FRONT-END DEVELOPMENT PROGRAM

Lecture 01: CSS Intro

BeaconFire

OUTLINE

- CSS Selectors
- Fundamentals (Inheritance, Cascade, Specificity, Variables, Color, Units)
- Basic Properties (Display, Visibility, Text, Font, Icons)
- Box Model (Content, Padding, Border, Margin)

CSS

- Cascading Style Sheets (CSS) is a style sheet language used to describe the way that HTML elements should be displayed on screen.
- External stylesheets are stored in .css files and can be imported into an HTML document. This lets us reuse styles across several distinct webpages.

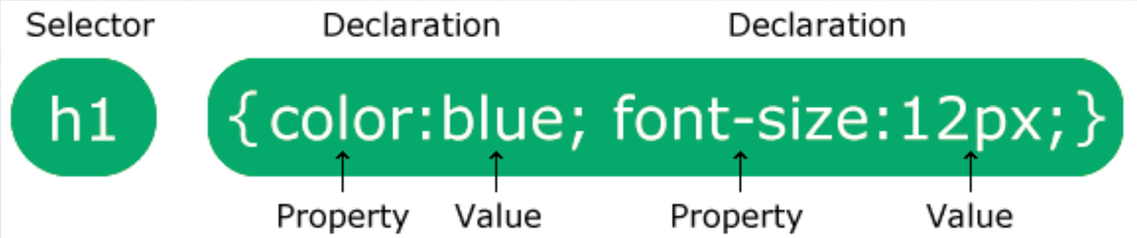
INSERTING CSS

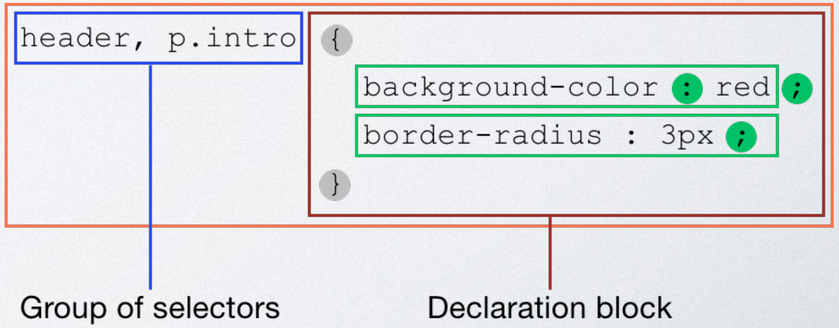
- **External CSS**: *<link rel="stylesheet" href="CSS/style.css">*
- **Internal CSS**: *<style type="text/css">*
- **Inline CSS**: *<h3 style="color:blue">Inline CSS </h3>*

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>

  <link rel="stylesheet" href="CSS/style.css" />
  <style type="text/css">
    /* CSS Comments */
  </style>
</head>

<body>

  <h3 style="color: blue">Inline CSS</h3>
```

CSS SYNTAX

- **Selector**: refers to words that you use to target HTML elements to apply styles to
- **Declaration**: includes a CSS property name and its value, separated by colon (:), ending in semi-colon (;)
  https://www.w3schools.com/cssref/default.asp
- **Declaration block**: contains multiple declarations, wrapped in curly braces { }
- **Ruleset/rule**: a selector group & their declaration block





CSS SELECTORS

- **Simple selectors:** select elements based on tag name, id, class, etc.
- **Combinator selectors**: use combinators to select elements based on their relationship (child or descendant elements)
- **Pseudo-class selectors**: select elements based on their state (hover)
- **Pseudo-element selectors**: select part of an element (first character)
- **Attribute selectors**: select elements based on attributes or their values

SIMPLE SELECTORS

- **Universal (*)**: apply styles to all elements
- **Element (tag)**: apply styles to all elements with that tag (div)
- **Class (.class)**: apply styles to all elements with that class attribute
- **ID (#id)**: apply styles to the element with that **unique** id
- You can apply a declaration block to multiple selectors **(,)**.
- You can chain selectors to be more specific about the elements.

