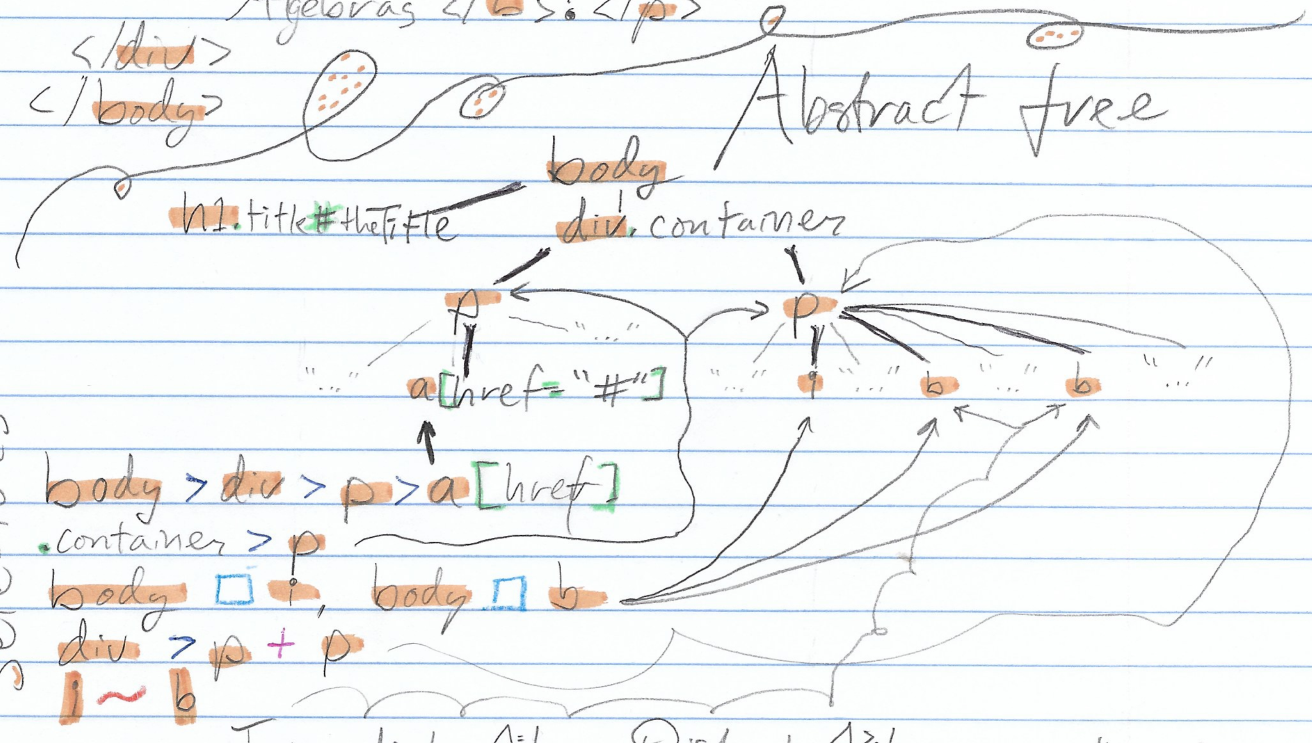
Introduction

CSS Selectors & HTML trees

Combine
Match only

(X)(HT)ML:

```
<body>
  <h1 class="title" id="theTitle">Title </h1>
  <div class="container">
    <p> Welcome to this <a href="#">presentation</a>:)</p>
    <p> Hopefully you will learn at least <i> something</i>
        about <b> CSS selectors</b> and <b> Boolean
        Algebras </b>!</p>
  </div>
</body>
```



Selectors

body > div > p > a[href]
.container > p
body □ i , body □ b
div > p + p
i ~ b

	Immediate $\Delta=1$	Distant $\Delta\geq 1$
Vertical	>	□
Horizontal	+	~

technically a space in CSS

Observations

Can add more attributes : div.container.special[title]
but not more elements, since a node has only
one element type : div and p never overlap.
Can match more things with , commas
Can negate a selector : :not(div > p)
Other pseudo-classes & elements , :hover ::selection
Always: * , never: :not(*)

Excluded Middle: tt = a || not a

Introduction (continuuuuuuuuued)

Boolean Algebra

everywhere in language and mathematics

$\mathbb{B}$

```
data Bool = T | F
Boolean Algebra Bool where
    tt = T, ff = F, not T = F, not F = T        tt = not ff
                                                 not . not = id
    T && T = T , _ && _ = F     "conjunction"
    F || F = F , _ || _ = T     "disjunction"
Lift over Tuple as expected.
Even predicates: ∀a, b. Boolean Algebra b. a → b
  e.g. isKeyword = (eq "let") || eq "in" || eq "where" || ...
       isOperator = eq "→" || eq "⇒" || ...
       classify = isKeyword &&& isOperator
          (&&&) :: ∀abc. (a→b) → (a→c) → (a → Tuple b c)
```

B-tree Assoc. → List Comm. → Multiset Idem. → Set

Simplify structure

Subsets

Properties:

Conjunction forms a idempotent, commutative monoid with tt as identity, ff as annihilator.

Disjunction: similar story, tt and ff swap roles.

Distributivity:
$(a \&\& b) \| c = (a \| c) \&\& (b \| c)$
$(a \| b) \&\& c = (a \&\& c) \| (b \&\& c)$

monoid homomorphism

De Morgan:
not (a && b) = not a || not b
not (a || b) = not a && not b

coherence / sensicality

Big Reveal... (Disjunctive) Normal Form

BNF did it again

```
DNF    := clause | clause '||' DNF
clause := term | term '&&' clause
term   := 'not' var | var
var    := 'a' - 'z'
```

actually, this is whatever datatype.

```
newtype DNF var = Set (Map var Boolean)
```

set of ↑ disjoined clauses

each variable may appear negated or unnegated in a conjunctive clause