

30th ESUG Conference
European Smalltalk Users Group

WebST

Lille, France, 8 - 11 July 2024



Smalltalk 2024 Events

ESUG 2024 has started

[Checkout the Program!](#)

Smalltalk Meetup Zurich

I want to make this web page

107 days	22 hours	56 minutes	49 seconds
--------------------	--------------------	----------------------	----------------------

Join Us in Zurich, Switzerland!



Smalltalk 2024 Events

ESUG 2024 has started

[Checkout the Program!](#)

Smalltalk Meetup Zurich

Updates every second

107 days	22 hours	56 minutes	49 seconds
--------------------	--------------------	----------------------	----------------------

Join Us in Zurich, Switzerland!



Smalltalk 2024 Events

ESUG 2024 has started

[Checkout the Program!](#)

Smalltalk Meetup Zurich

Different View States

107 days	22 hours	56 minutes	49 seconds
--------------------	--------------------	----------------------	----------------------

Join Us in Zurich, Switzerland!



Smalltalk 2024 Events

ESUG 2024 has started

[Checkout the Program!](#)

Smalltalk Meetup Zurich

But, I am
Lazy!

107 days	22 hours	56 minutes	49 seconds
--------------------	--------------------	----------------------	----------------------

Join Us in Zurich, Switzerland!



Smalltalk
ESUG

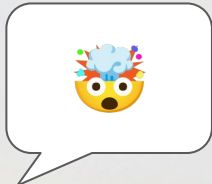
**Web Components
you reuse must**

[Checkout the Program!](#)

Smalltalk Meetup Zurich

107 days	22 hours	56 minutes	49 seconds
--------------------	--------------------	----------------------	----------------------

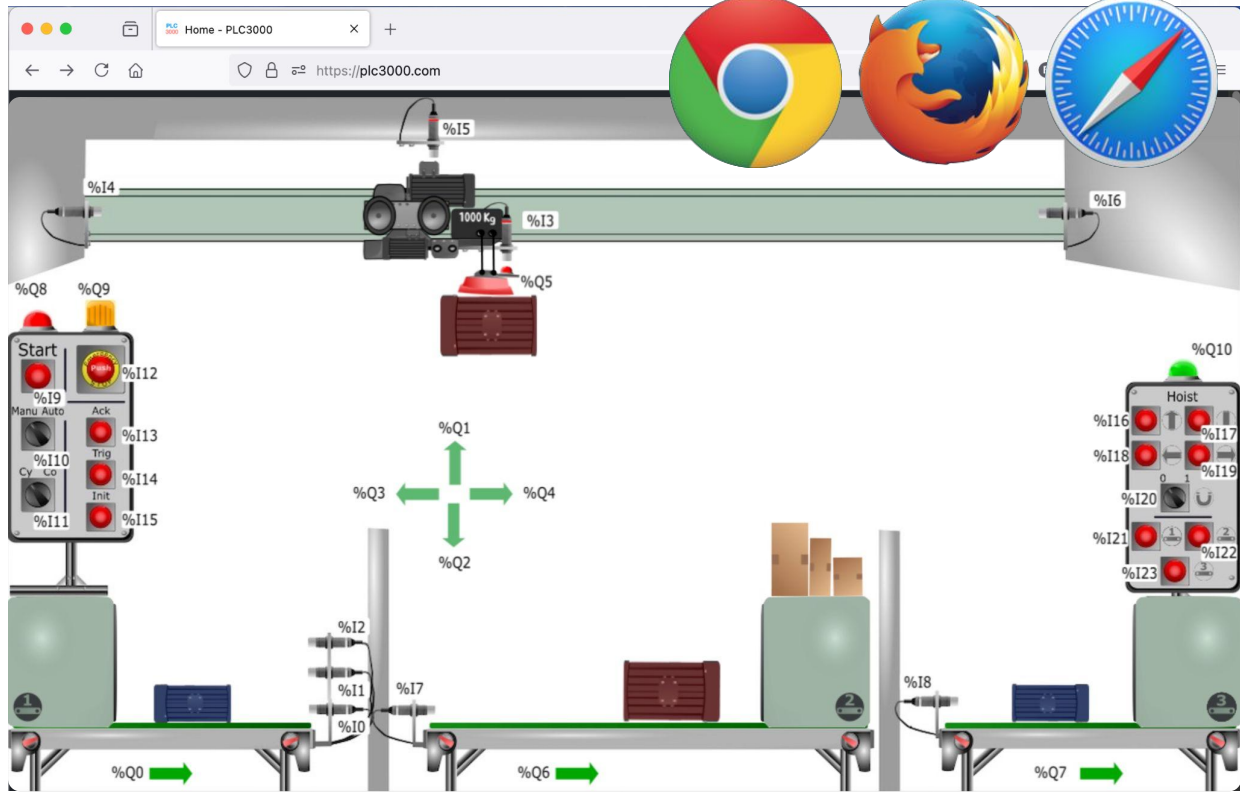
Join Us in Zurich, Switzerland!





