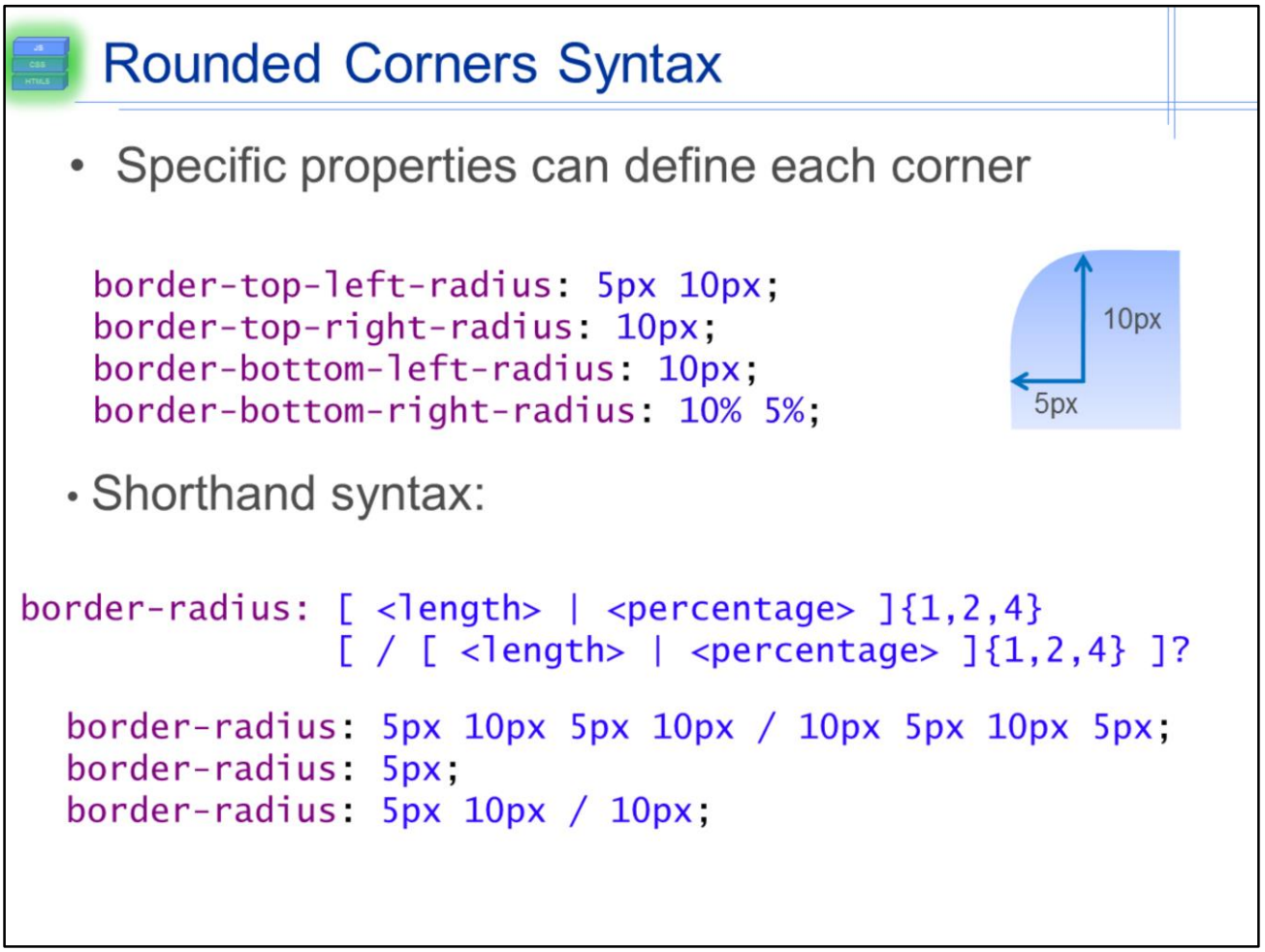
Note that the –ms-filter opacity (and the standard opacity) are slightly different in meaning than the background-color alpha transparency value. The later will only make the background-color transparent, while the –ms-filter (or opacity) option will make the entire element transparent including foreground text color, border, padding, etc.



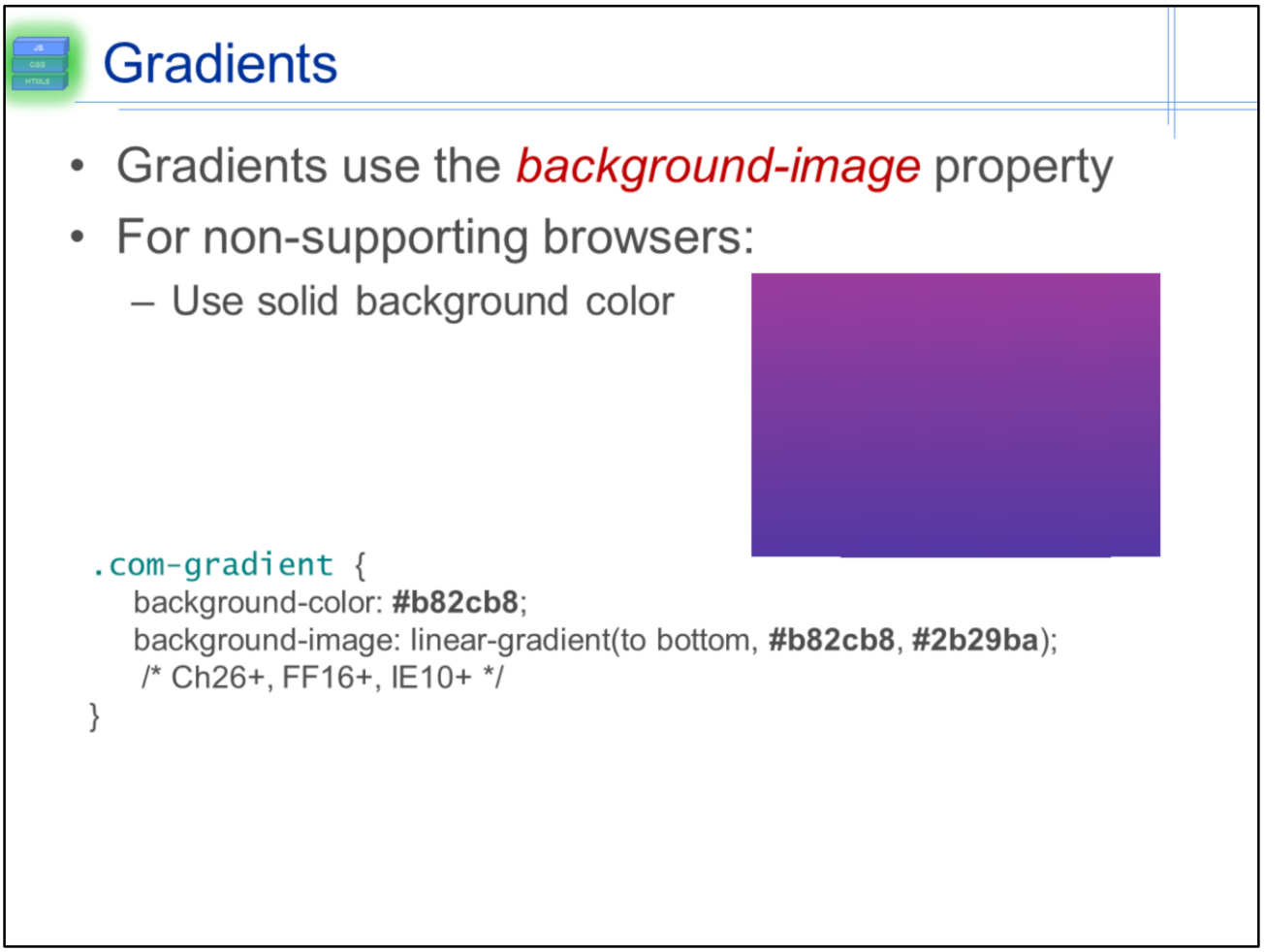


http://www.w3.org/TR/css3-background/

Most browsers today support the std property and extensions are generally not necessary.



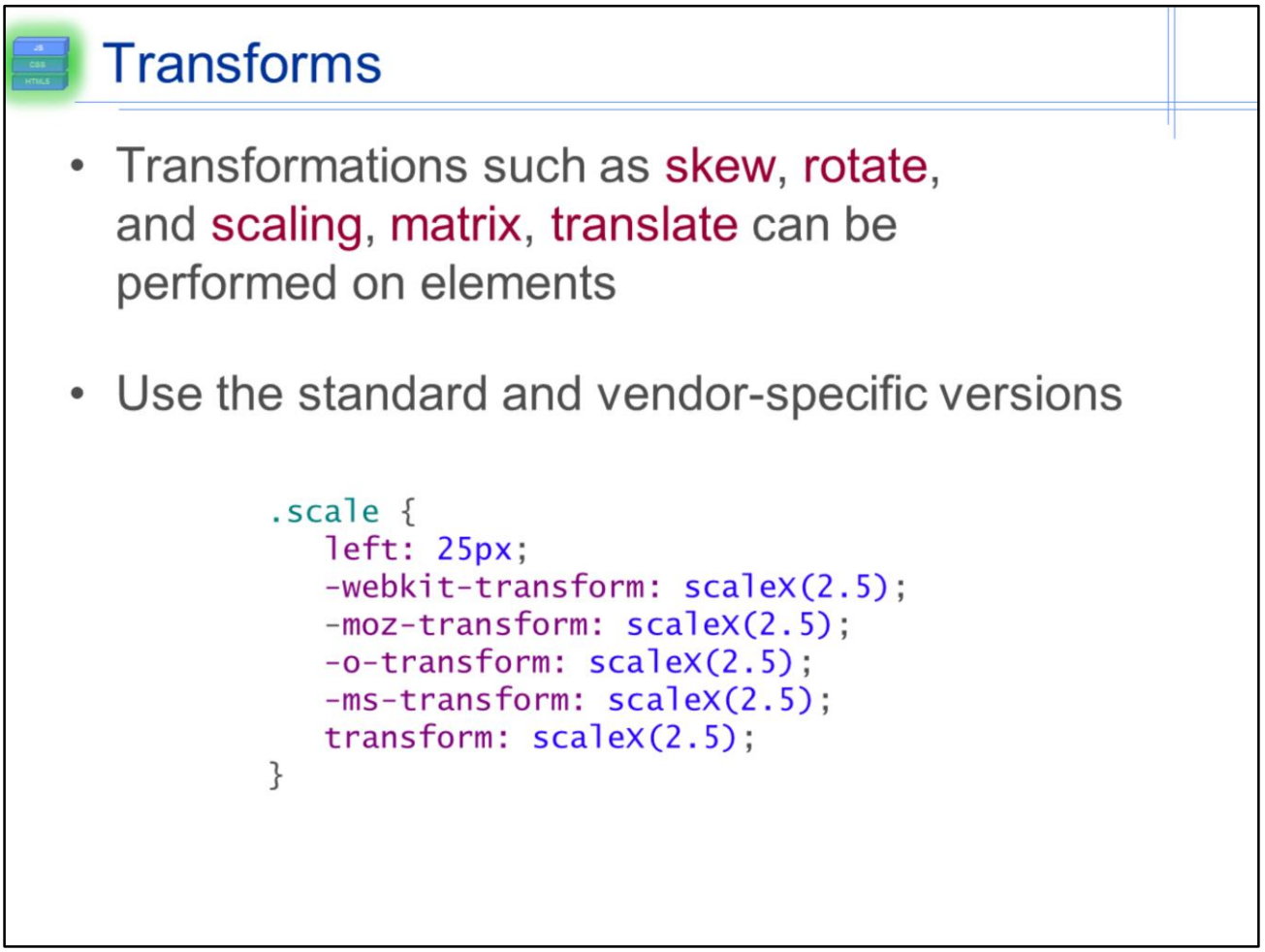
There are several ways to specify rounded-corner values. Either as one or two values for each corner or via the use of shorthands (shown at the bottom). Use the / operator to separate the x-values from the y-values.



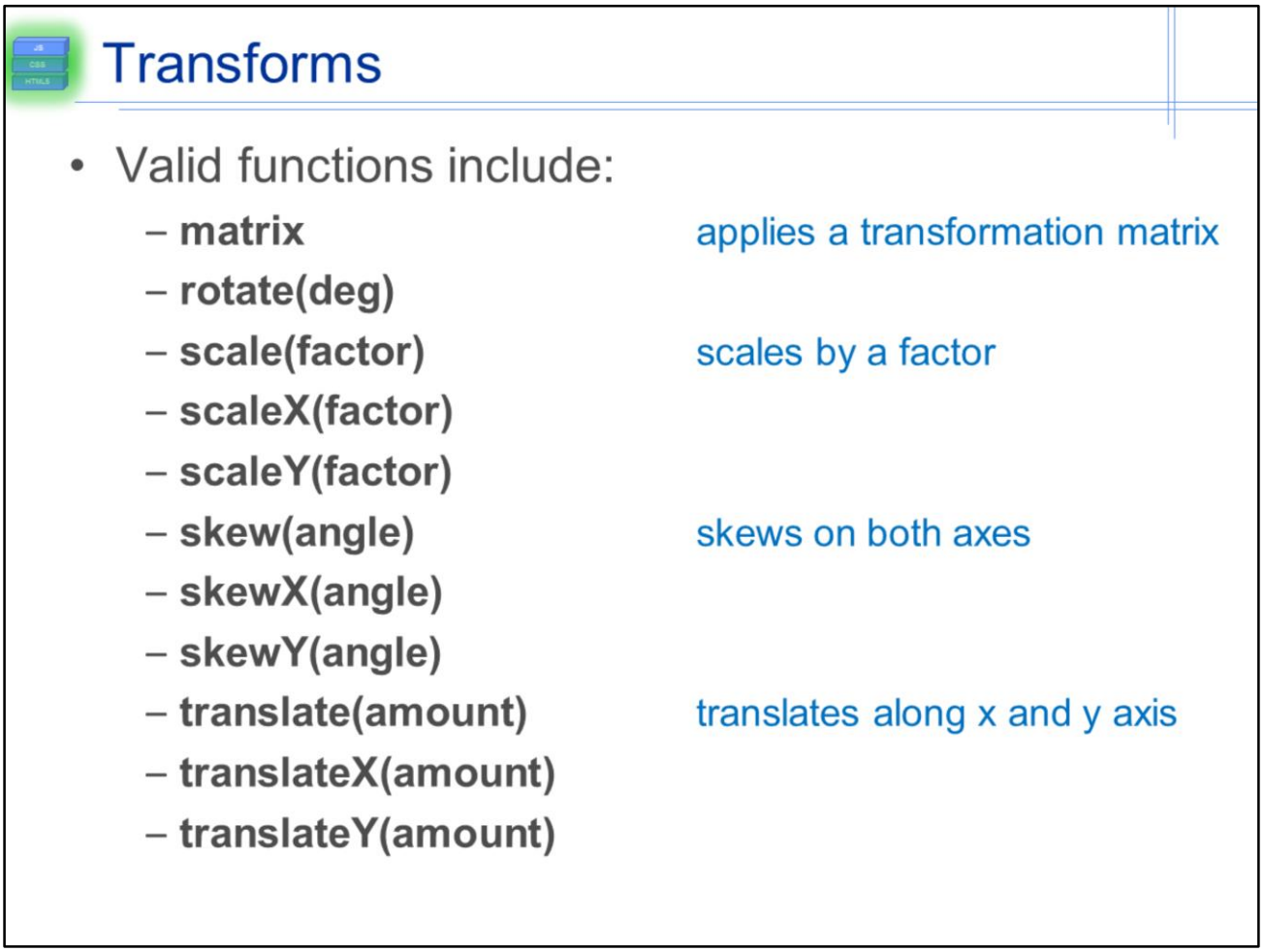
Gradients receive various levels of support in browsers. Because of this a determination has to be made as to the need for gradients within non-supporting browsers. If the gradient is important to the design of the UI or presentation, then an appropriate solution must be achieved for all browsers. If it is not important, an option to simply degrade to a solid color background can be made.

Our rule of thumb continues, specify the generic "all-supported" value first, in this case the standard color value. Then "enhance it" with the browser-specific extensions, and finally specify the standards-based version last.

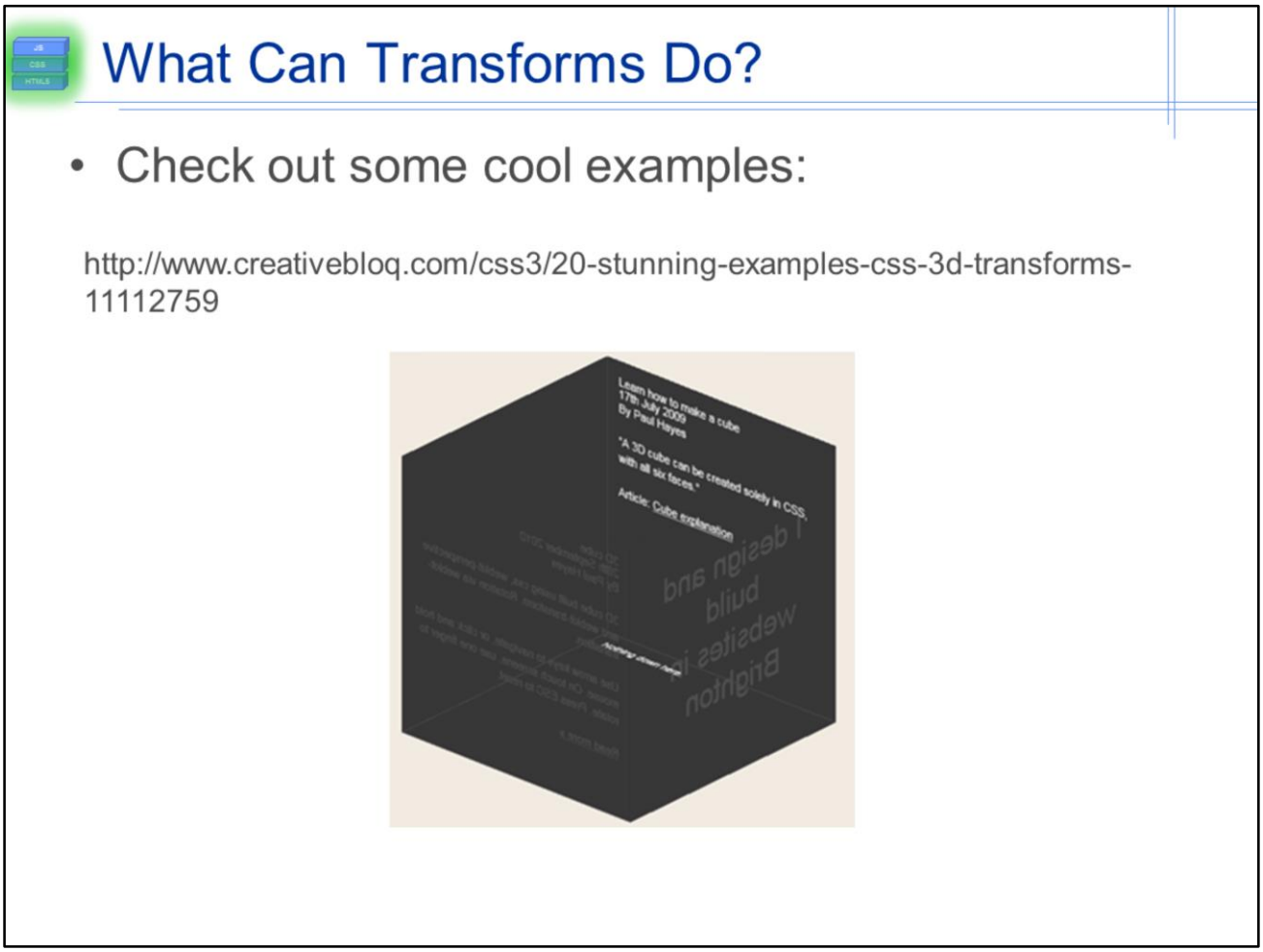
CSS3Please.com provides a good way to remember the proper progressive enhancement order of elements in CSS. The page shows the supported browser version and the correct order in which the rules need to be placed.



The transform property allows elements to be skewed, rotated, scaled, and translated. Implementations may use this technique as long as the non-transformed browser versions are fully usable without being transformed.



The following provides a list of the transform functions available to supporting browsers. Be sure to indicate the proper CSS property prefix for specific browsers.



There are numerous galleries that showcase great effects that can be generated with CSS animations, transforms, and transitions.

Exercise 5 - It's Your Turn

A CSS 3 Dialog Popup

Locate step-by-step instructions for this lab in the back of your manual

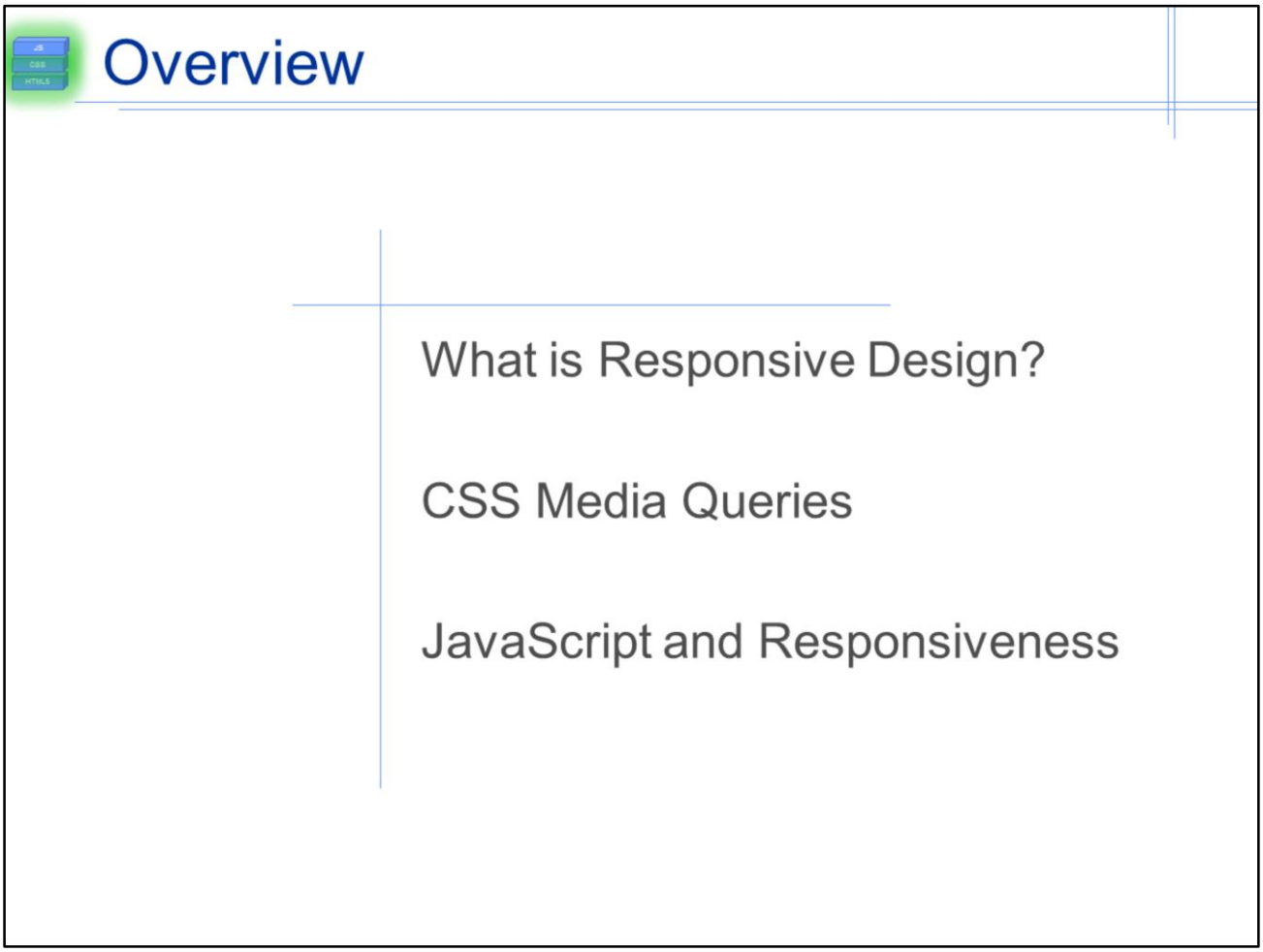
Summary

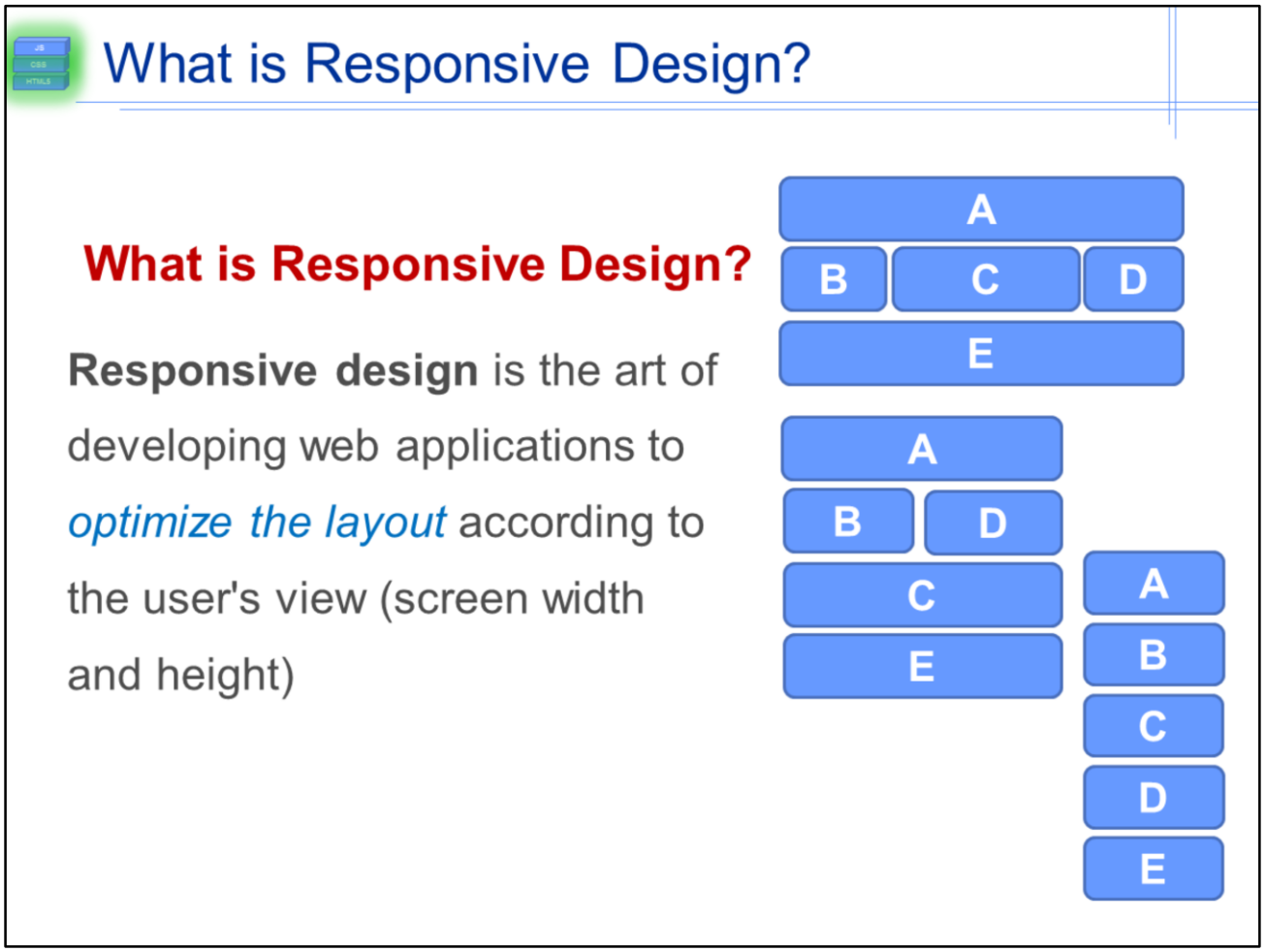
- CSS 3 features have become very broadly supported by modern browsers
  - If IE6-8 support is not needed, most CSS 3 features may be implemented
- Older browsers might need proper fallback
- Place properties in a specific order to achieve progressively enhance newer browsers