```
<style type="text/css">
  * {
    background-color: gainsboro;
  }

  h1 {
    color: red;
  }

  .h1 {
    color: #4caf50;
  }

  h2,
  h3 {
    color: black;
  }

  h1.h1.otherClass {
    color: purple;
  }

  #h1 {
    color: blue;
  }
</style>
</head>
<h1>Element/tag selector</h1>
<h1 class="h1">Class selector</h1>
<h1 class="h1 otherClass">Chained (tag, class, id) selector</h1>
<h2>Grouped tag selector h2</h2>
<h3>Grouped tag selector h3</h3>
<h1 id="h1">Id selector</h1>
```

Element/tag selector

Class selector

Chained (tag, class, id) selector

Grouped tag selector h2

Grouped tag selector h3

Id selector

COMBINATOR SELECTORS

- **Combinator**: a symbol that defines the relationship between two selectors
- **Child (s1 > s2)**: look for all elements that match s1, select all children that match s2
- **Descendant (s1 s2)**: look for all elements that match s1, select all descendants that match s2
- **Adjacent sibling (s1 + s2)**: look for all elements that match s1, select the first following sibling that matches s2
- **General sibling (s1 ~ s2)**: look for all elements that match s1, select all following siblings that match s2

```
  <style type="text/css">
    .list>li {
      color: green;
    }

    .list li {
      background-color: yellow;
    }

    li+ol {
      color: purple;
    }

    li~span {
      color: blue;
    }
  </style>
</head>

<body>
  <ul class="list">
    <li>Li-1</li>
    <li>Li-2</li>
    <ol>
      <li>Ol-1-Li-1</li>
    </ol>
    <li>Li-2</li>
    <ol>
      <li>Ol-2-Li-2</li>
    </ol>
    <span>Span1</span>
    <span>Span2</span>
  </ul>
</body>
```

- Li-1
- Li-2
  1. Ol-1-Li-1
- Li-2
  1. Ol-2-Li-2

Span1 Span2

PSEUDO-CLASS & PSEUDO-ELEMENT SELECTORS

- **Pseudo-class (:)**: defines a special state of an element (hover, focus)

  https://www.w3schools.com/css/css_pseudo_classes.asp

- **Pseudo-element (::)**: defines a specific part of the selected element (first letter/line, insert content before/after element content)

  https://www.w3schools.com/css/css_pseudo_elements.asp

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

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

```
  <style type="text/css">
    /* Pseudo-classes */
    dt:first-child {
      color: red;
    }
    dt:nth-child(2n) {
      color: blue;
    }
    a:visited {
      color: purple;
    }
    a:hover {
      color: red;
    }
    a:active {
      color: green;
    }

    /* Pseudo-elements */
    p::first-letter {
      font-size: 50px;
    }
    p::selection {
      background-color: red;
    }
  </style>
</head>

<body>
  <dl>
    <dt>1</dt>
    <dt>2</dt>
    <dt>3</dt>
    <dt>4</dt>
    <dt>5</dt>
  </dl>

  <a href="#">LINK LINK LINK</a>

  <p>paragraph</p>
</body>
```

BeaconFire

ATTRIBUTE SELECTOR

- **Attribute selector ([attr])**: apply styles to all elements that have this attribute or attribute value
- [**attr**]: select all elements that have an attribute named "attr".
- [attr**=**value]: select all elements with an attribute named "attr" whose value **is** "value"
- [attr**^=**value]: select all elements with an attribute named "attr" whose value **starts with** "value"
- [attr**$=**value]: select all elements with an attribute named "attr" whose value **ends with** "value"
- [attr***=**value]: select all elements with an attribute named "attr" whose value **contains** "value" **at least once**

```
<style type="text/css">
  button[value] {
    background-color: red;
  }

  button[value="Submit"].btn {
    color: white;
  }

  /* Internal links, beginning with "#" */
  a[href^="#"] {
    background-color: gold;
  }

  /* Links with "example" anywhere in the URL */
  a[href*="example"] {
    background-color: silver;
  }

  /* Links with "insensitive" anywhere in the URL, regardless of capitalization */
  a[href*="insensitive" i] {
    color: blue;
  }

  /* Links with "cAsE" anywhere in the URL, with matching capitalization */
  a[href*="cAsE"] {
    color: pink;
  }

  /* Links that end in ".org" */
  a[href$=".org"] {
    color: red;
  }

  /* Links that start with "https" and end in ".org" */
  a[href^="https"][href$=".org"] {
    color: green;
  }
</style>
```

Submit Redo

- #internal
- http://example.com
- #InSensitive
- #cAsE
- http://example.org
- https://example.org

```
a[target] {
  background-color: yellow;
}
```

```
<body>
  <button value="Submit">Submit</button>
  <button value="Redo">Redo</button>

