Seaside

Building Complex Web Applications Simply

What are the problems?

HTML is goto programming

Reusability is difficult

Hard to debug

Goto programming

```
<form action="second.html">
   <input type="text"
  name="value1">
   <input type="submit" value="OK">
</form>
```

```
 <form action="result.html">
    <input type="hidden" name="value1" value="<% value1 %>" >
    <input type="text" name="value2">
    <input type="submit" value="OK">
 </form>
```

```
<p>
    <% value1 + value2 %>
</p>
```

Seaside flow

```
| value1 value2 |
value1 := self request: 'first number'.
value2 := self request: 'second number'.
self inform: value1 + value2
```

Call/Answer

*Call a **component** (widget) and answer its returned value.*

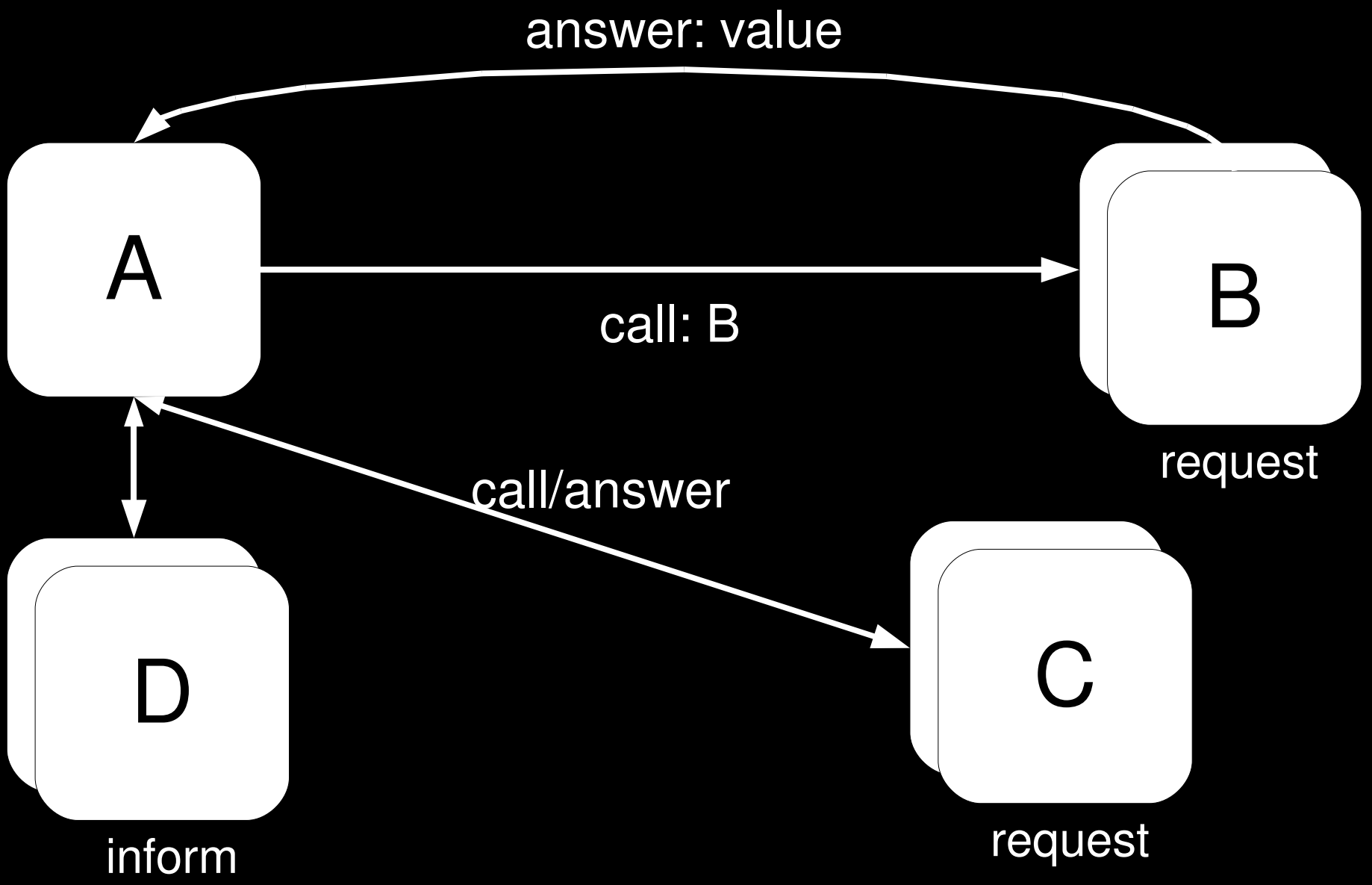
```
value1 := self request: 'first number'.
```

Call a component

Answer its value

answer: value

A flow of components



More examples

Event model

goto not in our world

```
Example2>>renderContentOn: html
    html form: [
        html submitButton
            callback: [ self inform: 'Hello'] ;
            text: 'Say Hello' ]
```