Noury
Bouraqadi

PLC3000.com



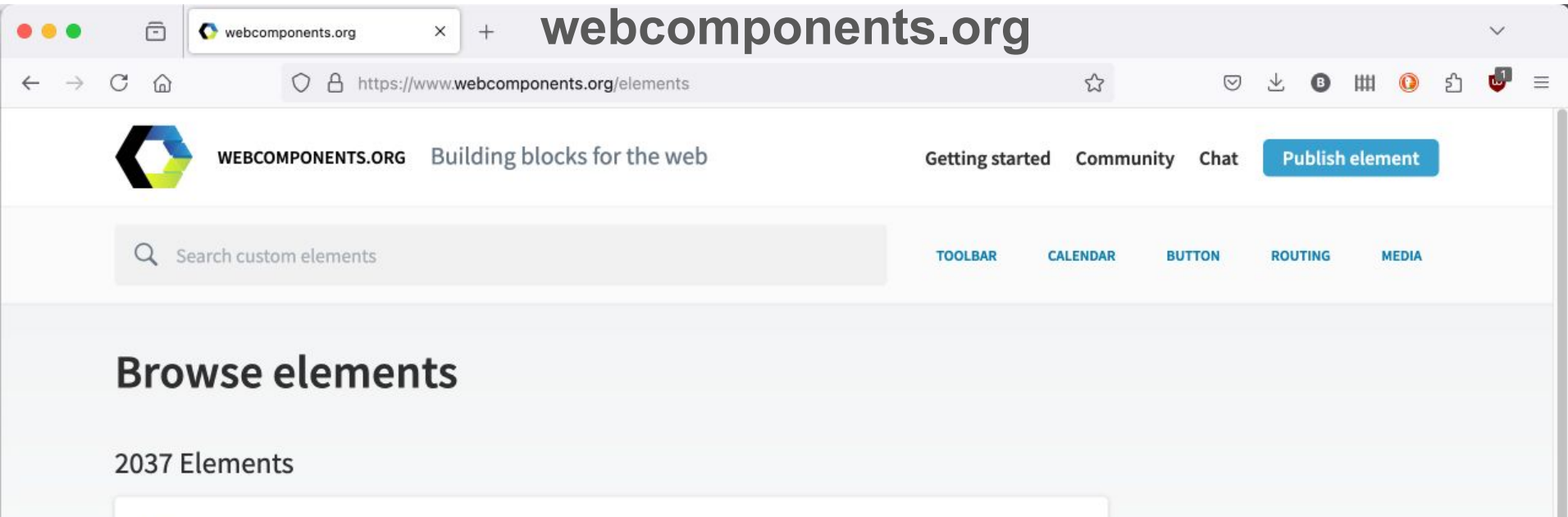
**Noury
Bourağadi**



Web Components & Smalltalk

Web Components = W3C Standard

- Custom HTML tags
- HTML + CSS + JS
- Encapsulation = No collisions



The screenshot shows a web browser window with the URL `https://www.webcomponents.org/elements`. The page header includes the Web Components logo, the text "WEBCOMPONENTS.ORG Building blocks for the web", and navigation links for "Getting started", "Community", "Chat", and a "Publish element" button. A search bar is present with the text "Search custom elements". Below the search bar, a list of element categories is displayed: "TOOLBAR", "CALENDAR", "BUTTON", "ROUTING", and "MEDIA". The main heading is "Browse elements" and it indicates "2037 Elements".



Smalltalk 2024 Events

ESUG 2024 has started

[Checkout the Program!](#)

Smalltalk Meetup Zurich



```
<countdown-timer date="2024-07-08T09:00:00"  
heading="ESUG 2024" ...></countdown-timer>  
  
<countdown-timer date="2024-10-18T18:00:00"  
heading="Smalltalk ...></countdown-timer>  
  
<script src="./countdown-timer.min.js"></script>
```

Smalltalk 2024 Events

ESUG 2024 has started

[Checkout the Program!](#)

Smalltalk Meetup Zurich

But, what if I have special requirements?

7	22	56	49
s	hours	minutes	seconds

Join Us in Zurich, Switzerland!



