Client/browser productivity language (for layout)

Ras Bodik, Thibaud Hottelier, James Ide,
Doug Kimelman(IBM), and Leo Meyerovich

Millions of Designers Struggle With CSS

Berkeley Par Lab

"I need help sorting out the problem with a website I designed which uses DIV tags to allow me to use a background image with layers of editable text over it. What I have works fine in all browsers EXCEPT when the screen resolution changes and/or the browser is resized. Then the text no longer properly or predictably lines up with the background image. [...]"

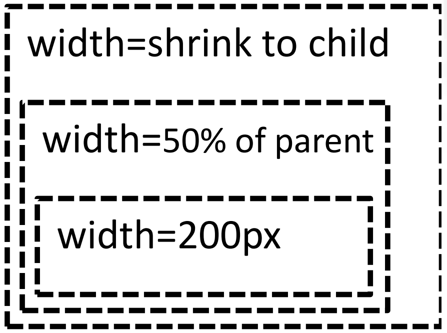
CSS is hard. Why?

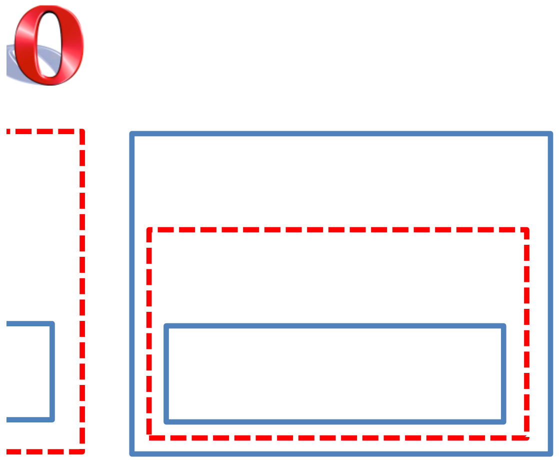
- Is it too large, bloated?
- Or is something missing?
- Do the language concepts map onto how users think?

→ Brokenness by Example

CSS Spec is Contradictory

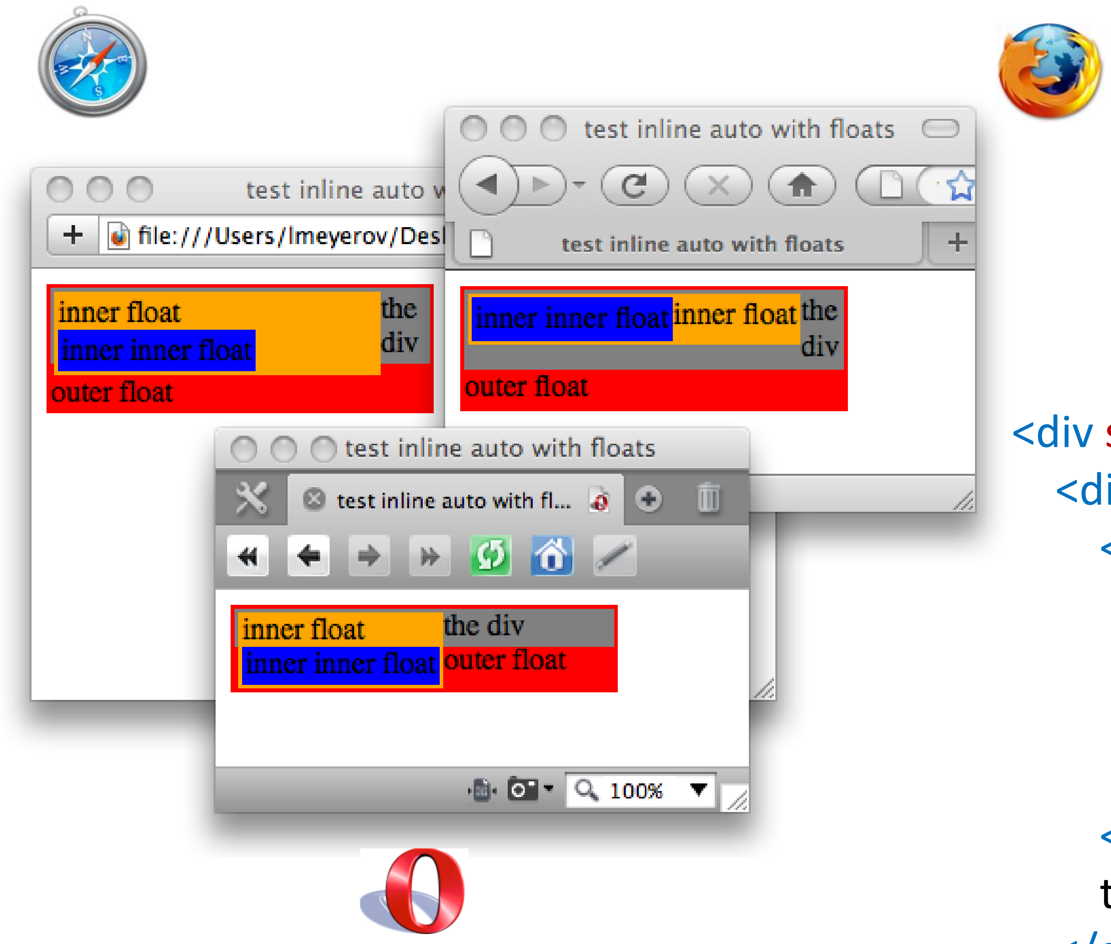
- Browser tries to guess user’s intent
  - Deviate from Spec
- CSS is too lo
  - Does not raints yourself