  <ul>
    <li><a href="#internal">#internal</a></li>
    <li><a href="http://example.com">http://example.com</a></li>
    <li><a href="#InSensitive">#InSensitive</a></li>
    <li><a href="#cAsE">#cAsE</a></li>
    <li><a href="http://example.org">http://example.org</a></li>
    <li><a href="https://example.org">https://example.org</a></li>
  </ul>
</body>
```

INHERITANCE, CASCADE, SPECIFICITY

- **Inheritance**: some styles are inherited throughout the HTML document tree (font properties, text color)
  - The **inherit** keyword gives a declaration the same value as its parent.
  - http://www.javascriptkit.com/dhtmltutors/cssreference.shtml

- **Cascade**: an algorithm that determines which style/stylesheet to prioritize and apply to a document (think, which is closest to the element?)
  1: inline style (author)
  2: internal style (author)
  3: external style (author)
  4: online external style (user)
  5: default browser

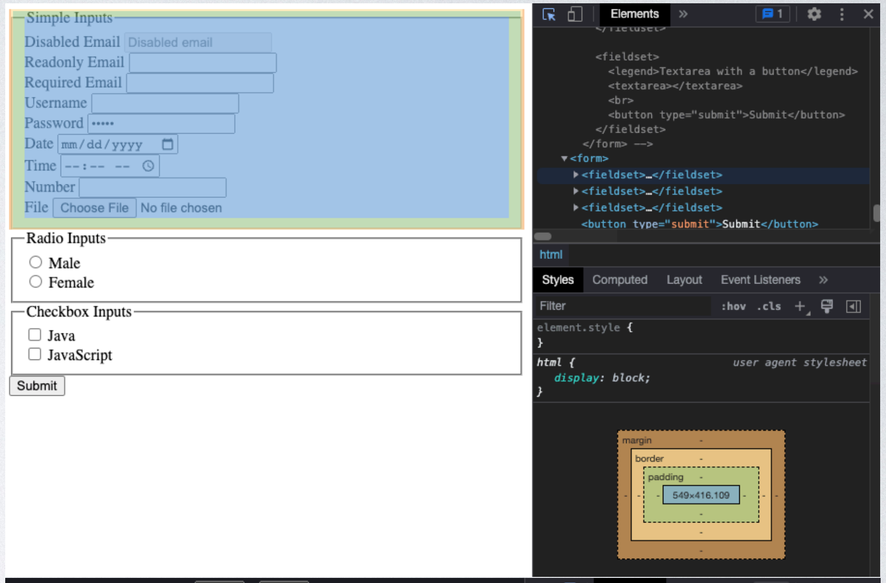
- **Specificity**: an algorithm that determines which rules to prioritize and apply to an element, represents how specific a selector is
  1: id selectors
  2: class & pseudo-class selectors
  3: element selectors

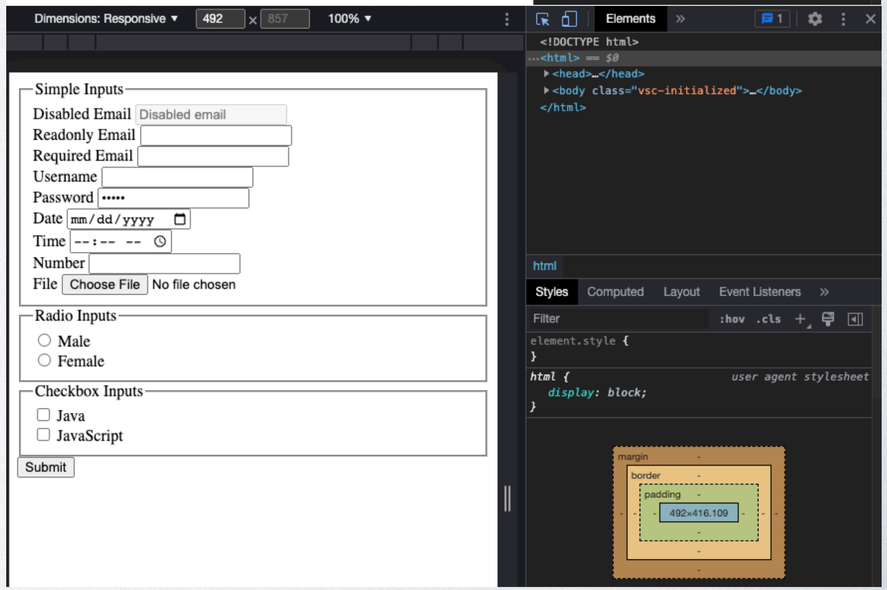
- **ORDER MATTERS**: In the same stylesheet, if two rules have the same specificity, we apply the rule that was declared **last.**
  - In all cases, the applied rule doesn't overwrite all declarations, only those that conflicted.
  - Inline styles are the highest priority & specificity doesn't apply (you don't use selectors).
  - The **!important** keyword makes a declaration have the highest specificity, so it overrides all other declarations (be careful).

CHROME DEV TOOLS: INSPECT

- Inspect elements on the webpage to debug elements, test layouts on different screen sizes, temporarily edit CSS, view the console, etc.

- How?
  - Right-click > Inspect
  - MacOS: Command + Shift + C
  - Windows: Control + Shift + C





VARIABLES

- Variables are useful for storing colors, which can be reused throughout the stylesheet and be easily updated.

- Variables can have a global scope (used throughout the entire document) or a local scope (only inside the selector where it is declared).
  - **Global**: declared in the :root selector
  - **Local**: declared in any selector

- They must start with "--" and are case-sensitive. Not supported in IE

