Introduction to HTML and CSS

A Hands-on Course

April 3, 2018

Peter J. Jones

pjones@devalot.com

@devalot

http://devalot.com

DEVALOT

What's In Store

Before Lunch	After Lunch
HTML Syntax and Structure	Classes and Pseudo Classes
Semantic Tags and HTML5	Commonly Used Selectors
Most Commonly Used Tags	Background Images and Fonts
CSS Syntax and Structure	Flexible Box Layout
Basic Selectors and Properties	Responsive Design

Word Processors and WYSIWYG

- Many content creation tools are What You See is What You Get (WYSIWYG)
- Want some *emphasized* text? Select the text and click a button
- The tool will store your text with styling information

Markup Languages

- A markup language requires you to explicitly provide information along with the content:

```
<p>
  This is a paragraph with <em>emphasized</em> text.
</p>
```

What is HTML?

- Hyper Text Markup Language
- Uses “<”, “>”, and “&” as *control* characters
- To use those characters in your text they need to be encoded:

```
<p>
  2 &gt; 1
</p>
```

- Failure to encode these characters will result in errors (best case) or very **serious security issues** (XSS)

Features of HTML

- HTML is very error tolerant (browsers are very forgiving)
- That said, you should strive to write good HTML
- All about content and structure
- Parsed as a tree of nodes (A.K.A. elements or tags)

HTML is Semantic

Example semantic tags:

- `<p>`: Paragraph
- `<table>`: Table
- `<ul>`: Unordered list

These tags tell you something about their content. This is *very* important for accessibility.

Tags That Are Not Semantic

Compare the semantic tags to these:

- `<div>`: Generic block division
- `<span>`: Generic inline content

These tags tell us nothing about their content.

Current Version: HTML5

- Introduced a lot of new semantic tags
- Focus on content and not style
- Simplified the developer experience

Anatomy of an HTML Element

Also known as: nodes, elements, and tags:

```
<element key="value" key2="value2">
  Content or children of element.
</element>
```

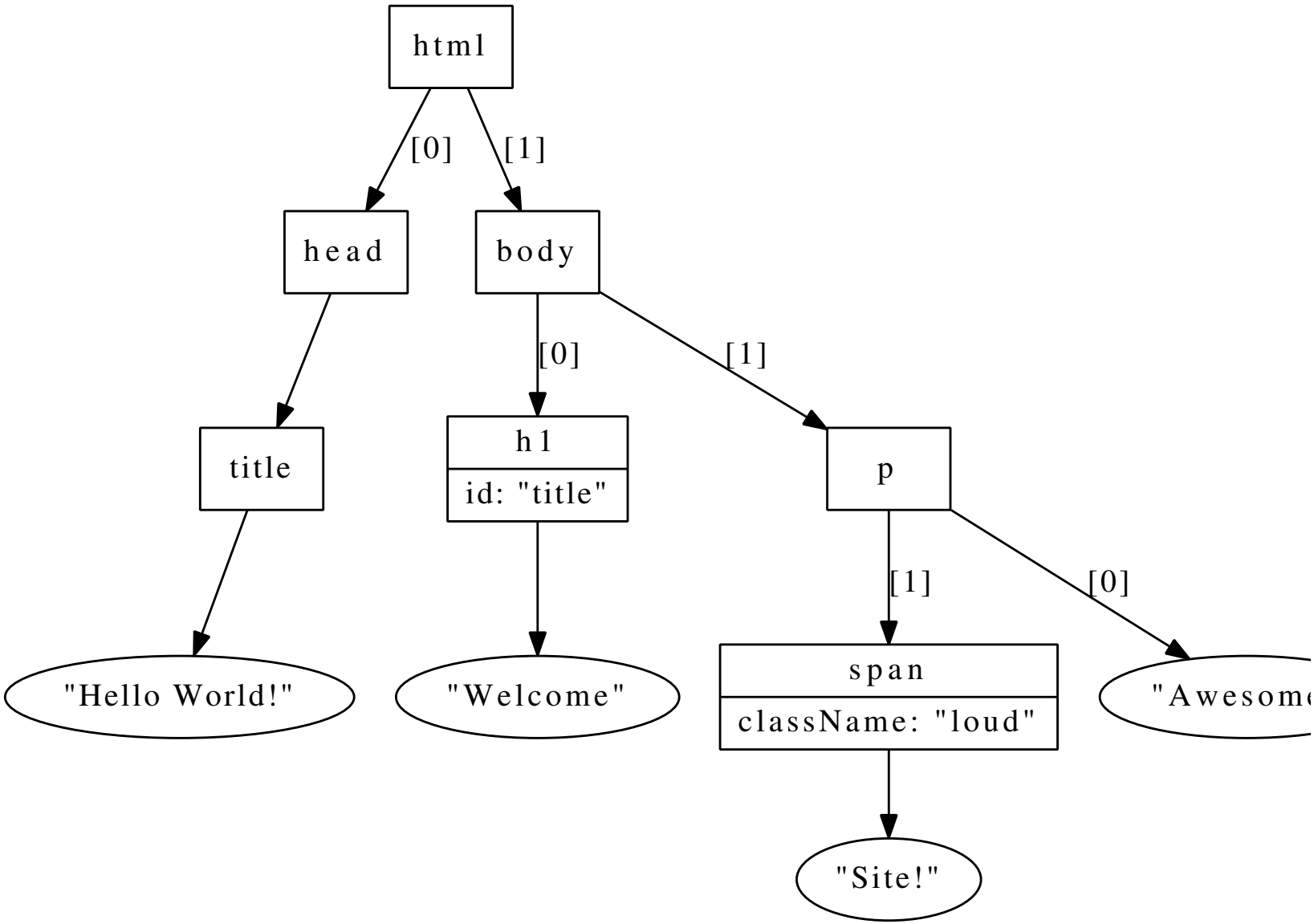
Example HTML Document

```
<html>
  <head>
    <title>Hello World!</title>
  </head>

  <body>
    <h1 id="title">Welcome</h1>

    <p>
      Awesome <span class="loud">Site!</span>
    </p>
  </body>
</html>
```