→ Silently dropped constraints lead to unpredictability

CSS Spec is Ambiguous



```
<div style="float: left ; width:200px">
   <div>
      <div style="float: left; ">
         inner float
         <div style="float: left;">
            inner inner float
         </div>
      </div>
      the div
   </div>
   outer float
</div>
```

Summary of Motivation

Users are confused

- Limited Expressiveness; results are unpredictable.

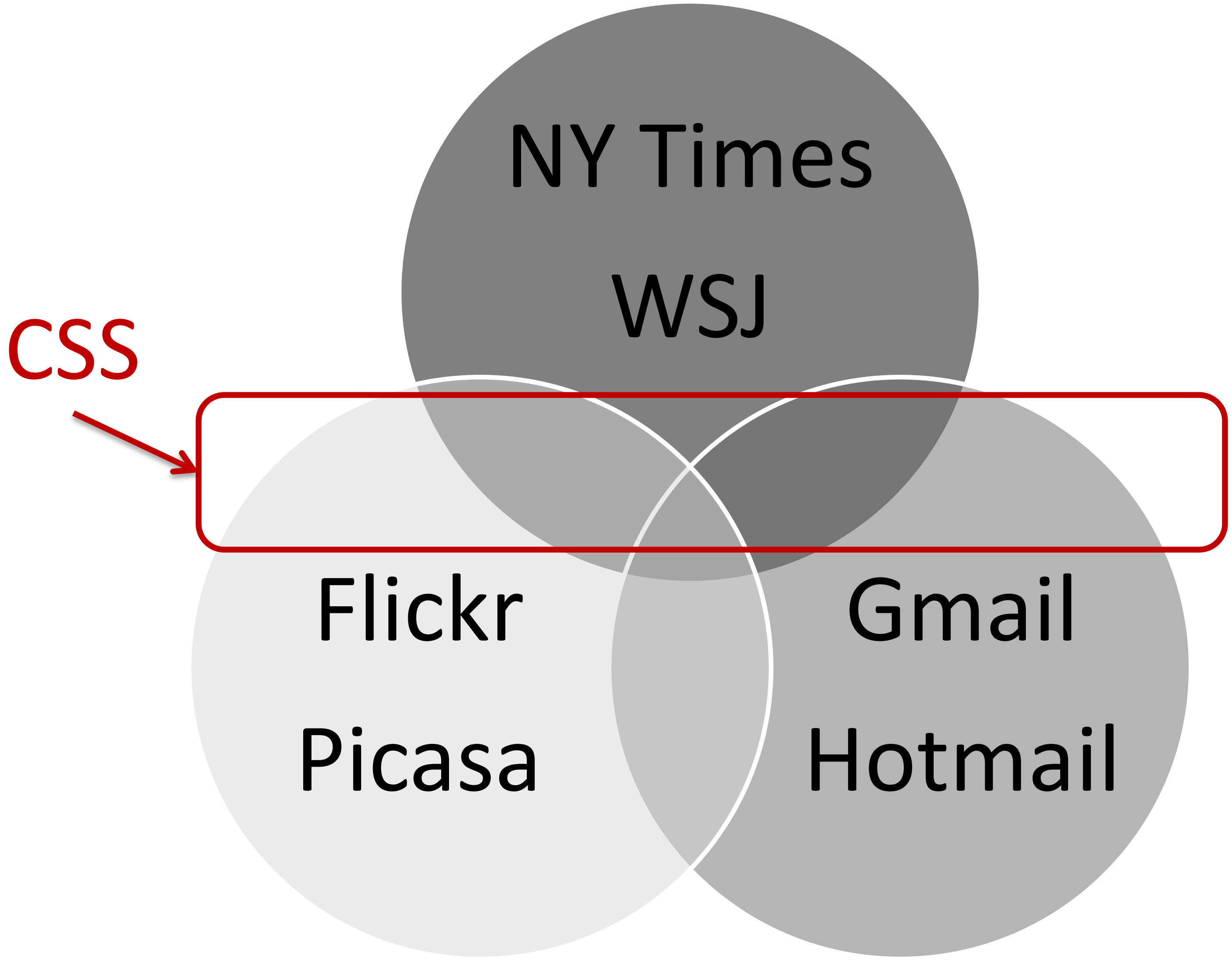
CSS Spec confusing because

- Contradictory, constraints silently dropped
- Ambiguous, diverging browsers
- Complicated, hard to implement

We address these by

- Simpler, domain languages
- Tool support for checking specs
- Tool for generating layout engine