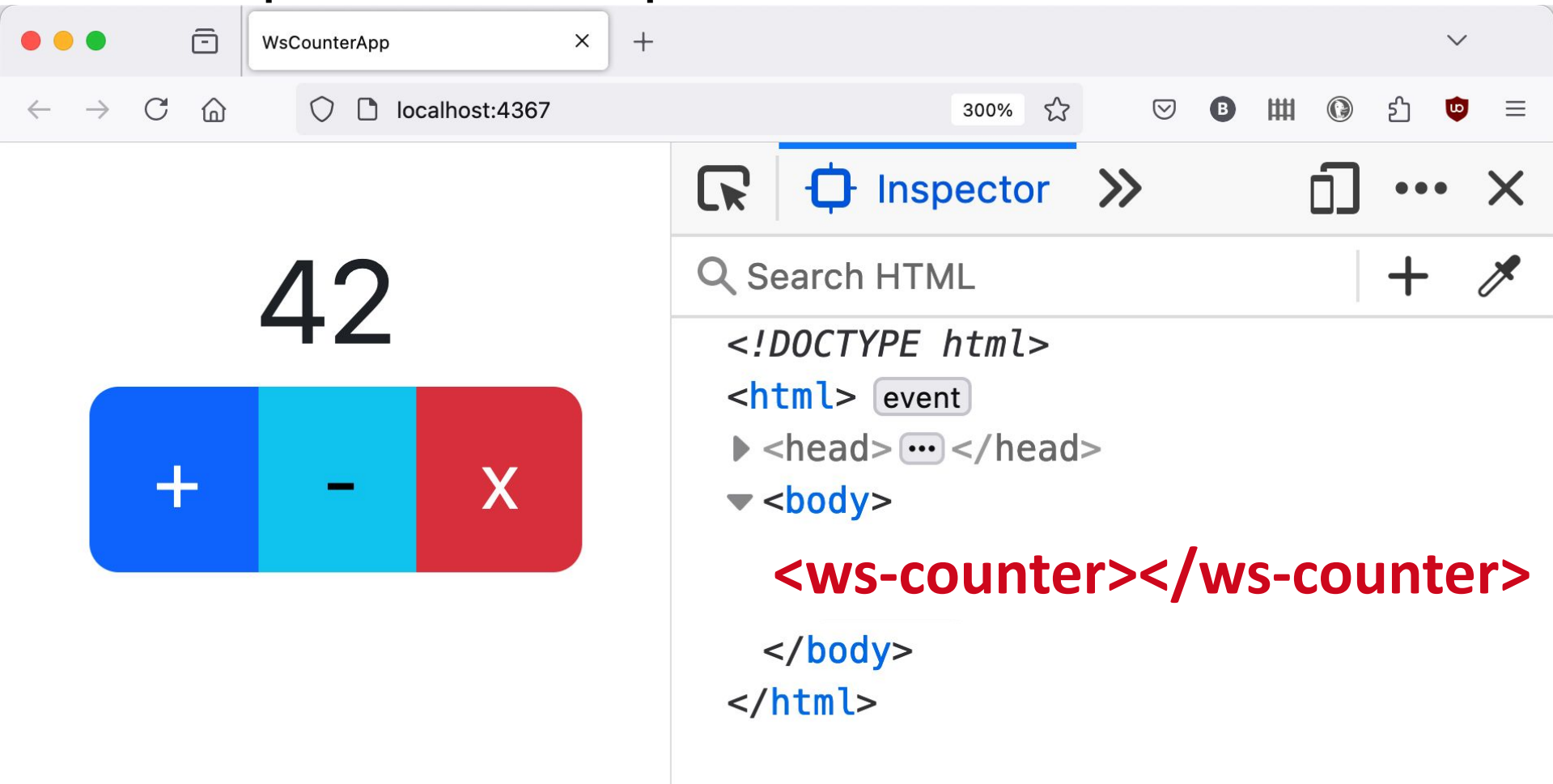
Smalltalk 2024 Events

ESUG 2024 has started

In Smalltalk
Web
Components
you make!