Chapter 5

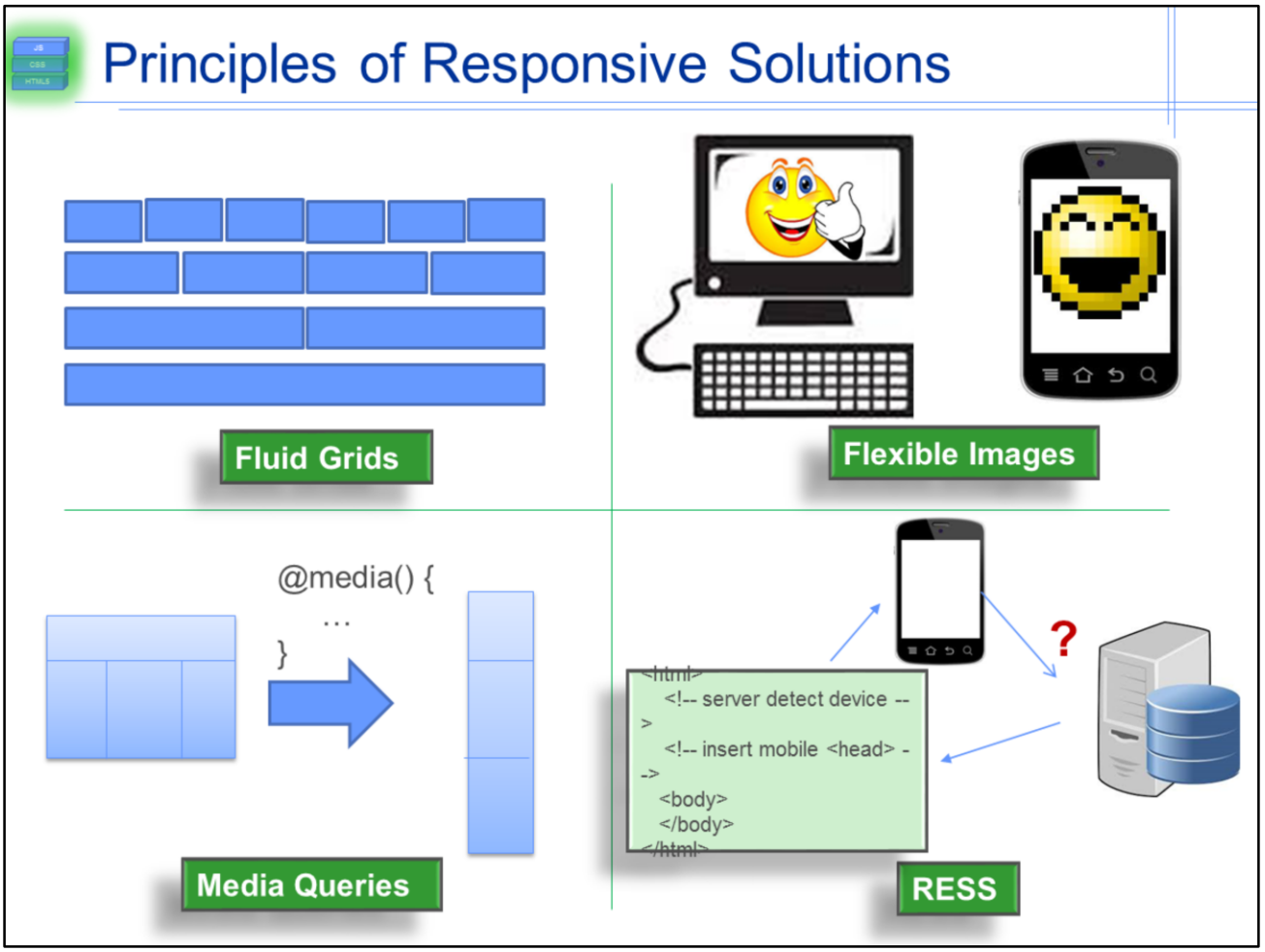
Responsive Solutions

Creating single solutions for multiple devices

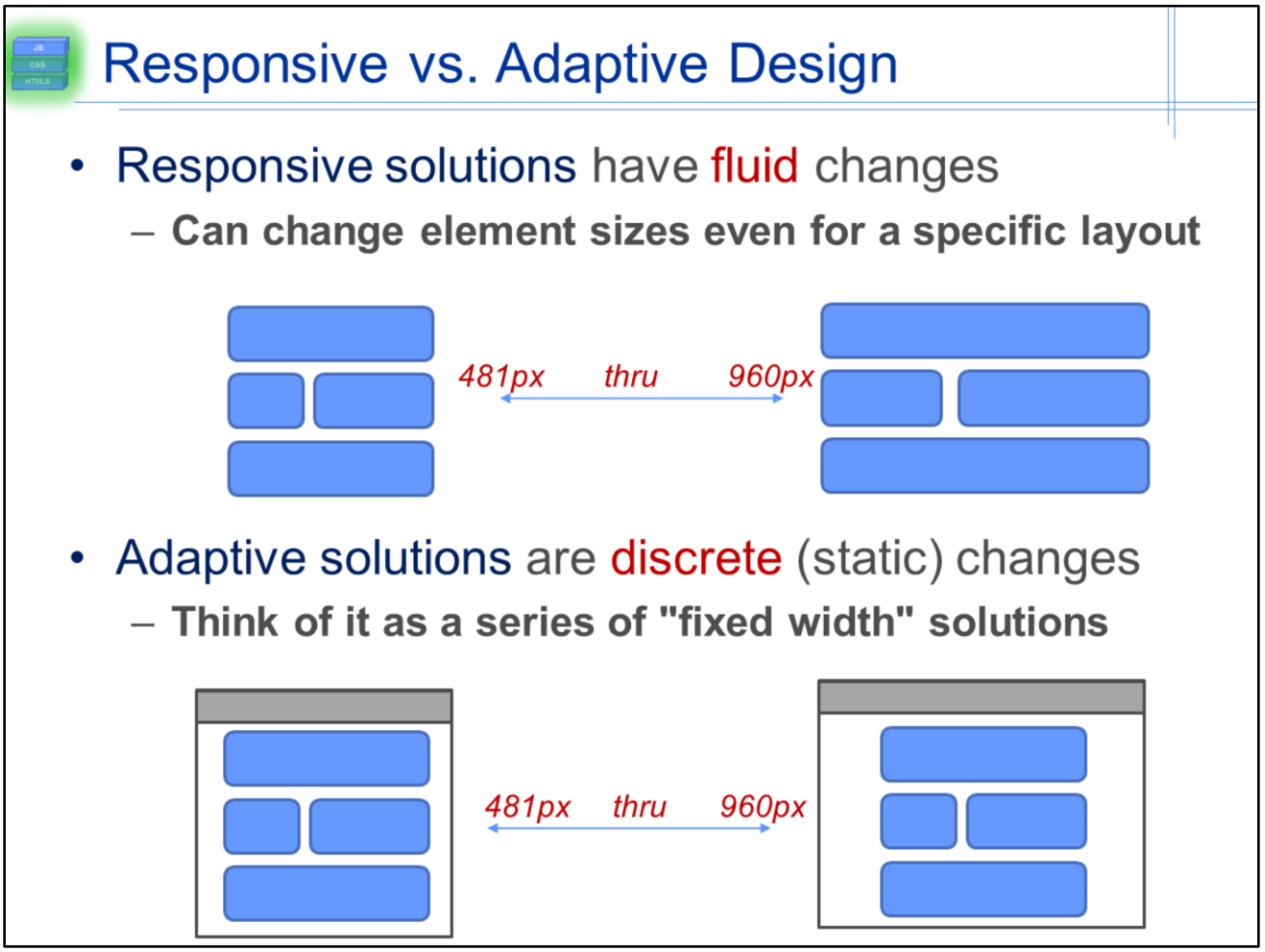




Responsive design is not a single, specific technology. In fact, it is the combination of several techniques (discussed next) that provide for the variations on the solution in the browser.

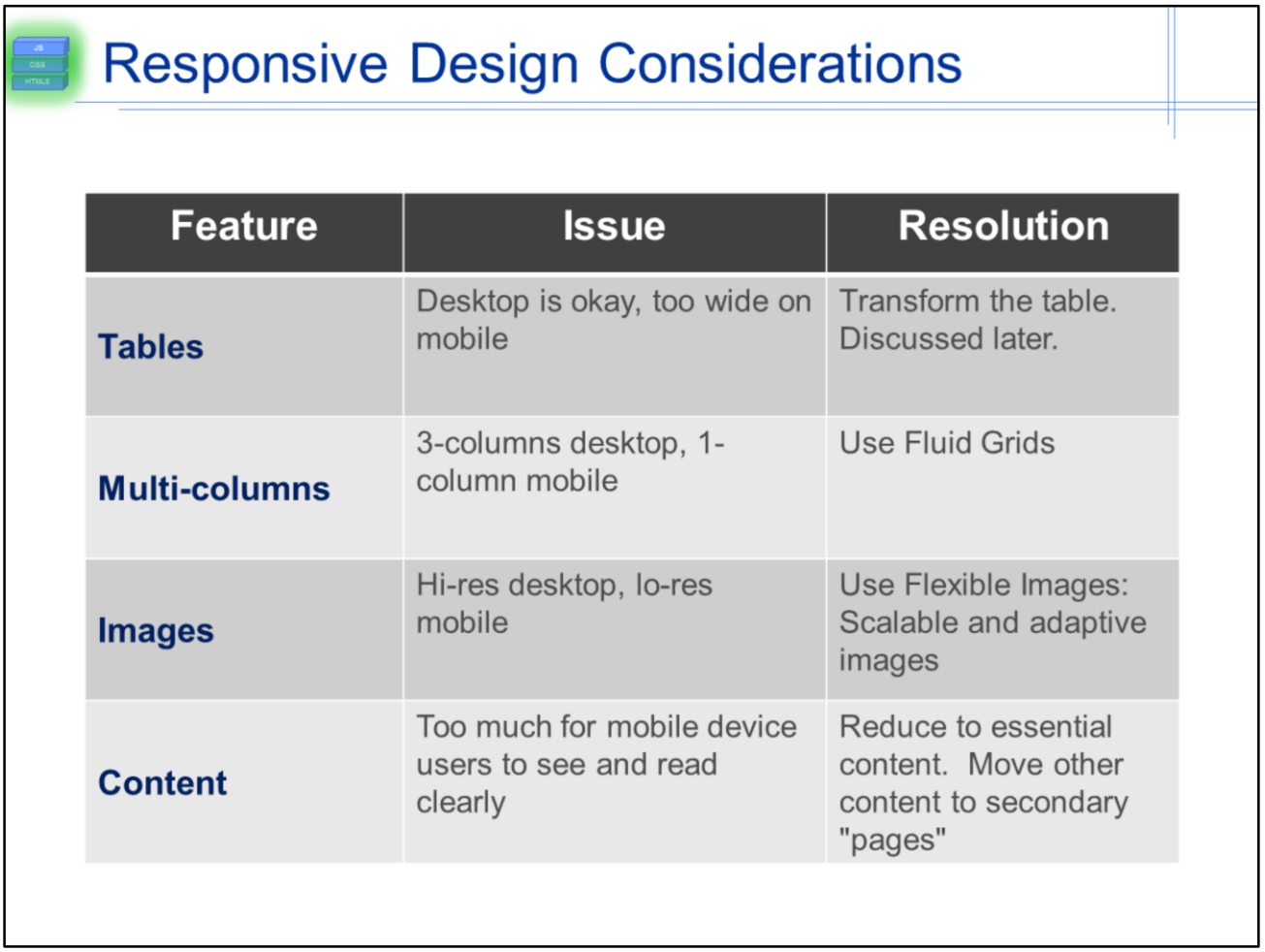


RESS = RWD + Server-side components. See slide 3 slides ahead.

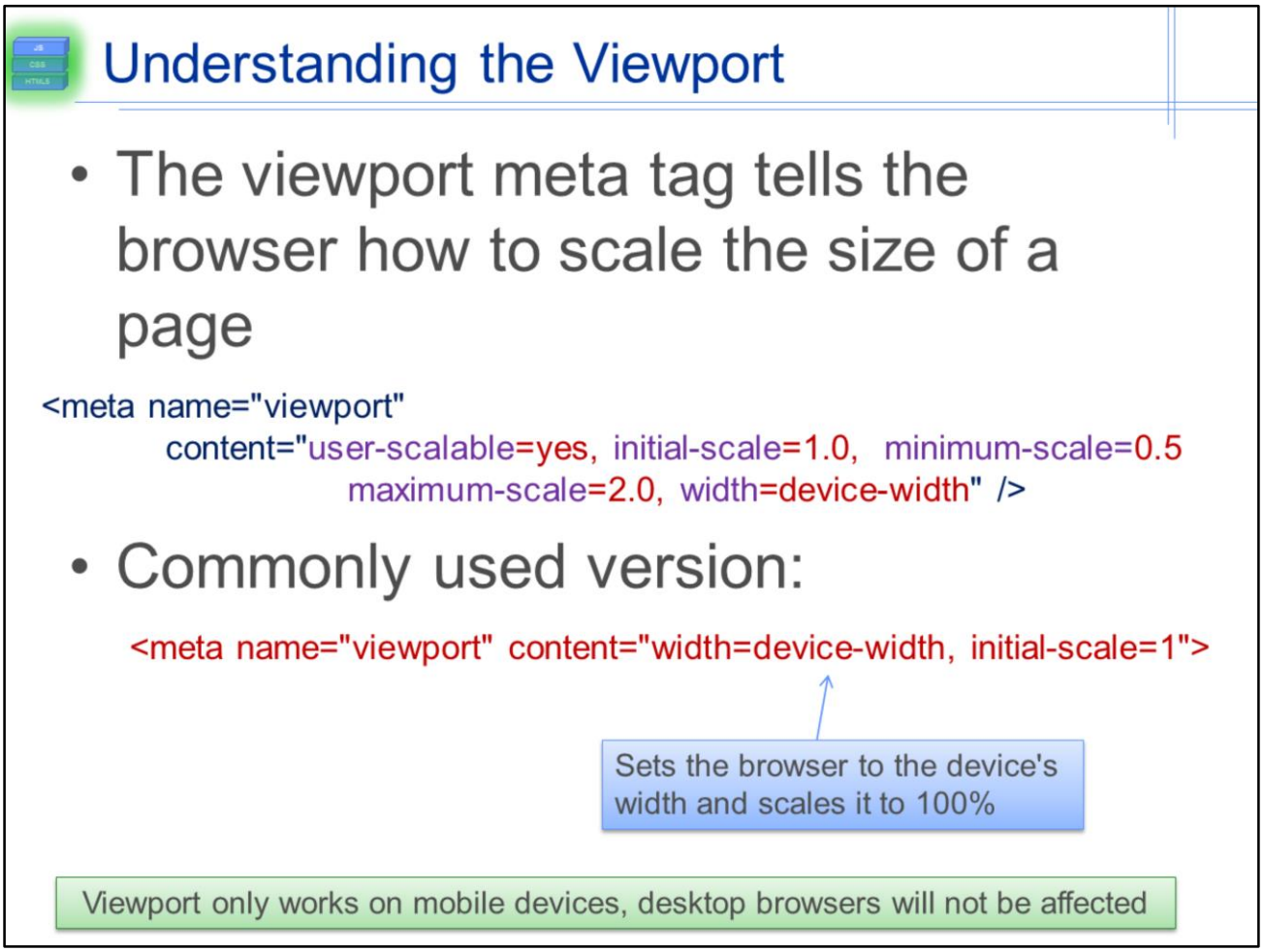


Both techniques will typically use media queries and support multiple layouts. The adaptive approach will "snap" into a particular layout and stay that way even as your browser resizes within the range of that layout. The example above shows what happens to each type of layout in an example (made up) range where the layout doesn't change.

Check out a further example here:
http://liquidapsive.com/

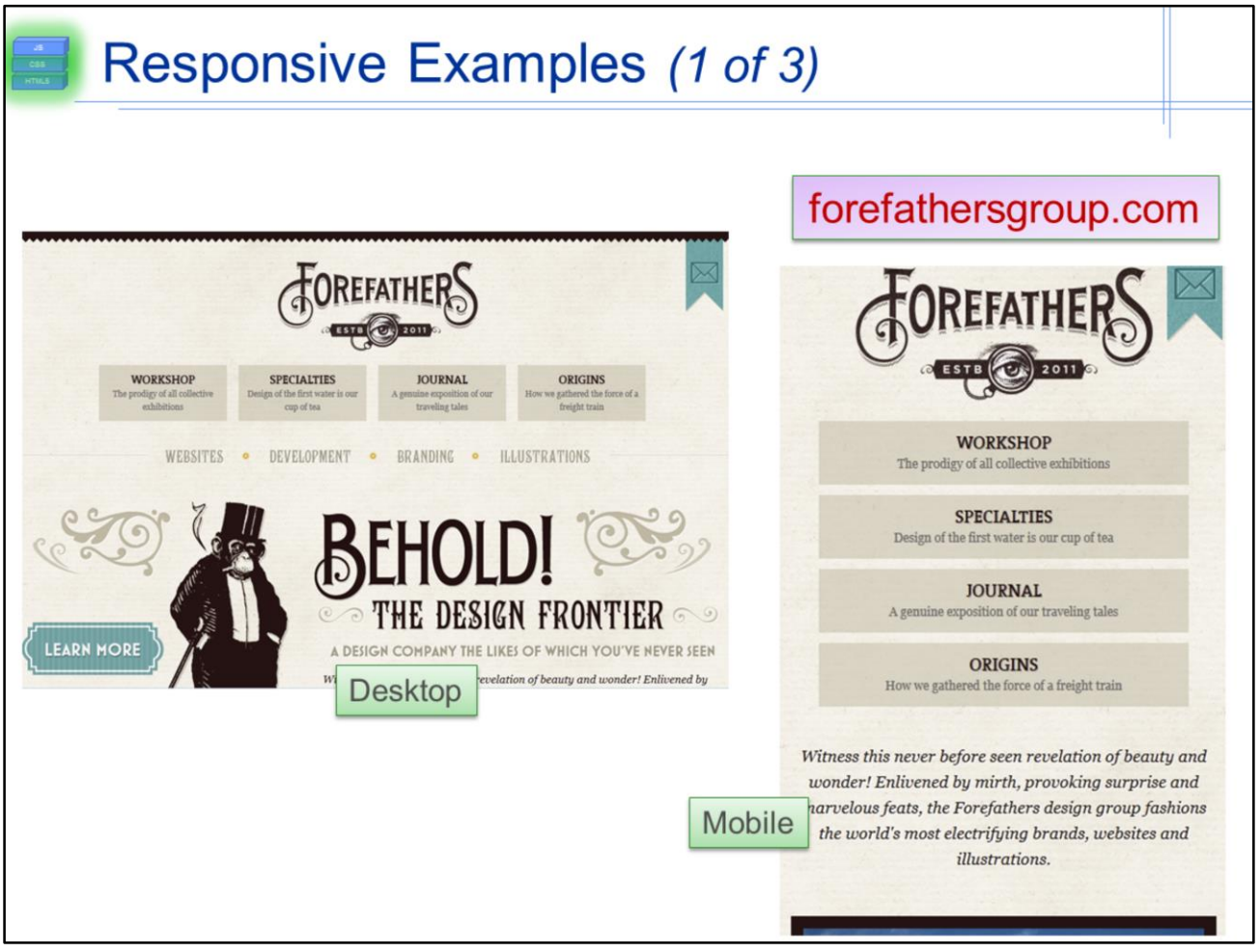


The 3 cornerstones to responsive solutions are: flexible grids (adjustable not fixed width columns), media queries, and flexible images (images that use CSS to resize them by specifying max-width: 100%, which causes them to fill only the size of the container).



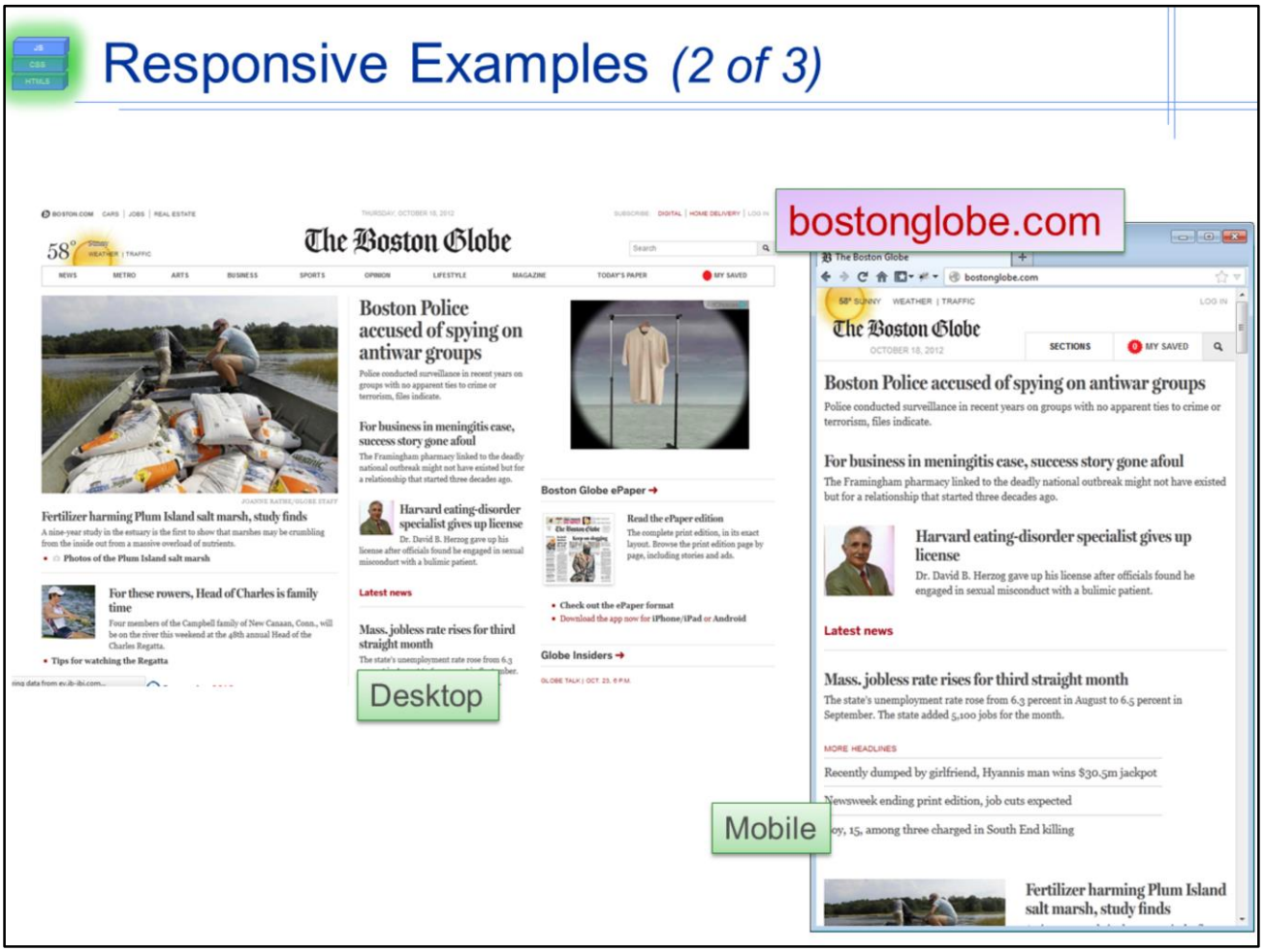
The width can be set to the device-width in most mobile devices now. The initial-scale determines the amount of zoom.

Check out this article for more info:
http://dev.opera.com/articles/an-introduction-to-meta-viewport-and-viewport/

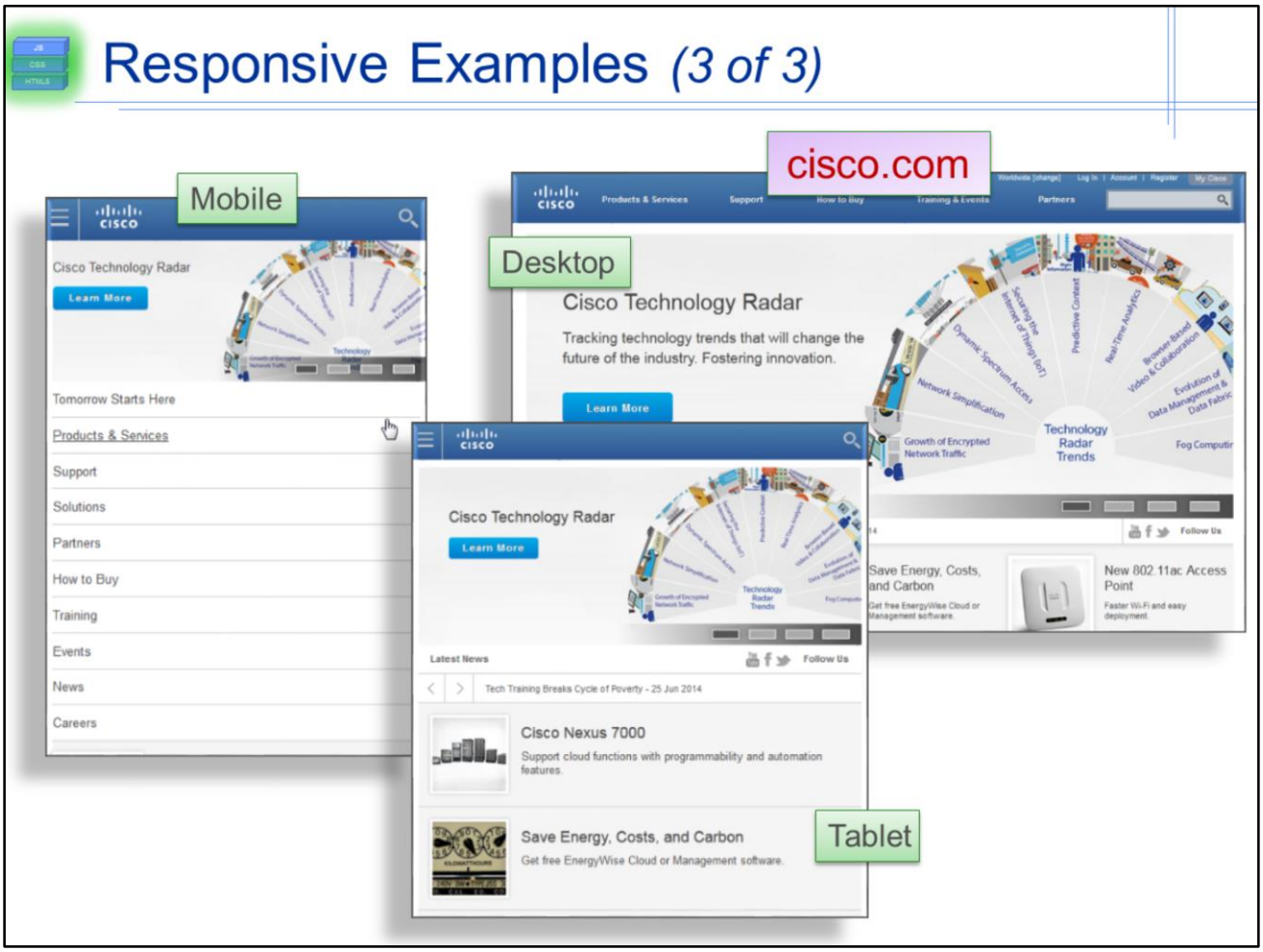


forefathersgroup removes unessential images when content is squeezed down.

Responsive Examples *(2 of 3)*

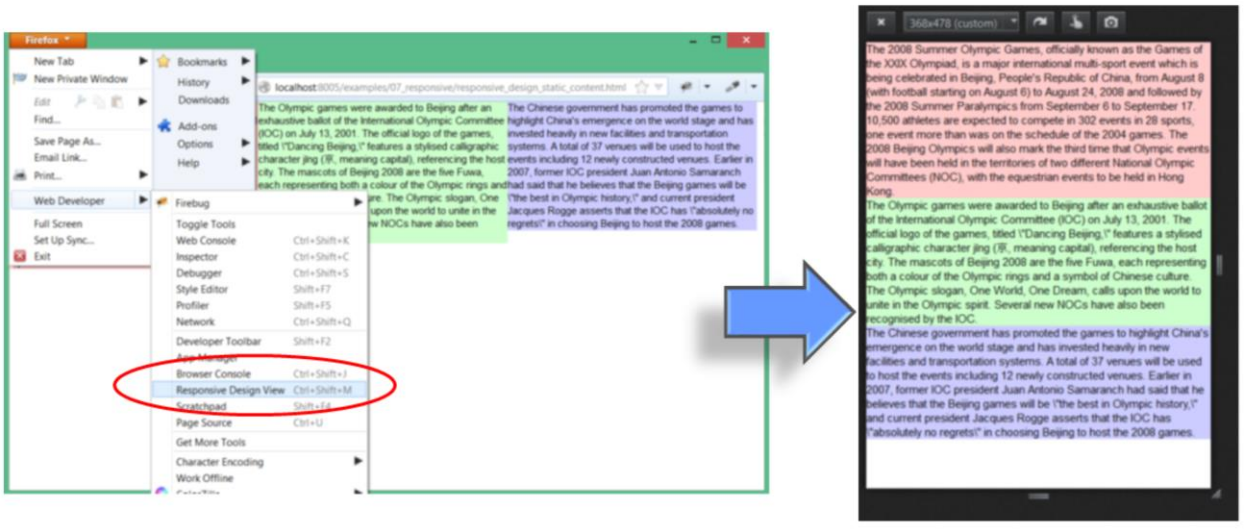


Notice the dynamic layout of columns based on screen resolution.



Firefox Responsive Design View

- Responsive Design View can simulate applications at various sizes

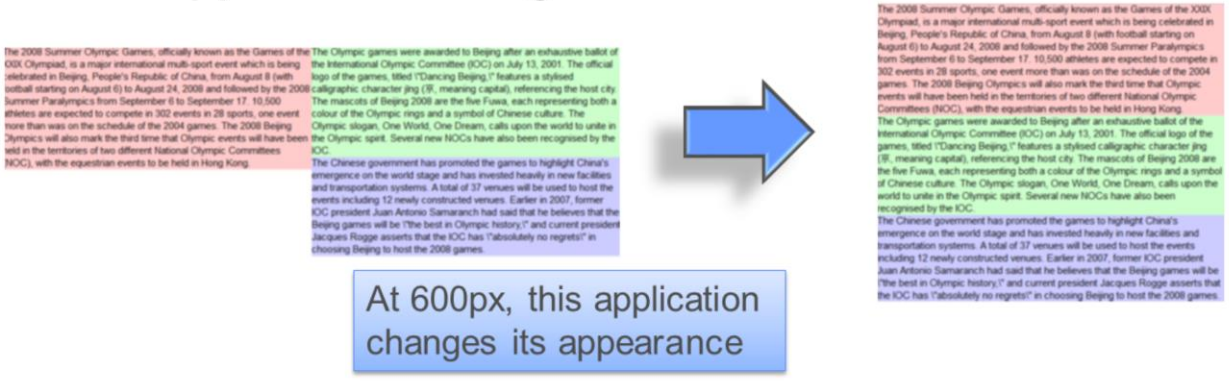


It also rotates from portrait to landscape and simulates touch events

While not a perfect replacement for the real devices, the Responsive Design View can go a long way in providing how an application will look when applied to custom heights and widths.