Different needs -> Different languages



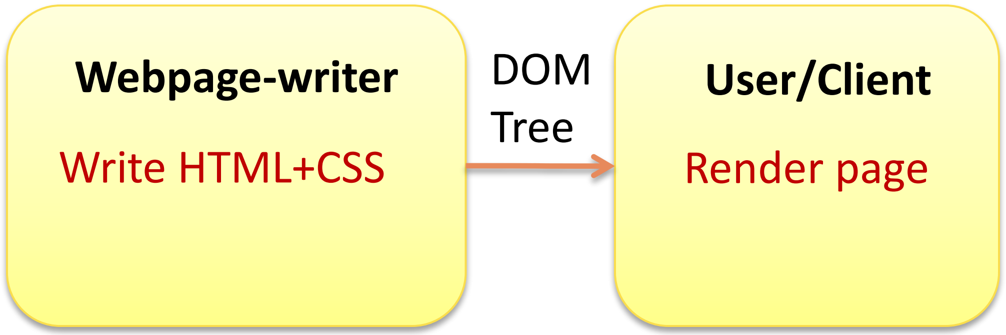
Eventually

Successful if we can embed in our model

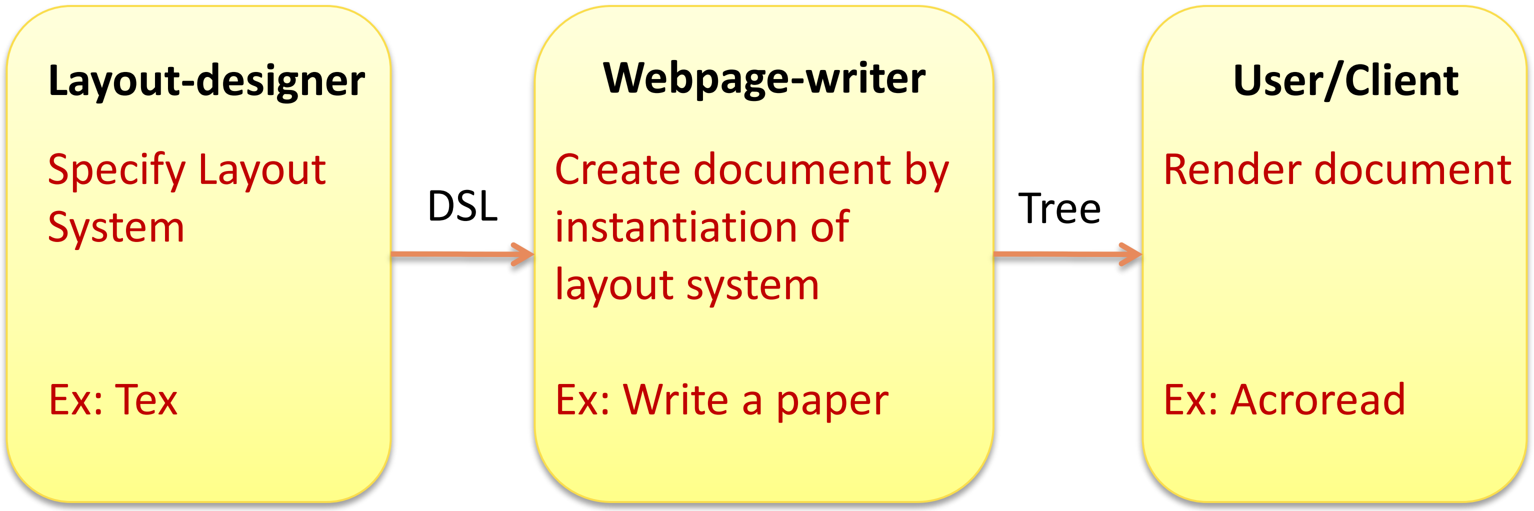
- GUIs: QML, XAML, etc.
- New Grid-Based Layouts
- Core/Subsets CSS [Meyerovich'09]