HTML Parsed into a Tree Structure



Exercise: Writing Some HTML

1. Open the following file:

   `src/www/html/body.html`

2. Let's write some HTML!

Ordered Lists

List items are displayed as sequentially numbered items.

```
<ol>
  <li>Write an HTML document
  <li>Upload it to a web server
  <li>Ask friends to visit site
</ol>
```

See: `src/examples/html/ordered-list.html`

Unordered Lists

List items are displayed as bullet points.

```
<ul>
  <li>Lemon Juice
  <li>Hydrogen Peroxide
  <li>Latex Gloves
</ul>
```

See: `src/examples/html/unordered-list.html`

Tables

```
<table>
  <caption>History of HTML
  <thead>
    <tr>
      <th>Version
      <th>Year
  <tbody>
    <tr>
      <td>1.0
      <td>1991
    <tr>
      <td>2.0
      <td>1995
</table>
```

See: `src/examples/html/table.html`

Forms

```
<form action="https://www.google.com" method="get">
  <label>
    Search: <input type="search"
                   name="q"
                   placeholder="Type Here"
                   required>
  </label>

  <input type="submit" value="Search">
</form>
```

See: `src/examples/html/form.html`

Form Input Types

- `button`
- `checkbox`
- `color`
- `date`
- `datetime-local`
- `email`
- `file`
- `hidden`
- `image`
- `month`
- `number`
- `password`
- `radio`
- `range`
- `reset`
- `submit`
- `tel`
- `text`
- `time`
- `url`
- `week`

See the HTML spec for details on the input types.

Exercise: Writing Forms

1. Open the following file:

   `src/www/html/form.html`

2. Let's write some HTML!

Audio

There is no standard audio format. You should provide various formats based on the browsers you support. See the MDN compatibility chart for more details.

```
<audio controls>
  Your browser doesn't support audio tags.
  <source src="audio.ogg" type="audio/ogg; codecs=vorbis">
  <source src="audio.mp3" type="audio/mpeg">
</audio>
```

See: `src/examples/html/audio.html`

Video

There is no standard video format. You should provide various formats based on the browsers you support. See the MDN compatibility chart for more details.

```
<video poster="../../www/css/images/haskell.png"
       width="640"
       height="450"
       controls>
  Your browser doesn't support the video tag.
  <source src="video.webm" type="video/webm">
  <source src="video.mp4"  type="video/mp4">
  <source src="video.ogv"  type="video/ogg">
</video>
```

See: `src/examples/html/video.html`

Sections

Separating the content in your HTML:

- `<article>`: Content that can stand alone
- `<main>`: Indicates the main content
- `<aside>`: Tangential content
- `<section>`: Generic section (useful in an `article`)

Headings

In addition to the existing <h1> through <h6> tags, HTML5 adds:

- <header>: Introduce a section header
- <hgroup>: Group headings or introduce subheadings/subtitles
- <footer>: Information about the containing section.
- <address>: Contact information for the containing article.

Navigation

```
<section>
  <header>
    <hgroup>
      <h1>Geology</h1>
      <h2>A topic that rocks</h2>
    </hgroup>
    <nav>
      <ul>
        <li><a>Rocks, they're old</a>
        <li><a>Like your mom</a>
      </ul>
    </nav>
  </header>
  <p>I love rocks.
</section>
```

See: `src/examples/html/nav.html`

Viewing the Source Code of a Page

- Browsers allow you to see the source of a page using the *View Source* menu item
- This is the content the browser pulled from the server
- Does not reflect any changes made by JavaScript

Inspecting Elements

- Most modern browsers allow you to see a live view of the HTML using the *Inspect Element* menu item
- This provides a wealth of information, and even allows you to edit the HTML (internal to the browser only)

Selector Quiz

What do these selectors match, and what's the difference between them?

```
p span  { /* ... */ }
```

```
p, span { /* ... */ }
```

Selector Quiz

What does this selector match?

```
li.active.tracked span { /* ... */ }
```

Selector Quiz

What does this selector match?

```
ul li:first-child { /* ... */ }
```

Selector Quiz

What does this selector match?

```
li:nth-child(3n+1):not(:only-child) { /* ... */ }
```

The ID Selector

- HTML:

```
<section id="advertisement">
  <p>Buy now!</p>
</div>
```

- CSS:

```
#advertisement {
  font-weight: bold;
}
```

The Class Selector

- HTML:

```
<div class="info admin report">
  <p>Admin Report (blue).</p>
</div>

<div class="info report">
  <p>Normal Report (green).</p>
</div>
```

- CSS:

```
div.info.report {
  color: green;
}

div.info.admin.report {
  color: blue;
}
```

Descendant Selector

- HTML:

```
<article>
  <ul><li>...</li></ul>
  <section>
    <ul><li>...</li></ul>
  </section>
</article>
```