```
<style type="text/css">
  :root {
    --blue: #6495ed;
  }

  body {
    background-color: var(--blue);
  }

  h3 {
    --white: #faf0e6;
    color: var(--white);
  }
</style>
</head>

<body>
  <h3>Inline CSS</h3>
</body>
```

Inline CSS

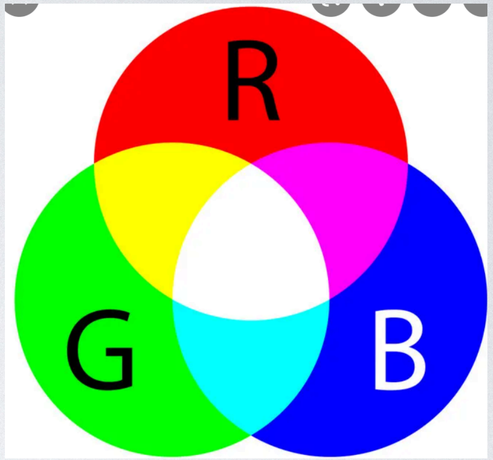
COLOR

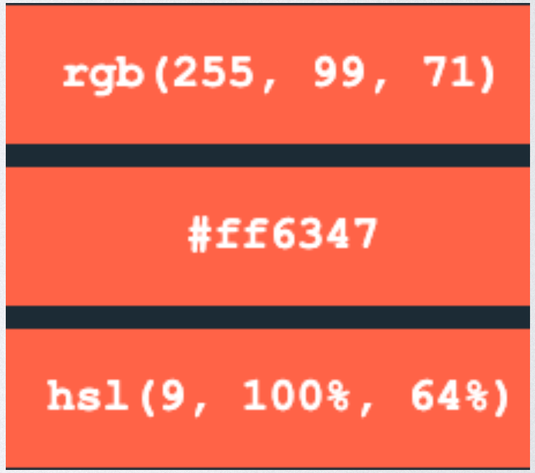
- Colors are specified using predefined color names, or values for RGB, HEX, HSL, RGBA, HSLA.
  - https://www.w3schools.com/cssref/css_colors.asp
  - https://www.w3schools.com/css/css_colors.asp
  - https://www.w3schools.com/css/css3_colors.asp

- **RGB**: (red, green, blue) <=> (0, 0, 0) - (255, 255, 255)
  RGBA: (…, alpha) <=> (…, 0-1), transparency

- **HEX**: #rrggbb <=> #000000 - #ffffff

- **HSL**: (hue, saturation, lightness) <=> (0, 0%, 0) - (360, 100%, 100%)
  HSLA: (…, alpha) <=> (…, 0-1), transparency





BeaconFire

CSS UNITS

- Many CSS properties (width, margin, padding) take "length" values which are numbers followed by units.
  - Cannot have whitespace between the number and unit
  - If a length of 0, you can omit the units
  - https://www.w3schools.com/css/css_units.asp

- **Absolute**: fixed lengths and not recommended for screens since size varies
  - **px**: pixels, relative to the viewing device

- **Relative**: lengths are relative to other lengths and scale better between screen sizes
  - **rem**: a factor of the root element's font-size **(default: 16px)**
  - **em**: on font-size, it's a factor of the parent element's font-size
    - for all other properties, it's a factor of the selected element's font-size
  - **vw** & **vh**: viewport width & height, specified as a percentage

BASIC PROPERTIES

- Display & Visibility
- Text
- Font
- Icons

DISPLAY & VISIBILITY

- **display**: specifies how an element should be displayed
  - **none:** element is invisible and doesn't take up space on the page
  - **block**: display like a block element
  - **inline**: display like an inline element
  - **inline-block**: no line break, affected by top & bottom margin and width & height (block element without line break)

- **visibility**: specifies whether or not an element should be visible
  - **hidden**: element is invisible but takes up space on the page
  - **collapse**: for table rows, removes them without affecting table layout