Roles

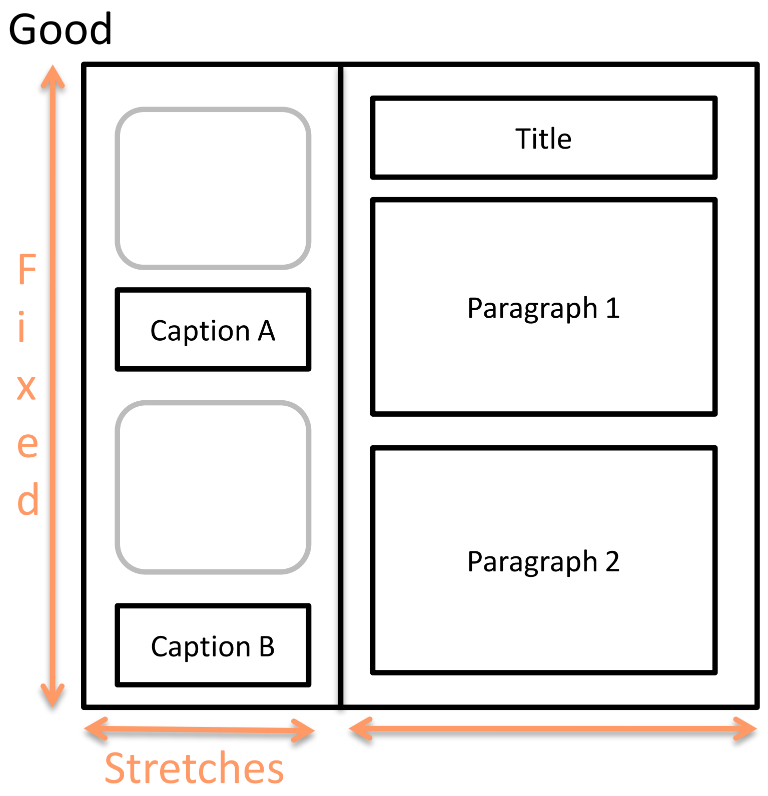
Today

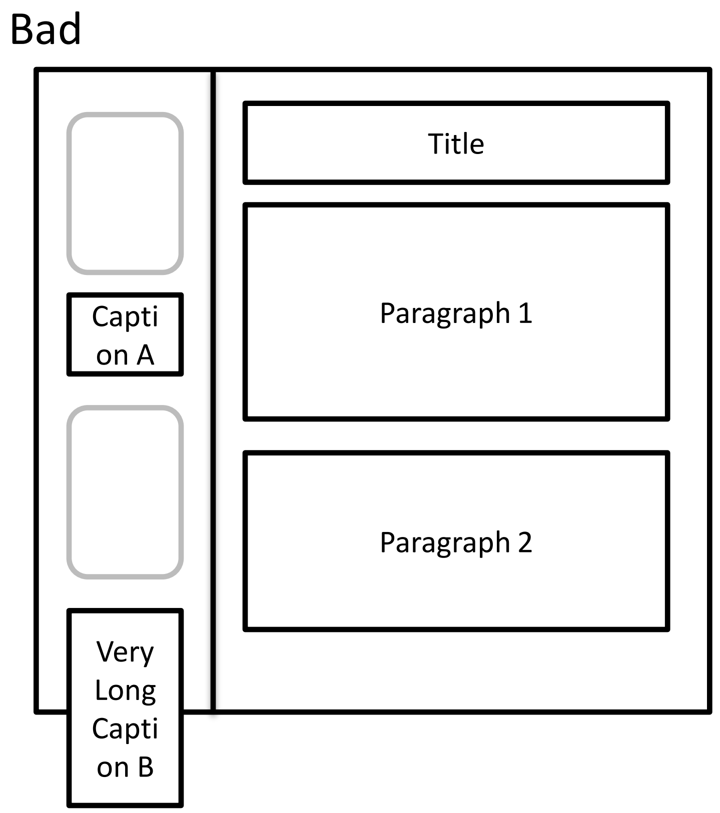


Tomorrow



Example





Designer Intent: Pictures + Captions all on first page.

Computation: Left: width := F(height)

Right : height := G(width)

Free The Constraints

Let the designer express declaratively his intent via constraints.