- CSS:

```
article ul { /* ... */ }
```

- Match all `<ul>` decedents of `<article>`

Child Selector

- HTML:

```
<article>
  <ul><li>...</li></ul>
  <section>
    <ul><li>...</li></ul>
  </section>
</article>
```

- CSS:

```
article > ul { /* ... */ }
```

- Match `<ul>` decedents of `<article>` that are direct children

Sibling Selector

- HTML:

```
<h2>Hello There!</h2>
<p>Paragraph 1.</p>
<p>Paragraph 2.</p>
```

- CSS:

```
h2 + p { /* ... */ }
```

- Match the <p> elements that immediately follow a <h2> (next sibling)

General Sibling Selector

- HTML:

```
<p>Hello!</p>
<ul><li>...</li></ul>
<ol><li>...</li></ol>
```

- CSS:

```
p ~ ol { /* ... */ }
```

- Match all <ol> siblings that come after a <p>

Exercise: Siblings, Children, and Descendants

1. Open (and edit) the following file in your text editor:

   `src/www/css/selectors/part-01.css`

2. Review (but don't edit) the following file:

   `src/www/css/selectors/index.html`

3. Follow the directions in the CSS file
4. Open the HTML file in your browser and confirm your changes

Pseudo What?

- Advanced selectors that use the element's state or relative location
- Can also select non-elements (e.g., paragraph text)
- Begin with a colon (`:`) instead of a dot (`.`)
- (Pseudo elements now start with two colons (`::`))

Pseudo Class Example

```
input:focus {
  border: 3px solid blue;
}
```

Pseudo Element Example

```
/* First (visible) line: */
p::first-line {
  color: red;
}

/* First character: */
p::first-letter {
  font-size:   4em;
  font-weight: bold;
}
```

Partial List of Pseudo Classes and Elements

Classes	Elements
`:link`	`::first-line`
`:visited`	`::first-letter`
`:active`	`::after`
`:checked`	`::before`
`:focus`	`::selection`
`:hover`	
`:enabled`	
`:disabled`	
`:root`	

Exercise: Pseudo Classes and Elements

1. Open (and edit) the following file in your text editor:

   `src/www/css/selectors/part-02.css`

2. Review (but don't edit) the following file:

   `src/www/css/selectors/index.html`

3. Follow the directions in the CSS file
4. Open the HTML file in your browser and confirm your changes

Selecting the First or Last Child

- HTML:

```
<ul>
  <li>First</li>
  <li>Second</li>
  <li>Third</li>
  <li>Forth</li>
</ul>
```

- CSS:

```
li:first-child, li:last-child {
  background-color: #eee;
}

li:only-child {
  color: #f00;
}
```

Selecting First or Last Based on Type

- HTML:

```
<section class="products">
  <header><h2>Products</h2></header>
  <p>First</p>
  <p>Second</p>
  <p>Third</p>
</section>
```

- CSS:

```
.products p:first-of-type {
  border-top: 1px solid #ddd;
}

.products p:last-of-type {
  border-bottom: 1px solid #ddd;
}
```

Exercise: Child Pseudo Selectors

1. Open (and edit) the following file in your text editor:

   `src/www/css/selectors/part-03.css`

2. Review (but don't edit) the following file:

   `src/www/css/selectors/index.html`

3. Follow the directions in the CSS file
4. Open the HTML file in your browser and confirm your changes

Selecting Any Grouping of Children

```
:nth-child(value) { /* ... */ }
```

Example uses of `nth-child`:

- Select even or odd children
- Select every third child
- Select the first 5 children
- Select the last 8 children

Even or Odd Children

```
li:nth-child(even) { /* ... */ }
li:nth-child(odd)  { /* ... */ }
```

(The first child is odd.)

The Nth Child

Select the third (and only the third) child:

```
li:nth-child(3) { /* ... */ }
```

Every Nth Child

Select the third child and every third child after that:

```
li:nth-child(3n) { /* ... */ }
```

Every Nth Child Starting at X

Select every third child, starting at the first child:

```
li:nth-child(3n+1) { /* ... */ }
```

Selecting All Previous or Following Children

Select all children after (and including) the second child:

```
li:nth-child(n+2) { /* ... */ }
```

Select all child before (and including) the second child:

```
li:nth-child(-n+2) { /* ... */ }
```

Nth Child Variations

`:nth-last-child`: Starts from the bottom of the child list.
`:nth-of-type`: Filters the child list by a type selector.
`:nth-last-of-type`: `:nth-of-type` + `:nth-last-child`

Negating a Selector

```
ul:not(.products) {
  background-color: #eee;
}
```

Simple Selectors

The `:not` pseudo-class can only be used with *simple selectors*:

- Type selector
- Universal selector
- Attribute selector
- Class and pseudo-class selectors
- ID selector

Type selector example:

```
.products:not(ul) {
  background-color: #f00;
}
```

Exercise: Pseudo Classes that Take Values

1. Open (and edit) the following file in your text editor:

   `src/www/css/selectors/part-04.css`

2. Review (but don't edit) the following file:

   `src/www/css/selectors/index.html`

3. Follow the directions in the CSS file
4. Open the HTML file in your browser and confirm your changes

Selecting Based on Arbitrary Attributes

Writing a selector for the `id` or `class` attributes is easy. What about the other attributes?

```
/* Attribute exists */
input[placeholder] {
  color: #eee;
}

/* Attribute has exact value */
input[type="number"] {
  border: none;
}

/* Attribute contains substring */
a[href*="salesforce.com"] {
  font-weight: bold;
}
```