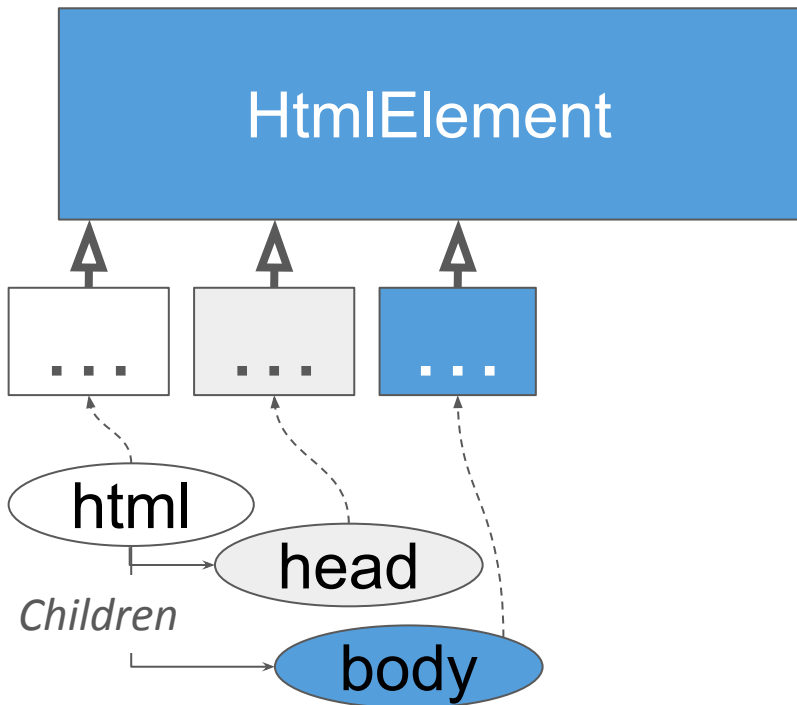
A Component Example



The image shows a web browser window with the title "WsCounterApp" and the address bar "localhost:4367". The browser is displaying a counter application with the number "42" and three buttons: a blue "+" button, a cyan "-" button, and a red "X" button. The browser's developer tools are open, showing the "Inspector" tab. The HTML structure is visible, including the following code:

```
<!DOCTYPE html>  
<html> event  
  <head> ... </head>  
  <body>  
    <ws-counter></ws-counter>  
  </body>  
</html>
```

Web Browsers Build Objects from HTML

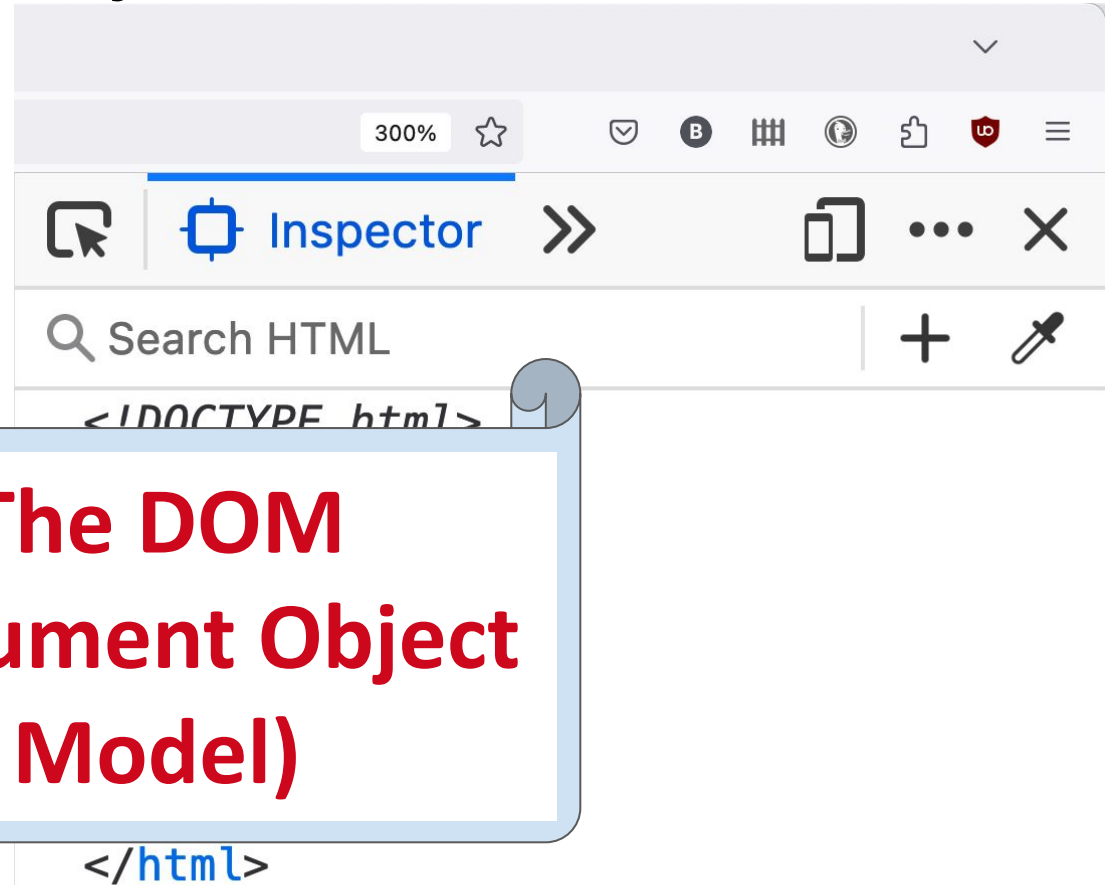
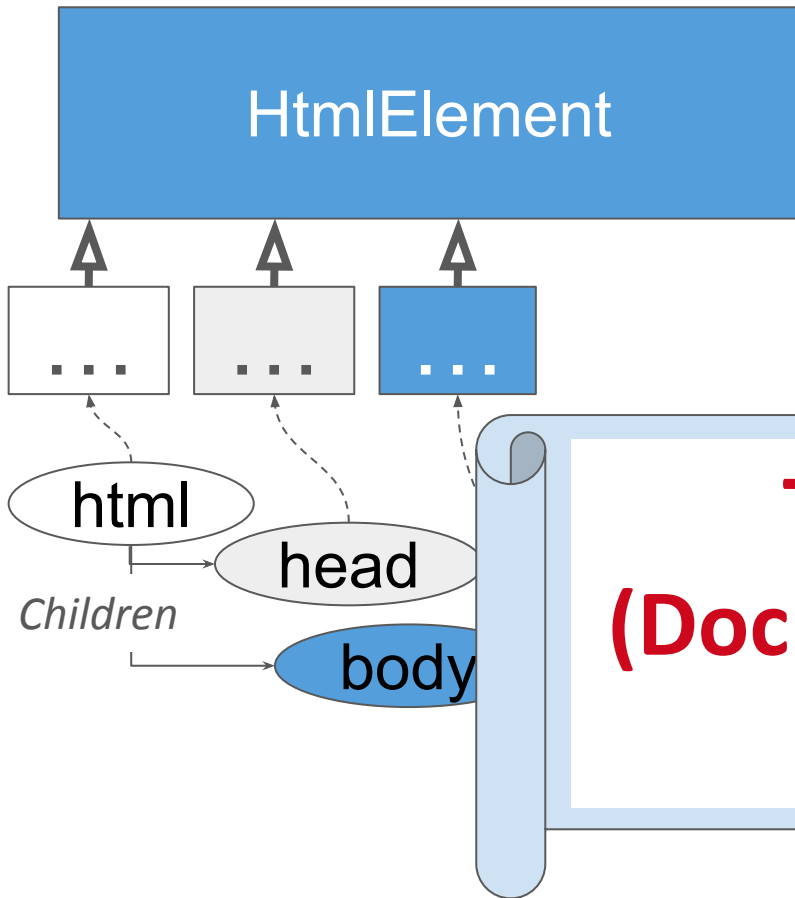