box.width == box.height

Bi-directional constraints:

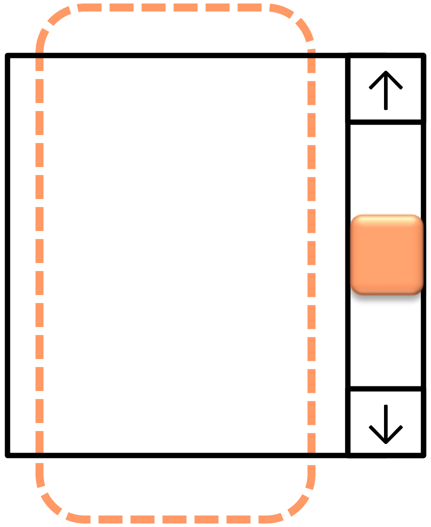
- Conciseness
- Split specified behavior and computation
  - You specify, We Solve

Bi-directional Constraints

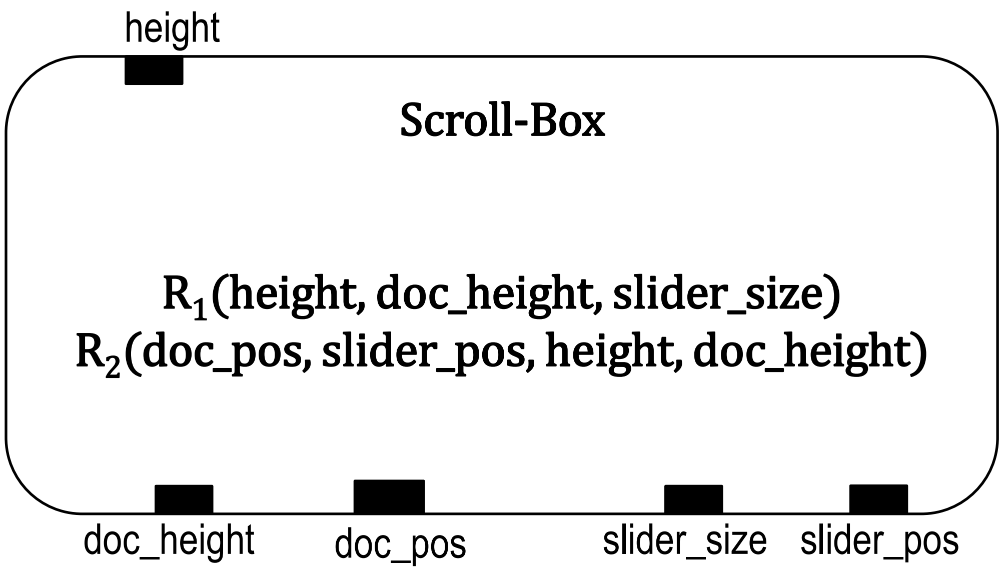
Redundancy in GUI:

Multiple knobs/indicator for a single variable.

