CSS and IMAGES

More on selectors, web graphics, and image manipulation

Lecture 6: Jan. 19 2016

Advanced Selectors

Attribute and Pseudo selectors

Attribute Selectors

Selector	Description
existence [attr] ex:`a[target]`	matches a specific attribute (regardless of value)
equality [attr="value"] ex:`a[target="_blank"]`	matches a specific attribute with a specific value (needs to match exactly)
substring [attr*="value"] ex:`img[alt*="art"]`	matches a specific attribute whose value contains a specific substring (the specified string of letters appears somewhere in the value)

```
selector[attribute] {
    property: value;
}
```

If you want to select elements based on their attributes, you can use an attribute selector.

See full list of attribute selectors.

Note: The difference between `h1#page-title` and `h1[id="page-title"]` is that the latter is more specific; with conflicting rules, the latter will override the former.

Pseudo-Classes

```
selector:pseudo-class {
    property: value;
}
```

A CSS pseudo-class is a keyword added to selectors that specifies a special state of the element to be selected.

The most popular use of pseudo selectors are in links. Browsers typically show text links in blue with an underline text decoration; visited links are purple. To customize this, you can target those elements using pseudo-classes.

```
a:link {
        color: red;
        text-decoration: underline;
}
a:visited {
        color: black;
}
a:hover {
        color: green;
        text-decoration: none;
}
a:active {
        color: blue;
        text-decoration: underline;
}
```

Note: The `:hover` class can be applied to any element, but will not work in mobile. (You cannot “hover” with your finger.)

Structural Pseudo-classes

Selector	Description

First child element
`:first-child`
 	
`li:first-child`
	matches the first child element, regardless of type, in a parent element

Last child element
`:last-child`
 	
`li:last-child`
	matches the last child element, regardless of type, in a parent element

Nth child element
`:nth-child( )`
 	
`li:nth-child(2)`
	matches a child element in relation to its position within the parent element — accepts numerals, formulas, and keywords

You can use pseudo selectors to target specific children of parent elements.

More on pseudo classes.

A common use is to style alternate rows in a table:

Table Header		
cell	cell	cell
cell	cell	cell
cell	cell	cell
cell	cell	cell

```
tr:nth-child(even){
  background: #EEE;
}
```

Pseudo-Elements

```
selector::pseudo-element {
    property: value;
}
```

Pseudo-elements allow you to style specific parts of an element or generate content around an element. More on pseudo-elements and things you can do with them.

CSS

```
p.note::after {
      content: url(images/icons/nemo.png);
}

p.note::before {
      content: "Note: ";
      font-weight: bold;
}
```

HTML

```
<p class="note">This is a
paragraph with a “note”
class. </p>
```

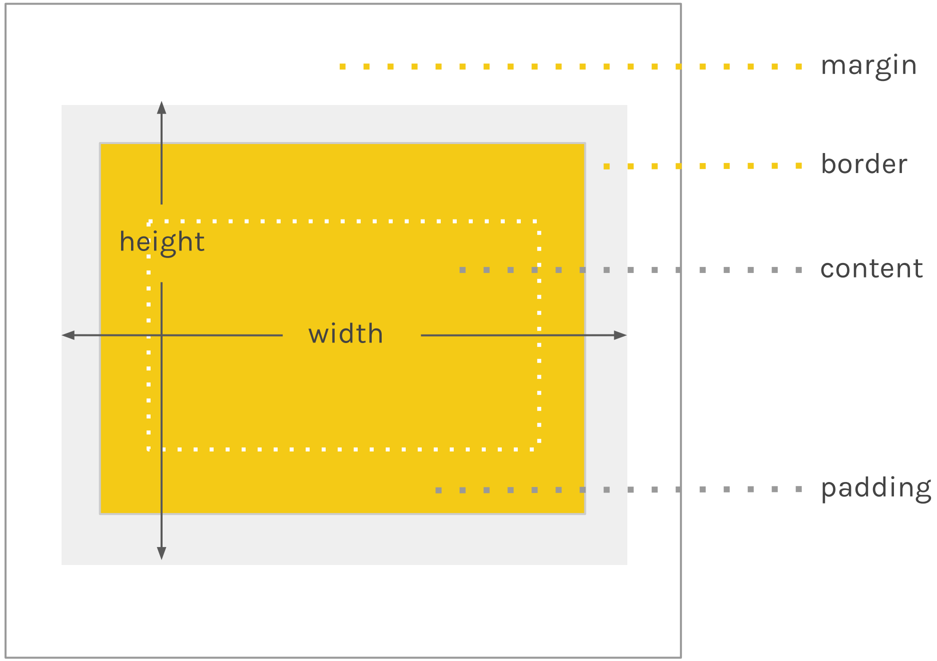
Output

Note: This is a paragraph with a “note” class.