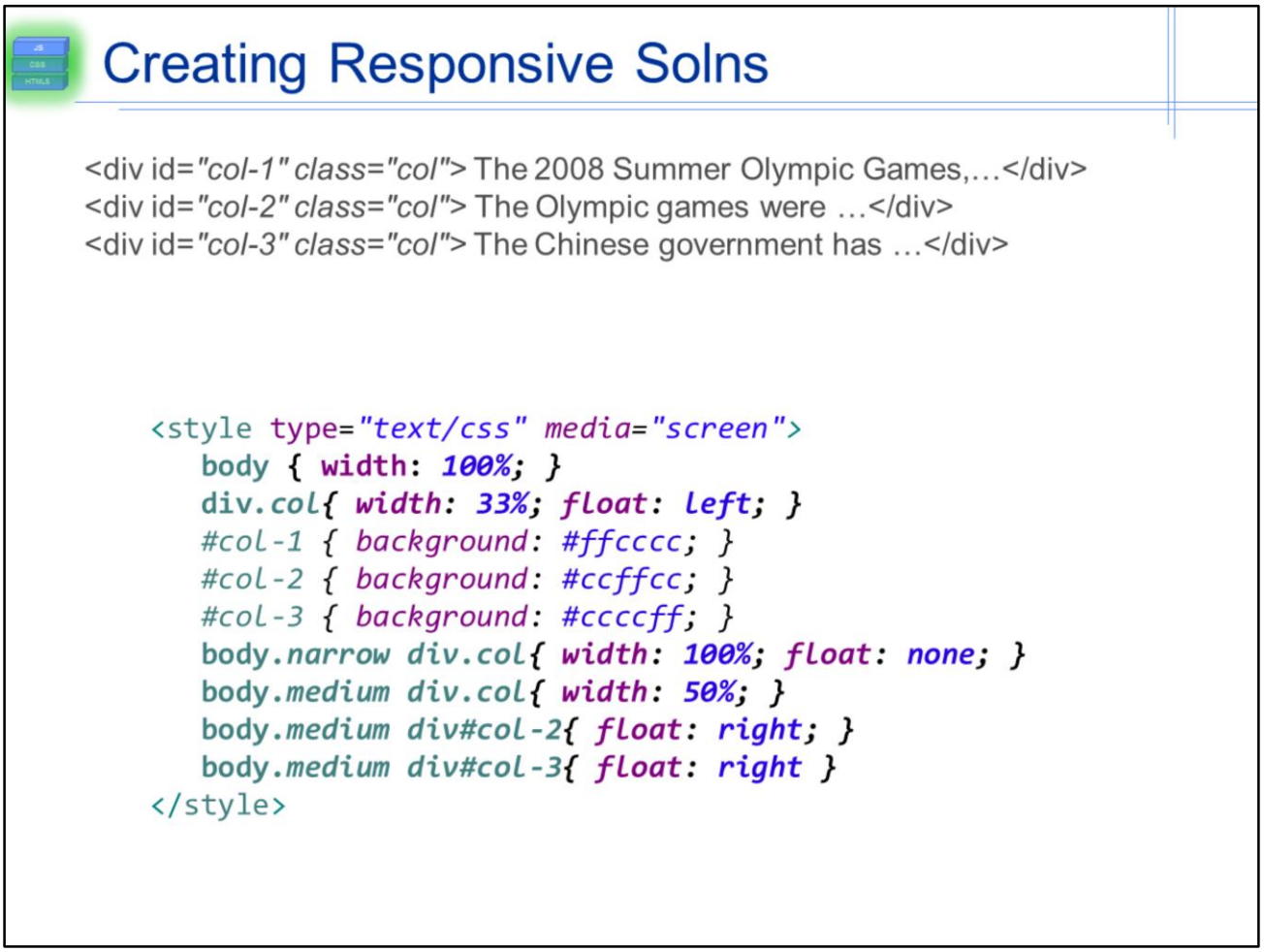
Breakpoints

- *Breakpoints* are pre-defined *pixel widths* at which an application changes occur

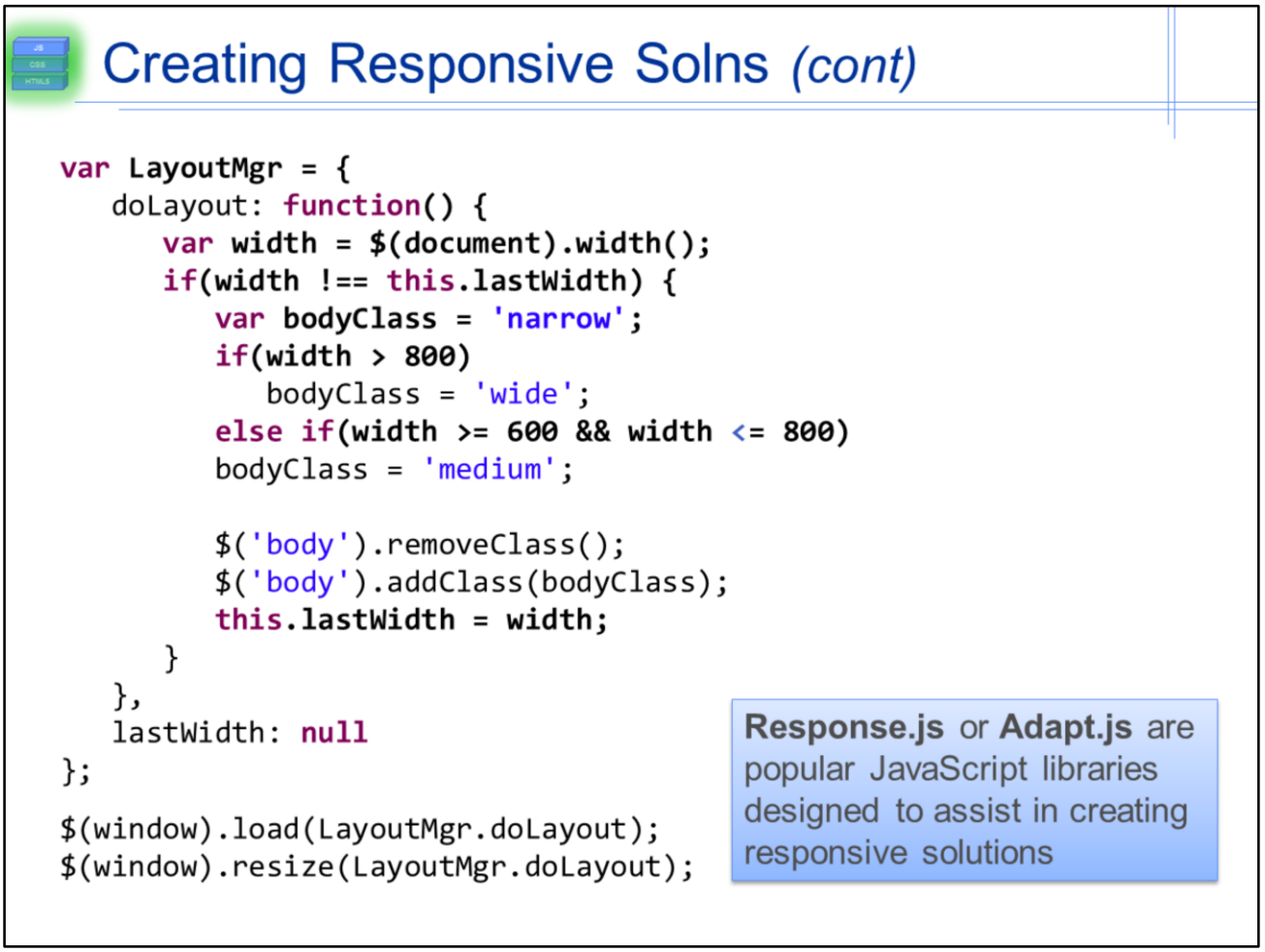


- Breakpoints are determined by the developer
  - These should be based on best look for the application, not based on common device sizes

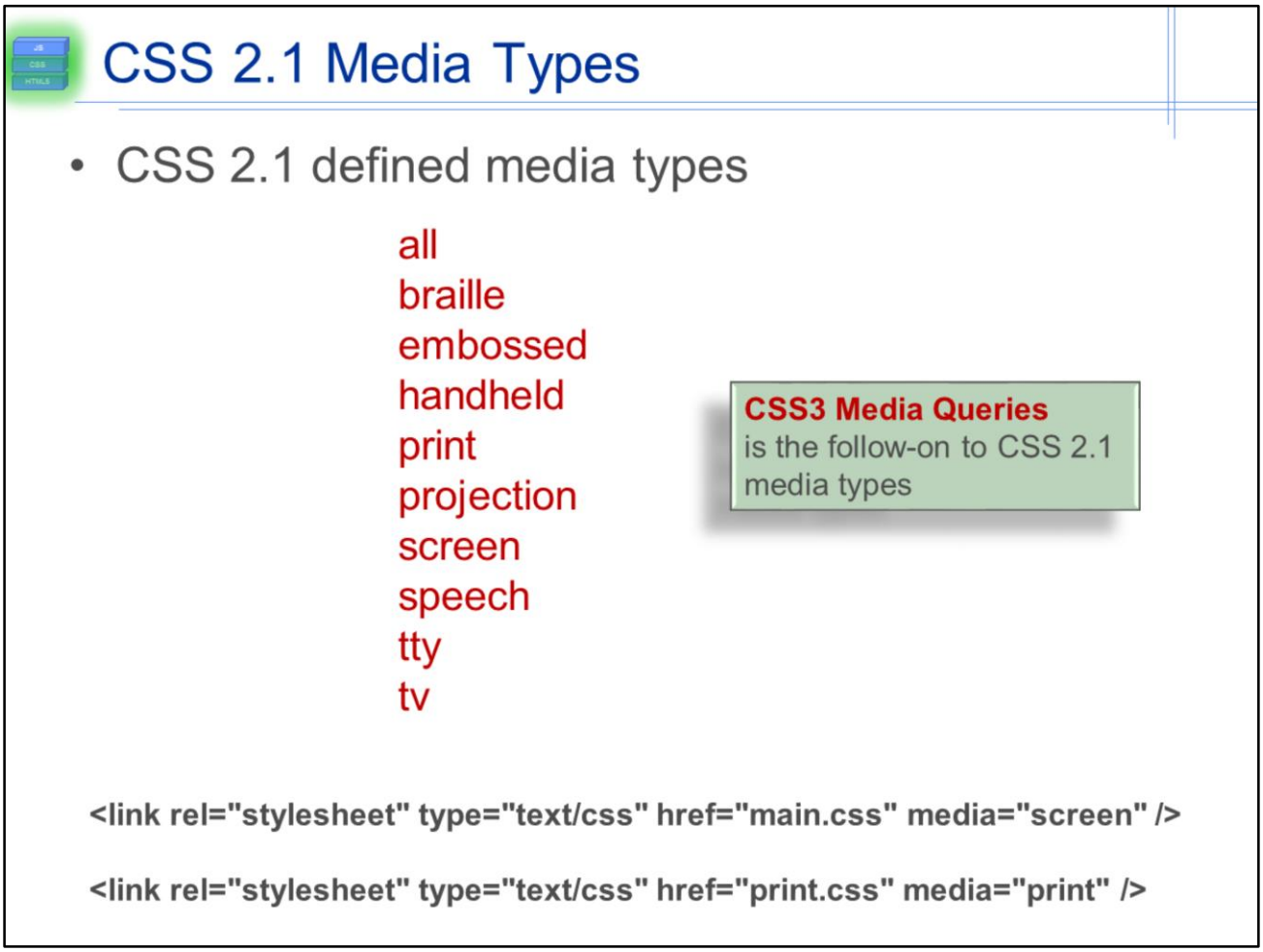
The term is a bit confusing, but breakpoints have nothing to do with the debugging term for breakpoints. Breakpoints should not be based on common device sizes--meaning



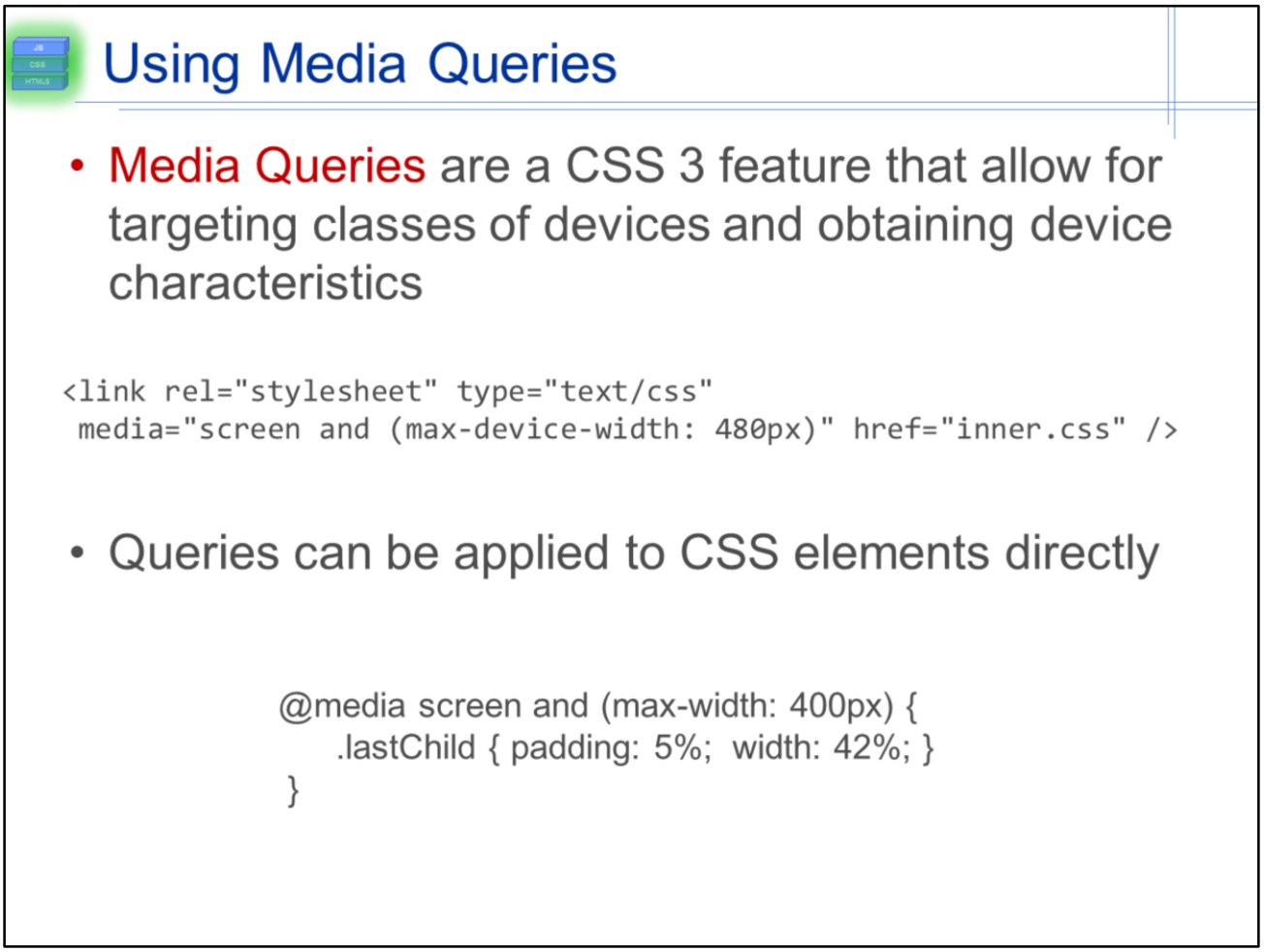
Refer to examples/05_responsive/01_responsive_design_static_content.html



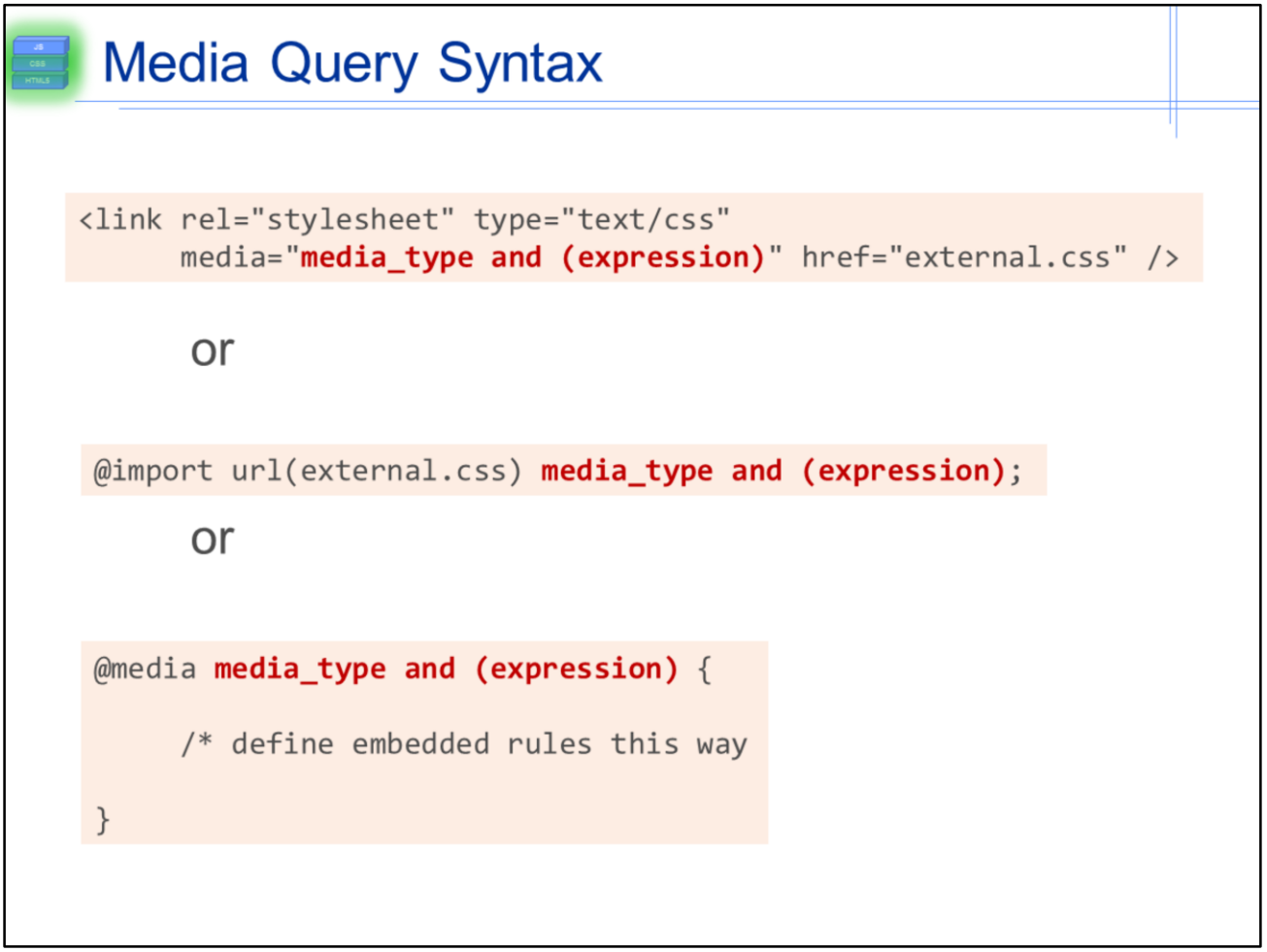
Refer to examples/05_responsive/01_responsive_design_static_content.html



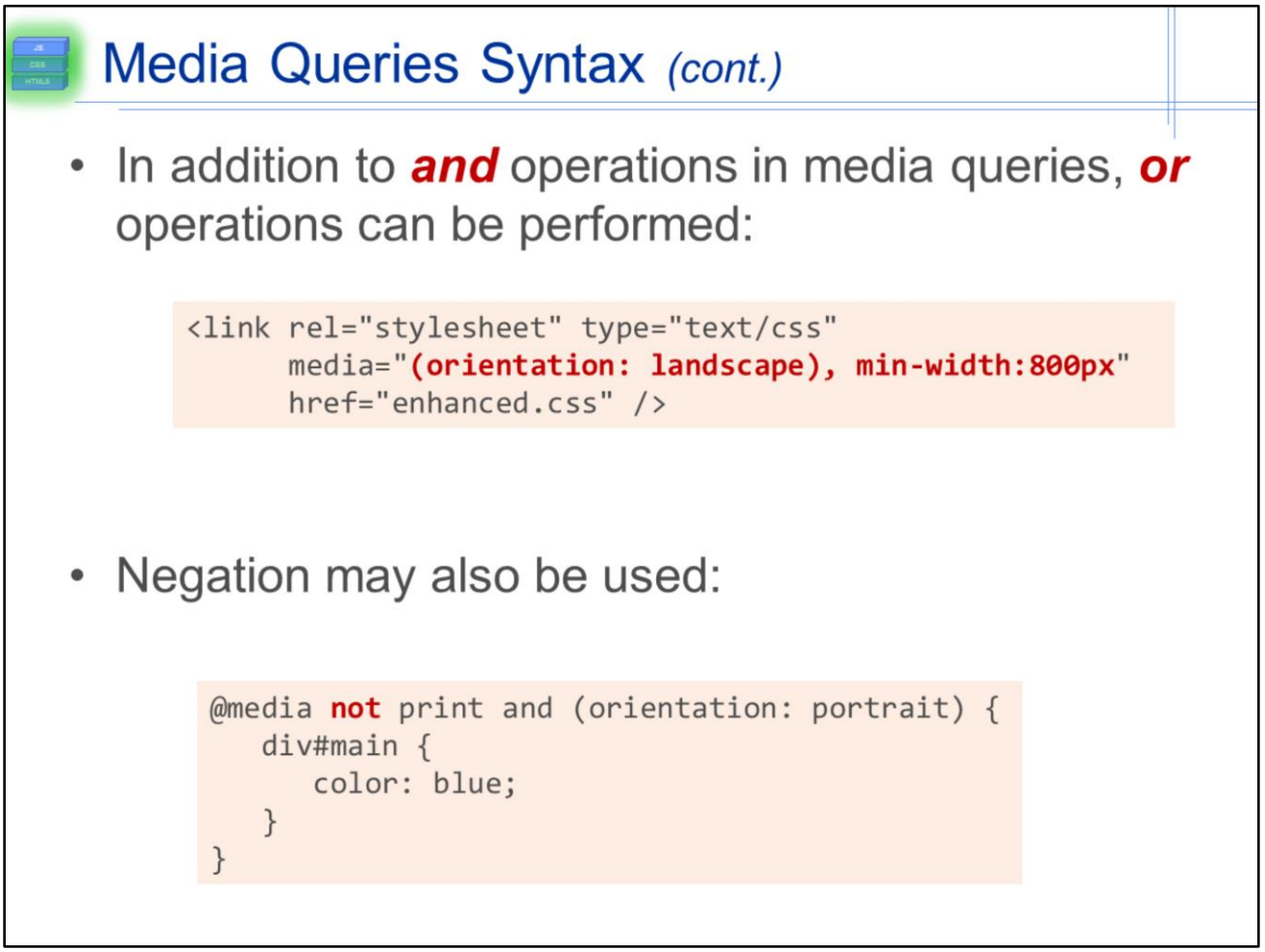
http://www.w3.org/TR/CSS21/media.html



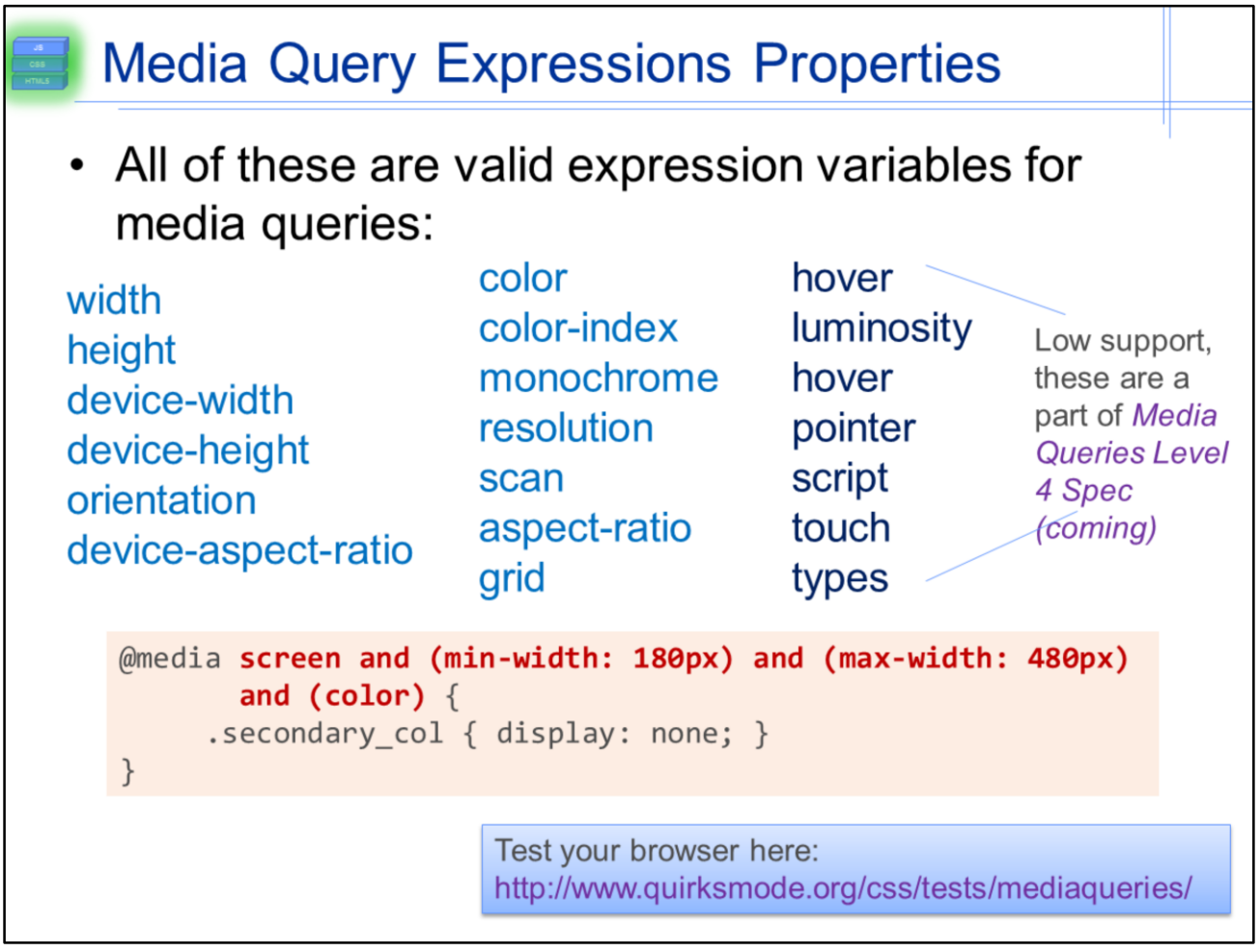
In the media query above, the statement checks the device's max width, if it is less than 480px, load the inner.css file. The second example applies the .lastChild class to HTML elements only if the max-width is no more than 400px (it must also be a screen-based device).



Multiple expressions may be applied by using another 'and (expression)' after the previous one. The first and second forms above apply external css files when the media_type and expressions are met. The third syntax above assumes rules will be embedded within the @media{} braces.



The not and or operators are less commonly used.

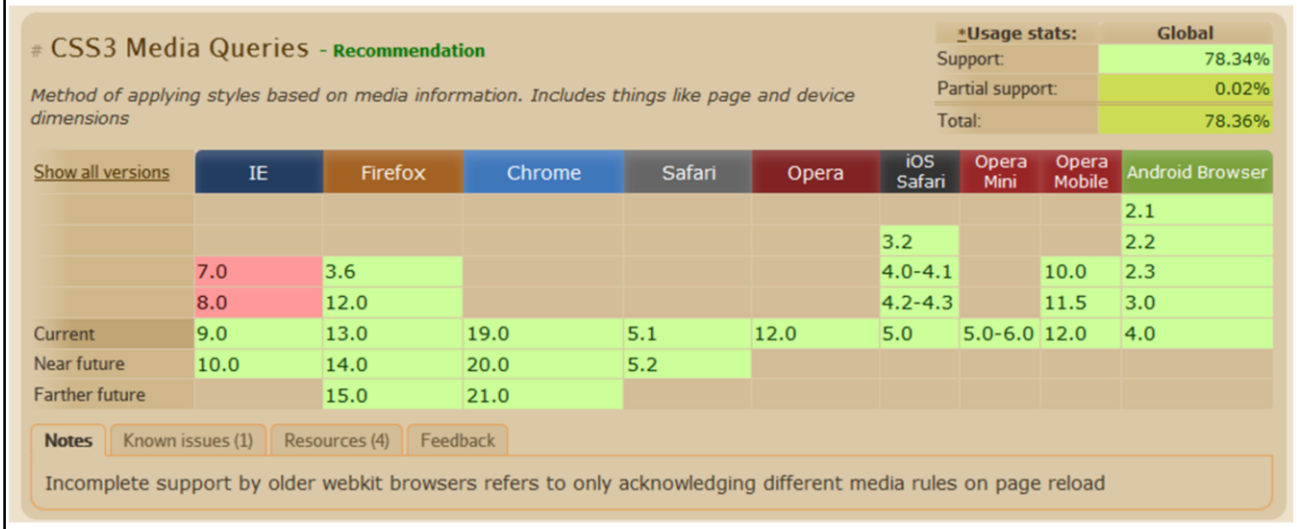


width, height, device-width, device-height, color, resolution all support a min- and max- prefix as well.
orientation can be 'portrait' or 'landscape'
color reports whether or not it supports color or how many bits depth color is supported. (e.g. @media all and (color) {….})

resolution supports min-resolution, max-resolution. Units for this can be dpi (dots per inch)

CSS Media Queries Level 4 Spec: http://dev.w3.org/csswg/mediaqueries4/

Media Query Device Support



CSS3 Media Queries - Recommendation

Method of applying styles based on media information. Includes things like page and device dimensions

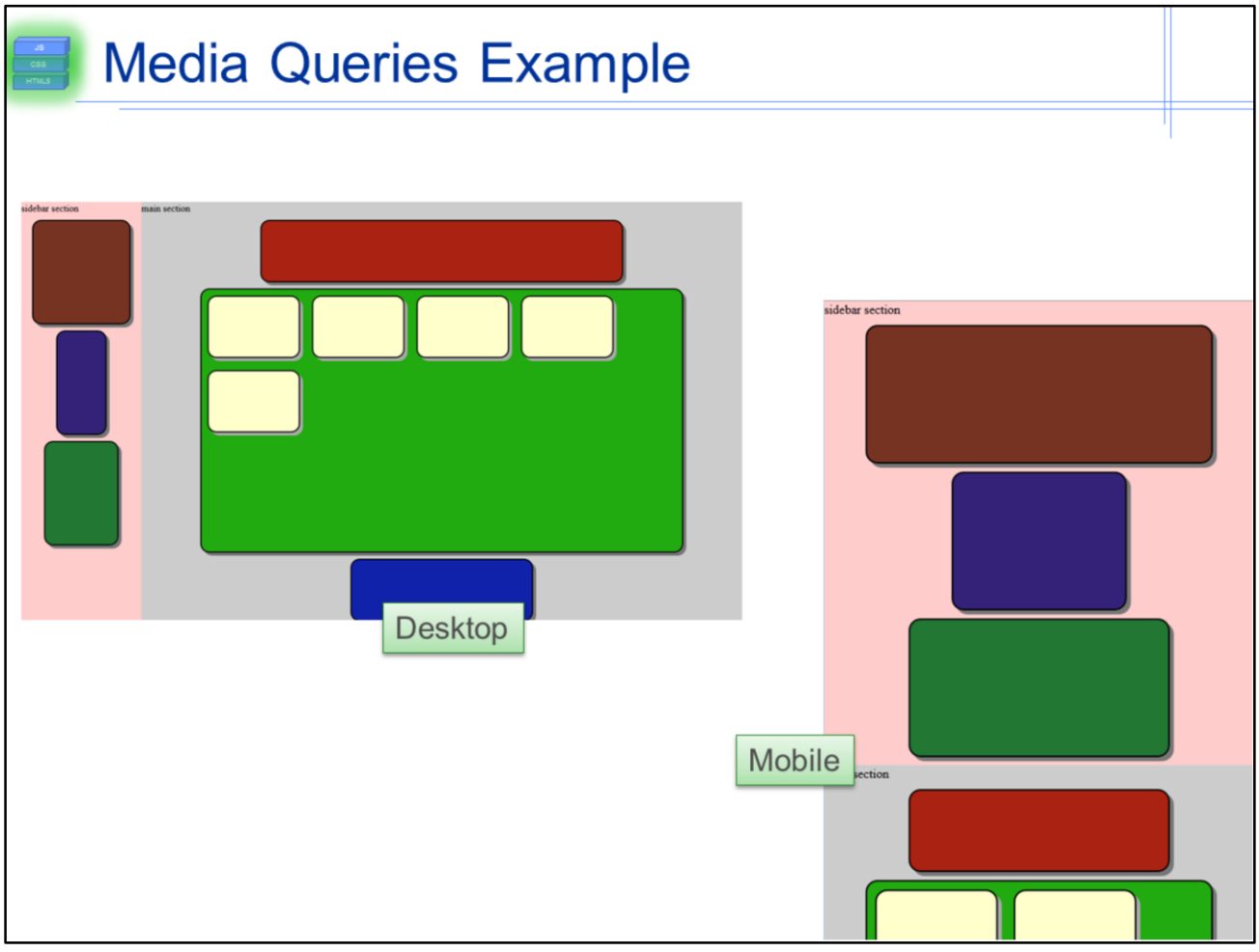
*Usage stats:	Global
Support:	78.34%
Partial support:	0.02%
Total:	78.36%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
									2.1
						3.2			2.2
	7.0	3.6				4.0-4.1		10.0	2.3
	8.0	12.0				4.2-4.3		11.5	3.0
Current	9.0	13.0	19.0	5.1	12.0	5.0	5.0-6.0	12.0	4.0
Near future	10.0	14.0	20.0	5.2					
Farther future		15.0	21.0						