Thus, many ways to update it.



With bi-directional constraints:


height
Scroll-Box
R_1(height, doc_height, slider_size)
R_2(doc_pos, slider_pos, height, doc_height)
doc_height
doc_pos
slider_size
slider_pos

Summary of Design Choices

Our proposed solution is

- Domain-specific Layout Languages (DSLL).
- Bi-directional constraints exposed to the document writer.

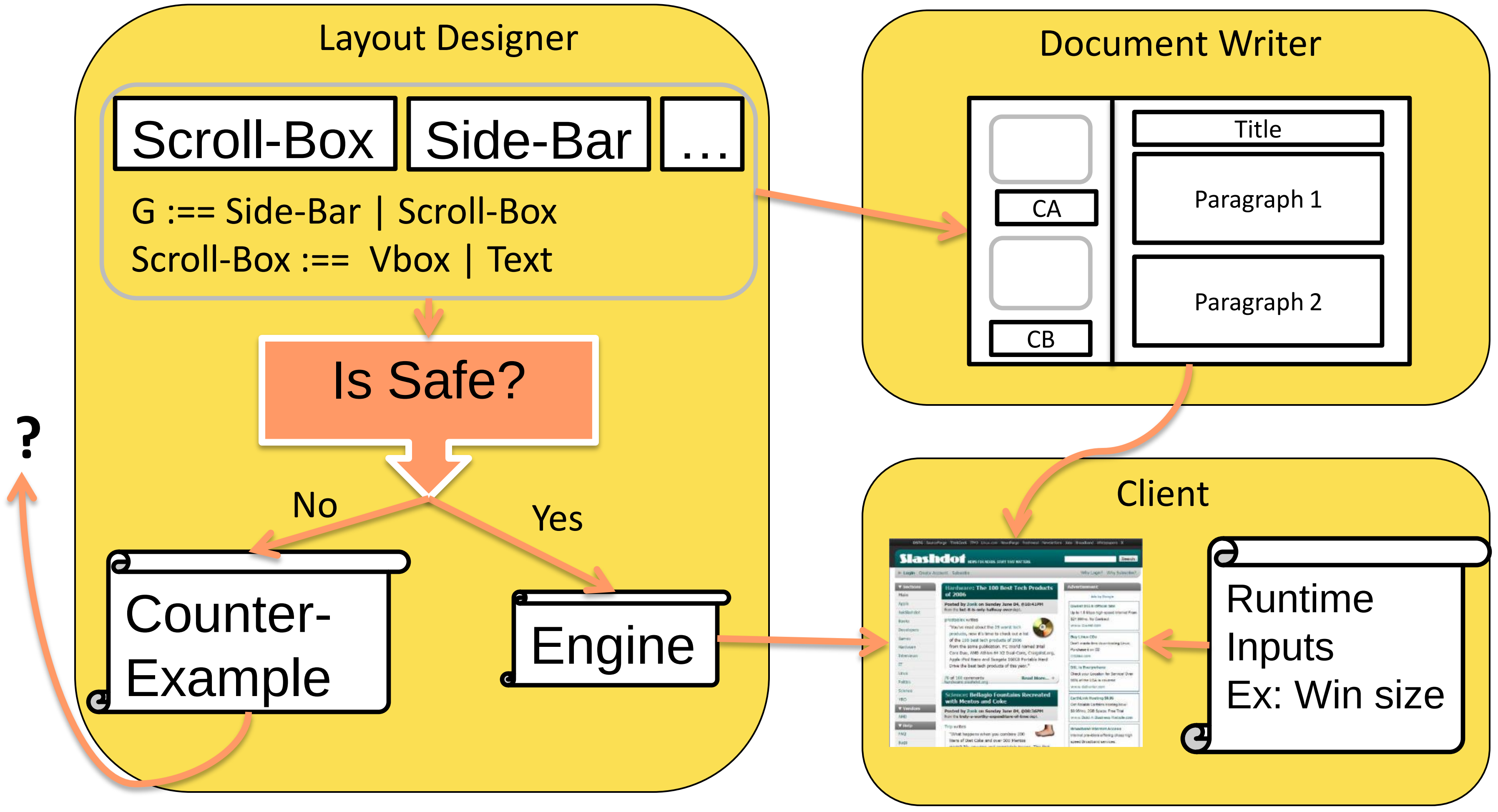
We want all documents in a DSLL to be

- Fast to solve.
- Always well defined: Can always layout.

We need to

- Generate efficient solver (layout engine).
- Check DSLL is "Good"
  - Compilation to tree traversals (AGs)
    - With synthesis of local evaluation rules.

3-Stage Architecture



Safe: Forall Tree in G, Forall Input in Tree, Tree(input) is Satisfiable and the solution can be found with propagation only.