Components

the counter example

Write your own reusable component

- data model
- component rendering

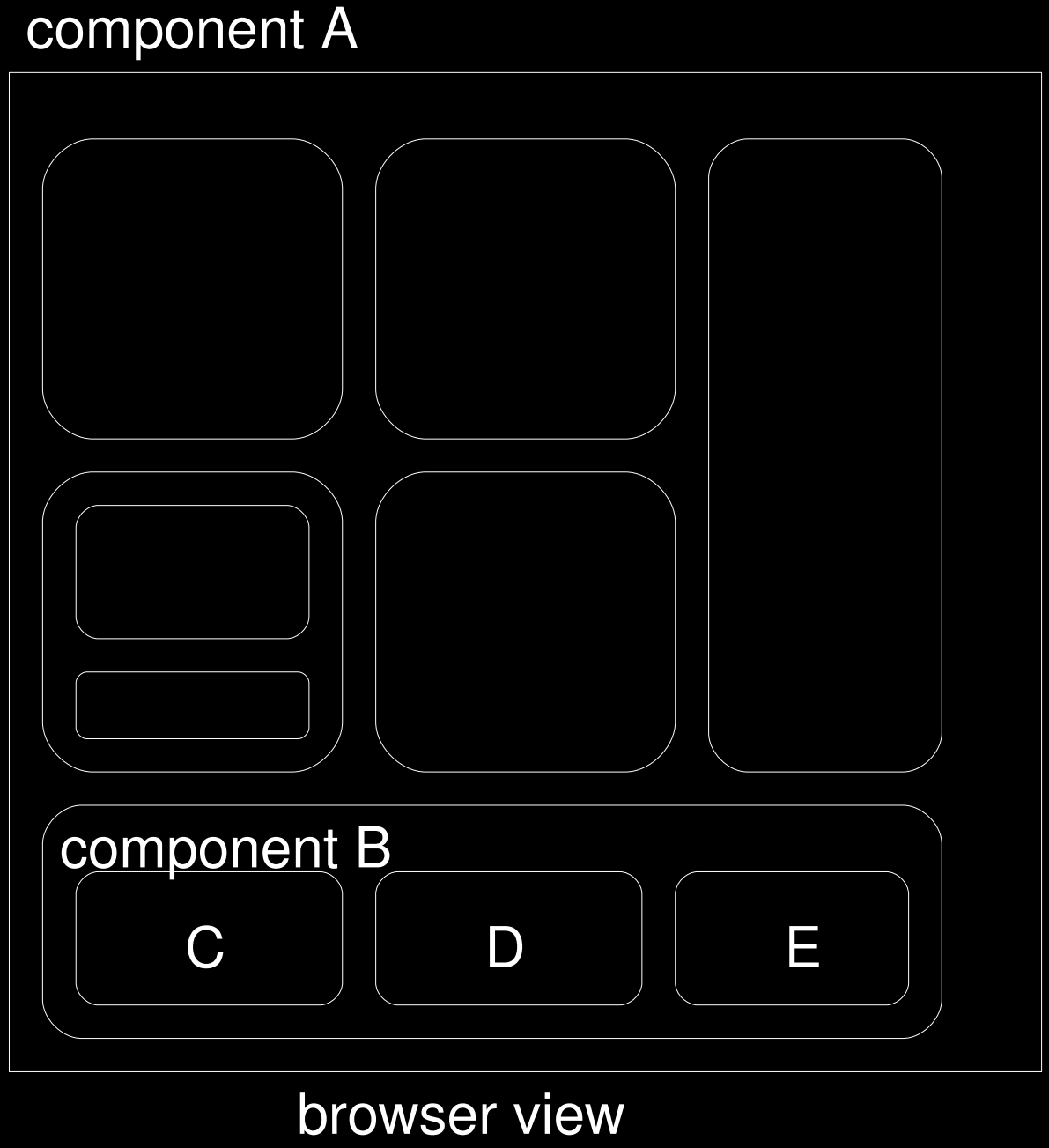
Reusability

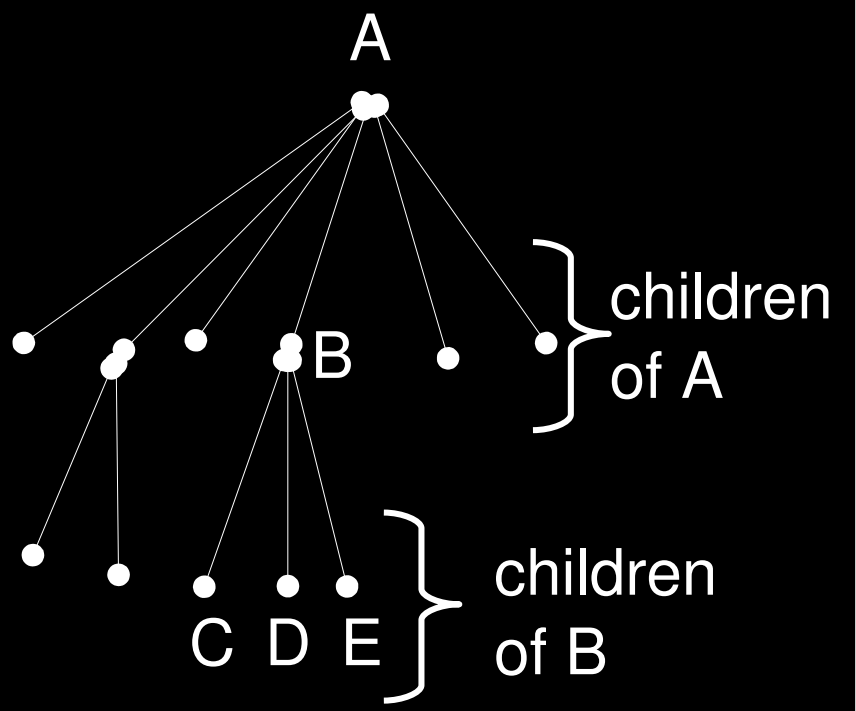
compose your components

There is more than one note

in a music partition, right?

Nesting components





How to nest components

A component :

- declares publicly its children components
- asks the html canvas to render its children

Debug

How do you do it?

Debug

Hum, better to not remember it...

Debug tourism on the Seaside

All the **Smalltalk** machinery is there for debugging,

and it is hudge!

The **Smalltalk image** holds the state
of your web application

The **Smalltalk environment** is the place
to debug your web application

Free tour to debug land...

AJAX is there as well

A real example with *i*Stoa.net