```html
<!-- display: none vs visibility: hidden -->
<h4 style="display: none">Hidden Header</h4>
<h4 style="visibility: hidden">Hidden Header</h4>

<!-- display: block vs inline -->
<h4>Header <span style="display: block">span display:block</span></h4>

<h4>
  Header
  <div style="display: inline; height: 30px; border: 1px solid blue">
    div display: inline
  </div>
  <div style="display: inline-block; height: 30px; border: 1px solid red">
    div display: inline-block
  </div>
</h4>
```

Header
span display:block

Header **div display: inline** **div display: inline-block**

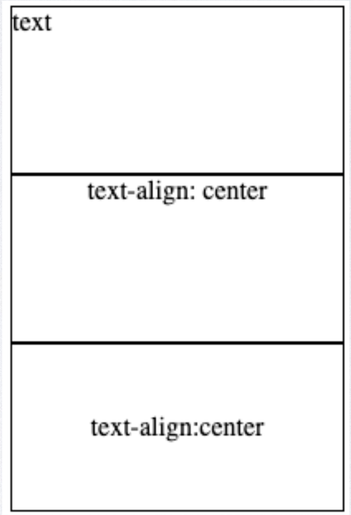
TEXT

- **text-align**: specify the horizontal alignment of text in an element
  - left (usually default), right, center, justify

- **line-height**: specify the height of a line with a length
  - set this equal to the element's height to vertically center text
  - https://www.w3schools.com/css/css_align.asp

- Other ways to style text: https://www.w3schools.com/css/css_text.asp
  - Decorate with overline/line-through/underline
  - Transform characters into uppercase, lowercase, capitalization
  - Add spacing between characters, words, lines
  - Text shadows or special effects

```
<div style="border:1px solid black; height:100px; width: 200px;">text</div>
<div style="border:1px solid black; height:100px; width: 200px; text-align: center;
">text-align: center</div>
<div style="border:1px solid black; height:100px; width: 200px; text-align: center;
line-height: 100px;">
  text-align:center</div>
```



FONT

```
body {                                                    dls-lite_ce…a6c84.css:1
  font-family: Circular, -apple-system, BlinkMacSystemFont, Roboto, Helvetica
     Neue, sans-serif;
  font-size: 14px;
  line-height: 1.43;
  color: #484848;
  background-color: #fff;
  margin: 0;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
```

- **font-family**: CSS has 5 generic font families; all fonts come from one of these families.
  - **serif**: the edges of each letter have small strokes, "formal, elegant"
  - **sans-serif**: clean lines without small strokes, "modern, minimal"
  - **monospace**: all letters have equal, fixed with, "mechanical"
  - **cursive**: like human handwriting
  - **fantasy**: decorative/playful

- **font**: family, size, style, variant, weight
  - https://www.w3schools.com/css/css_font_shorthand.asp

- **Web-safe fonts**: universally installed across all browsers and devices

- **Fallback fonts**: backup fonts provided to the "font-family" property to ensure maximum cross-browser & operating system compatibility

- Google fonts: https://fonts.google.com/

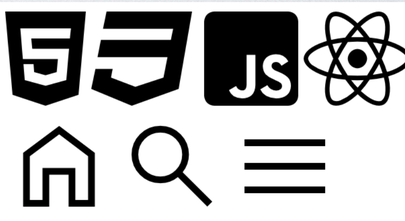
Generic Font Family	Examples of Font Names
Serif	Times New Roman Georgia Garamond
Sans-serif	Arial Verdana Helvetica
Monospace	Courier New Lucida Console Monaco
Cursive	Brush Script MT Lucida Handwriting
Fantasy	COPPERPLATE Papyrus

ICONS

- Font Awesome
  - https://cdnjs.com/libraries/font-awesome
  - https://fontawesome.com/icons

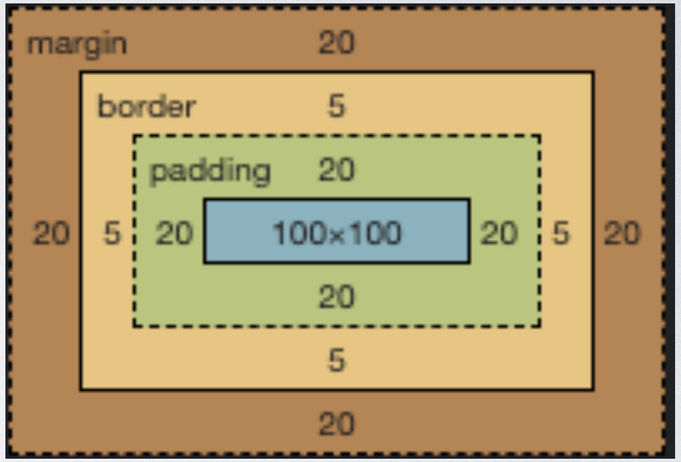
- Material Icons
  - https://fonts.google.com/icons