CSS3

Using experimental properties

Box-Model Adjustment



If you found the box model (in which the padding and border are outside of the element width) to be unintuitive, you can use the `box-sizing` property. Setting this to `border-box` includes the padding and border width in the element width. More on box-sizing.

```
-webkit-box-sizing: border-box;
   -moz-box-sizing: border-box;
        box-sizing: border-box;
```

(old notation; prefixes no longer needed)

Vendor Prefixes and Browser Support

`-ms-`	Microsoft Internet Explorer
`-moz-`	Mozilla Firefox
`-o-`	Opera
`-webkit-`	Safari, Chrome, Android

- **Vendor prefixes** (such as `-webkit-` and `-moz-`)allow us to use CSS properties before they become part of official specifications.
- As the experimental properties become standardized, prefixes will no longer be necessary. If you'd prefer to keep your CSS prefix-free, you can use the prefix-free plugin and have the browser handle adding necessary prefixes or process them before uploading with Autoprefixer. More on the prefixing practice.
- caniuse.com helps map out browser support for these new properties. See also: Modernizr, a plugin that helps detect browser features to support older versions.

Responsive CSS

The following CSS properties and values may be useful for scalable designs.

- **Flex:** "Flex" is an experimental CSS property that allows for dynamic layouts. By declaring `display: flex` on a container element, child elements could be made to shrink or expand according to specified properties. See how it works and read more on flex-box properties.

- More relative length units:
    - **rem** font size of the root element (html) (ex. `font-size: 2rem;`)
    - **vw** 1% of viewport's width (ex. `width: 100vw;`)
    - **vh** 1% viewport's height (ex. `height: 100vh;`)

- **Columns** allows you to declare a column width or count and **column-gap** allows you to specify the gutters, flowing the content accordingly. (ex. `columns: 4; column-gap: 2.5rem;`) See it in action.

The Display Property

While `display: flex` introduced in CSS3 may be unreliable in older browsers, other `display` properties may be useful. See full list of properties.

- **inline** displays an element as an inline element (ex. `li{ display: inline; }`)
- **block** displays an element as a block element (ex. `img{ display: block; }`)
- **inline-block** displays an element as an inline-level block container. The inside of this block is formatted as block-level box, and the element itself is formatted as an inline-level box (ex. `div.columns{ display: inline-block; }`) and allows you to define a width and height. This is often used as an alternative to using `float`.
    - Note: Because it is an inline element, white spaces between elements are treated in the same way as white spaces between words of text. You may get unwanted gaps appearing between elements.

Shapes and Images

CSS Graphics, SVG, and Image manipulation

CSS Shapes

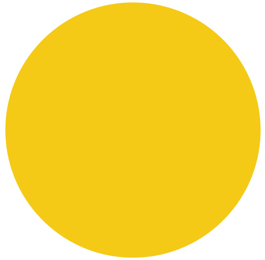
By creatively using CSS properties, we can generate shapes. This may aid in avoiding non-scalable image use.

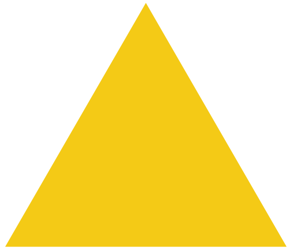
For example, the `border-radius` property allows you to round the corners of block elements. These may not necessarily need to be the same across all corners.