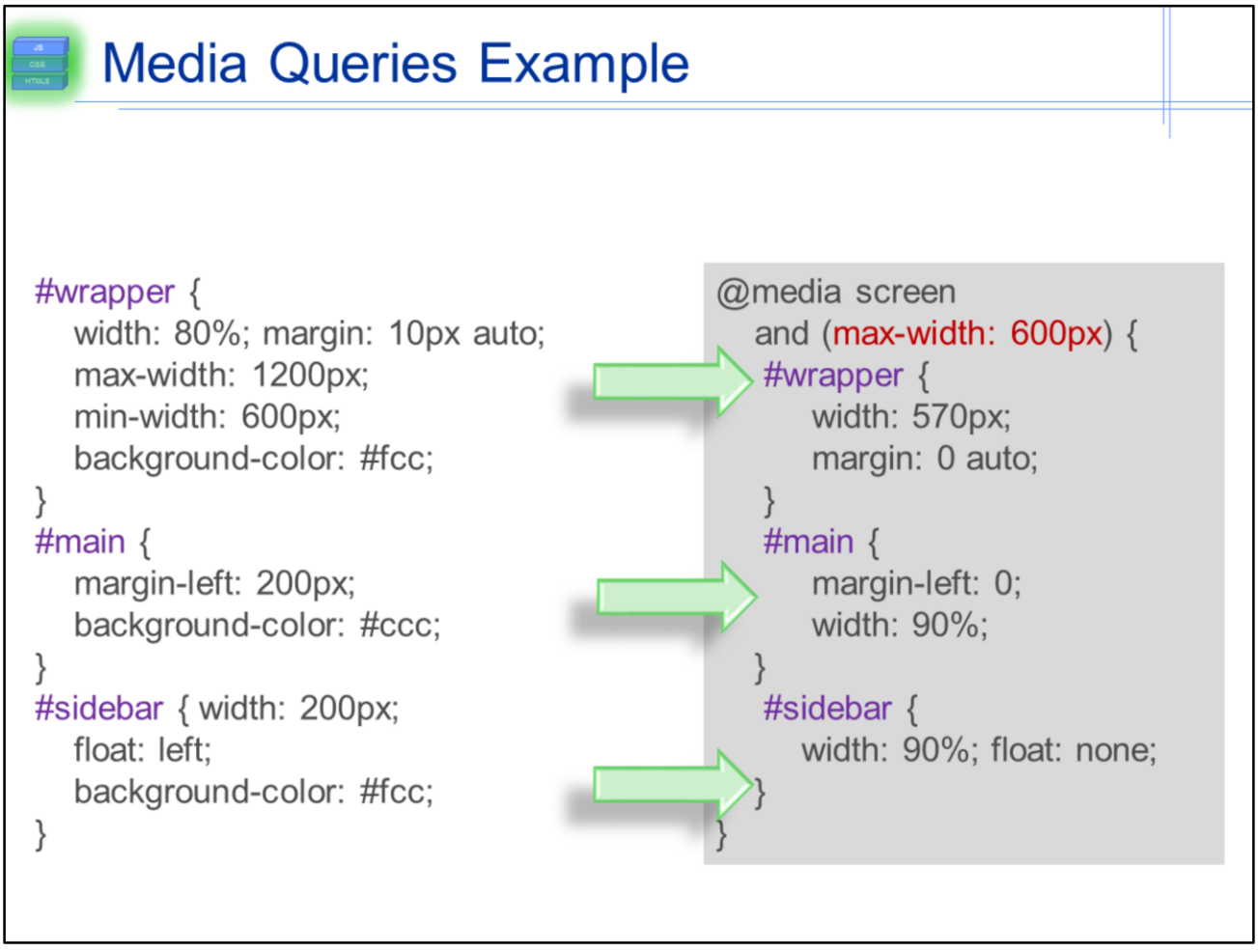
Notes | Known issues (1) | Resources (4) | Feedback

Incomplete support by older webkit browsers refers to only acknowledging different media rules on page reload

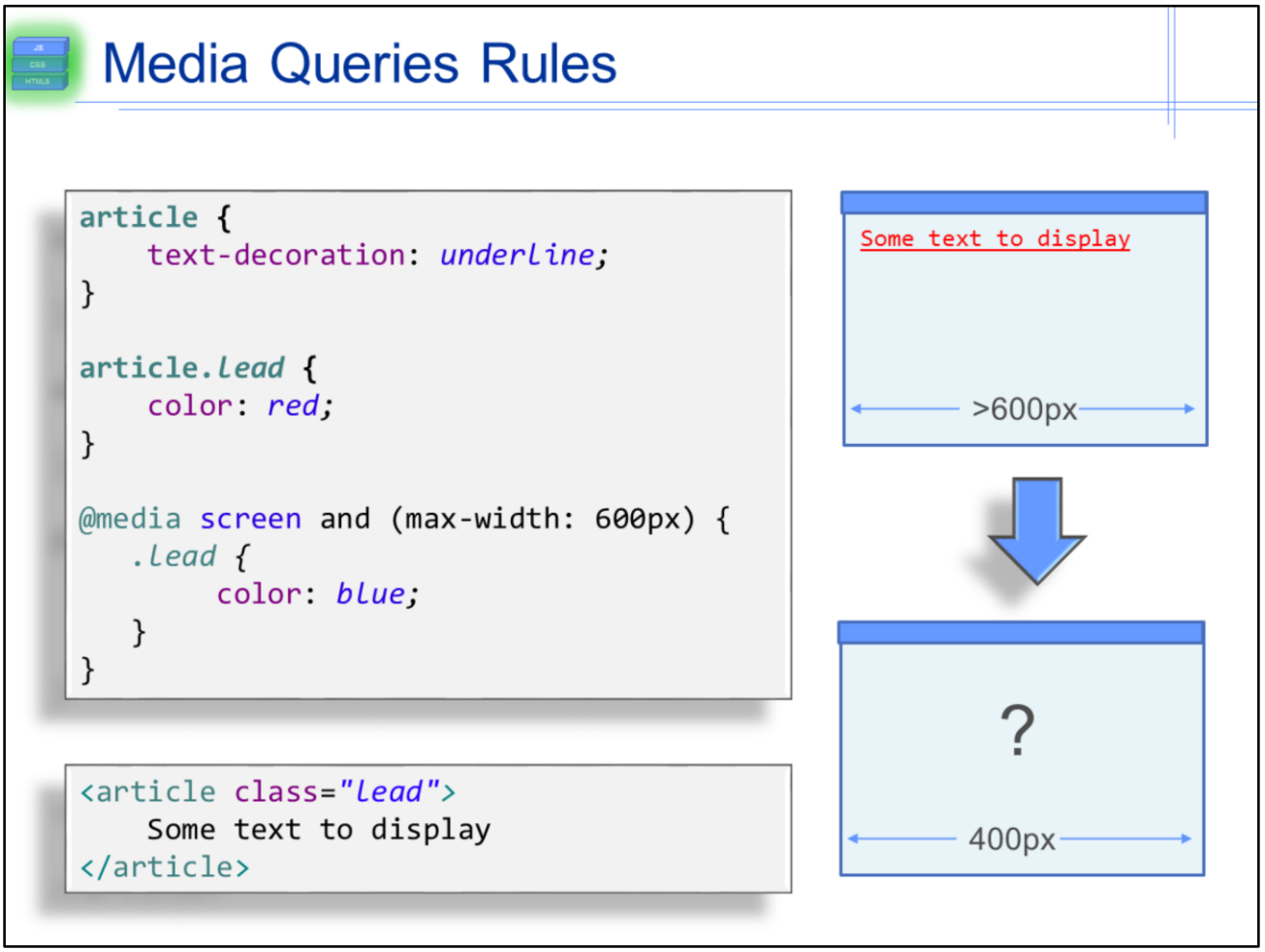
Media queries are supported by all modern devices. Only legacy IE browsers will not. However, this is generally okay as IE legacy browsers are usually run within standard desktop/laptop devices which are often the default implementations that do not require a media query-based solution.



examples/05_responsive/02_media_queries.html

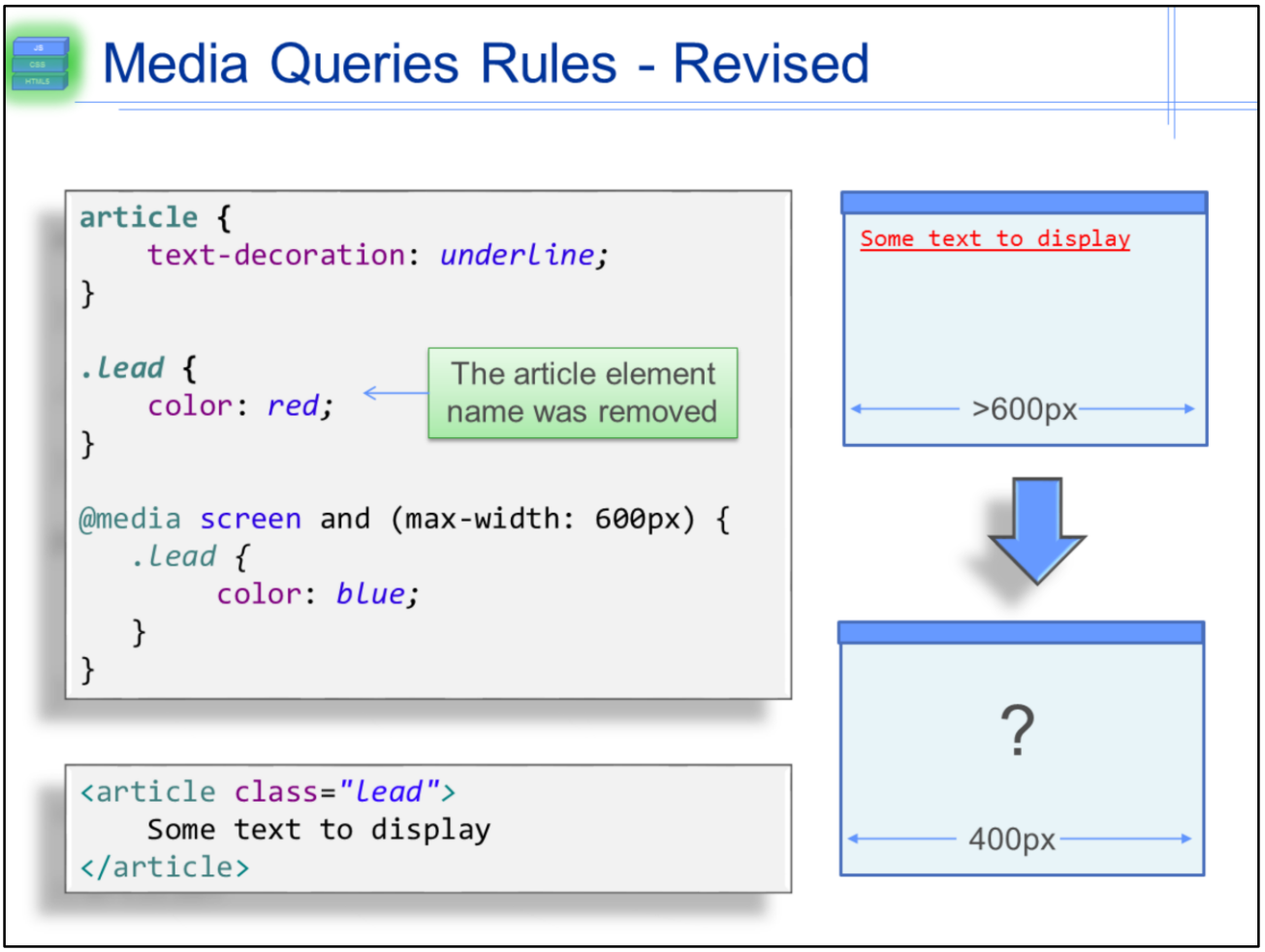


examples/05_responsive/02_media_queries.html



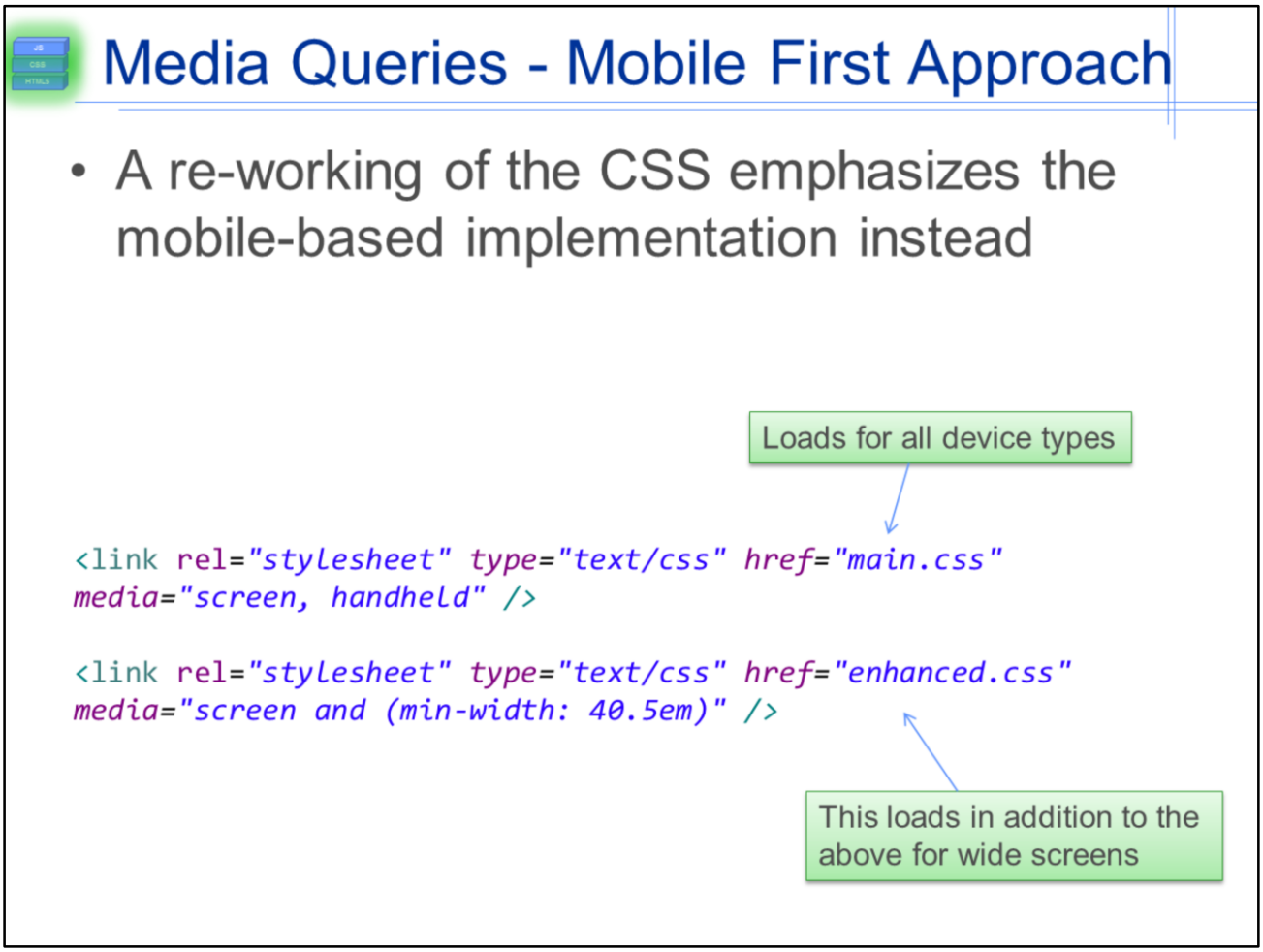
examples/05_responsive/03_media_queries_rules.html

The media query MUST follow the same rules of CSS. In this case, when the browser is 1000px, the default CSS kicks-in but the media query doesn't get evaluated. Therefore, the text is rendered in red and underlined. When the device is less than 600px, the media query is valid. But the results are perhaps not what you think. The default CSS is still valid, therefore, the article.lead selector is still more specific than the .lead selector within the query. Therefore the color red will override the color blue.

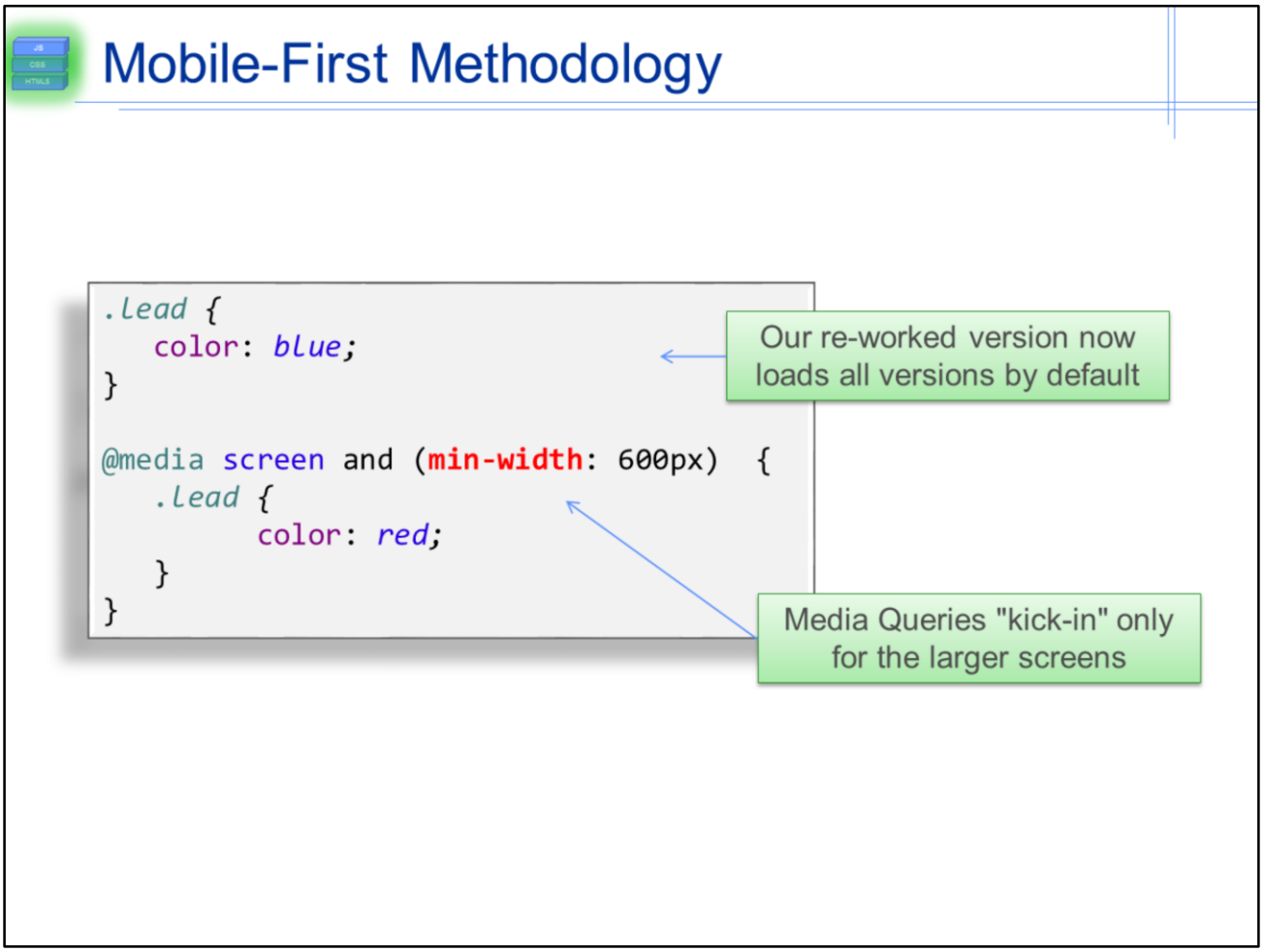


examples/05_responsive/03b_media_queries_rules.html

Here the value changes to blue because both .lead rules are evenly weighted + the .lead rule came later in the document.



While it may seem subtle, the default CSS will now apply to all versions, while only enhanced stuff will be loaded in the desktop environment.



examples/05_responsive/04_media_queries_mobile_first.html

The change in emphasis is subtle but loads default values for all devices first, then layers CSS for additional devices on top of this.

More Media Queries

```
nav ul li span {
    background-color: #ffc73d;
    border: 1px solid #aa5522;
    padding: 5px;
    display: block;
}
.products li {
    float: left;
    width: 50%;
}
@media screen and (min-width: 600px)  {
    .products li {
        width: 33.3333332%;
    }
}
@media screen and (min-width: 800px)  {
    .products li {
        width: 16.6666667%;
    }
}
```

```
<nav id="main-nav">
    <ul class="products">
        <li><span>Item 1</span></li>
        <li><span>Item 2</span></li>
        <li><span>Item 3</span></li>
        <li><span>Item 4</span></li>
        <li><span>Item 5</span></li>
        <li><span>Item 6</span></li>
    </ul>
</nav>
```

Item 1	Item 2
Item 3	Item 4
Item 5	Item 6

Item 1	Item 2	Item 3
Item 4	Item 5	Item 6

Item 1	Item 2	Item 3	Item 4	Item 5	Item 6

examples/05_responsive/05_media_queries_evolving.html

This version shows 3 different layouts that change at the 600 and 800px breakpoint levels. The media query has the capability to completely change the look of an app short of actually changing the DOM.

Media Queries in JavaScript

- To perform media-queries dynamically within JavaScript use window.matchMedia()

matchMedia - Working Draft

*Usage stats: Global

Support: 80.63%

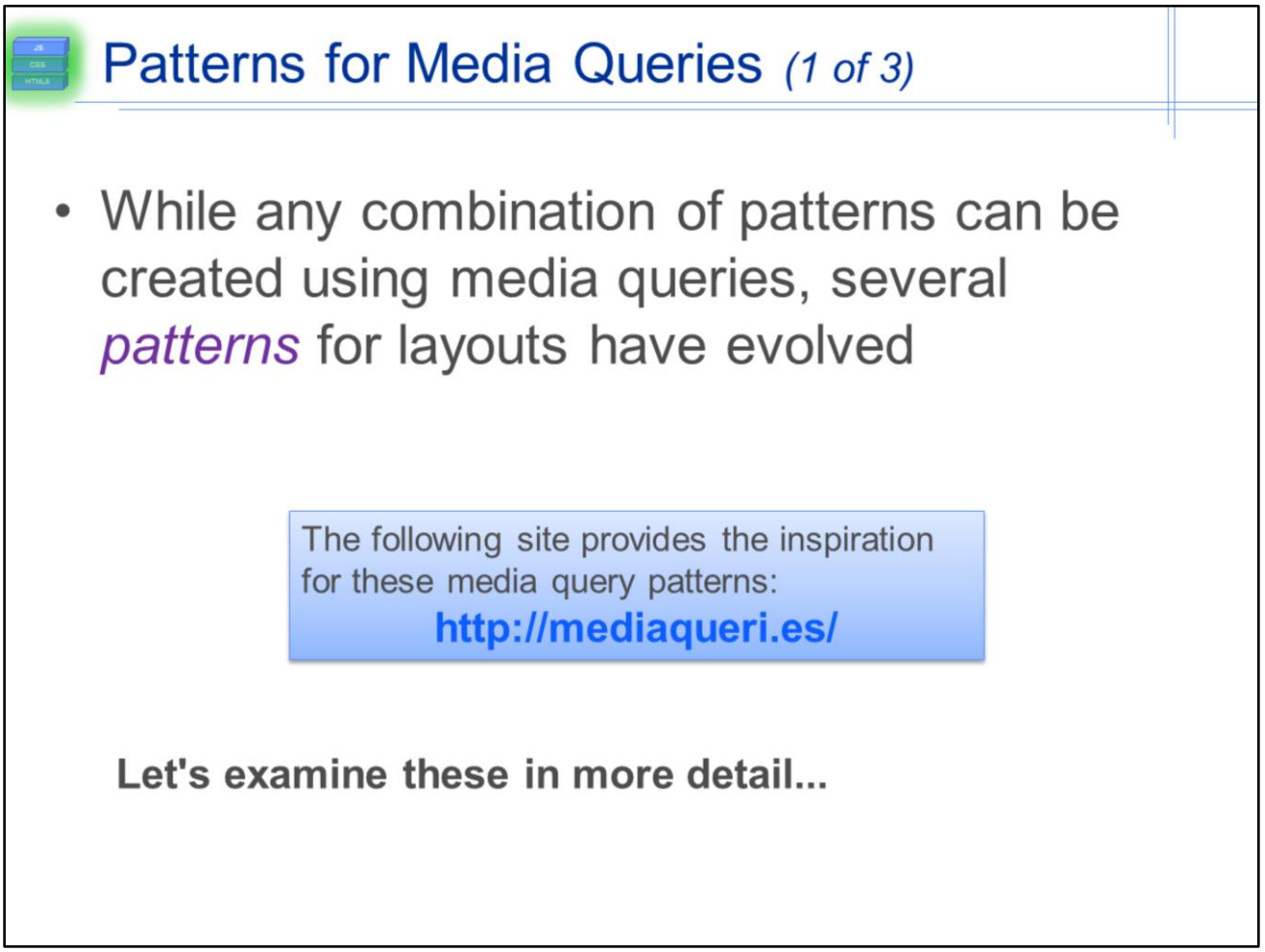
API for finding out whether or not a media query applies to the document.

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1		
								2.2		
						3.2		2.3		
						4.0-4.1		3.0		
	8.0					4.2-4.3		4.0		
	9.0	28.0	33.0			5.0-5.1		4.1		
	10.0	29.0	34.0			6.0-6.1		4.2-4.3	7.0	
Current	11.0	30.0	35.0	7.0	22.0	7.0-7.1	5.0-7.0	4.4	10.0	10.0
Near future		31.0	36.0	8.0	23.0	8.0		4.4.3		
Farther future		32.0	37.0		24.0					
3 versions ahead		33.0	38.0							

Notes | Known issues (0) | Resources (5) | Feedback | Edit on GitHub

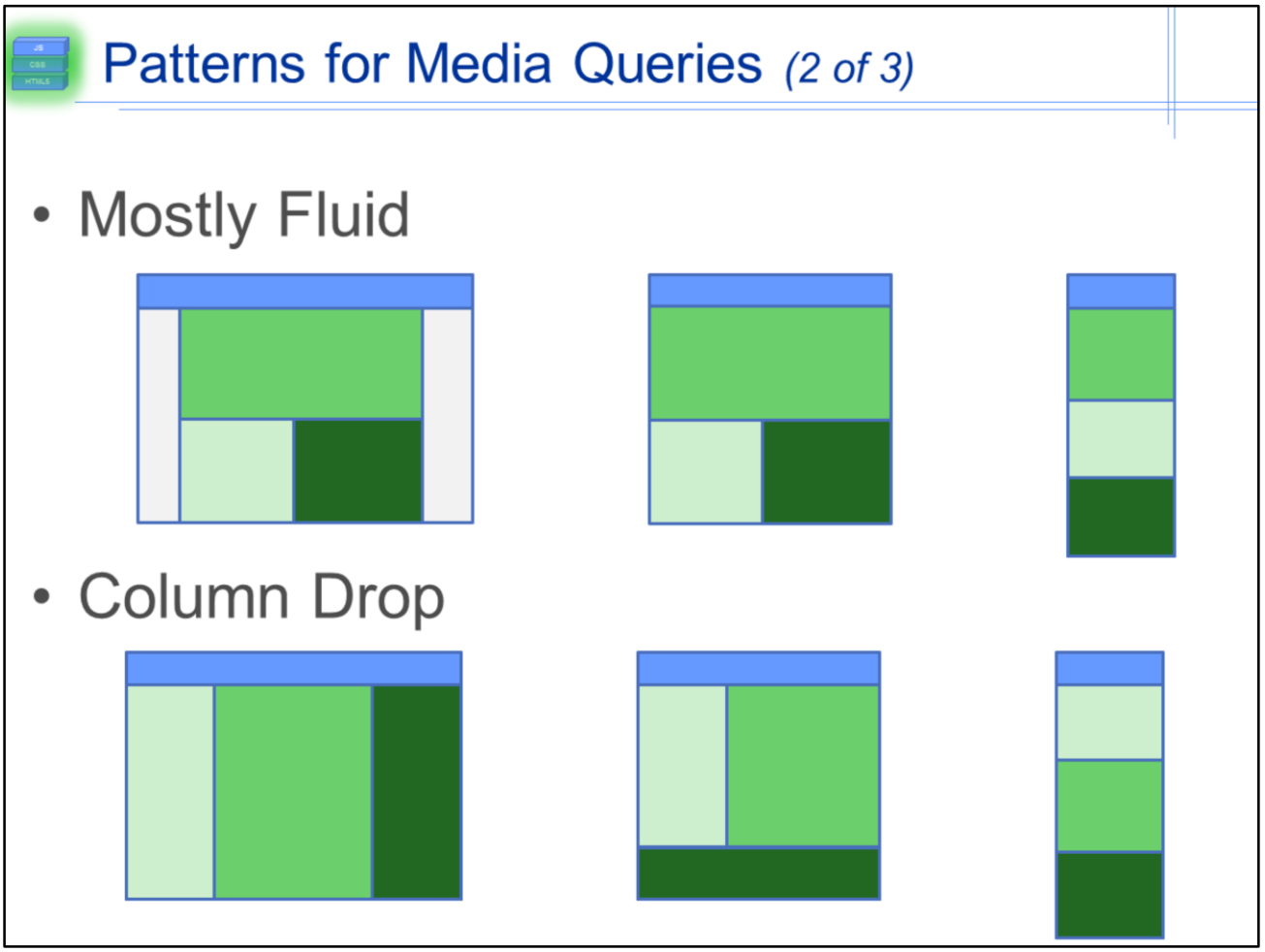
No notes

matchMedia() is a way to perform Media Query checks from within your JavaScript. It works the same way the CSS except within JavaScript.



The patterns mentioned here give ideas on how sites can change responsively. The patterns here were named by Luke Wroblewski, author of the recommended book: "Mobile First" development. These patterns are listed in order of popularity.

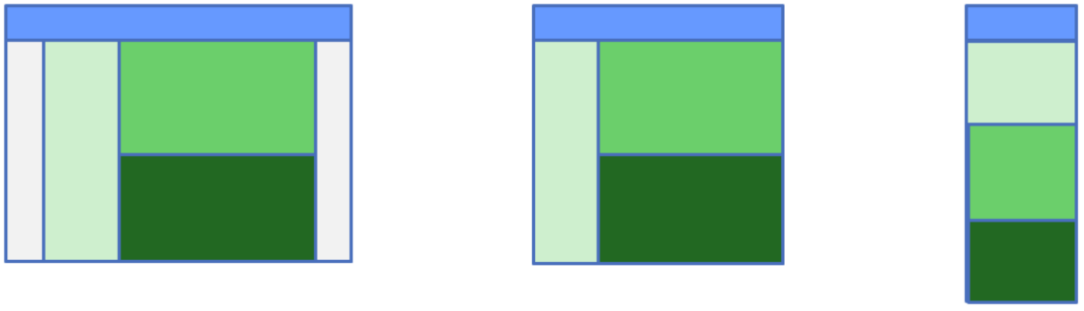
For the following patterns, the site: http://mediaqueri.es was used as a reference.