Related Work On Solving

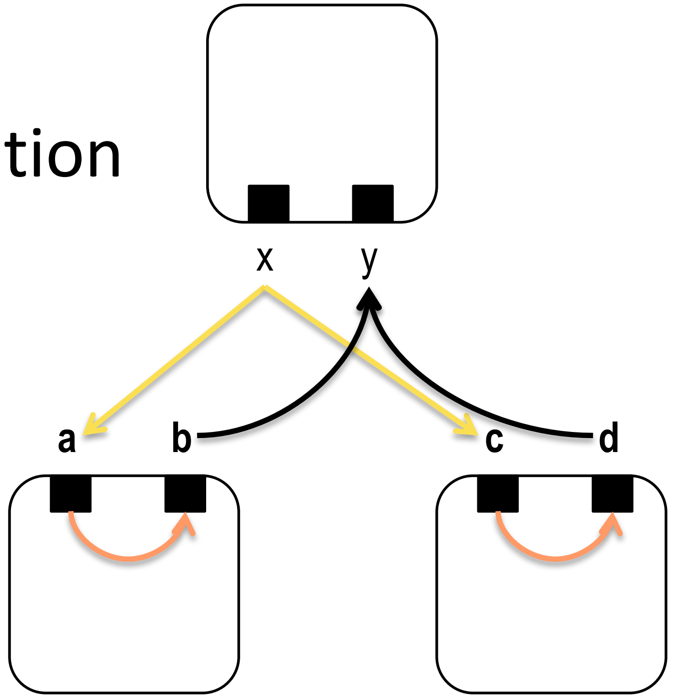
- What would you do?
- Use a generic solver
  - Cassowary [Badros]: Analyze documents online and figures out layout.
- For performance, we want
  - Reduce runtime work by doing offline pre-computation.
  - Modular & Specialized solver.

On Solving

What is the fastest solver ?

- Set of traversals on Tree
- This is given by scheduling an AG
    - Can do parallel traversal
    - Can do incremental evaluation
    - …

  [Leo & Adam]



Example

Relations (input)

```
Hbox ::= Box1 Box2          Box1.x + Box2.x == Hbox.x
                            Box1.x == Box2.x
                            Box1.y == Box2.y == Hbox.y
```

Functions

Box1.x := Hbox.x / 2   ~~Box2.x := Hbox.x / 2~~
~~Hbox.y := Box1.y~~   ~~Hbox.y := Box2.y~~
Box1.y := Box2.y   Box2.y := Box1.y

[Leo & Adam]