Available Operators

Operator	Description	Example
`=`	Exact match	`[type="text"]`
`~=`	Contains word	`[class~="foo"]`
`|=`	Prefix before dash	`[lang|="en"]`
`^=`	Begins with	`[href^="http://"]`
`$=`	Ends with	`[href$=".pdf"]`
`*=`	Contains substring	`[href*="salesforce.com"]`

Exercise: Attribute Selectors

1. Open (and edit) the following file in your text editor:

   `src/www/css/selectors/part-05.css`

2. Review (but don't edit) the following file:

   `src/www/css/selectors/index.html`

3. Follow the directions in the CSS file
4. Open the HTML file in your browser and confirm your changes

Form Validation in the Browser

Validation attributes:

- `max`: Maximum number or date
- `maxlength`: Maximum number of characters
- `min`: Minimum number or date
- `minlength`: Minimum number of characters
- `pattern`: Regular expression `value` must match
- `required`: Input must have a value
- `title`: Describe the `pattern` conditions

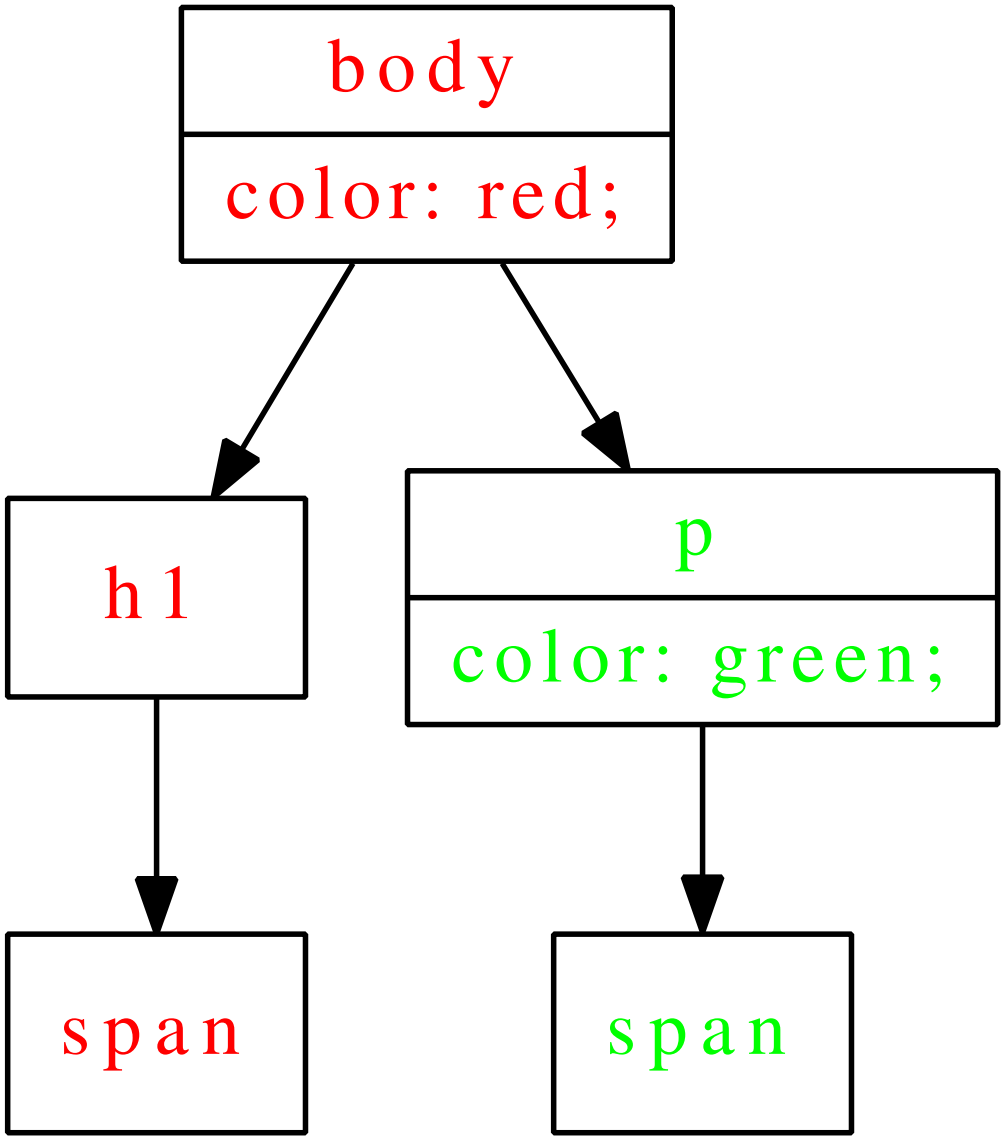
CSS Pseudo Classes:

- `:valid`: Element's value is valid
- `:invalid`: Element's value is invalid
- `:optional`: No value is required
- `:required`: A value is required

Exercise: Form Styling

1. Open (and edit) the following files in your text editor:
   - `src/www/css/form/form.css`
   - `src/www/css/form/index.html`
2. Follow the directions in the CSS file
3. Open the HTML file in your browser and confirm your changes

Inheriting Styles from Ancestors



Inheritable Properties

An (incomplete) list of inheritable properties

- `line-height`
- `color`
- `text-align`
- `letter-spacing`
- `font-family`
- `font-style`
- `font-variant`
- `font-weight`
- `font-size`

Forcing Inheritance

You can inherit any value from a parent as long as it's set on the parent and you use the `inherit` value keyword:

```
/* Set the value on the parent:       */
p          { border: 1px solid blue;   }

/* Inherit from the parent:           */
p > span { border: inherit;           }

/* Or change the property's behavior */
*          { border: inherit;          }
```

Conflicting Properties

What happens when properties conflict?