```html
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.2/
  css/all.min.css"
    integrity="sha512-1sCRPdkRXhBV2PBLUdRb4tMg1w2YPf37qatUFeS7zlBy7jJI8Lf4VHwWfZZfpXtYSLy8
    5pkm9GaYVYMfw5BC1A=="
    crossorigin="anonymous" referrerpolicy="no-referrer" />

  <link rel="stylesheet"
    href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,
    FILL,GRAD@48,400,0,0" />
  <style type="text/css">
    i {
      font-size: 5rem;
    }

    .material-symbols-outlined {
      font-size: 5rem;
    }
  </style>
</head>

<body>
  <i class="fa-brands fa-html5"></i>
  <i class="fa-brands fa-css3"></i>
  <i class="fa-brands fa-square-js"></i>
  <i class="fa-brands fa-react"></i>

  <br>
  <span class="material-symbols-outlined">
    home
  </span>
  <span class="material-symbols-outlined">
    search
  </span>
  <span class="material-symbols-outlined">
    menu
  </span>
```

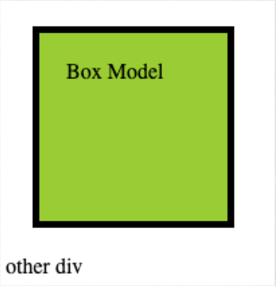


BOX MODEL

- Every HTML element has a layered structure, often referred to as the "box model". It's an important concept in website design and layout. You need to understand it to be able to set element widths and heights correctly.
- **Content**: the content of the element, where text and images usually appear
- **Padding**: the transparent space surrounding the content
- **Border**: the border surrounding the padding and content
- **Margin**: the transparent space outside the border, separating an element from other HTML elements




```
  <style type="text/css">
    #box {
      width: 100px;
      height: 100px;
      background-color: yellowgreen;
      margin: 20px;
      padding: 20px;
      border: 5px solid black;
    }
  </style>
</head>

<body>
  <div id='box'>Box Model</div>
  <div>other div</div>
</body>
```



BeaconFire

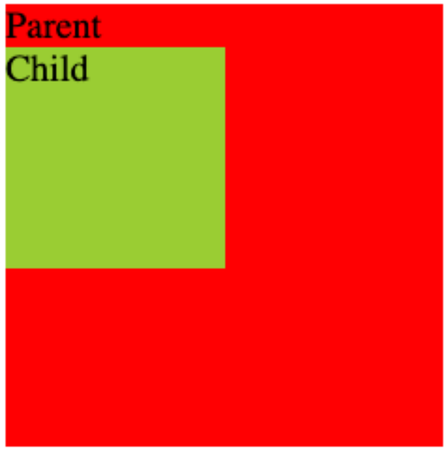
CONTENT

- **width & height**: usually sets the width and height of an element's content
  - **auto** (default): browser calculates it based on element content
  - **length**: some number and units
  - **n%**: a percentage of an element's parent/ container block width & height

- max-width/height & min-width/height

```css
#content-parent {
  background-color: red;
  /* Responsiveness: parent and child size depend on viewport/screen size */
  width: 50vw;
  height: 25vh;
}

#content {
  height: 50%;
  width: 50%;
}
```

```html
<div id="content-parent">
  Parent
  <div id="content">Child</div>
</div>
```



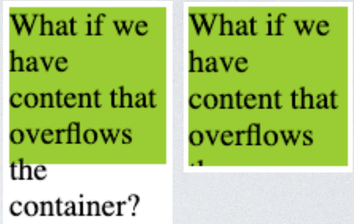
CONTENT: OVERFLOW

- **overflow**: affects the appearance of content too big to fit in the element
  - **visible** (default): excess content leaks out of the element
  - **hidden**: excess content is clipped
  - **scroll, auto**: excess content is clipped but can be scrolled

  https://www.w3schools.com/css/css_overflow.asp

```
.content-overflow {
  margin: 1em;

  /* overflow: visible; */
  /* overflow: scroll; */
  /* overflow: hidden; */
  /* overflow: auto; */
  overflow: visible;
}
```