The screenshot shows a web browser's developer tools interface. The top bar indicates a zoom level of 300%. The "Inspector" tab is active, showing the HTML structure of the page. The code is as follows:

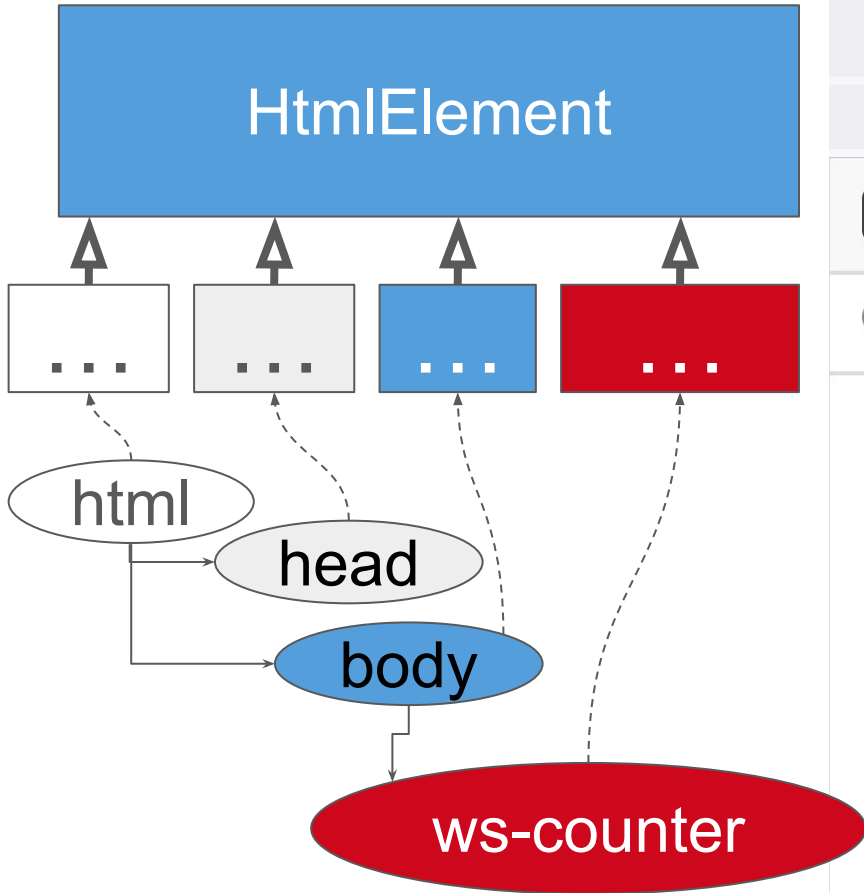
```
<!DOCTYPE html>
<html> event
  <head> ... </head>
  <body>

    </body>
  </html>
```