- HTML:

```
<div id="main" class="fancy">
  What color will this text be?
</div>
```

- CSS:

```
#main {color: red;}

#main.fancy {color: blue;}

div.fancy {color: green;}
```

Specificity Chart

Selector	Points	Examples
Universal selector	0	`*`
Type selectors	1	`p`, `a`, `h1`, etc.
Pseudo elements	1	`::before`, `::after`, etc.
Classes	10	`.sidebar`
Pseudo classes	10	`:nth-child`
Attribute selectors	10	`[type="number"]`
ID selectors	100	`#main`

- Inline styles add 1,000 points.
- Tie breaker: last defined style wins.
- Force highest specificity with `!important`.

Specifying Fonts

```
html {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 16px;
}
```

Using Web Fonts

```
/* Create a new font-family */
@font-face {
  font-family: "My Font Name";
  src: url(/fonts/myfont.woff);
}

/* Then use it */
html {
  font-family: "My Font Name";
}
```

Web Font Services

Example using Google Fonts:

- HTML:

```
<link
 href="https://fonts.googleapis.com/css?family=Indie+Flowe
 rel="stylesheet">
```

- CSS:

```
html { font-family: "Indie Flower"; }
```

Background Image Properties

`background-image`: The URL of the image.

`background-position`: Absolute or relative position of background.

`background-origin`: Controls where the background image is initially placed. That is, it's upper-left origin.

`background-size`: Constrain the size of an image, or scale the image up or down.

`background-repeat`: How to tile images smaller than the container.

`background-clip`: Which bounding box the background (image or color) will be clipped to.

`background-attachment`: Control image location when scrolling.

Exercise: Background Images

1. Open (and edit) the following file in your text editor:

   `www/background-image/index.css`

2. Review (but don't edit) the following file:

   `www/background-image/index.html`

3. Modify the CSS so that your browser shows the same page as the one on the instructor's screen
4. Open the HTML file in your browser and confirm your changes

Image Sprites

- Several images stored in a single file
- Size the parent element to the size of a single image
- Changing `background-position` will change which image is show
- Can be animated with CSS or JavaScript

Exercise: Icons

1. Open (and edit) the following file in your text editor:

   `www/icons/index.css`

2. Review (but don't edit) the following file:

   `www/icons/index.html`

3. Make each `<LI>` show only it's designated icon.
4. Open the HTML file in your browser and confirm your changes

Embedded Images

Images can be encoded using Base64 and directly embedded in a CSS file:

```
.logo {
  background-image:
    url(-DATA-GOES-HERE);
}
```

Layout Engines

CSS layout is concerned with moving HTML boxes around on the screen to make the site look the way you want. By default the browser will simply stack all the boxes on top of one another.

The Default: Block/Inline

- Elements are either block or inline
- Block elements stack on top of one another
- Inline elements flow inside a block element
- This is the default layout engine
- Good for articles, not so good for applications

Introduction to the Box Model

Open the following file in your web browser:

`www/box-model/index.html`

- Block elements have newlines before and after their content
- Inline elements flow in the content of a block element.

Floated Boxes

- Boxes can be floated so they are side-by-side with their siblings
- Sibling boxes will wrap around the floated box
- Boxes can be floated to the left or the right

Using the Floating Layout

Float boxes with the `float` property:

```
.sidebar {
  float: left; /* left, right, or none  */
  width: 25%;  /* remember to set width */
  margin-right: 1em; /* Push the main content away */
}


footer {
  clear: both; /* Stop the floating */
}
```

Problem: Float Drop

- Boxes are dropping below the floated box instead of side-by-side
- Set a width for all of the floated boxes
- Keep the box model in mind (border, margin, padding, etc.)
- You can also make the browser include the entire box in the width:

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

Problem: Floating Siblings

- Floated boxes can escape their parent and continue to float other boxes (when the floated box is the biggest child)
- Make the parent enclose and clear the float:

```
.container::after {
  content: " ";
  display: table;
  clear:   both;
}
```

A Layout Engine for the Modern Web

- Easy to use with visually pleasing defaults
- Similar to a grid system, but easier to use
- No weird CSS tricks or class names to learn
- Universally supported (IE >= 11)

Flexible Boxes: The Basics

- Mark a container element as a flexible box:

```
.container { display: flex; }
```

- All children then become *flex items*
- Flex items can be laid out in rows or columns
- Wide range of alignment, sizing, and wrapping options

Flex Item Layout

- Items side-by-side, left to right (default):

```
.container {
  display: flex;
  flex-direction: row;  /* or row-reverse */
}
```

- Items stacked top to bottom:

```
.container {
  display: flex;
  flex-direction: column; /* or column-reverse */
}
```

Flex Direction Orientation

Since flex can layout items in a row or a column it uses generic terms to refer to its axes:

- Main axis vs. cross axis
  - For row: main is horizontal, cross is vertical
  - For column: main is vertical, cross is horizontal
- Main start and end, vs. cross start and cross end
  - For row: main start is on the left
  - For row-reverse: main start is on the right

Flex Item Wrapping

- Items must all be on the same line (row):

```
.container {
  display: flex;
  flex-wrap: nowrap; /* This is the default */
}
```

- Items are allowed to wrap onto the next line:

```
.container {
  display: flex;
  flex-wrap: wrap; /* or wrap-reverse */
}
```

Flex Item Sizing

The `flex-grow`, `flex-shrink`, and `flex-basis` properties control the width of flex items relative to their siblings.