```
<div class="content-overflow">
  What if we have content that overflows the container?
</div>
```




```
.box { overflow: scroll; }
```

Setting overflow to scroll will give you scrollbars BOTH horizontally and vertically. (even if they don't need it)

```
.box { overflow: auto; }
```

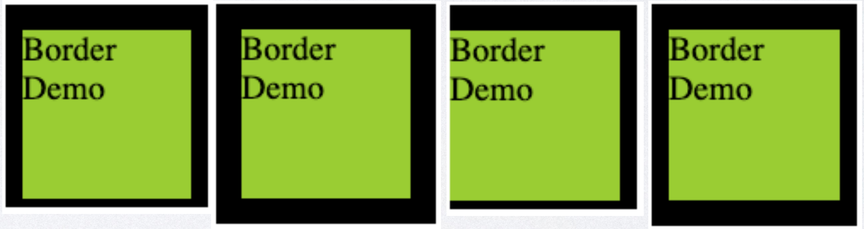
Auto overflow only gives the box the scroll bars it needs. (even none at all)

BORDER

- **border** (shorthand): width color style
  - style (required): dotted, dashed, solid, …

```
/* setting it with shorthand, all-around width */
border: 10px solid black;

/* setting individual border widths
border-width: 15px 10px 5px 0px;
border-width: 15px 10px 5px;
border-width: 15px 5px;
border-width: 15px
*/
```

- **border-width**: specify between 1-4 lengths
  - **4**: up, right, down, left
  - **3**: up, right & left, down
  - **2**: up & down, right & left
  - **1**: all-around



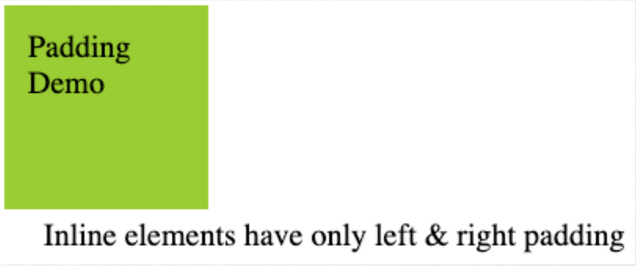
- **border-radius**: used to add rounded borders to an element, accepts a length

- border-color, border-style, border-top/right/bottom-left

PADDING

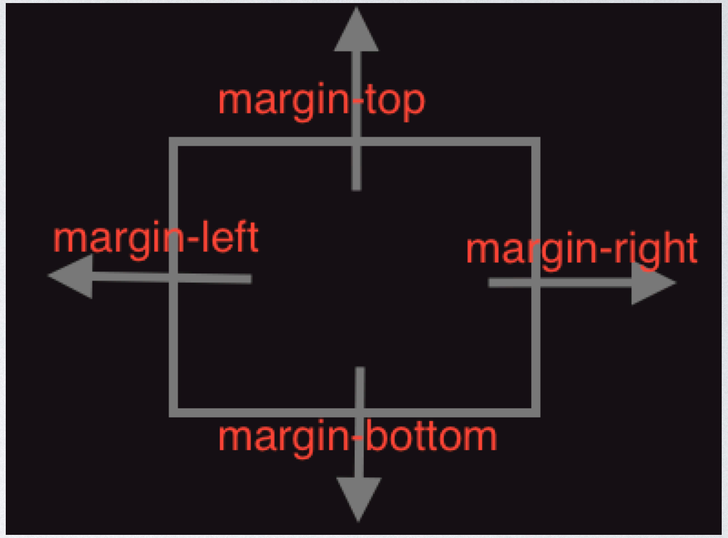
- **padding** (shorthand): can also set the sides in a clockwise fashion, following the same pattern as border - padding-top, padding-right, padding-bottom, padding-left
- Since padding is *transparent*, you see the content's background color where the padding is located.
- Inline elements don't have padding-top or padding-bottom.

```
<div id='padding'>Padding Demo</div>

<span style="padding-top: 30px; padding-left: 30px;">Inline elements have only left &
right padding</span>
```



MARGIN

- Margin can have positive or negative lengths. Positive lengths are like adding space to that side outside of the element, and negative lengths are like removing space from that side.



- top/left: moves the element in the opposite direction
  - **margin-top**: (+) moves the element down. (-) moves it up
  - **margin-left**: (+) moves the element right. (-) moves it left

- bottom/right: moves sibling elements in that direction
  - **margin-bottom**: (+) moves sibling elements down. (-) moves them up
  - **margin-right**: (+) moves sibling elements right. (-) moves them left