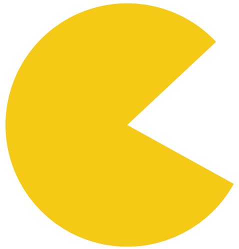
```css
.button {
     border-radius:  15px;
}
```

```css
.button {
     border-radius:  15px 15px 0 0;
}
```

Border uses







```
.circle {
      background: #f4ca16;
      height: 100px;
      width: 100px;
      border-radius: 50%;
}
```

```
.triangle {
      width: 0;
      height: 0;
      border-left: 50px solid transparent;
      border-right: 50px solid transparent;
      border-bottom: 100px solid #f4ca16;
}
```

```
.pacman {
  width: 0px;
  height: 0px;
  border-right: 50px solid transparent;
  border-top: 50px solid #f4ca16;
  border-left: 50px solid #f4ca16;
  border-bottom: 50px solid #f4ca16;
  border-radius: 50px;
 }
```

See extensions of this from John Caserta's programming class

SVG

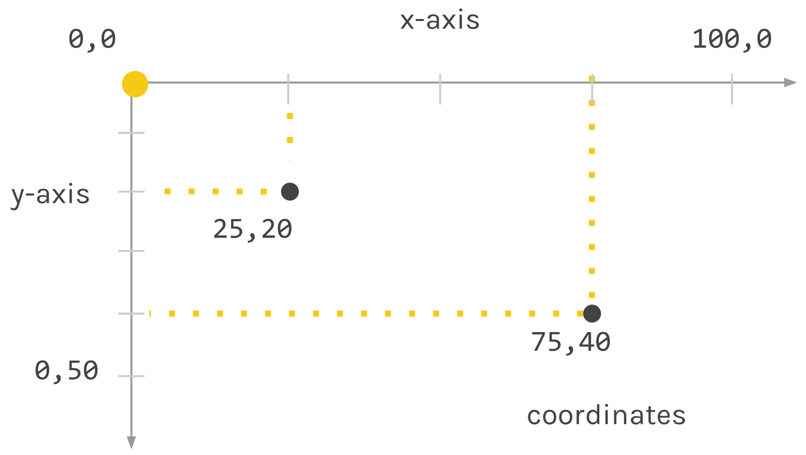
Scalable Vector Graphics is an image format defined using markup code similar to HTML. SVG code can be included directly within any HTML document. Every web browser supports SVG, except IE 8 and older.

Before you can draw anything, you must create an SVG element. Think of the SVG element as a canvas on which your visuals are rendered.

```
<svg width="500" height="50">...</svg>
```

If the width and height are not specified, the SVG will take up as much room as it can within its enclosing element.

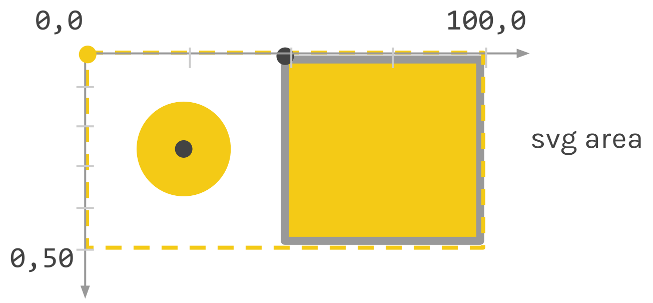
Drawing SVGs



SVG usually consist of `rect`, `circle`, `ellipse`, `line`, `text`, and/or `path` elements. They are drawn on a plane with reference to the coordinate origin at the top left corner of the defined SVG area. Multiple shapes can be combined into one SVG image.

SVGs components can also be styled in your stylesheet (for inline SVGS, but not with CSS properties.) This is especially useful to manipulate data-generated SVGs. More on SVG shapes.

SVG Syntax



```
<svg width="100" height="50">

                center point     radius

     <circle cx="25" cy="25" r="10"/>
     <rect x="50" y="0" width="50" height="50" class="box"/>

          top-left coordinates         size

</svg>
```

```
svg {
    fill: #f4ca16;
 }

svg .box {
    stroke: #666666;
    stroke-width: 5;
 }
```