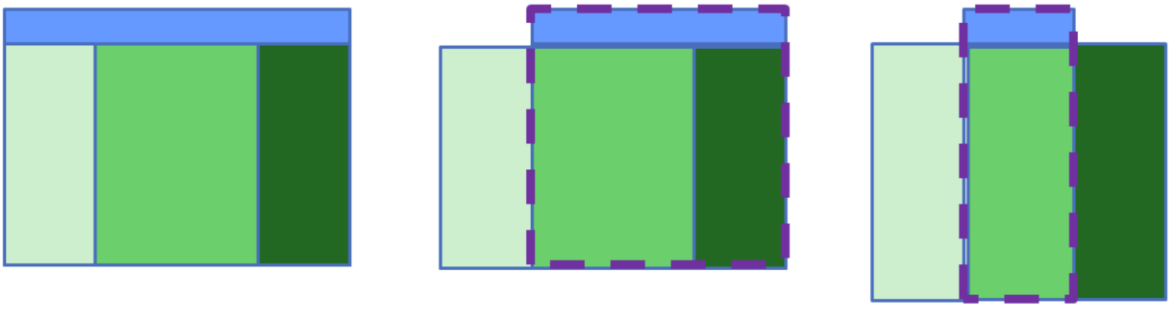
The mostly fluid solution is the most popular and one of the easiest to implement. It simply changes sizes at the lowest device width's into a stacked container.

Patterns for Media Queries *(3 of 3)*

- Layout Shifter



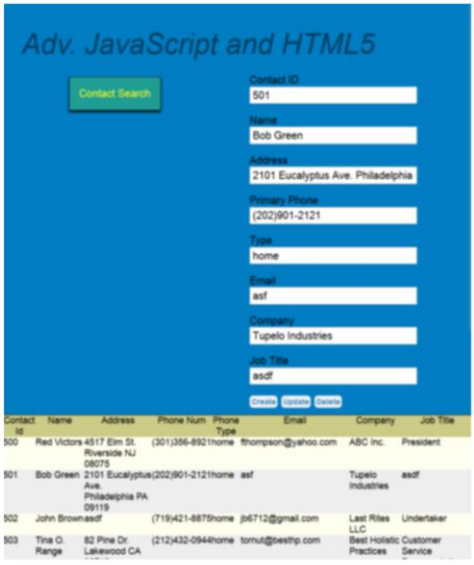
- Off Canvas



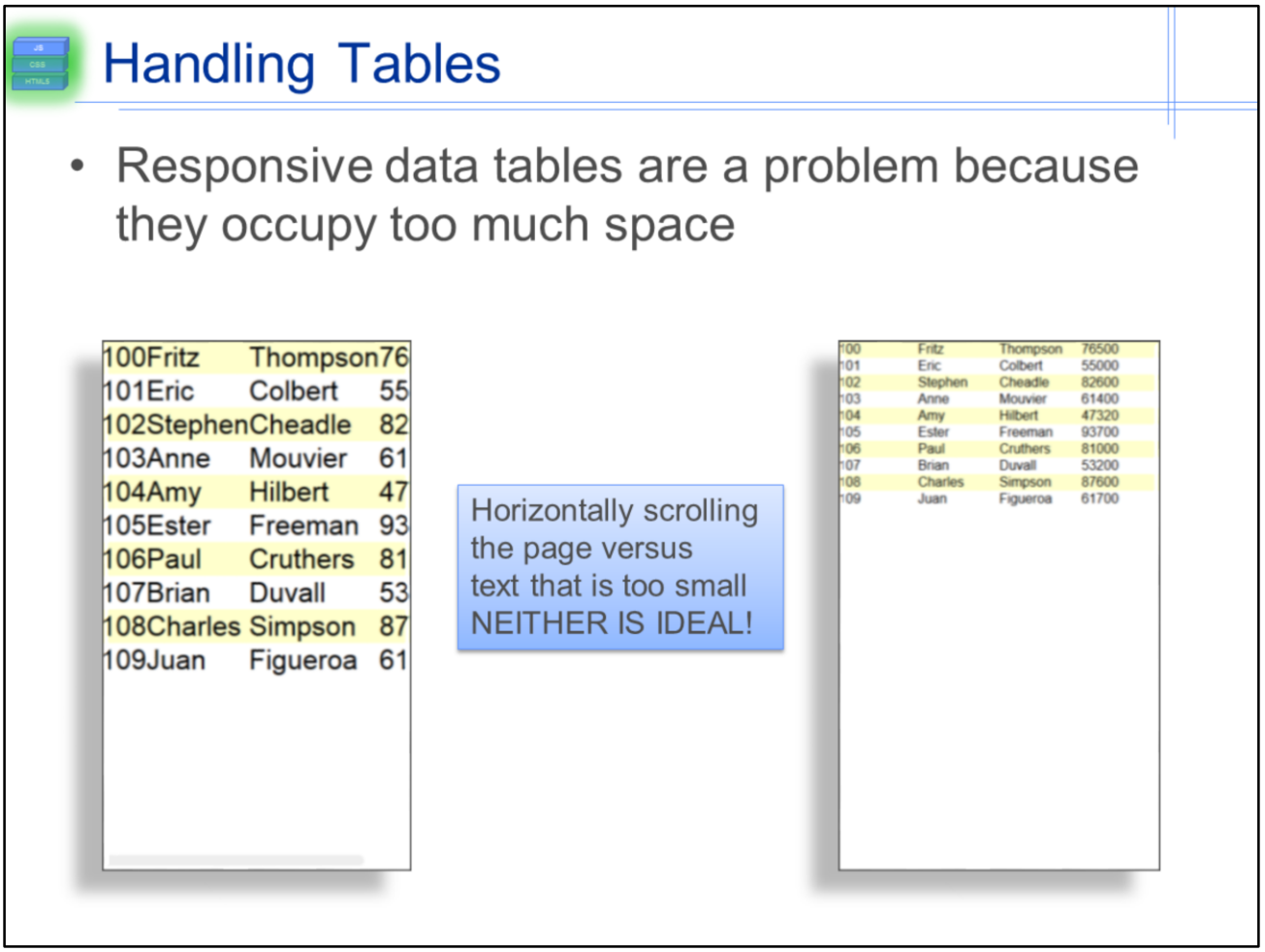
An explanation might be required for the off canvas pattern. When a button is tapped in the smaller sizes the off-screen piece slides into view. The example below is taken from: http://www.kaemingk.com/en/our-showroom/christmas/

Exercise 6 - It's Your Turn

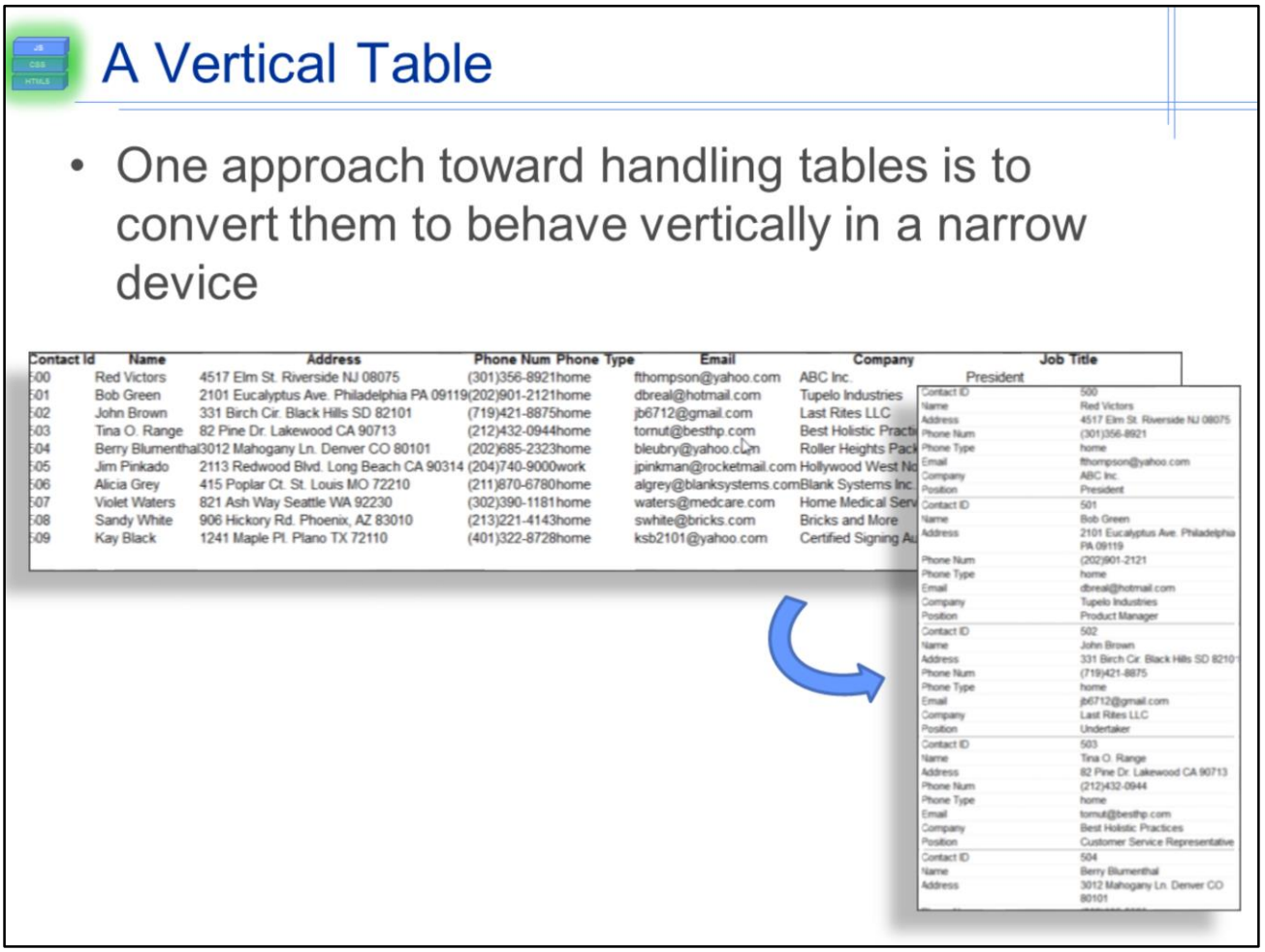
Responsive Solutions



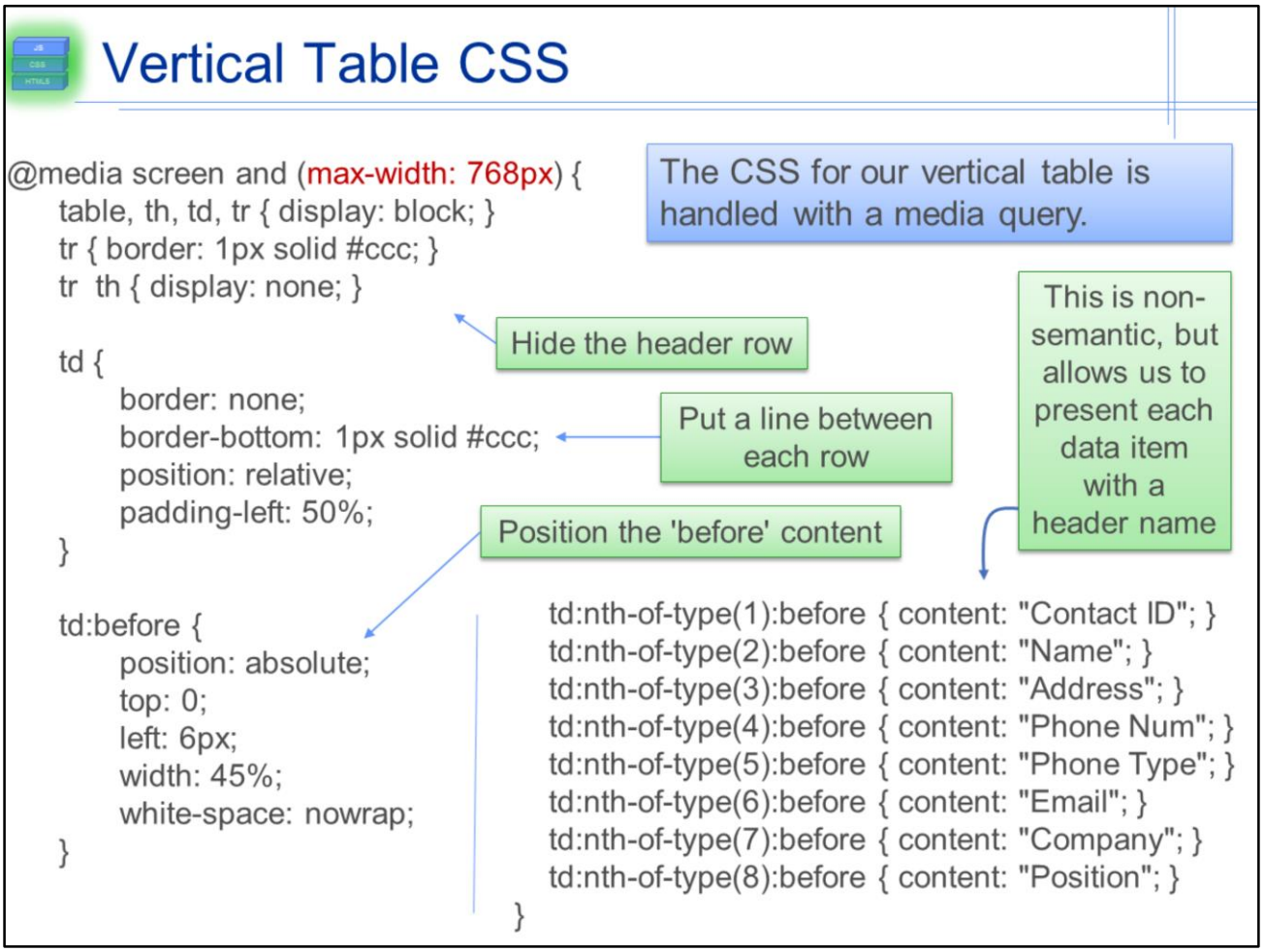
Instructions for this task are found in labs/lab06/starter/index.html



Handling tables can be tricky. There is no ideal solution. Scrolling tables to the side is an option for dealing with the width while making the text so small to make them fit is another (allowing pinching to expand the text).



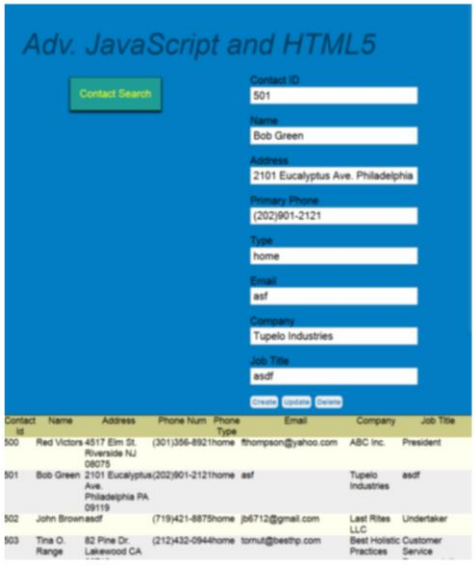
examples/05_responsive/08_tables.html



examples/05_responsive/08_tables.html

Exercise 6b - It's Your Turn

Responsive Tables

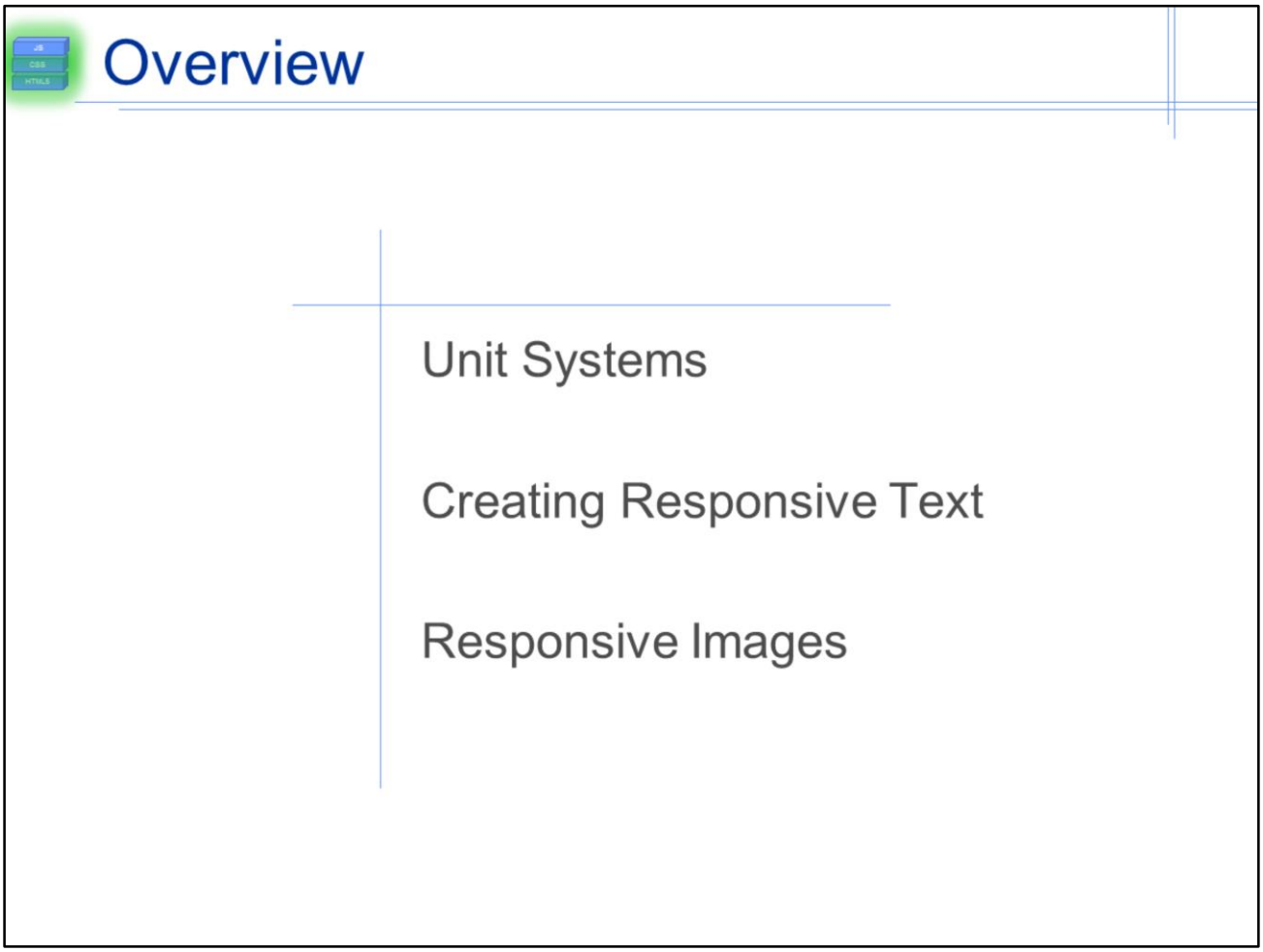


Instructions for this task are found in labs/lab06b/starter/index.html

Chapter 6

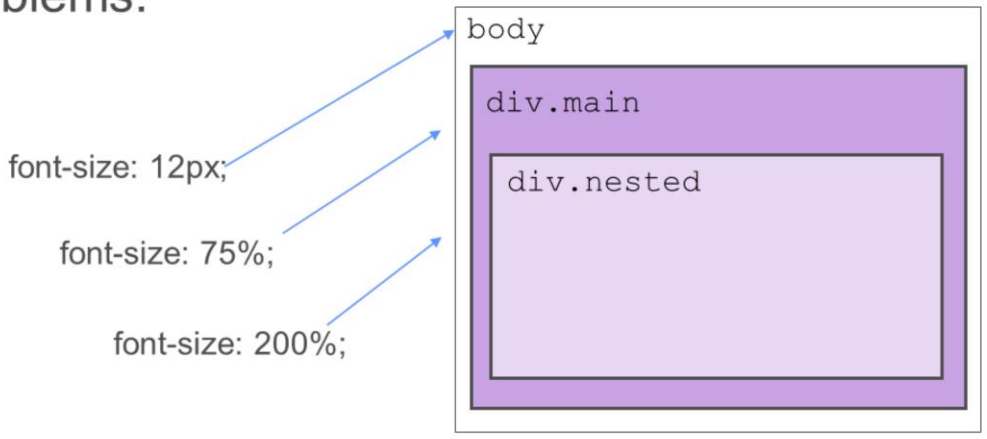
Responsive Typography and Images

Dealing with images and fonts in a responsive world



Problems with relative units: em, %

- Generally use relative fonts (percent, ems) when building scalable solutions
- However, these unit systems are not without problems:

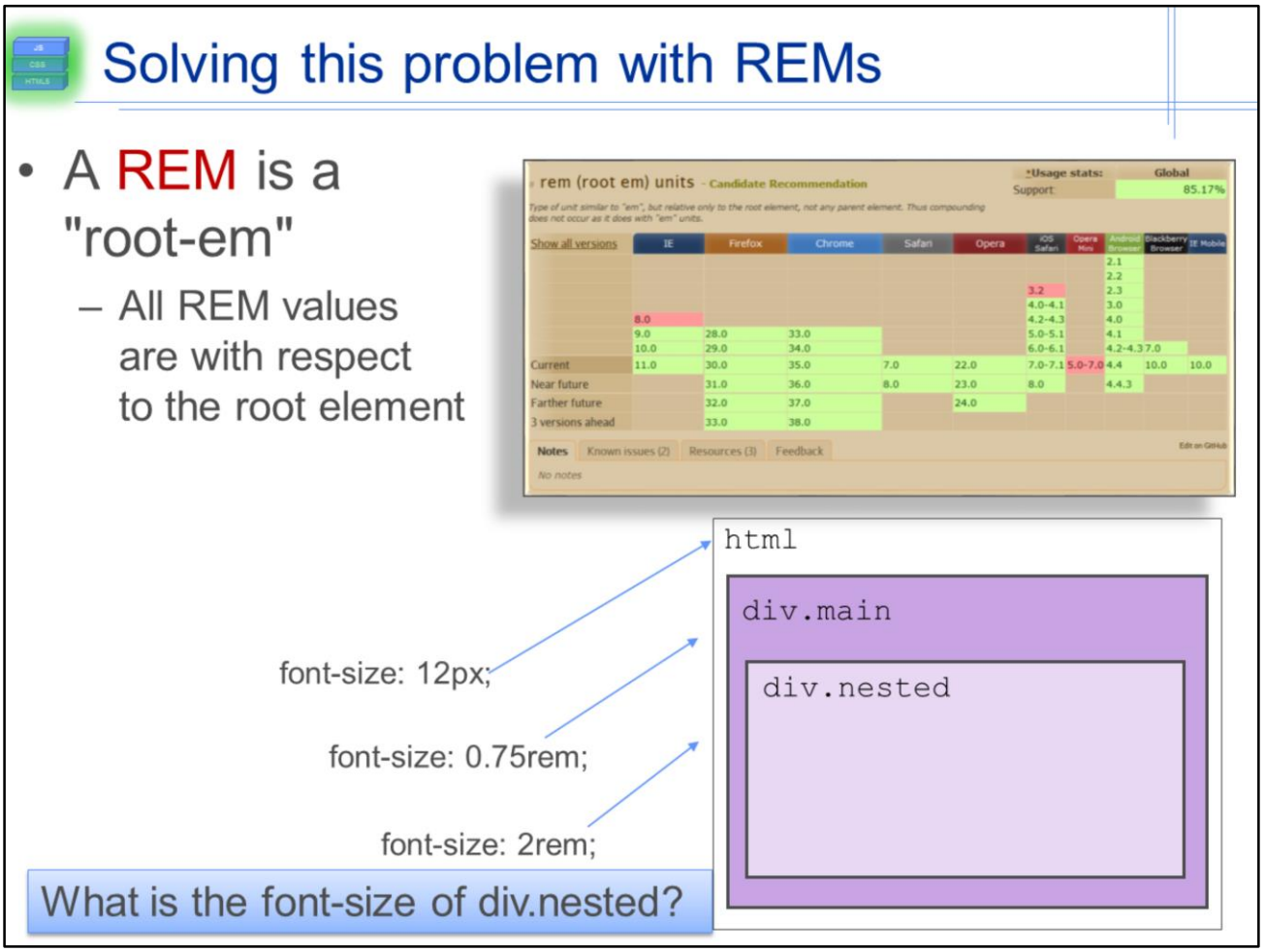


What is the font-size of div.nested?

examples/06_typography/01_relative_unit_issues.html

Answer: 18px.

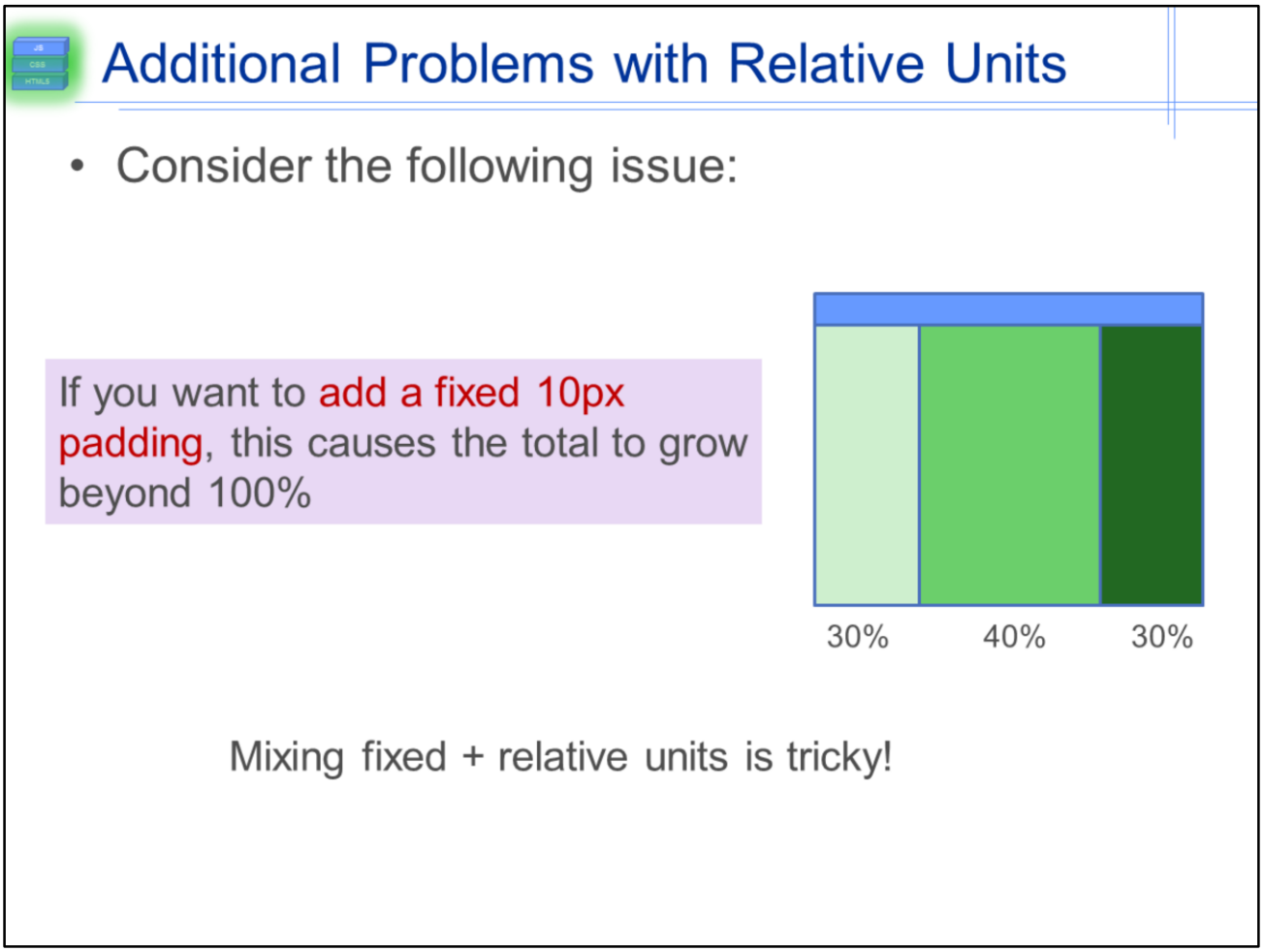
Relative fonts are always relative to their parent element. This can be tricky to calculate and track as your solution becomes increasingly complex.



examples/06_typography/02_rems.html

Answer: 24px

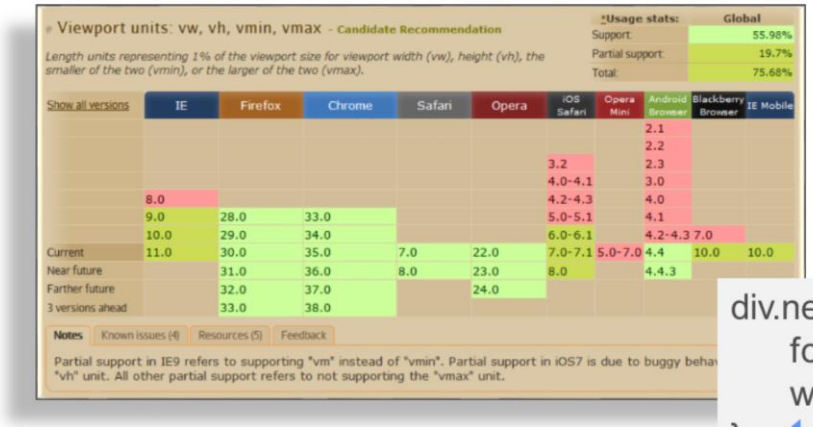
The root-em is a way to use relative fonts, but the root-em value is not inherited as an em value is.



Here we've defined a nice fluid layout at 30-40-30 percent. If you want to add a 10px padding to any or all columns, this will cause the columns to grow beyond 100%, thus ruining the solution. So, how do you do it? Do you convert the fixed 10px to a percentage? If so, what percentage do you use? The bottom line is that fixed units and percentages don't mix too well.

Viewport-Relative Units

- Rather than relying on a value relative to a parent element, it might make sense to rely on a unit relative to the viewport




```
div.nested {
    font-size: 10vh;
    width: 50vw;
}
```

50% of the viewport width

10% of the viewport height

examples/06_typography/03_viewport_units.html

Viewport-relative units are not in wide use yet. IE browsers do not properly support them and even numerous mobile devices have spotty support.