- Make all flex items the same width:

```
.container {
  display: flex;

  /* flex-grow flex-shrink flex-basis */
  flex: 1 1 250px;
}
```

- Make the second item take up twice as much space as the others:

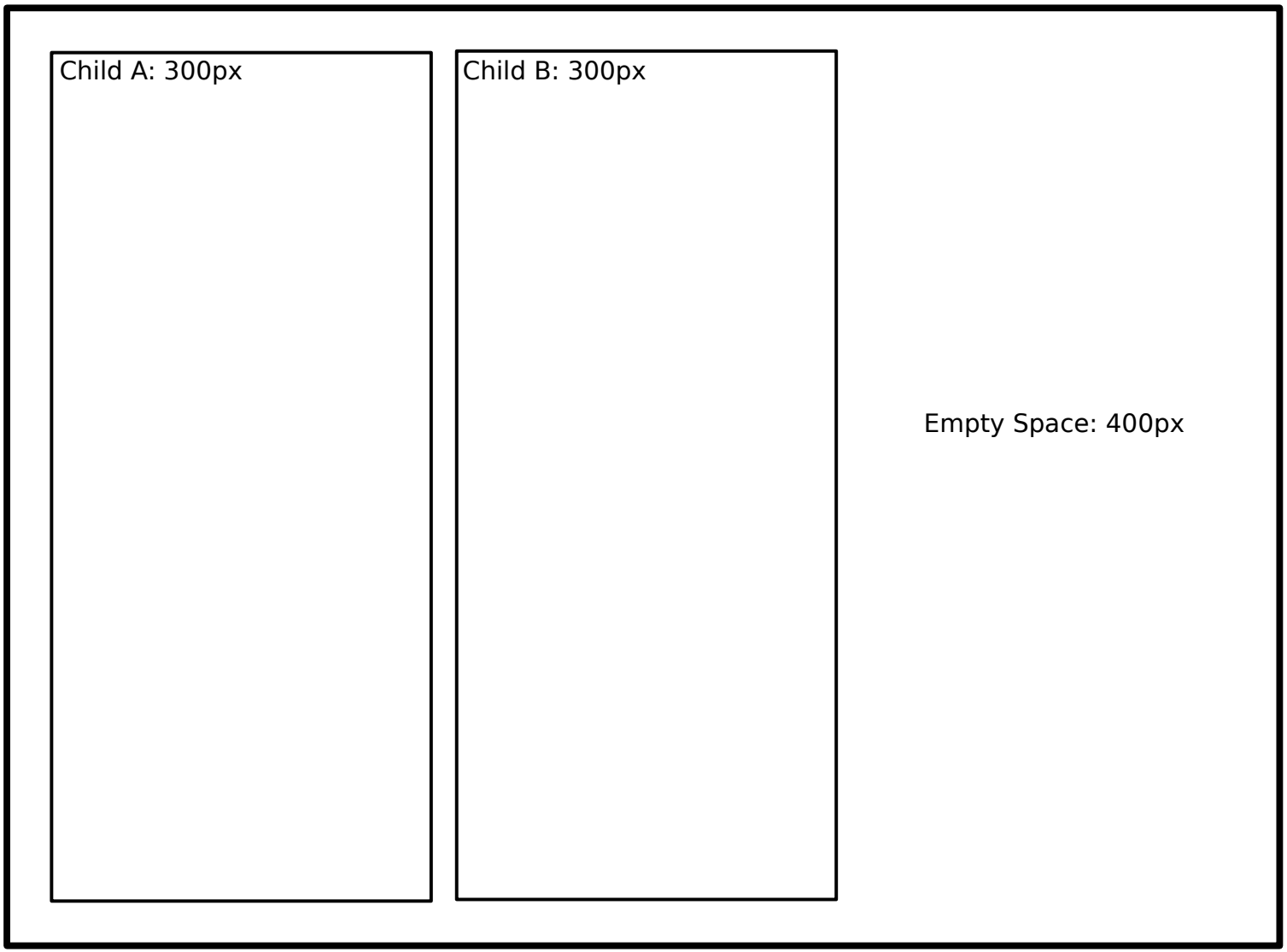
```
.container                { display: flex; }
.container > *            {    flex: 1;    }
.container :nth-child(2)  {    flex: 2;    }
```

A Word About `flex-basis`

This property can be a bit tricky to understand. It's not your fault, it's complicated!