Web Browsers Build Objects from HTML



Adding a Component = Adding a Subclass

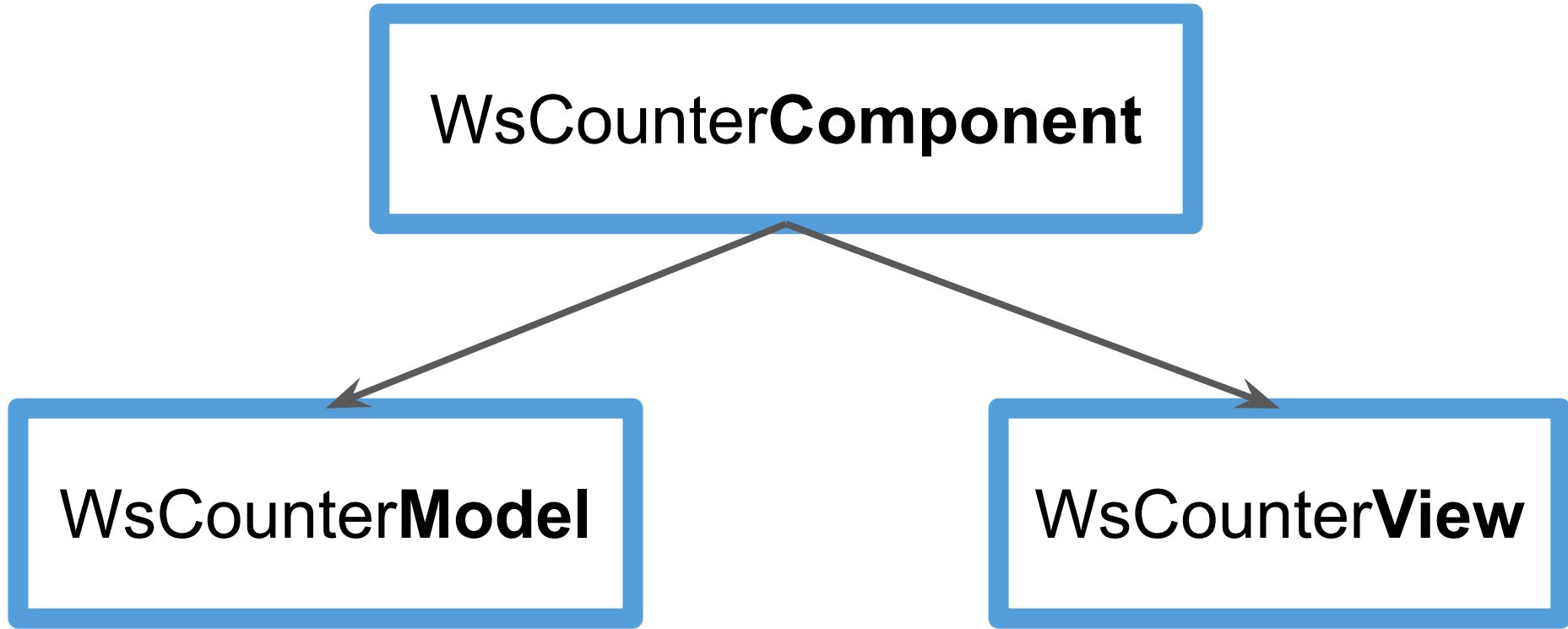


The screenshot shows a browser's developer tools interface. The top bar indicates a zoom level of 300%. The "Inspector" tab is active, showing the DOM tree. The search bar contains "Search HTML". The DOM tree shows the following structure:

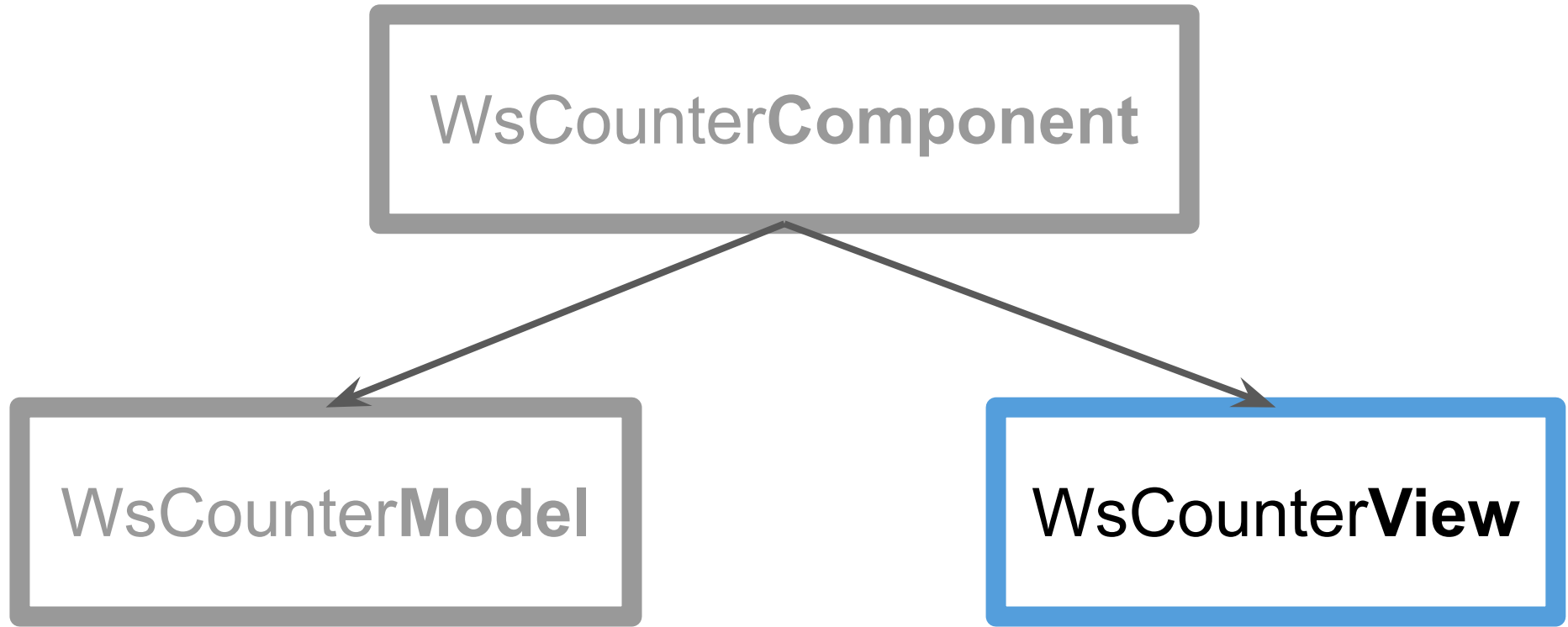
```
<!DOCTYPE html>  
<html> event  
  <head> ... </head>  
  <body>  
    <ws-counter></ws-counter>  
  </body>  
</html>
```