Making Text Responsive

- Viewport-relative units are not widely supported yet
  - **Text can be made scalable still:**

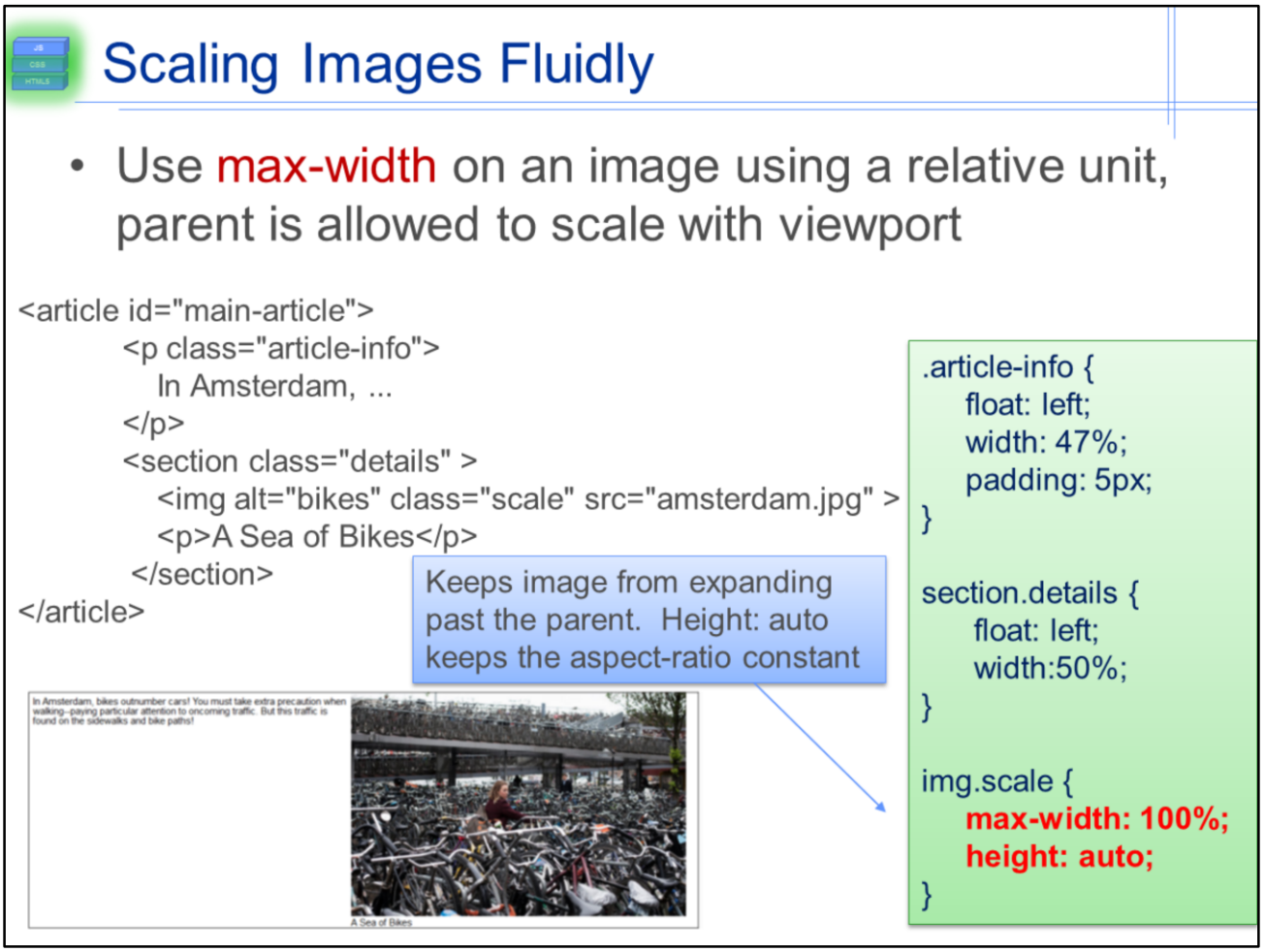
```
html{font-size:18px;}

@media screen and (orientation:portrait){
   div.nested{ font-size: 2rem; }
}
@media screen and (orientation:landscape){
   div.nested{ font-size: 1rem; }
}
```

Text size changes upon orientation changes

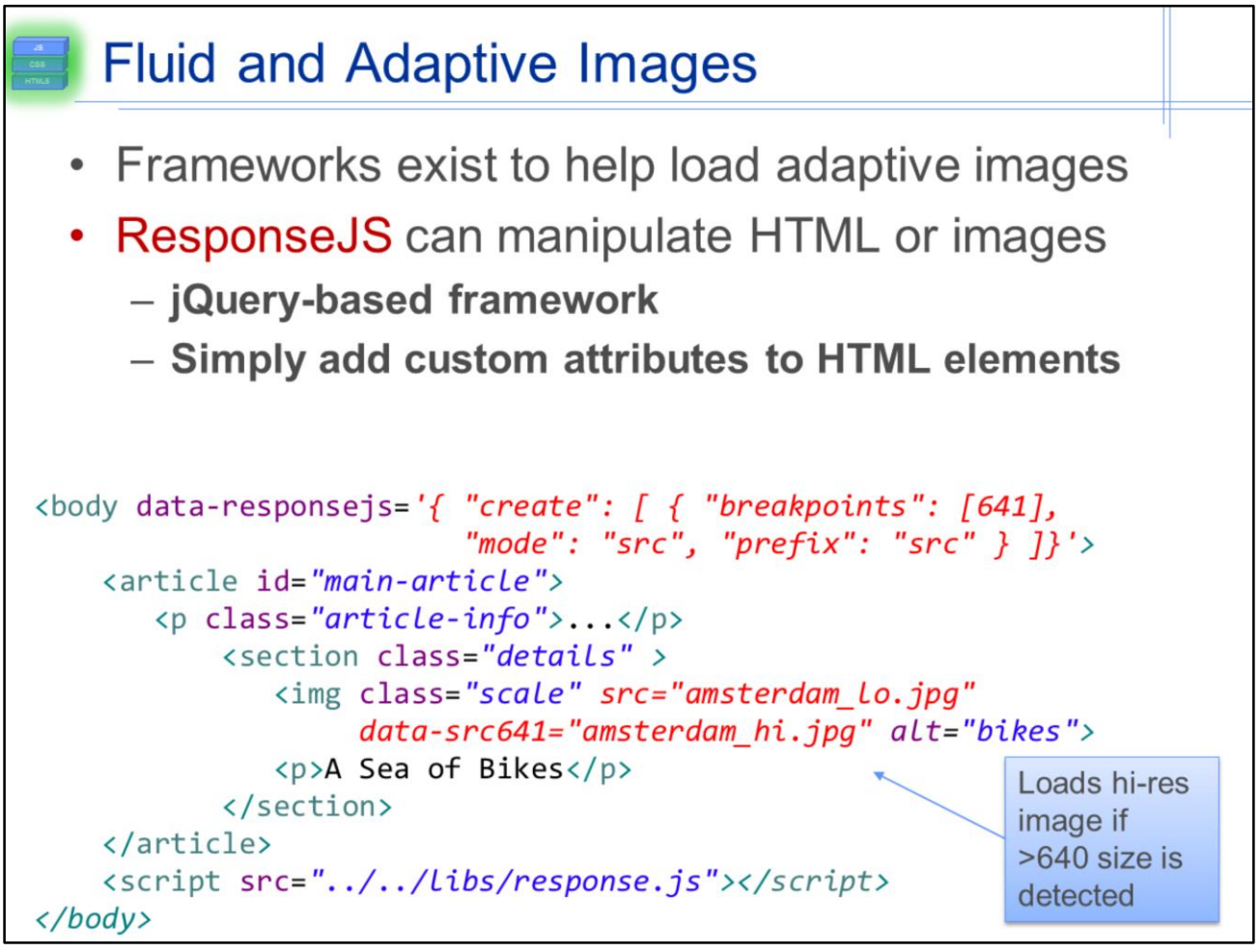
examples/06_typography/04_scaling_fonts.html

Making Images Flexible

- Images can be scaled or adapted
- *Scaling* involves controlling the viewport and max-width of the image
- *Adaptive* images will download a different image depending on the device size
  - Usually controlled through a media-query



examples/06_typography/05_responsive_images.html
In this example, the image is allowed to scale with the parent by fixing its size to its parent but allowing the parent to change sizes depending on the device size.



examples/06_typography/06_adaptive_and_scalable_images.html

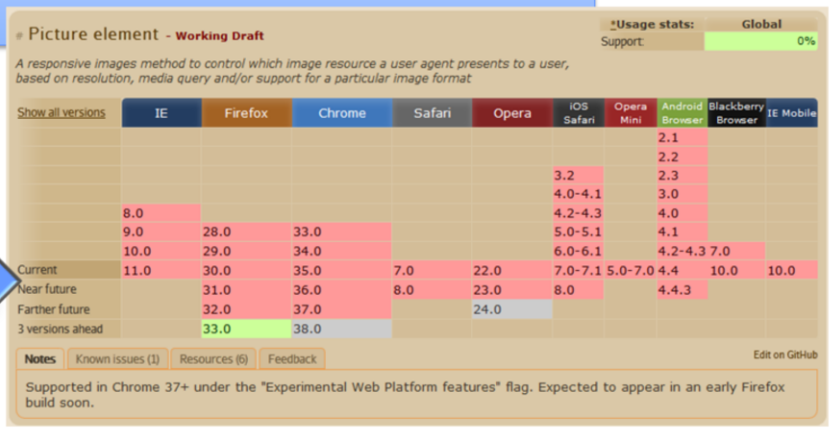
This example uses a JavaScript framework to load images by simply marking them with custom tags. Placing a bootstrap tag in the body and data-srcXXX tags in the <img> tags causes the library to automatically load the higher resolution image if / when the test becomes valid (true).

CSS Picture Elements

- Native support for adaptive images is coming…

```
<picture width="500" height="500">
   <source media="(min-width: 40.5em)" src="hi-res.jpg">
   <source media="(min-width: 20em)" src="med-res.jpg">
   <source src="lo-res.jpg">
   <img src="small.jpg" alt="">
   <p>Picture of bicycles</p>
</picture>
```

But not yet!



Picture element - Working Draft

*Usage stats: Global

Support: 0%

A responsive images method to control which image resource a user agent presents to a user, based on resolution, media query and/or support for a particular image format

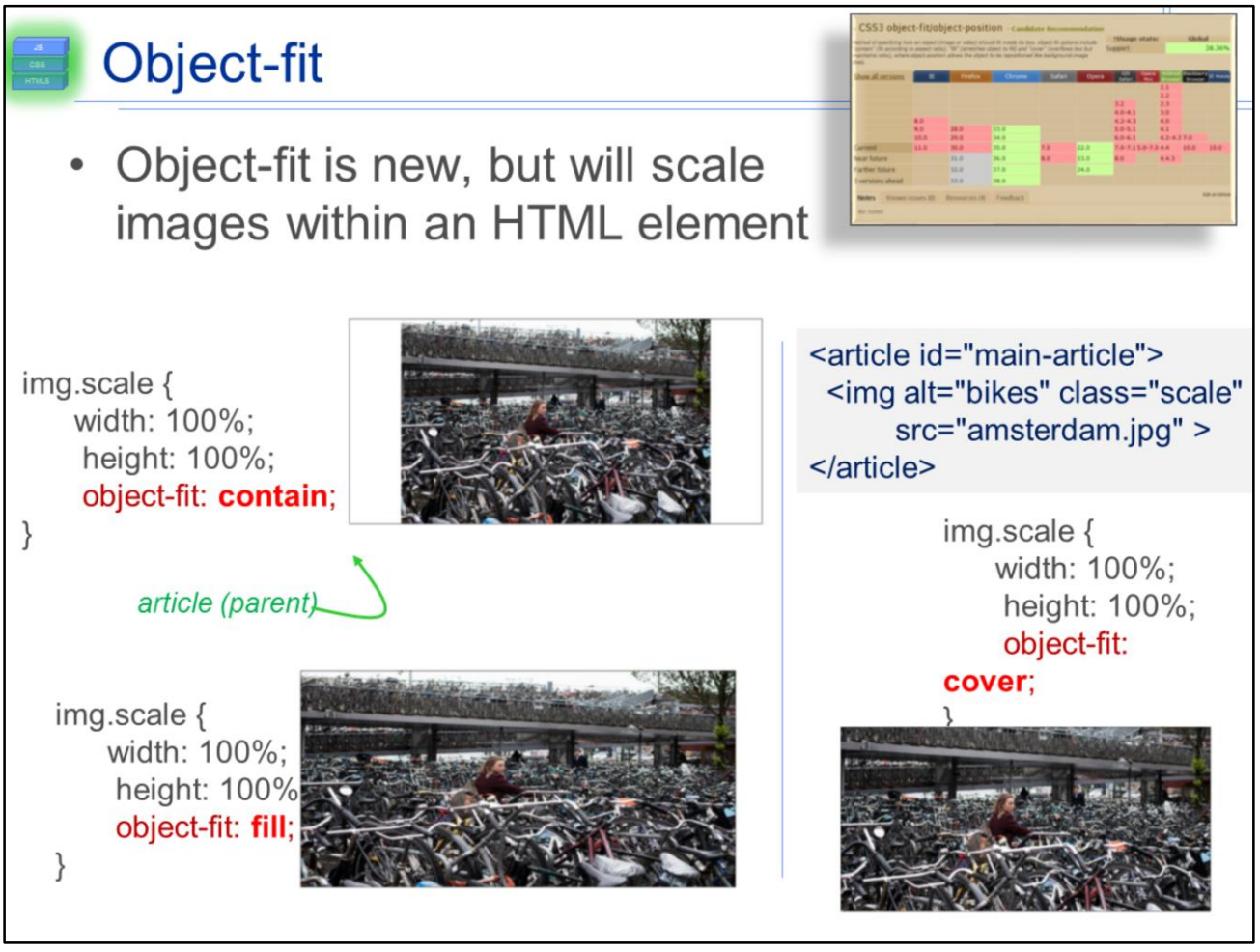
Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1		
								2.2		
						3.2		2.3		
						4.0-4.1		3.0		
	8.0					4.2-4.3		4.0		
	9.0	28.0	33.0			5.0-5.1		4.1		
	10.0	29.0	34.0			6.0-6.1		4.2-4.3	7.0	
Current	11.0	30.0	35.0	7.0	22.0	7.0-7.1	5.0-7.0	4.4	10.0	10.0
Near future		31.0	36.0	8.0	23.0	8.0		4.4.3		
Farther future		32.0	37.0		24.0					
3 versions ahead		33.0	38.0							

Notes | Known issues (1) | Resources (6) | Feedback | Edit on GitHub

Supported in Chrome 37+ under the "Experimental Web Platform features" flag. Expected to appear in an early Firefox build soon.

The spec is in working draft now: http://www.w3.org/TR/html-picture-element/

But essentially this will remove the need to have a ResponseJS framework.



examples/06_typography/08_object_fit.html
Object-fit is similar to the new features added to the background-image property in CSS3 except that these apply to <img> images. There are relatively new, but provide some control over how images scale. Using 'contain', the image will maintain its aspect-ratio, but perhaps not fill the entire parent. Using 'fill', the image will alter the aspect-ratio to fill the parent. Using 'cover', will also maintain the aspect-ratio, but will enlarge until the small dimension (either height or width) fills the parent. In this case the image enlarged until the width was completely covered. This causes clipping on the image. By default the object is centered. Control the positioning with object-position.

text-size-adjust

– **text-size-adjust** (coming soon) determines if text should enlarge and if so, by how much

```
/* says text on iPhone should not be adjusted, e.g. in an orientation change */
p { -webkit-text-size-adjust: none; }

/* says adjust text size downward by half */
p { -moz-text-size-adjust: 50%; }
```

This property currently requires vendor extensions (until it matures in the standards)

examples/06_typography/07_text_size_adjust.html
This property is not standard yet. Here is its current working draft:
http://dev.w3.org/csswg/css-size-adjust/

More on the property:
https://developer.mozilla.org/en-US/docs/Web/CSS/text-size-adjust

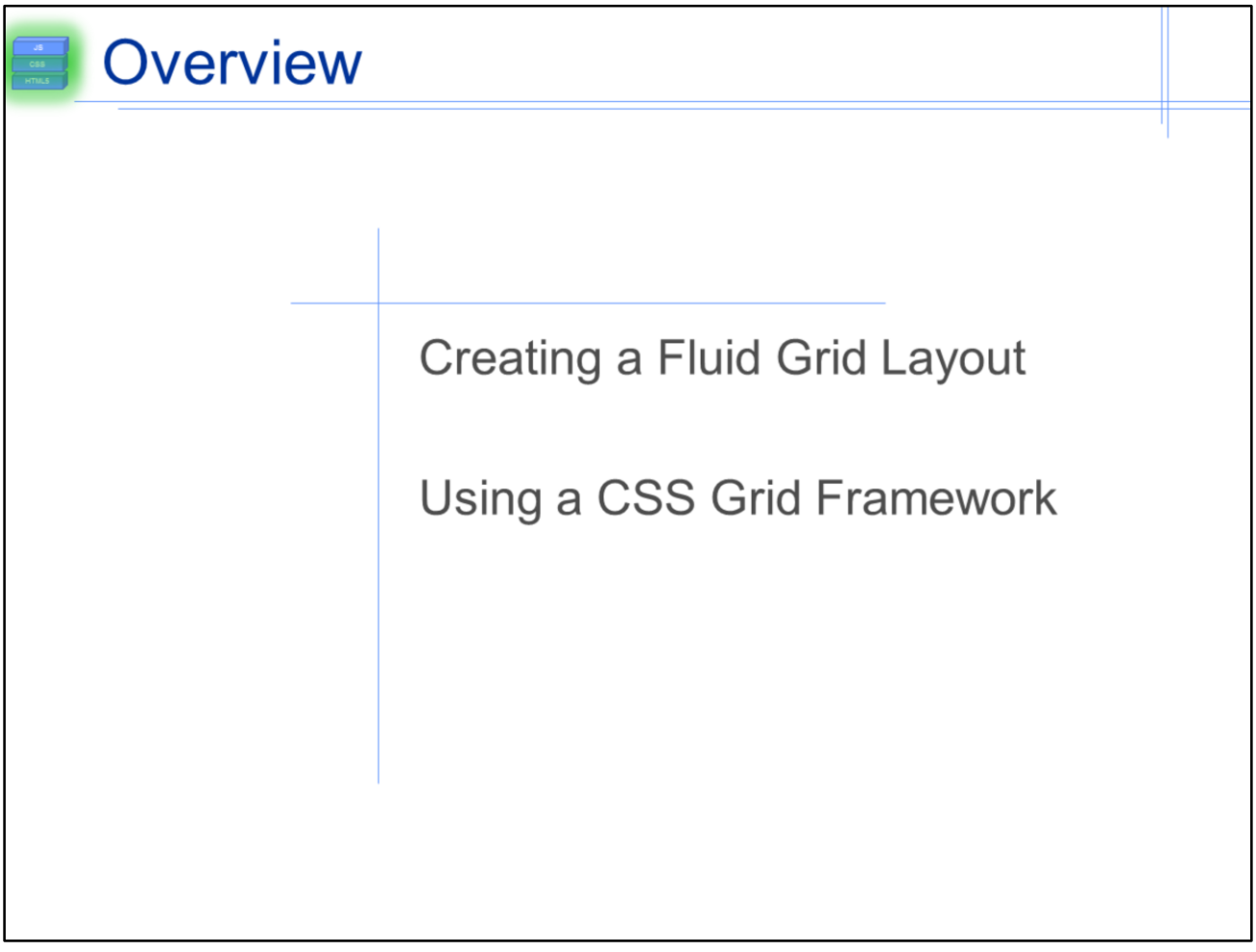
Summary

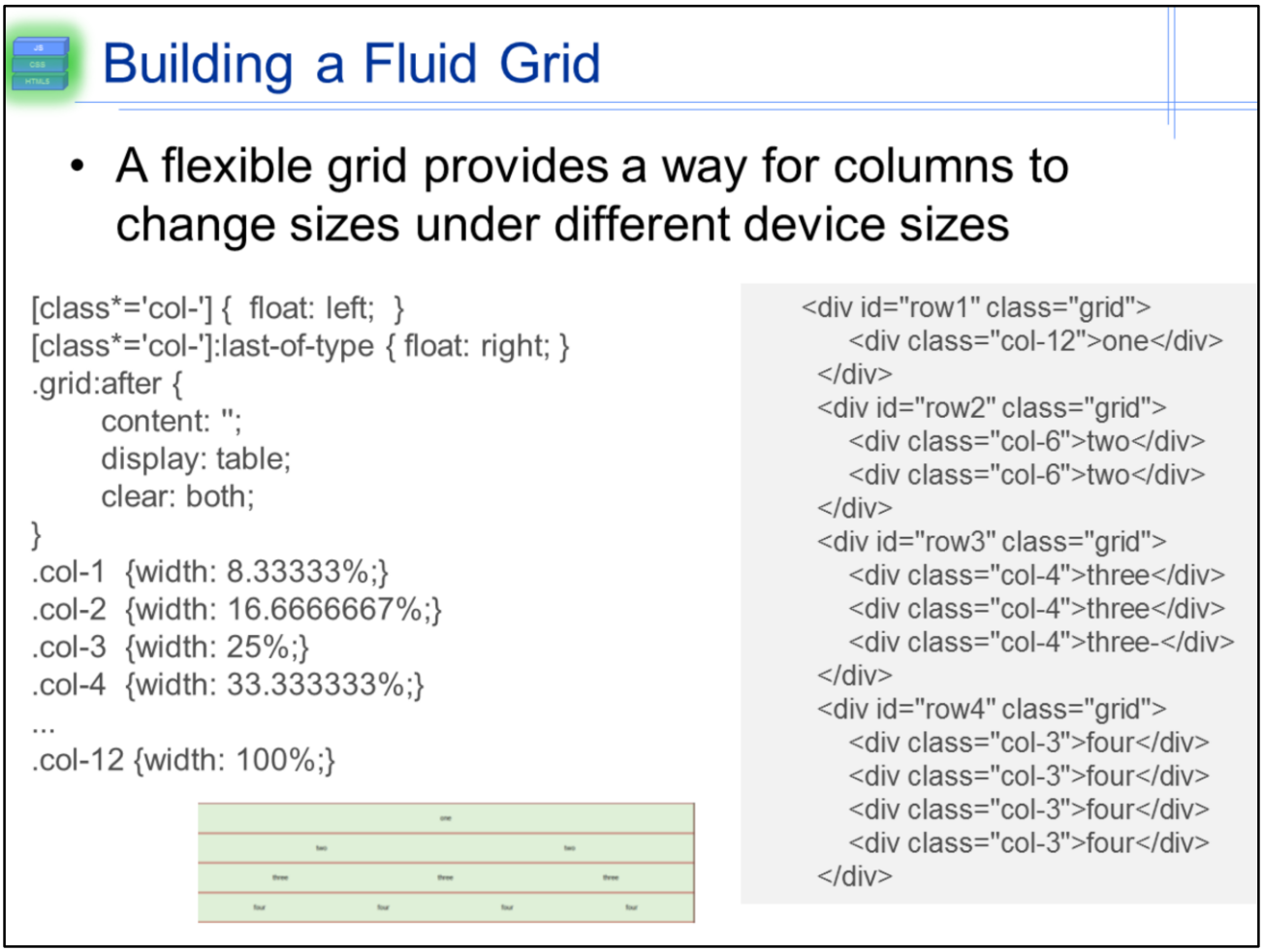
- Use height:auto, max-width: 100% to scale images fluidly
- Scale fonts using relative units
  - **Choose rems to scale according to the root**
  - **Viewport-relative units are a bit early to use**
- Use adaptive image frameworks to serve low-bandwidth images
- Be aware of new properties coming: object-fit, text-adjust-size, and <picture> elements
  - But we can't use these just yet!

Chapter 7

Fluid Layouts and Grid Systems

Implementing Layouts using Pre-built Frameworks

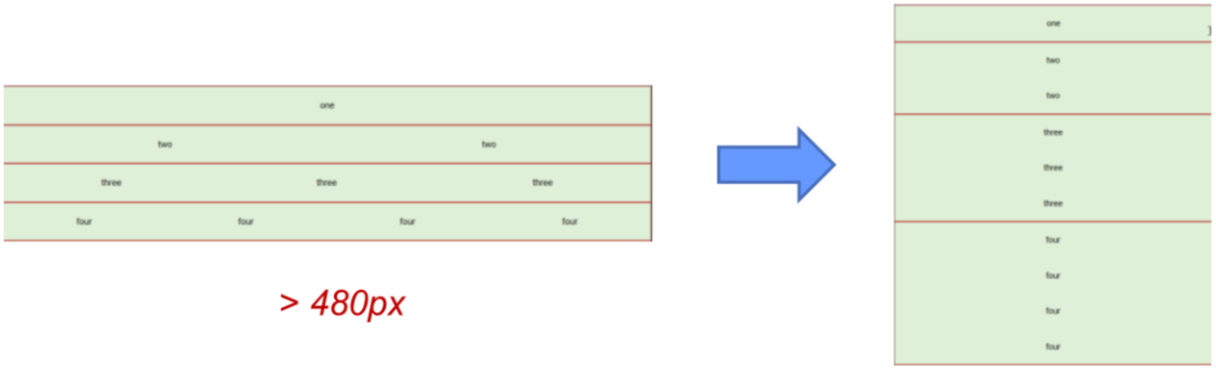




examples/07_fluid_layouts/01_flex_grid.html
This example shows that it is relatively easy to build a grid system.

Fluid Grids + Media Queries

- Our fluid grid can also be controlled by media queries



> 480px

< 480px

```
@media screen and (max-width: 480px) {
        .col-1, .col-2, .col-3, .col-4, .col-5, .col-6,
        .col-7, .col-8, .col-9, .col-10, .col-11, .col-12 { width: 100%; }
}
```

examples/07_fluid_layouts/01_flex_grid.html
Our grid solution breaks at 480px. Note: this is not a mobile-first implementation.

Flexbox

- Flex-box is a CSS-based way to lay out pages using giving high-level elements resizing capabilities

box2 box1

```
#box1, #box2 {
   width: 50%;
   height: 200px;
}
#box1 { background-color: papayawhip; }
#box2 { background-color: honeydew; }

#flexcontainer {
   display: flex;
   flex-direction: row-reverse;
}
```

```
<div id="flexcontainer">
   <div id="box1">
      box1
   </div>
   <div id="box2">
      box2
   </div>
</div>
```

examples/07_fluid_layouts/02_flexbox.html

Notice that the flex-direction can actually change the order of the boxes from the DOM! Legal values include row, row-reverse, column, column-reverse.

Other flexbox properties include the ability to completely re-order the columns (use the flex-order property on the child elements supplying an ordinal value).

Another property, called justify-content can align content along the main axis (the direction boxes are placed, such as row or column) or the cross-axis, which is the opposite axis.

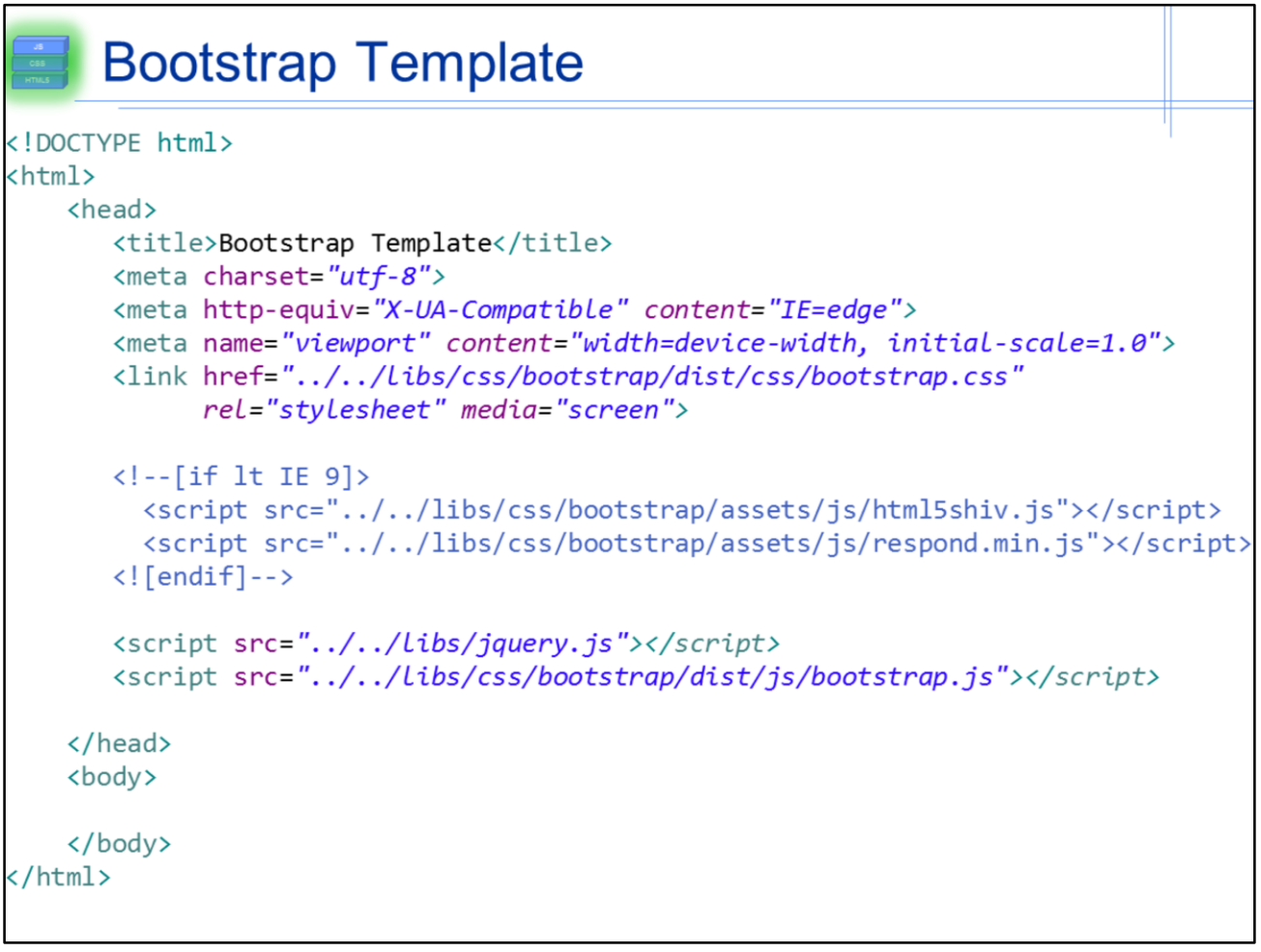
The justify-content can be either: flex-start, center, space-around which place boxes at the axis beginning, center, or evenly justified.
The align-items property can place justify items on the cross-axis using: stretch (fills the height of the parent), center (centers within the parent), and flex-end (bottom-aligns on the parent). You may also align items individually using align-self.

Furthermore, the ultimate in flexibility comes from the flex property which is a shorthand for three properties: flex-grow, flex-shrink, and flex-basis. The flex-basis value takes precedence over the width property that the browser applies to the box. Flex-grow and flex-shrink represent a ratio (of the value of all children) for how to grow and shrink. An example flex property might look like this: flex: 1 2 200px; More on this can be found in a

second flex example in the student files.