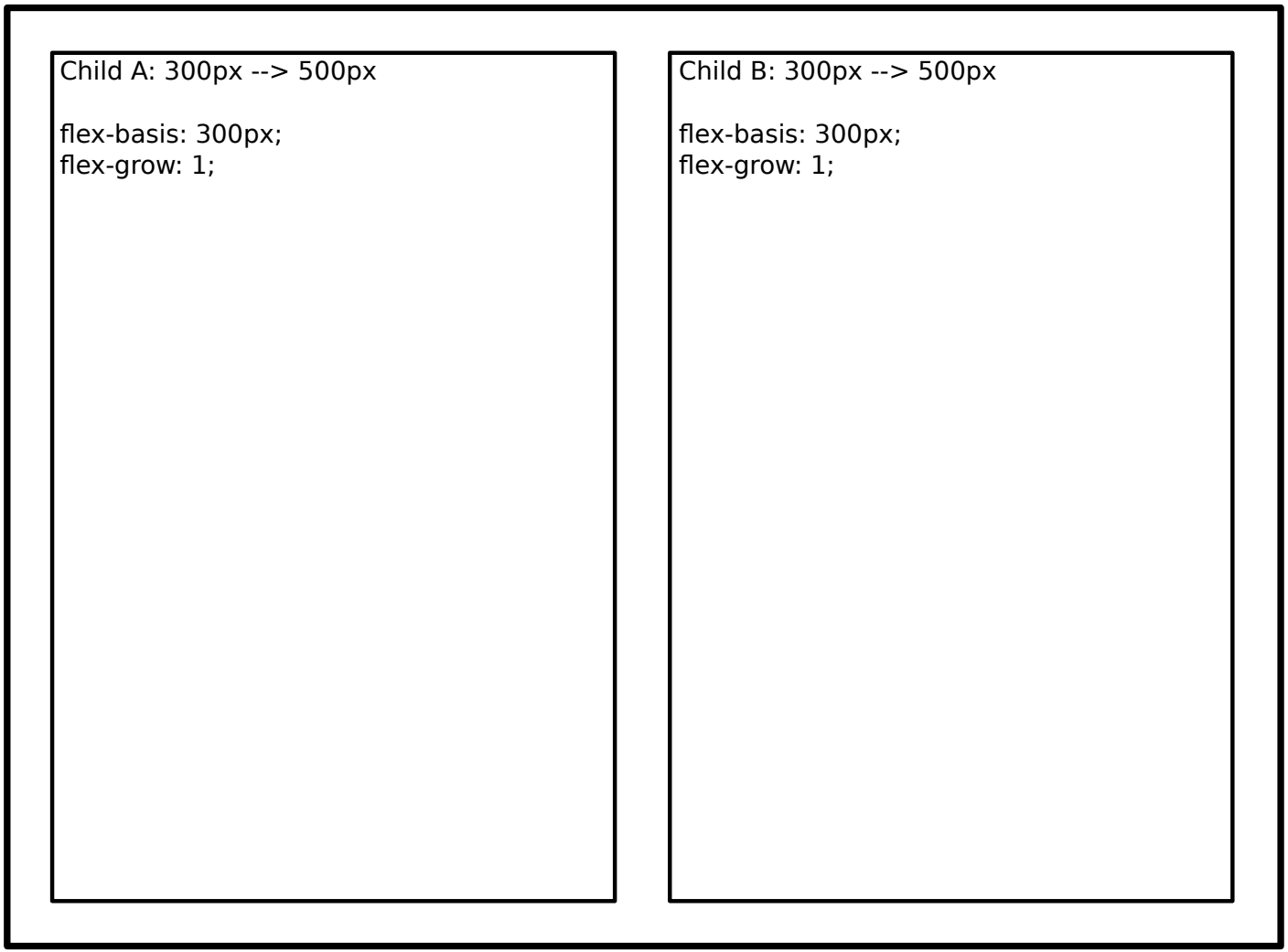
- Definition: The *initial* size of a flex item.
- The default value (today): `auto`
- Most common value: An absolute or relative measurement
- Future values: `min-content`, `max-content`, `stretch-fit`.