The `<ws-counter></ws-counter>` tag is highlighted in red in the DOM tree and in the source code below.

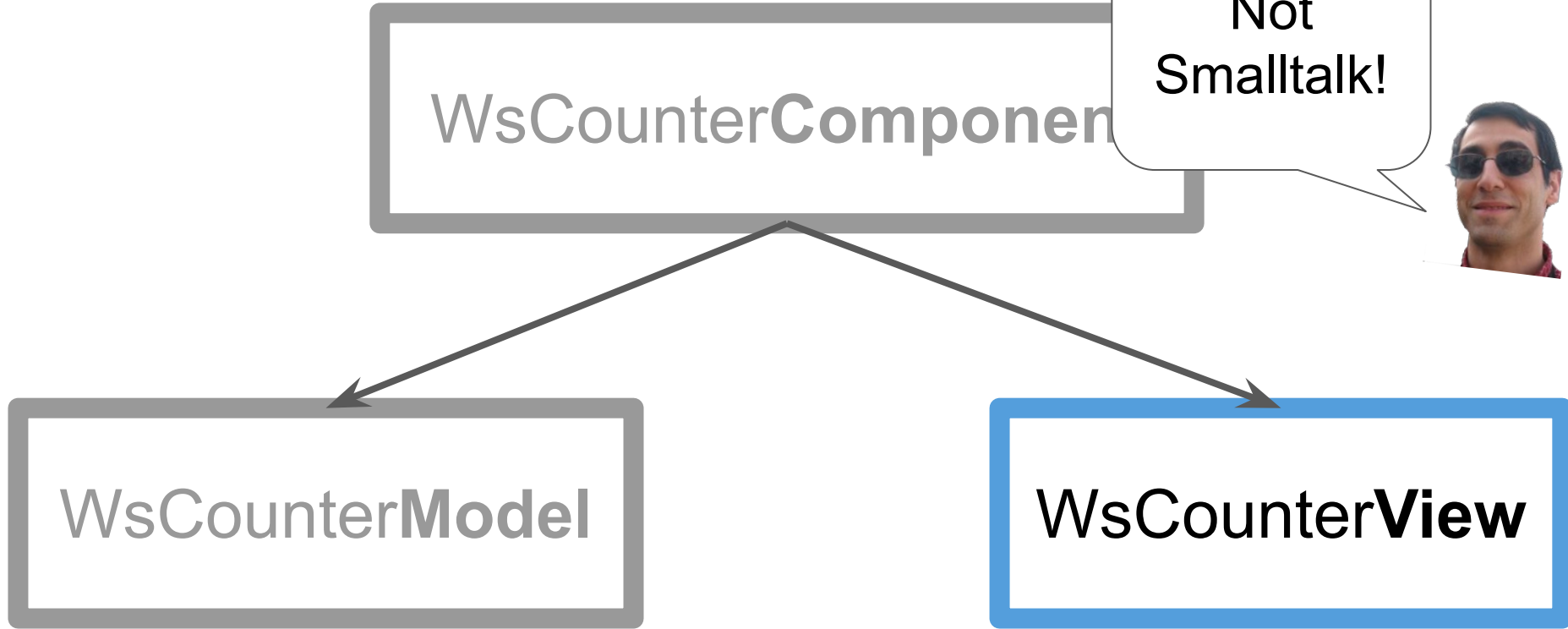
Component = View + Model



Component View = HTML + CSS 🙄



Component View = HTML + CSS 🙄





The
seaside ★ force
you must use





The
seaside ★ force
 you must use



Thanks to Master Johan Brichau
 & to the other Jedis!