Layout Frameworks

- CSS Frameworks can provide a faster time-to-production approach than building your own grid system
  - Provide legacy browser hacks
  - Often provide other responsive components
- Twitter's Bootstrap is a popular mobile-first, fluid grid layout



Bootstrap Template

```
<!DOCTYPE html>
<html>
    <head>
       <title>Bootstrap Template</title>
       <meta charset="utf-8">
       <meta http-equiv="X-UA-Compatible" content="IE=edge">
       <meta name="viewport" content="width=device-width, initial-scale=1.0">
       <link href="../../libs/css/bootstrap/dist/css/bootstrap.css"
             rel="stylesheet" media="screen">

       <!--[if lt IE 9]>
         <script src="../../libs/css/bootstrap/assets/js/html5shiv.js"></script>
         <script src="../../libs/css/bootstrap/assets/js/respond.min.js"></script>
       <![endif]-->

       <script src="../../libs/jquery.js"></script>
       <script src="../../libs/css/bootstrap/dist/js/bootstrap.js"></script>

    </head>
    <body>

    </body>
</html>
```

examples/07_fluid_layouts/04_bootstrap.html

Bootstrap Class Names

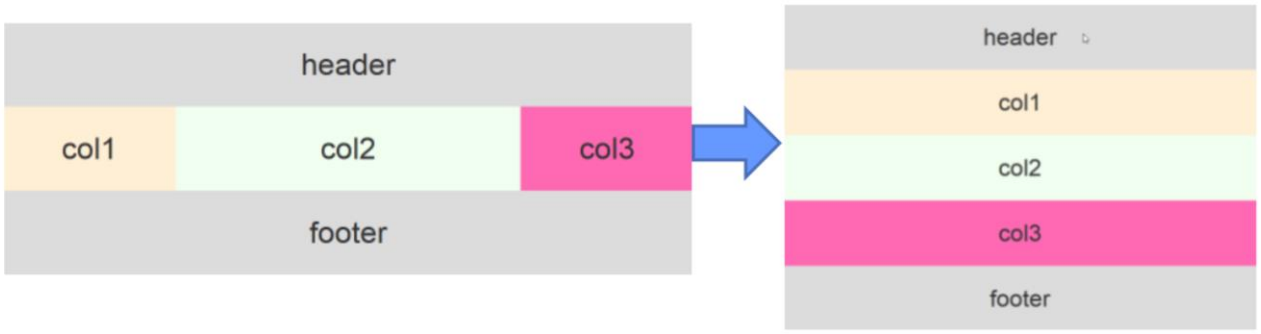
- Bootstrap provides a 12-column grid that can break at different predefined breakpoints
  - Its behavior is affected by the class names you use:

	Extra Small	Small	Medium	Large Devices
Breakpoint	< 768px	> 768px	> 992px	>1200px
Class Name	*.col-xs-*	*.col-sm-*	*.col-md-*	*.col-lg-*

To create 3 even columns in a 12-column grid that breaks at 992px use:

```
<div class="container">
  <div class="row">
    <div class="col-md-4"></div>
    <div class="col-md-4"></div>
    <div class="col-md-4"></div>
  </div>
</div>
```

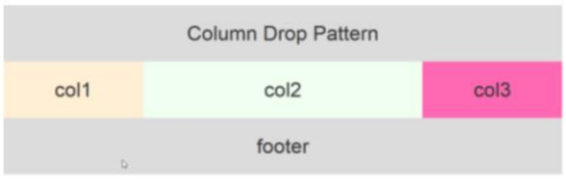
Bootstrap Layout Example 1



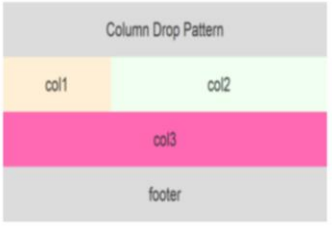

```
<div id="container" class="container">
   <header class="col-md-12">header</header>
   <section class="row">
      <article id="col1" class="col-md-3">col1</article>
      <article id="col2" class="col-md-6">col2</article>
      <article id="col3" class="col-md-3">col3</article>
   </section>
   <footer class="col-md-12">footer</footer>
</div>
```

examples/07_fluid_layouts/04_bootstrap.html

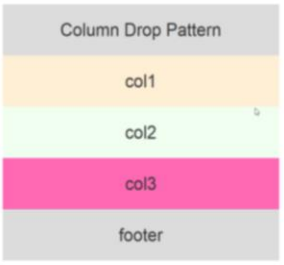
Bootstrap Layout Example 2



> 1200px



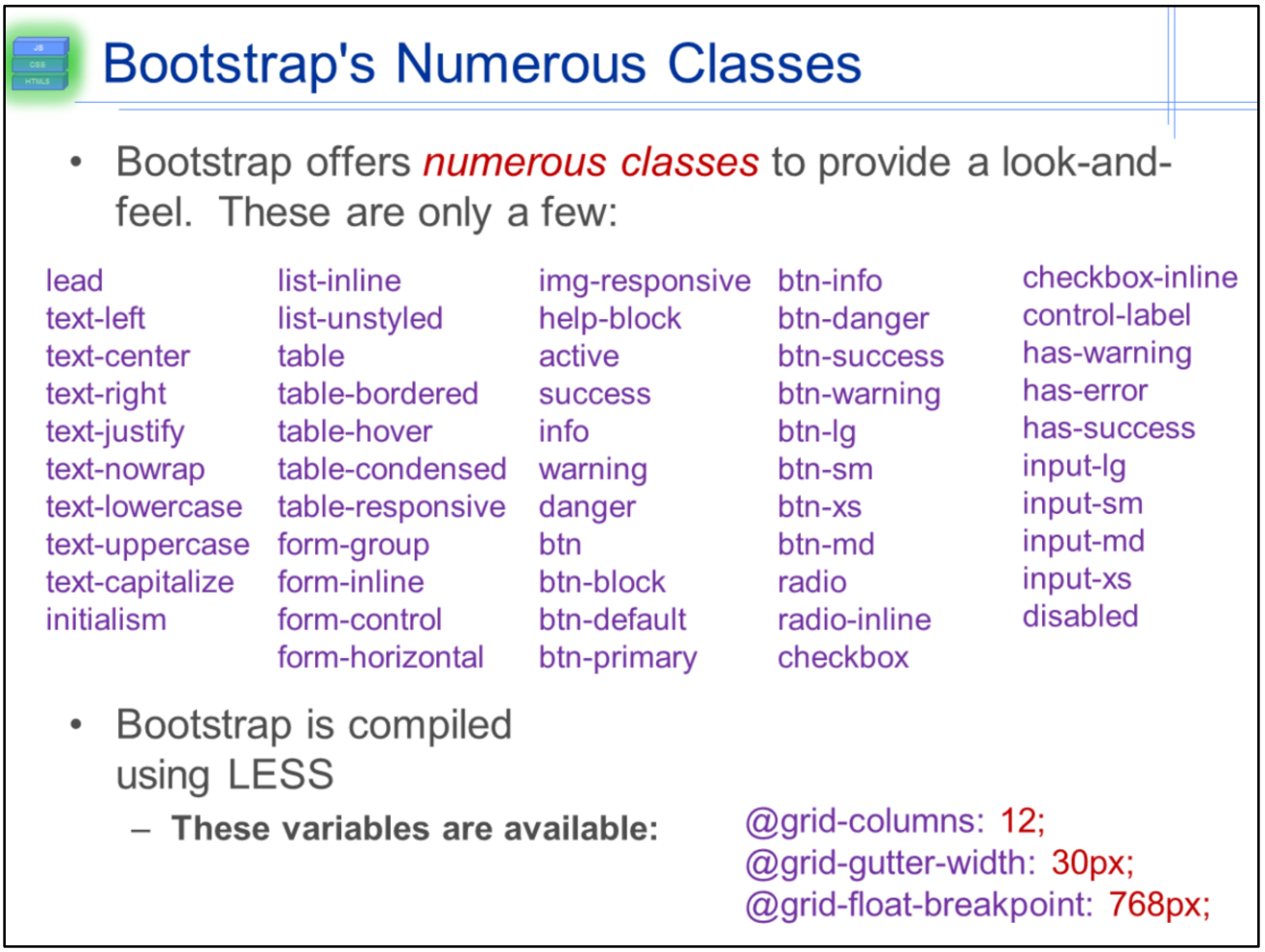
768-1200px



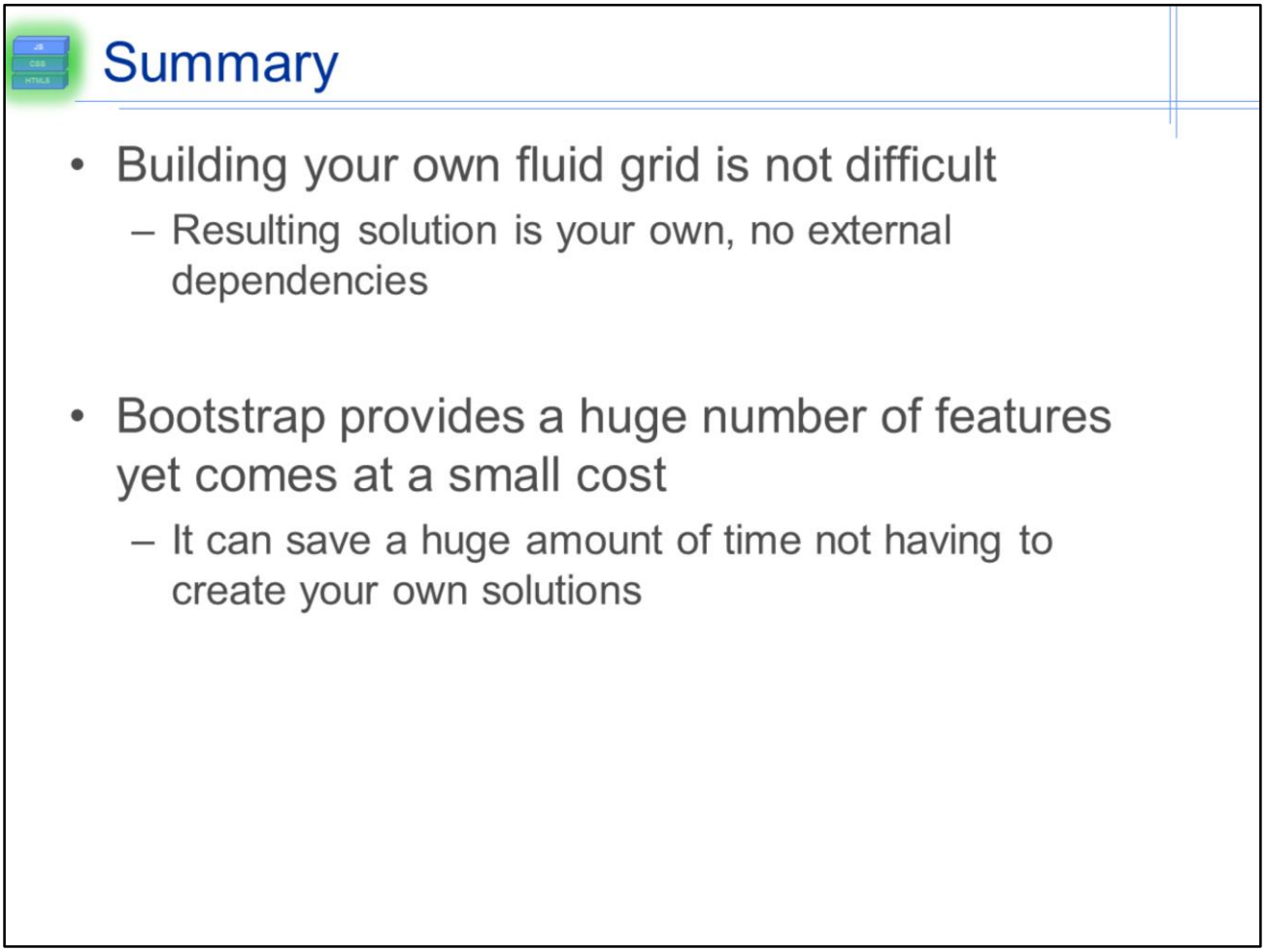
< 768px

```
<div id="container" class="container">
   <header class="col-md-12">header</header>
   <section class="row">
      <article id="col1" class="col-md-4 col-lg-3">col1</article>
      <article id="col2" class="col-md-8 col-lg-6">col2</article>
      <article id="col3" class="col-md-12 col-lg-3">col3</article>
   </section>
   <footer class="col-md-12">footer</footer>
</div>
```

examples/07_fluid_layouts/05_bootstrap_layout2.html



Bootstrap provides dozens of classes and LESS functions and variables that can help customize the bootstrap environment. It can be a very powerful tool when combined with LESS (discussed later).

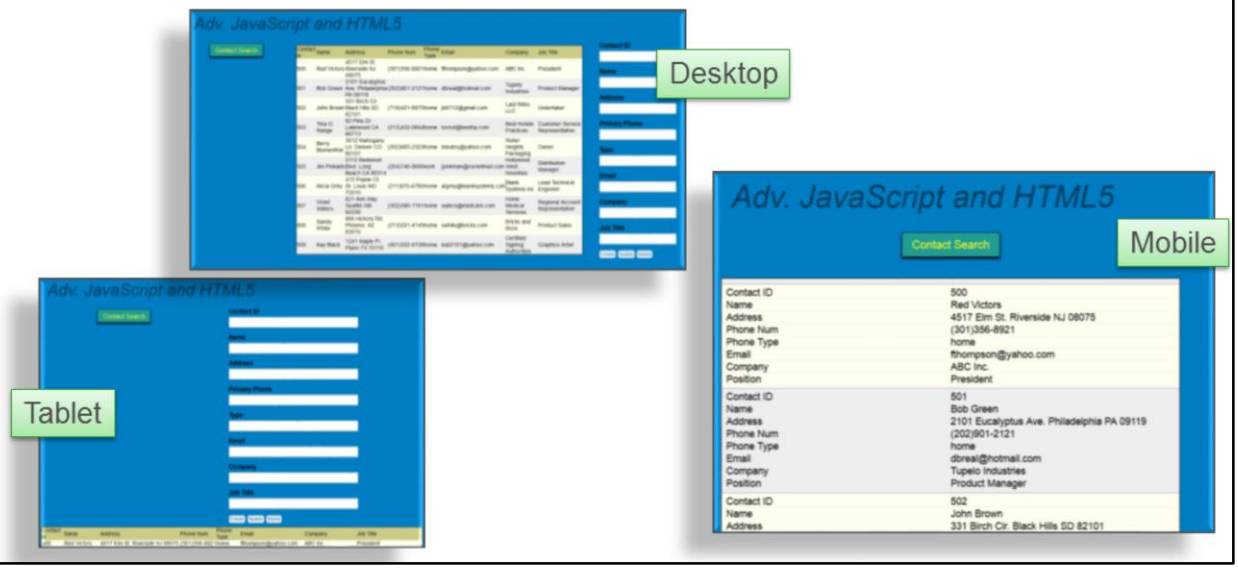


Exercise 7 - It's Your Turn

Implement the Column Drop Pattern with Bootstrap

Work from labs/lab07/starter/index.html.
Two steps:

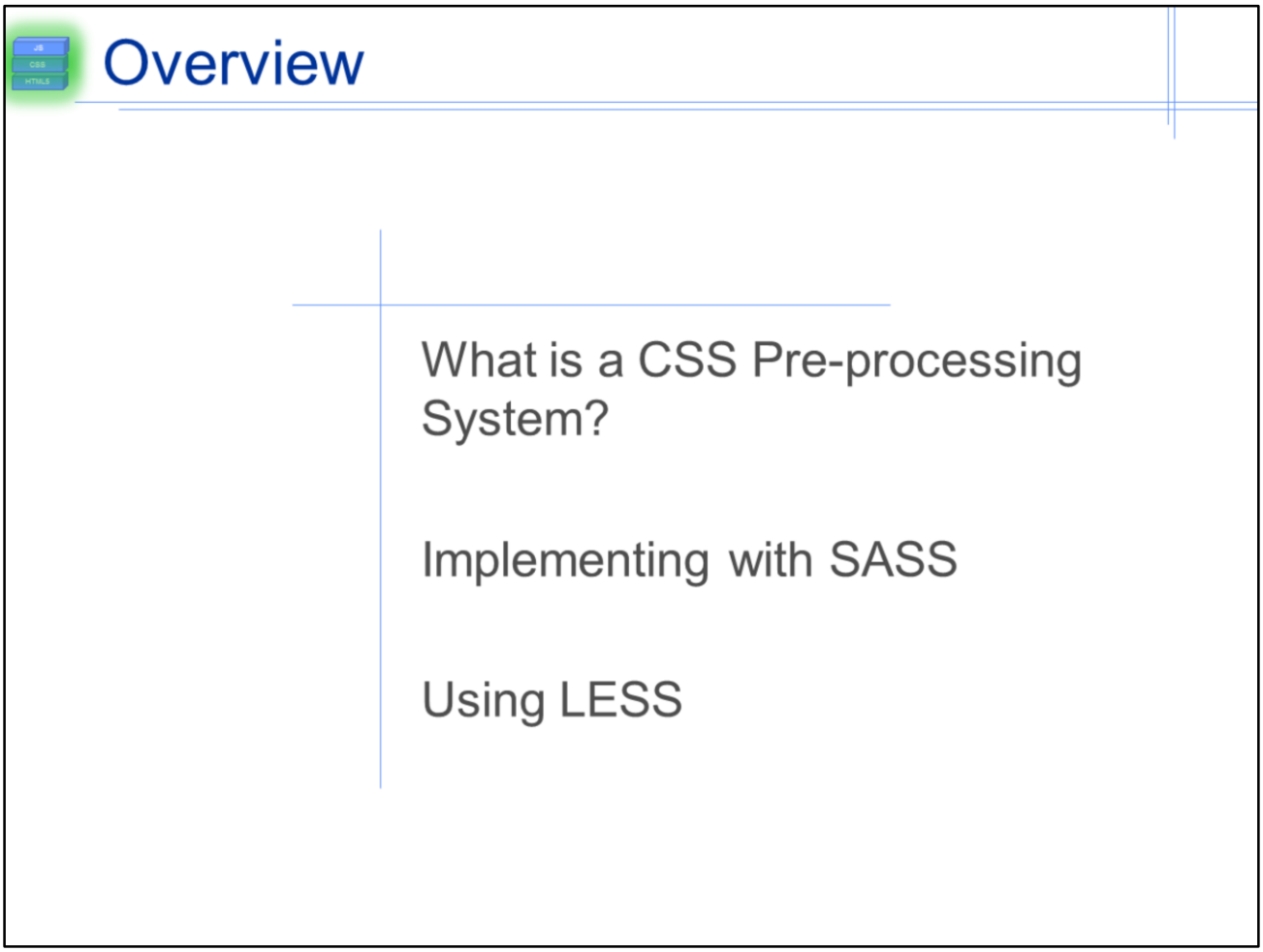
1. Add bootstrap to the implementation
2. Set the columns on leading, trailing, main divs to achieve the effect



Chapter 8

CSS Pre-processing Systems

Using SASS and LESS to make more



CSS Pre-processing

- Standard CSS has limitations
  - **Not very re-usable**
  - **Changes must be made throughout the CSS individually**
- CSS 3 did little to improve this
- Enter the CSS Pre-processing framework…

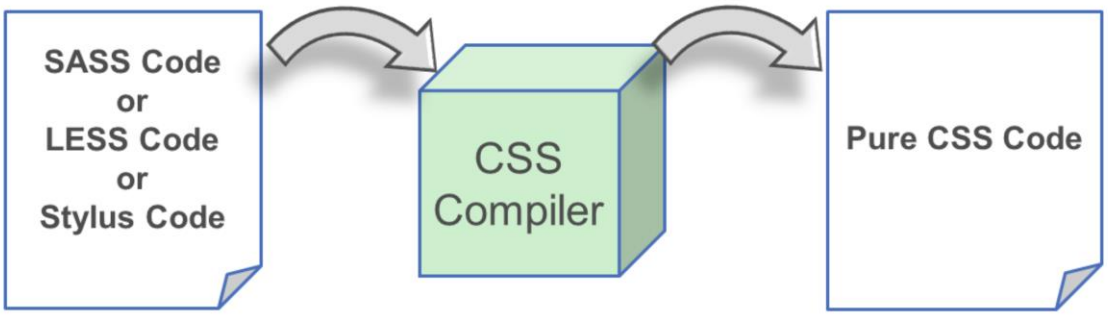
Pre-processing Frameworks

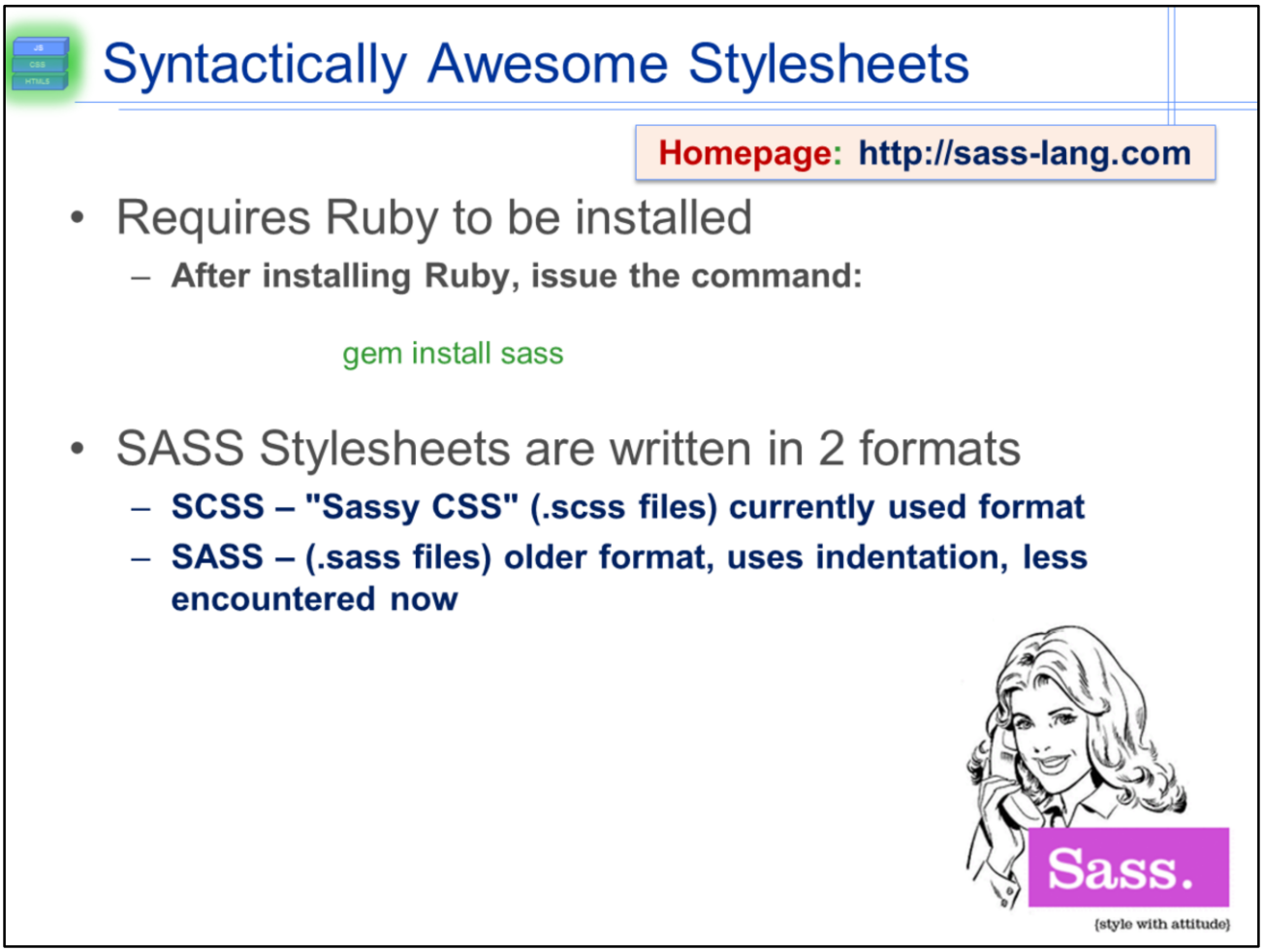
CSS Pre-processing Frameworks

- SASS
- LESS
- Stylus

Advantages
Less Redundancy
Variables and Mixins

Disadvantages
Updating CSS is one layer removed

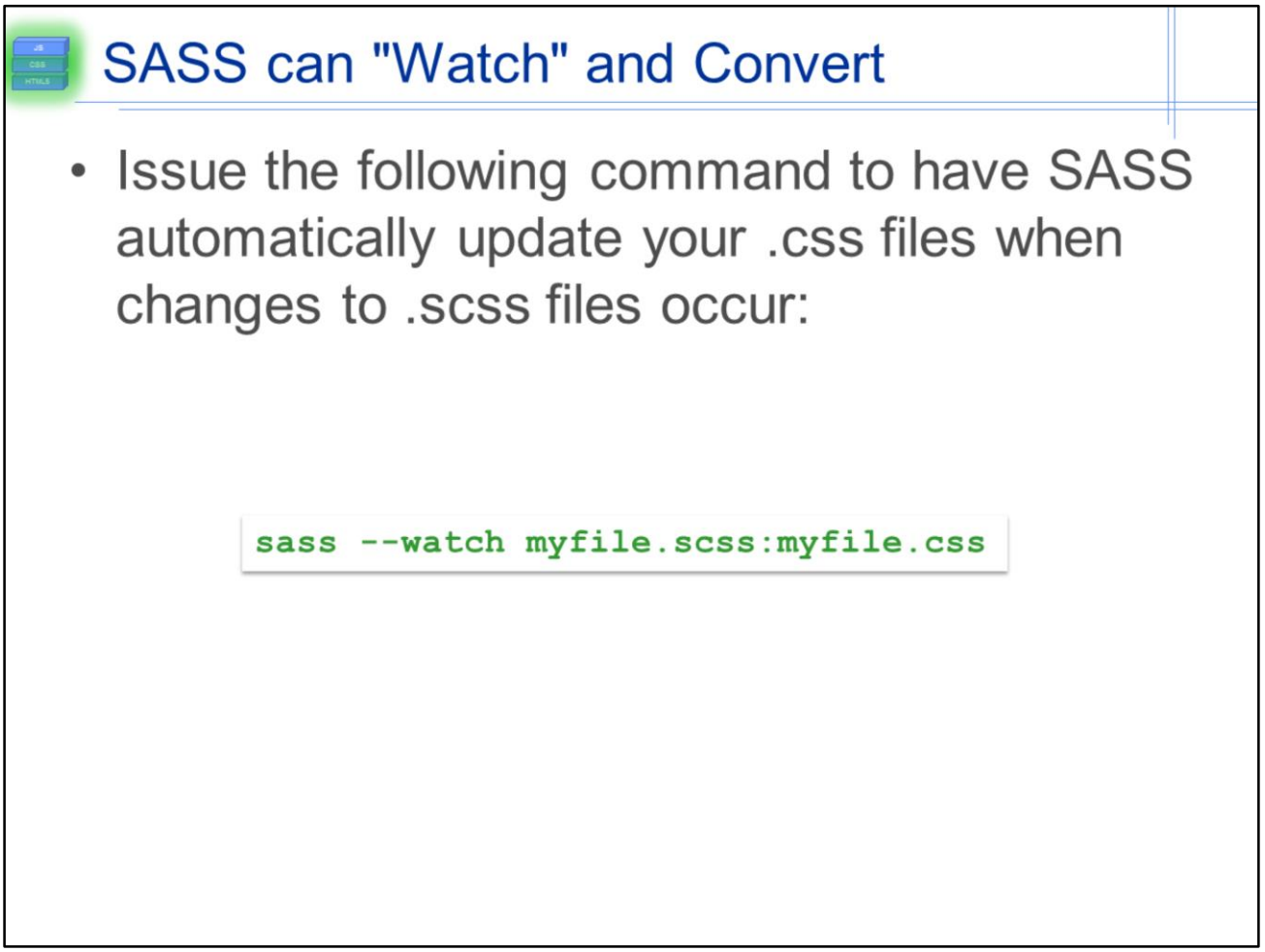




SASS requires Ruby to be installed first. Afterwards, simply executing the gem install sass command will cause SASS to be installed.

SASS stylesheets can be written using the classic (original) format, which uses files proper indentation instead of curly braces to indicate ownership. SCSS file formats support standard CSS3 syntax.

To convert .sass files to .scss files use the conversion tool sass-convert myfile.sass myfile.scss



SASS runs in the background. It actively monitors the files (directories) you specify. For example, in slide, issuing the described command will cause sass to "watch" for changes to your .scss file. When you save it, sass automatically rebuilds to CSS file.

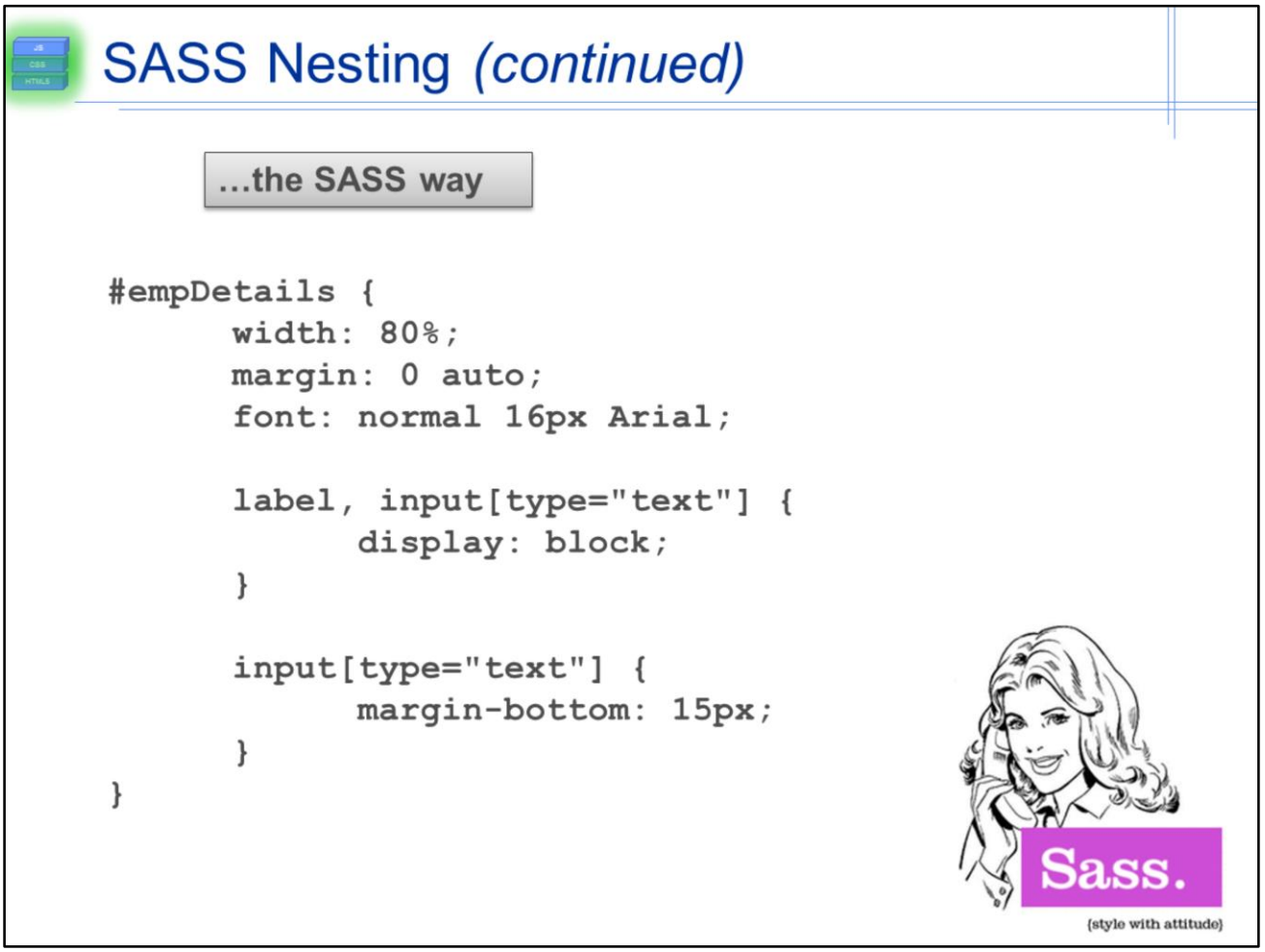
SASS can also monitor entire directories and convert .scss files within the directory as well.