```
<div>Sibling Element Before</div>
<div id='margin'> Margin Demo</div>
<div id="margin-below">Sibling Element After</div>
```

- Sibling elements' margins (m1, m2) visually collapse based on the margin sign
  - **same sign (both + or -)**: margin length = max(|m1|, |m2|)
  - **different sign**: margin length = m1 + m2

MARGIN (2)

Parent width = child's content width + padding + border + margin

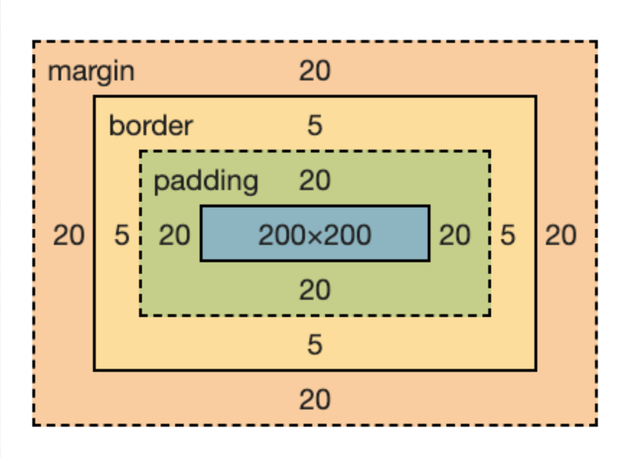
- The browser calculates a child element's margin based on its width and its parent's total size.
  * All sides of an element's margin are 0 (default).

- When width is constant, and:
  - left margin is 'auto' and right margin is not specified (0), **left margin fills the parent width.**
  - right margin is 'auto' and left margin is not specified (0), **right margin fills the parent width.**
  - both margins are 'auto', **margin-left = margin-right**.

- When width is 'auto' and both margins are unspecified or 'auto', **both margins are 0** (width 'auto' expands to fill the parent).

BOX-SIZING

- **box-sizing**: specify whether to include padding & border in element's total width/height
  - **content-box** (default): excludes, width & height applies to the content
  - **border-box**: includes, width & height applies to the border box



---

border box size = content + padding + border

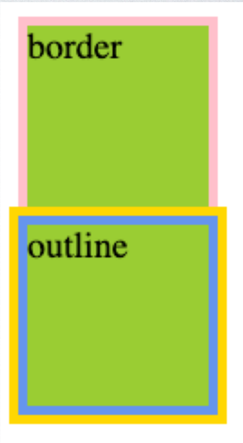
element size = content + padding + border + margin

```
*, *:before, *:after {                                  dls-lite_ce…a6c84.css:1
   -webkit-box-sizing: border-box;
   -moz-box-sizing: border-box;
   box-sizing: border-box;
   -ms-flex: 0 1 auto;
}
```

OUTLINE

- An outline is a line that is drawn around elements (outside the border) to make the element stand out, usually for highlighting text or focusing on inputs.
- The outline may visually overlap with other elements, since it's not part of the element's width & height.

```
  <style type="text/css">
    #border-demo {
      border: 5px pink solid;
    }

    #outline-demo {
      border: 5px cornflowerblue solid;
      outline: 5px gold solid;
    }
  </style>
</head>

<body>
  <div id='border-demo'>border</div>
  <div id='outline-demo'>outline</div>
</body>
```



BOX-SHADOW

- **box-shadow**: property attaches one or more shadows to an element (comma-separated). It can take several values.
  - **h-offset** (required, +/-): horizontal offset from the top-left corner
  - **v-offset** (required, +/-): vertical offset from the top-left corner
  - **blur**: makes the shadow blurry
  - **spread**: increases shadow range/size
  - **color**: shadow's color
  - **inset**: makes it an inner shadow

```html
<style type="text/css">
  #shadow-outset {
    margin: 1rem;

    /* values: h-offset v-offset blur spread color
    box-shadow: 20px 20px 10px 10px cornflowerblue;
    box-shadow: 20px 20px 10px 10px;
    box-shadow: 20px 20px 10px;
    box-shadow: 20px 20px;
    */
    box-shadow: 20px 20px 10px 10px cornflowerblue;
  }

  #shadow-inset {
    margin: 2rem 1rem;
    box-shadow: 20px 20px 10px 10px gold inset;
  }
</style>
</head>

<body>
  <div id="shadow-outset">Outset Shadow (Default)</div>
  <div id="shadow-inset">Inset Shadow</div>
</body>
```

BeaconFire

```
box-shadow: none|h-offset v-offset blur spread color |inset|initial|inherit;
```

ANY QUESTIONS?