SVG exports

```
<svg id="Layer_1" data-name="Layer 1" xmlns="
http://www.w3.org/2000/svg" viewBox="0 0 217.19
218.87"><title>RISD</title><path d="M117.
44,190.47c-36.47,0-69.77-31.5-69.77-70.93S77.
91,46.94,118,47c41.31,0.11,72,32.59,72,72s-
31.48,71.4-72.55,71.4h0Zm0.48-148.57C75,
41.79,42.57,77.34,42.57,119.6s35.68,76,74.77,76
c44,0,77.75-34.26,77.75-76.52S162.
18,42,117.92,41.91h0Zm-0.86,179.94c-52.42,0-
100.28-45.28-100.28-102S60.
25,15.52,117.84,15.68c59.
37,0.16,103.51,46.85,103.51,103.53S176.
11,221.84,117.06,221.84h0ZM117.77,9.33C56.
62,9.16,10.47,59.77,10.47,120S61.
29,228.2,116.94,228.2c62.68,0,110.72-
48.78,110.72-109S180.8,9.5,117.77,9.33h0Z"
transform="translate(-10.47 -9.33)"/><path d="
M73.42,188.76a21,21,0,0,1-.36,5l1.39,0.76,2-
3.37c0.56-.94-0.41-1.58,0-2.17a0.
47,0.47,0,0,1,.66-0.1l4.75,2.78a0.
47,0.47,0,0,1,.2.64c-0.34.53-1.25-.07-1.87,1l-
6,10.16c-0.65,1.14.14,2-.08,2.32a0.
45,0.45,0,0,1-.65.08l-4.85-2.9-1.88-1.21c-2.41-
1.54-3.92-4.2-2.22-6.91a5.33,5.33,0,0,1,4.89-
2.23,40.28,40.28,0,0,0-.2-6.48c-0.08-.65-0.88-
2.07-0.71-2.42h0.3a11.
43,11.43,0,0,1,2.37,1.11,4.79,4.79,0,0,1,2.31,4
h0Zm-3.23,12.31a0.24,0.24,0,0,0,.32-0.05l2.7-
4.43c-2.42-1.21-3.42-1.66-5,.78a2.
37,2.37,0,0,0,.63,2.88l1.32,0.82h0Z"
```

Programs such as Adobe Illustrator can export vector graphics into SVGs.

Logos, for example, can be exported as SVGs for scalable web use.

Adding SVGs

SVGs can be placed inline, inserted as an image, object, or background-image (each method has varying browser support.) As long as the `viewBox` attribute is defined in the SVG, and the SVG area isn't specified in absolute widths, the image coordinates can be scaled to fit its container.

ex.`<img src="img/RISD.svg" style="height: 80vh;">`

More on placing methods and scaling SVGs.

Backgrounds

Use background images for non-essential image content.

```
div {
    background: url(img/bgimg.jpg) no-repeat top center;
}
```

shorthand for

syntax	description	possible values
`background-image: url(img/bgimg.jpg);`	path to image, relative to css file	absolute / relative paths
`background-repeat: no-repeat;`	whether to tile, and along what axis	`repeat-x, repeat-y, repeat, no-repeat`
`background-position: top center;`	horizontal and vertical position	keywords / coordinates in any length unit

Background Scaling

The **background-size** CSS3 property may be useful for scaling background images for responsive layouts. See an example that also includes syntax for multiple backgrounds.

`background-size: cover;`
scales the background image to be as large as possible so that the background area is completely covered by the background image. Some parts of the background image may not be in view within the background positioning area.

`background-size: contain;`
scales the image to the largest size such that both its width and its height can fit inside the content area

Background Gradients

```
.gradient {
     height: 100px;
     background-color: red;      fallback
     background-image:
          linear-gradient(
             to right,
             red,
             #f06d06,
             rgb(255, 255, 0),
             green
          );
}
```

CSS3 allows you to create gradients as your background image. Colorzilla has a gradient editor to help generate the syntax with vendor prefixes. More on CSS gradients.

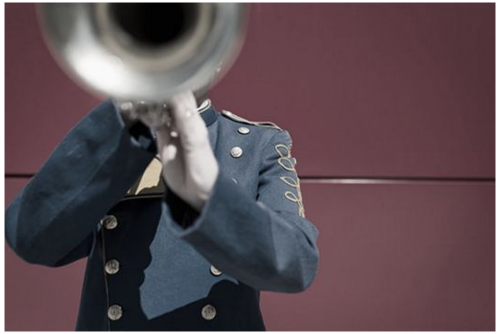
Combined with the `-repeating-` property and `background-blending-mode` properties, it's possible to create intricate patterns just in CSS.

Filters

#nofilter

CSS filters allow you to add “filter” effects, as you might adjust an image in Photoshop.

See a list of filter effects.



Gratisography

```
img {
     filter: grayscale(.8);
}
```

value between 0 and 1
determines strength of filter

CSS and Images: Review

You should now have an understanding of how to:

- ❏ Target elements with advanced selectors
- ❏ Use pseudo-classes and pseudo-elements
- ❏ Revisit techniques for responsive layouts
- ❏ Incorporate new CSS3 techniques
- ❏ Draw graphics in the browser
- ❏ Manipulate images