SASS Nesting (DRY)

```
#empDetails {
      width: 80%;
      margin: 0 auto;
      font: normal 16px Arial;
}

#empDetails label, #empDetails input[type="text"] {
      display: block;
}

#empDetails input[type="text"] {
      margin-bottom: 15px;
}
```

Take this original CSS...

SASS (like Ruby) encourages the DRY principle, don't repeat yourself. To this means, it uses the concept of "Nesting" rules.



SASS (like Ruby) encourage the DRY principle, don't repeat yourself. To this means, it uses the concept of "Nesting" rules.

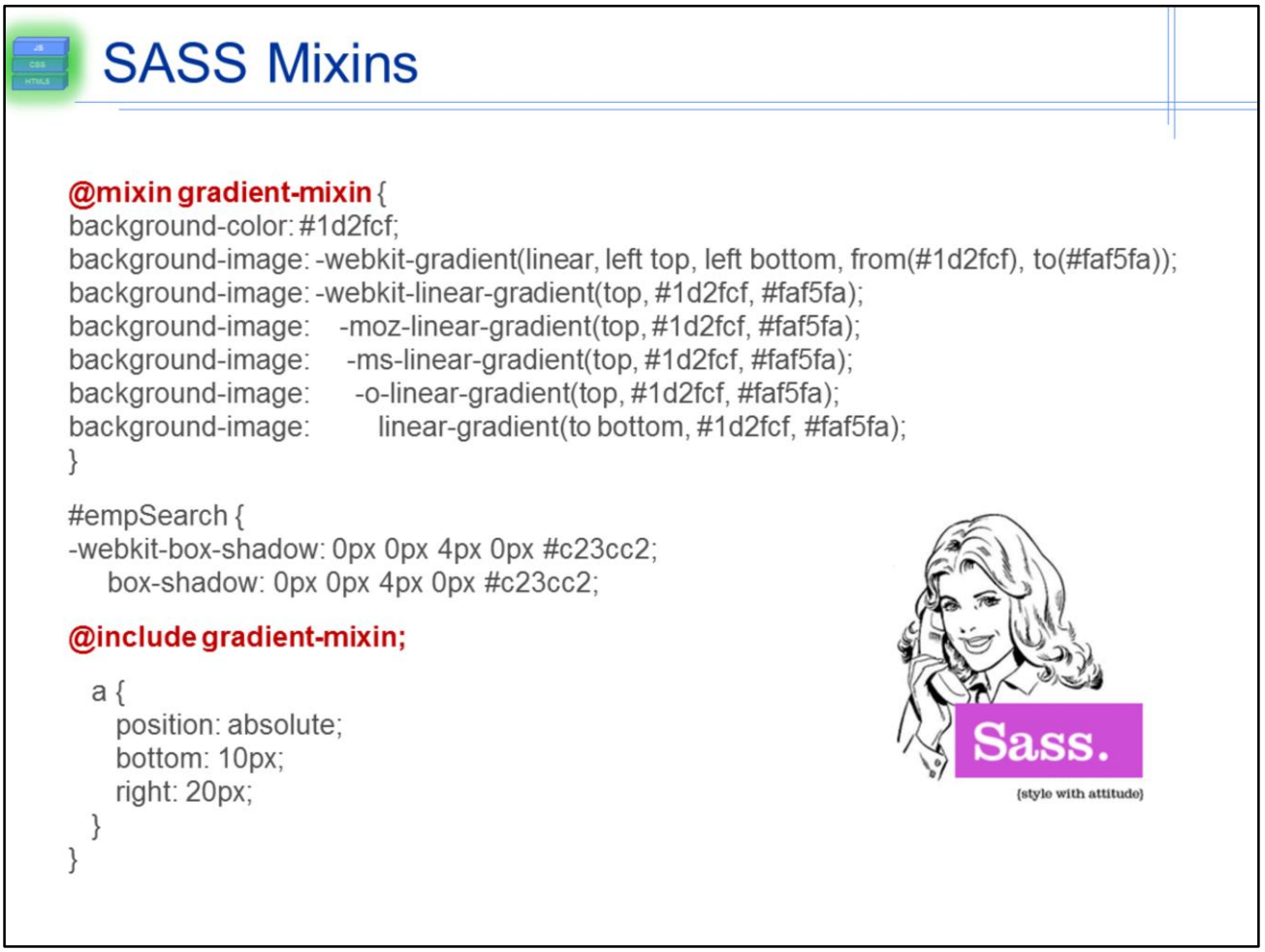
SASS Variables

```
$width: 80%;

#wrapper {
width: $width;
min-width: 800px;
margin: 10px auto;
}
```

emp.scss

```
#wrapper{
  width: 80%;
  min-width: 800px;
  margin: 10px auto;
 }
```

emp.css



Mixins allow for groups of properties to be "mixed in" or reused in various parts of CSS code. Declare a mixin using @mixin and then use it with an @include declaration.

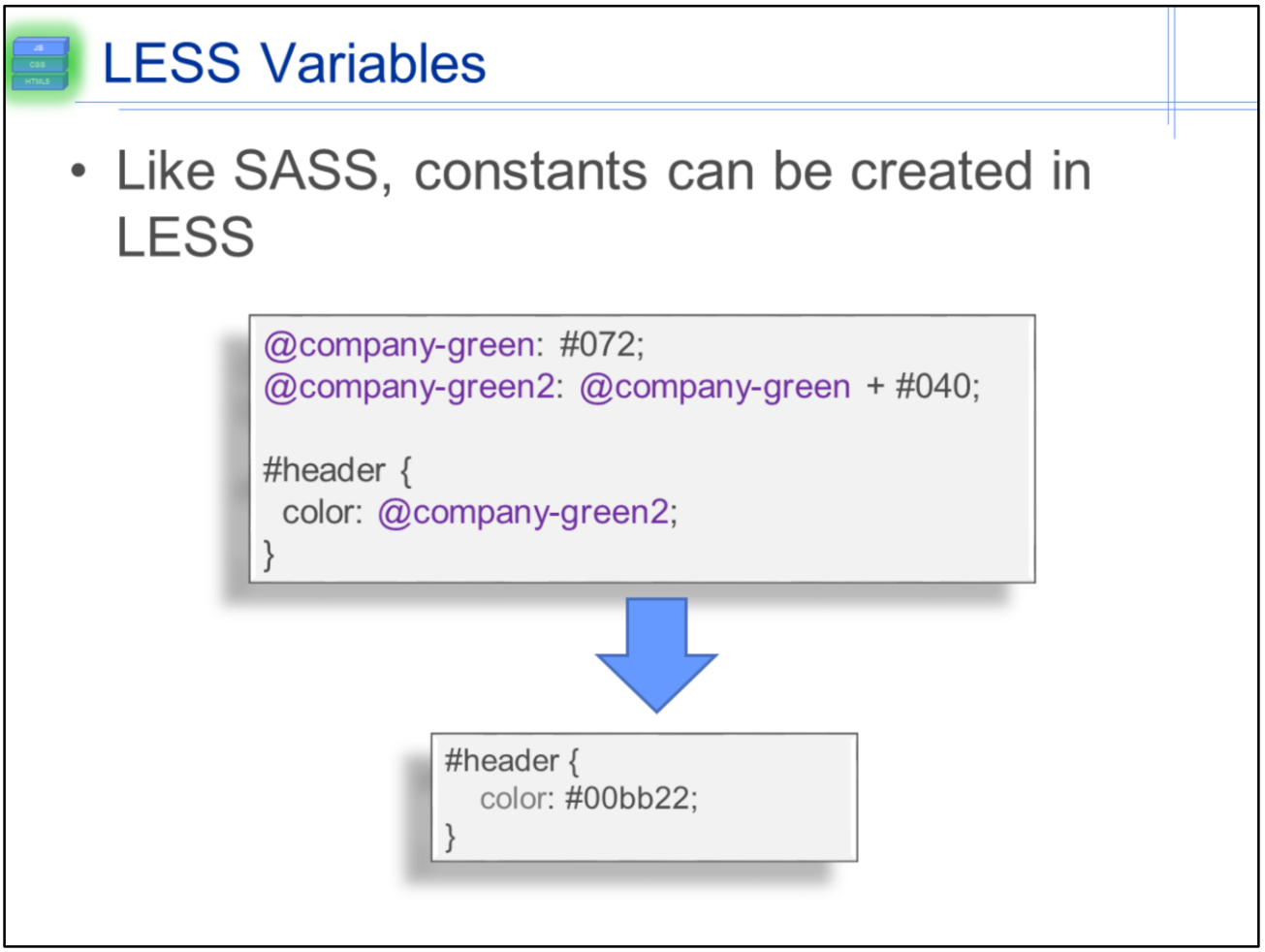
An additional reference for more you can do with SASS can be found here: http://sass-lang.com/tutorial.html

LESS

- LESS is another common CSS-Preprocessing framework

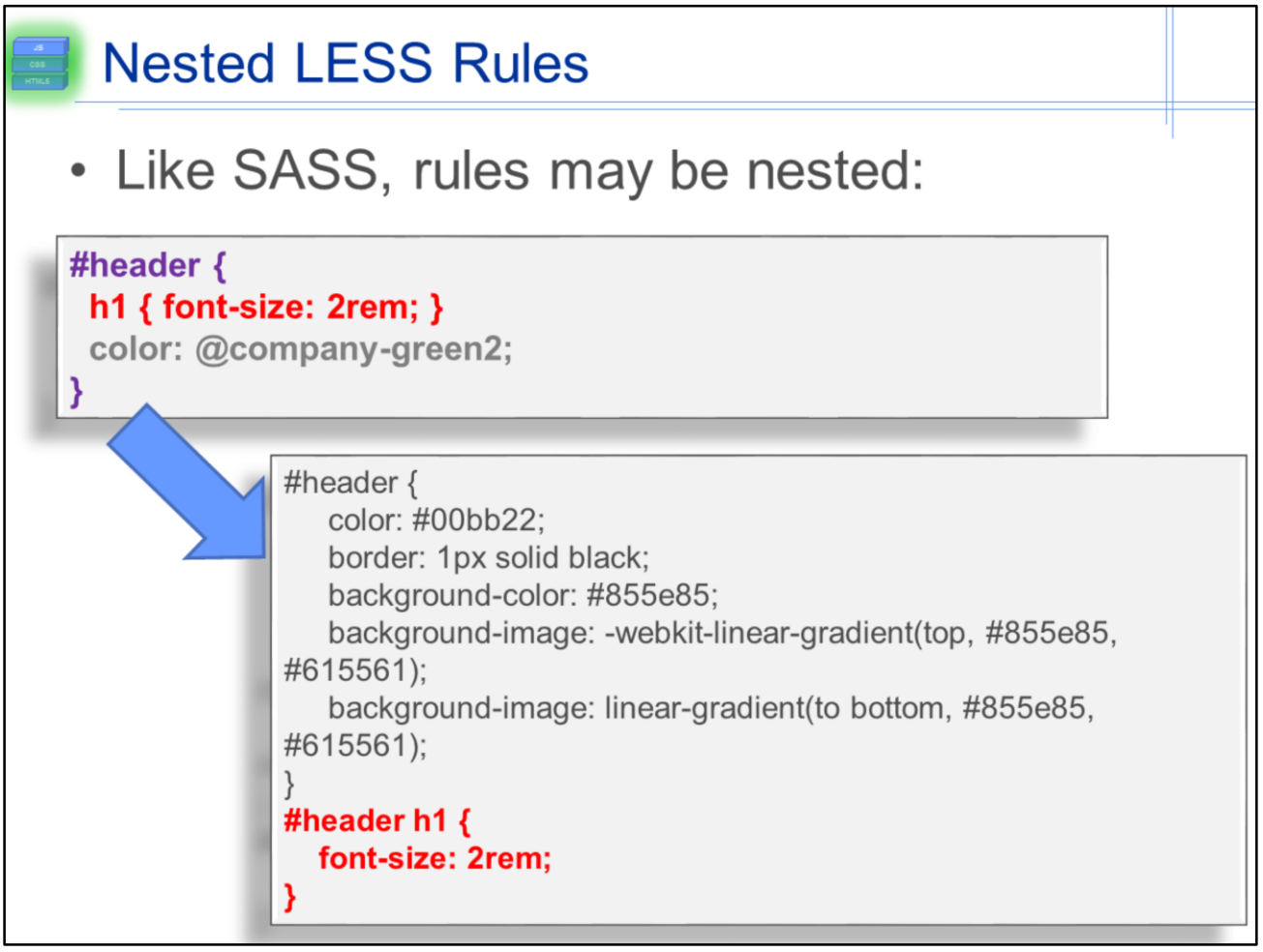
  - LESS requires **nodejs** to be installed first
  - Install LESS using **npm install -g less**
  - Run the LESS compiler using
    lessc options main.less main.css

Homepage: http://lesscss.org/

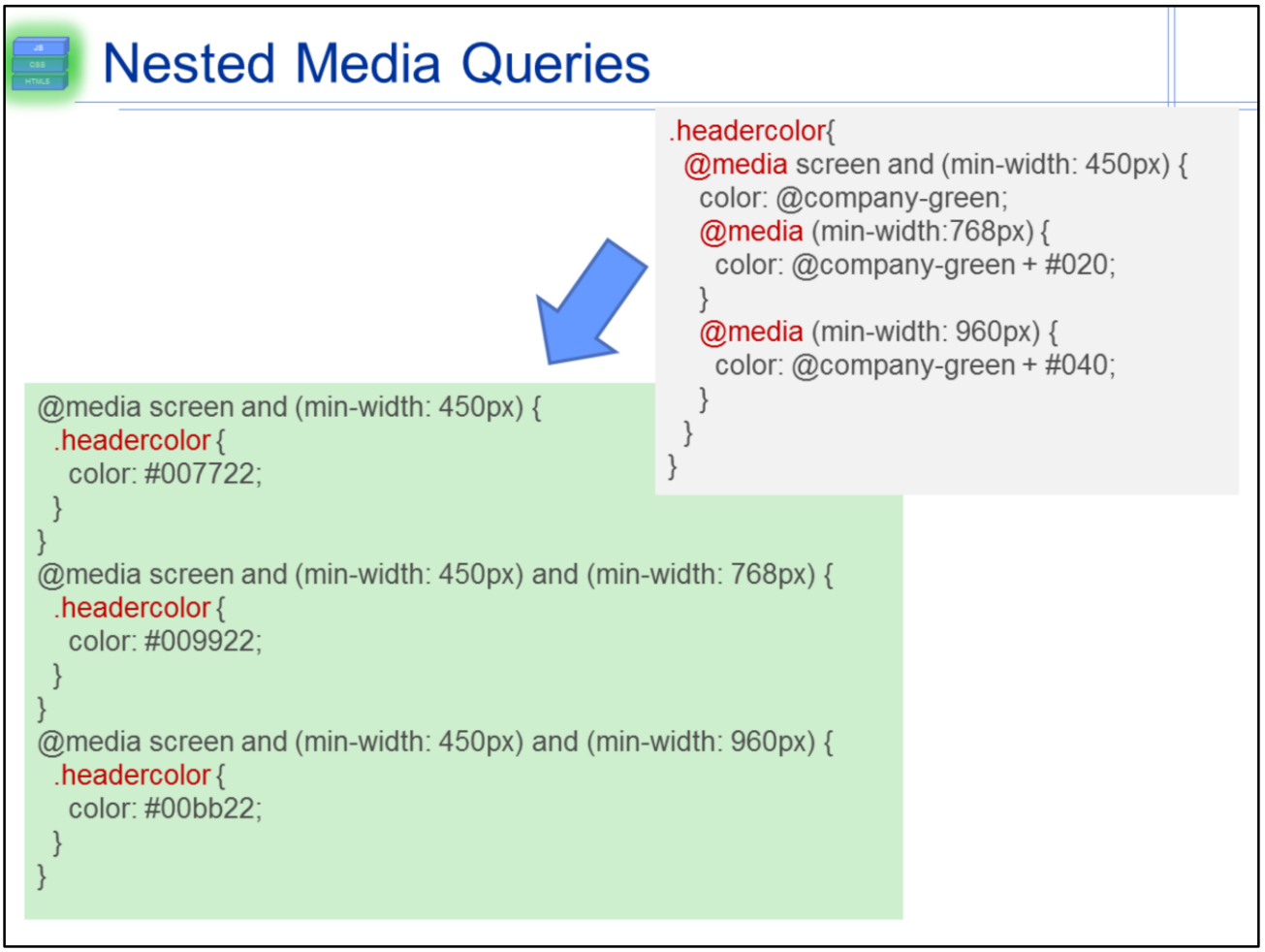


LESS variables, which are really more like constants can then be used throughout the LESS CSS.

LESS mixins follow the class creation format. Since classes are re-usable, this is also how LESS defines them. You may reuse classes in your CSS. Be sure to put them last. If something follows the class, it will be interpreted as being nested inside the class.



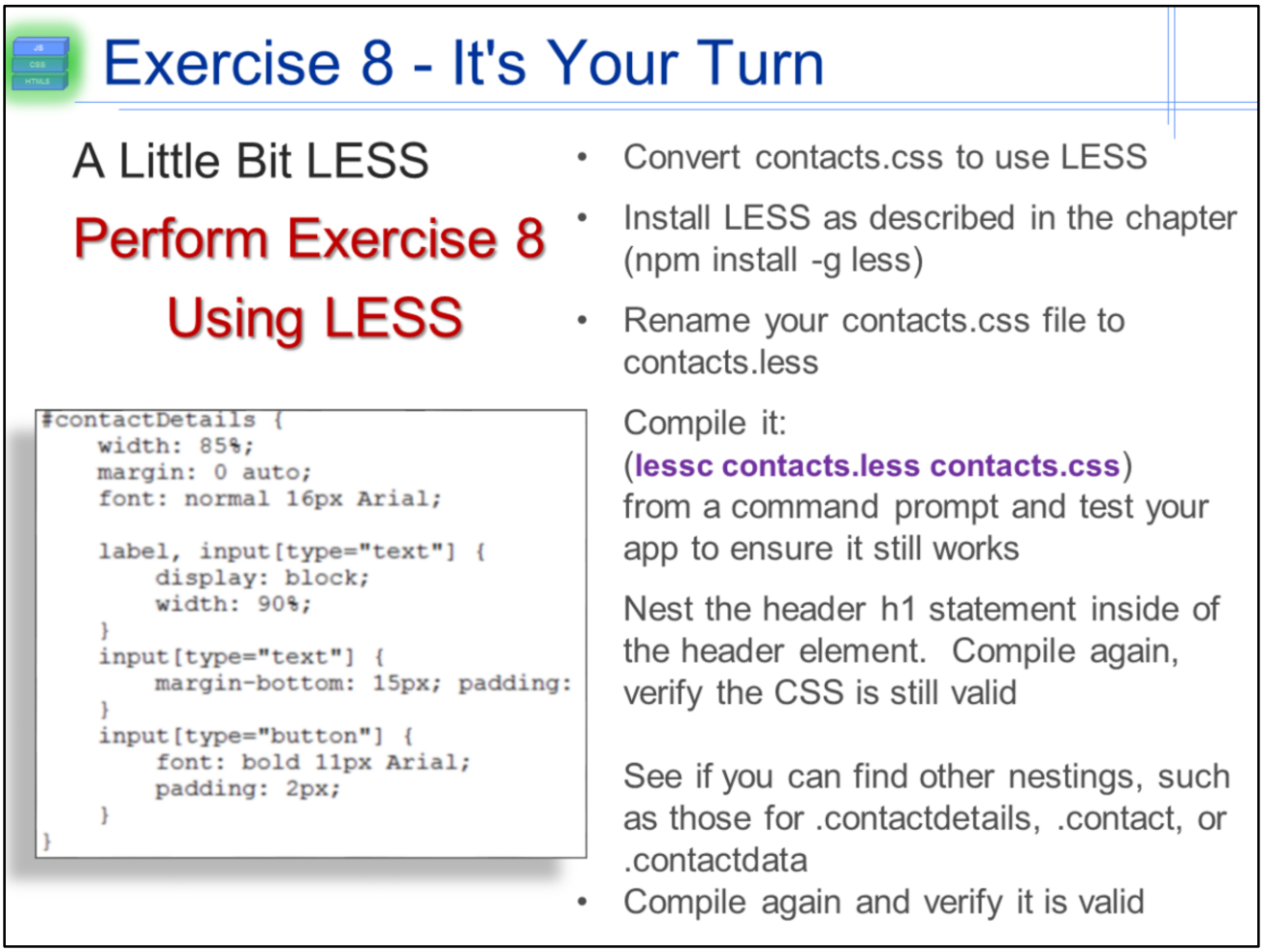
Here, as in SASS, rules may be nested and will compile into separate rules.



Care should be given to some of the rule systems within LESS. Readability should be kept in mind. Is the LESS version more readable in this case?

Summary

- CSS Pre-processing frameworks are becoming more and more popular, particularly in enterprise application development environments
- Neither SASS nor LESS enjoys any particular advantage over the other
- Choice of one of these systems is generally made based on current operating environments (does another framework you are using use SASS or LESS?, do you already use NodeJS?, etc.)



Note: the LESS install requires an internet connection.

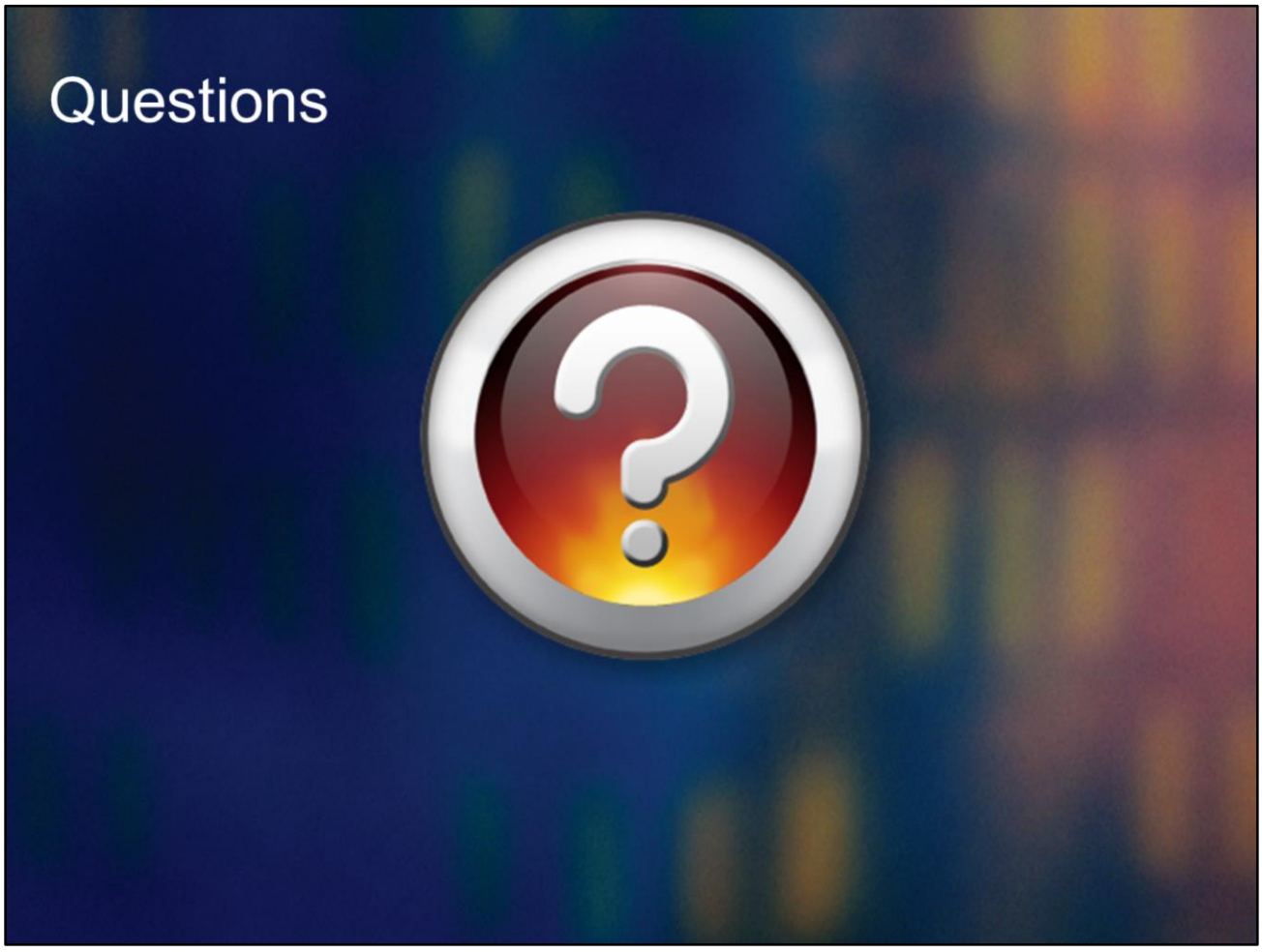
Course Summary

What Did We Learn?

- Layered Architectures
- Semantic HTML
- Browser Developer Tools
- CSS/CSS3 Selectors
- Specificity
- Absolute/Relative Positioning
- Floating, Clearing
- Laying out Pages
- CSS3 Colors, Gradients, Rounded Corners, Transforms
- Responsive Web Design
- CSS Media Queries
- Scalable Fonts
- Scalable & Adaptable Images
- Flexible Grids
- Unit Systems (relative, root-em)
- Transforming Tables
- CSS Fluid Grid Systems
- Working with Bootstrap
- CSS Pre-processing Solutions
- SASS and LESS

Evaluations

- Please take the time to fill out an evaluation
- All evaluations are read and considered



Modern CSS Techniques

Exercise Workbook

Exercise 1 - Browser Developer Tools

Overview

This exercise is designed to give you practice using either Chrome Developer Tools or Firebug.

Objectives

At the conclusion of this exercise, you should:

- Be able to select, view, modify/edit various CSS elements on a page

Step 1. Open Your Developer Tools *(as needed)*

Open either a Chrome Browser or Firefox Browser. Browse to **ESPN.com** (it will redirect you, that's okay).

Press F12 (or CTRL-SHIFT-I will also work in Chrome) to bring up Firebug or Chrome Developer Tools.

Note: If you do not have Firebug installed, you may either: 1) install it from getfirebug.com, or 2) switch to Chrome. Optionally, you may also use Firefox Developer Tools.

Step 2. Locate Elements

On the ESPN.com page, locate the menu bar that looks like the following:

Use the *inspector tool* in Chrome or Firebug to **select the MLB item**. At its innermost level, what kind of element is this?

Mouse over the parent element (the anchor) in the HTML source tree at the bottom in the developer tools. It should highlight in the page up above.

What color(s) do you see when it highlights in the page above?

You should see 2 colors. In Firebug these will be light blue, purple (there is also yellow--not present in this case).

In Chrome these colors are light blue, green(ish) (and orange, not present in this case).

What do these colors mean?

Step 3. Obtain Element Information

With the MLB anchor still selected, add a *margin-right* property to the first selector. Change it to **10px** and note the effect in the page.

What is the value of the padding? Why are there 2 values?

Add a font-size property to the first selector. Give it a value of 150%.

In the HTML source, select the <li> element (parent) that contains the anchor.

Turn off the *float: left* property associated with it. What happens? (floating is discussed later).

Turn the floating back on.

With the <li> still selected, view the Style(s) Tab opposite the HTML source in your developer tools.

In the Style(s) Tab, you should see CSS selectors that contain rules for the <li>. Find the selector that sets the float: left and line-height: 50px.

A selector is a shorthand notation for identifying one or more nodes. What selector did you see? What does the # (hash) mean? What does the . (period) indicate?

You should have answered an id attribute for the hash (#) and a class attribute for the period (.). Selectors will be discussed shortly.

Verify in the HTML source that the element does indeed match the selector by checking the id and class attributes.

Step 4. The Logo: A Last Task

Using the inspector tool, select the ***ESPN logo*** at the top of the page. What kind of element did you find?

The element should be an anchor <a> with the text of ESPN. Three questions:

1. Why don't we see this text (instead we see an image)? Where is the text? Hint: look at the CSS properties.

2. Why did they hide this text this way ?

3. Why did they wrap it inside of an <h1> ?

   Also, note: the <h1> has a transform property. Turn this off. What happens?

Answers to the previous questions:

1. Check the color (transparent) and font property (0px).
2. It is not intended to be seen the screen. It is there so that devices and machines (including screen readers) can understand the content.
3. Semantics. A text logo is typically the most important text. So, semantically, they want devices to understand what the most important text is on their page.

The logo is represented using an image. Yet there is no <img> tag here. Where is the logo image? Can you find it?

Hints:

- Look for background-images on the anchor

This concludes our Browser Developer Tools exercise. Feel free to use these newfound skills to explore further this page or other pages.

As you progress through this course, you should rely more and more on help from these tools and less on help from your instructor. Therefore, proper continued practice using these tools is essential!

Exercise 4 - Multi-column Layout

Overview

This exercise is designed to give you practice building a mult-column layout.

Objectives

At the conclusion of this exercise, you should:

- Be able to create any CSS layout using basic CSS properties

Step 1. Test the Solution First

This application provides actual data from the server. In order for this application to run properly, you must have the server running before testing it.

To run the server, open a command window, browse to the
<student_files_location>/nodejs folder.

Type the command: **node server.js**

You should see a message that your server is running on port 8005. (Assumes nodejs has been properly installed, a setup requirement of this class).

Now, in your browser, browse to localhost:8005/labs/lab04/solution/index.html.

You should see an approximation of what your solution should look like. It is fully functional in terms of the search and the table is populated with data from the server. The JavaScript used in this exercise is irrelevant to the presentation and can be ignored (but don't remove it!).

Step 2. Complete Your Solution

Your task is to create a 3 column layout such that the leading, trailing, and main divs are all columns.

Your choice of colors is up to you and largely unimportant.

Do not change the HTML or the order of this HTML as this might break the functionality.

You should only have to edit lab04/starter/contacts.css.

Some CSS already exists within this file, it styles the dialog box that appears at the bottom of this page. Don't worry about this dialog box, it will be styled in the NEXT exercise.

Tasks:

1. **Begin by getting your columns in place, don't forget about clearfix**
2. **Style the form (contactDetailsForm) in the trailing column.**
3. **Style the results div that displays search results.**
4. **Style the table, try to use alternating row colors. Hint: use tr:nth-child(odd)**
5. **Style the title in the <header>**
6. **Since our header and footer don't have any real content, you may artificially give them a height (100px is fine).**

Note: to test the search box, use contact ID values from 500 - 509.

Exercise 5 - CSS 3 Features

Overview

This exercise provides a chance to manipulate several CSS 3 features to observe their effects in an application.

Objectives

At the conclusion of this exercise, you should:

- Be able to create use gradients, rounded corners, and other CSS 3 features

Step 1. Style the Dialog Box

Note: if you wish to work from your previous solution, see the help file in the lab05/starter directory. It will help you with steps on how to use your lab04 solution that you created.

In this exercise, you should test the search dialog popup box. With the server still running, browse to **http://localhost:8005/labs/lab05/starter/index.html** and click the new "Contact Search" link. An ugly-looking dialog should pop up in the middle of the screen.

Your task: Using the provided contacts.css, style this dialog. You should incorporate a gradient for a background, rounded corners, and even a box-shadow if you wish. Your solution should vary from the solution found in the student files.