Flex Grow: Extra Space



Flex Grow: Uniform Growth

Parent: 1,000px

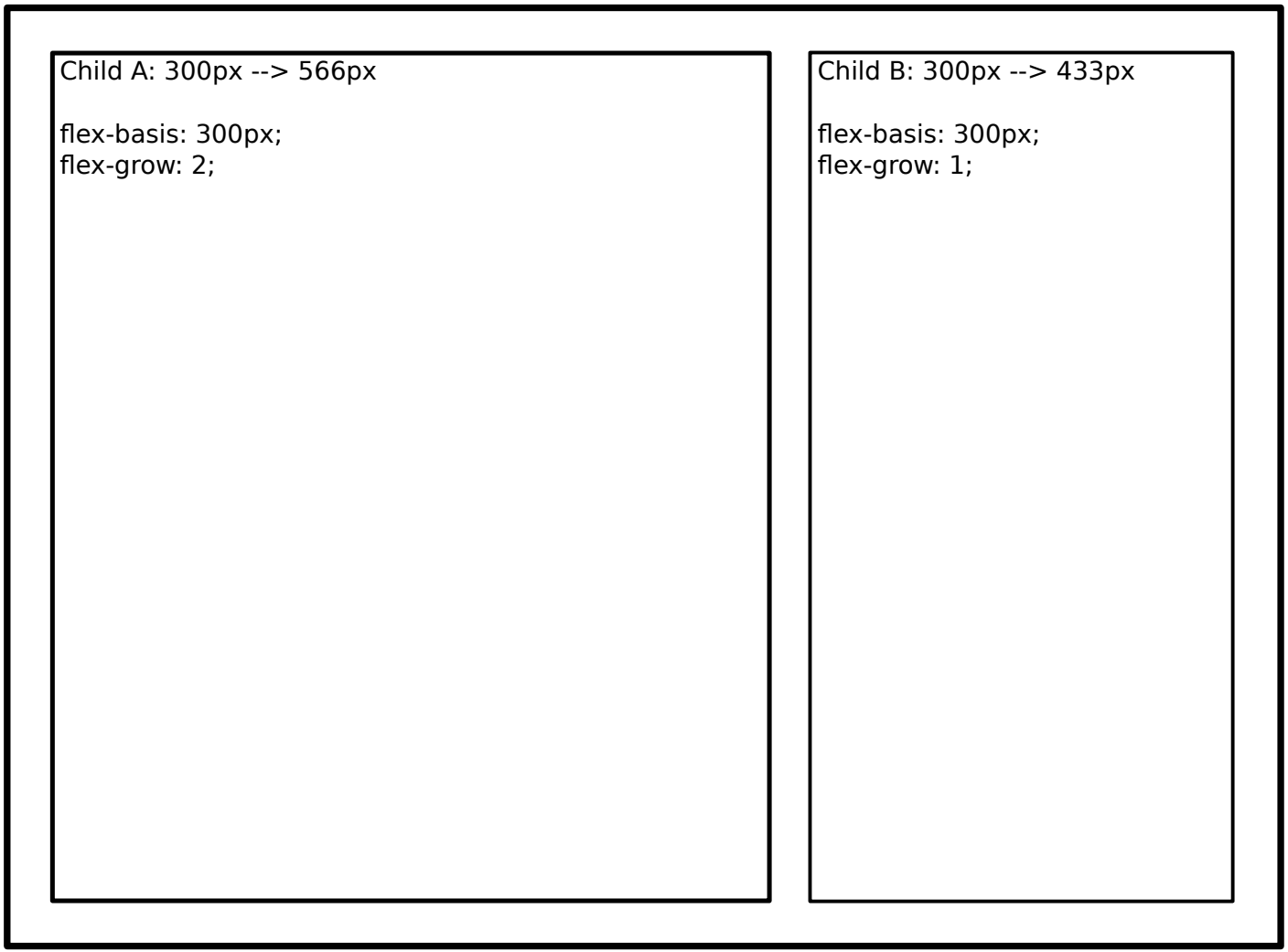


grow = Empty Space * (Child Flex Grow / Total Flex Grow)

Child Grow = 200px = 400px * (1 / 2)

Flex Grow: Nonuniform Growth

Parent: 1,000px



grow = Empty Space * (Child Flex Grow / Total Flex Grow)

Child A Grow = 266px = 400px * (2 / 3); Child B Grow = 133px = 400px * (1 / 3)

Flex Item Ordering

Items inside a flex container can be displayed in a different order than they appear in the HTML source code. This is done with the `order` property:

```
.container          { display: flex; }
.sidebar.primary    {   order: 1;    }
.main               {   order: 2;    }
.sidebar.secondary  {   order: 3;    }
```

Flex Alignment

`justify-content`: How space is distributed around and between items on the main axis. Requires that all flex items have a `flex-grow` of zero. Useful values; `space-between`, `space-around`, `space-evenly`, `flex-start` (the default), and `flex-end`.

`align-items`: How space is distributed around and between items on the cross axis. Used when flex items have different cross axis sizes. Useful values: `stretch` (the default), `center`, `flex-start` `flex-end`.

`align-content`: How space is distributed around and between multiple lines on the main axis created by wrapping. Useful values: `stretch` (the default), and all values form `justify-content`.

Flex Container and Item Properties

Container	Item
`flex-flow`	`order`
`flex-direction`	`align-self`
`flex-wrap`	`flex`
`justify-content`	`flex-grow`
`align-items`	`flex-shrink`
`align-content`	`flex-basis`

So Many Browser Sizes, One HTML File

- Fixed design: Treating the web like paper
- Liquid design: Better but more complicated
- Responsive: Adapt to each browser

Mobile Browsers and Zooming

- Mobile browsers automatically zoom out to show all content
- The first step to making a site responsive is to disable this
- Use the following `meta` tag in the `head` of your HTML:

```
<meta name="viewport" content="width=device-width">
```

- With that, browsers will respect width requests without zooming

Relative Measurements

- Avoid using absolute units such as `px`, `pt`, `cm`, `in`, `mm`, etc.
- Relative to current font size: `em`
- Relative to parent element size: `%`
- Percentages + Media Queries = Responsive Web Design

Introduction to Media Queries

- Media Queries are part of CSS
- They are like `if` statements in your CSS
- Example:

```
/* If the browser window is at least 400px wide... */
@media (min-width: 400px) {
  .sidebar {
    float: left;
    width: 25%;
  }
}
```

Compound Media Queries

Media queries can be combined with `and`:

```
@media (min-width: 400px) and (orientation: portrait) {
  /* ... */
}
```

Media Queries and Breakpoints

Set media query breakpoints—divisions of screen width that change the CSS:

```
@media (max-width: 480px) {
  /* Small screens */
}

@media (min-width: 481px) and (max-width: 768px) {
  /* Medium screens */
}

@media (min-width: 769px) {
  /* Larger screens */
}

/* Etc. */
```

Mobile First, or Desktop First?

There are two ways to approach responsive web design:

1. Design for small screens and use media queries to adapt the design for larger screens
2. Start with a design for large screens and use media queries to scale the design down to smaller screens

Fluid Images

- Automatically scale images to match the container width:

```
img { max-width: 100%; }
```

- For this to work, don't use `width` or `height` attributes on an `img` tag:

```
<img src="logo.jpg" alt="Logo">
```