Tree Traversals

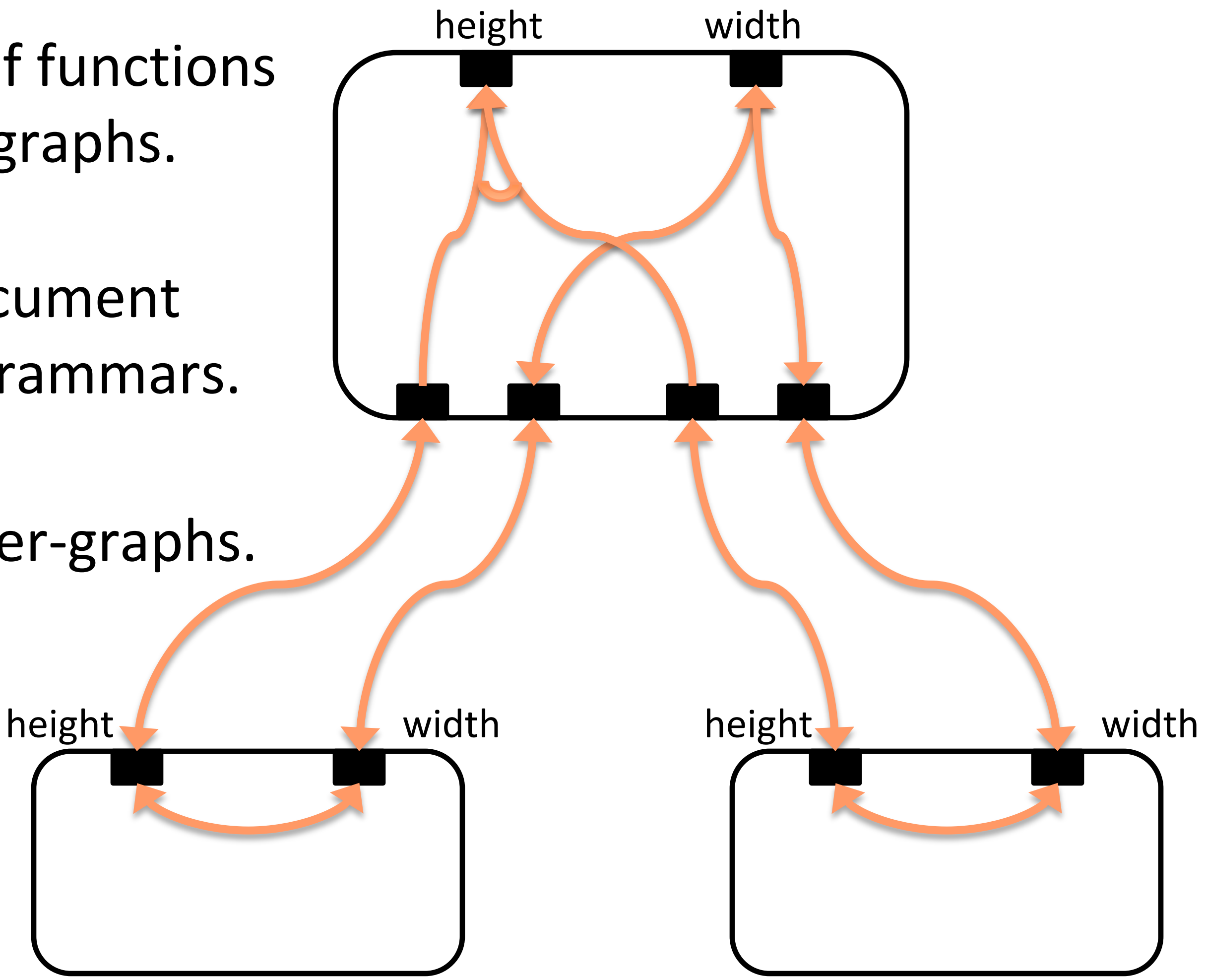
```
Hbox .visit() {
        Box1.y = this.y;
        Box1.visit();
        Box2.y = this.y;
        Box2.visit();
        thix.x = Box1.x + Box2.x
}
```

Which Functions Do We Choose

BERKELEY PAR LAB

- Picks some subset of functions to cover the whole graphs.
- Here on a single document but generalizes to grammars.

→ Reachability on hyper-graphs.



Future Work

- Events
  - Web-pages are dynamic (AJAX)
  - We are actively working on reactive semantics, ask me about it!
- Programming by demonstrations
  - Best paradigm for designer.
  - From a set of documents, infer the layout.
- Richer layout
  - Expressiveness vs. Speed trade-offs.

That is it!