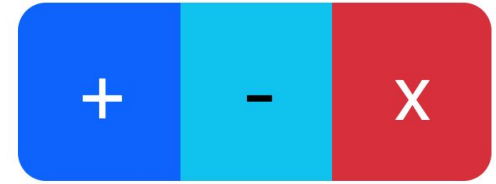
Web Apps in Smalltalk

- Client: HTML Generation
- Server: Requests Handling



Counter View Required HTML

42



```
<div class="display me-1 text-center fs-1">42</div>
```

```
<button class="increment btn btn-primary" type="button">+</button>
```

```
<button class="decrement btn btn-info" type="button">-</button>
```

```
<button class="reset btn btn-danger" type="button">x</button>
```


Counter View Definition



renderOuterHtmlOn: html

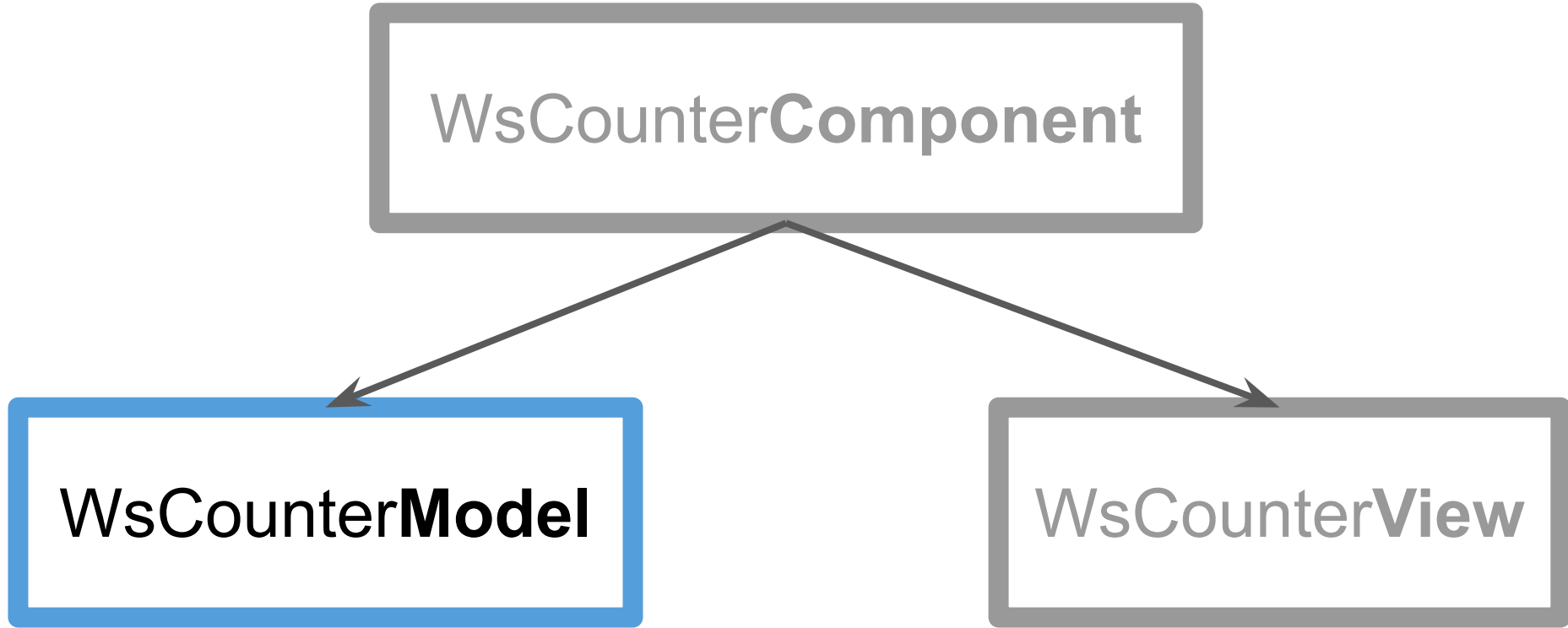
html **div** class: 'display me-1 text-center fs-1'.

html **button** class: 'increment btn btn-primary'; value: '+'.

html **button** class: 'decrement btn btn-info'; value: '-'.

html **button** class: 'reset btn btn-danger'; value: 'x'

Component Model = Business as Usual 🤗



Component Model

increment

self count: self count + 1.

count

[^]count

count: newCount

count := newCount.

countPort notifyWith: newCount

Trivial



Model = Subject in the **Observer Design Pattern**

increment

```
self count: self count + 1.
```

count

```
^count
```

count: newCount

```
count = newCount.
```

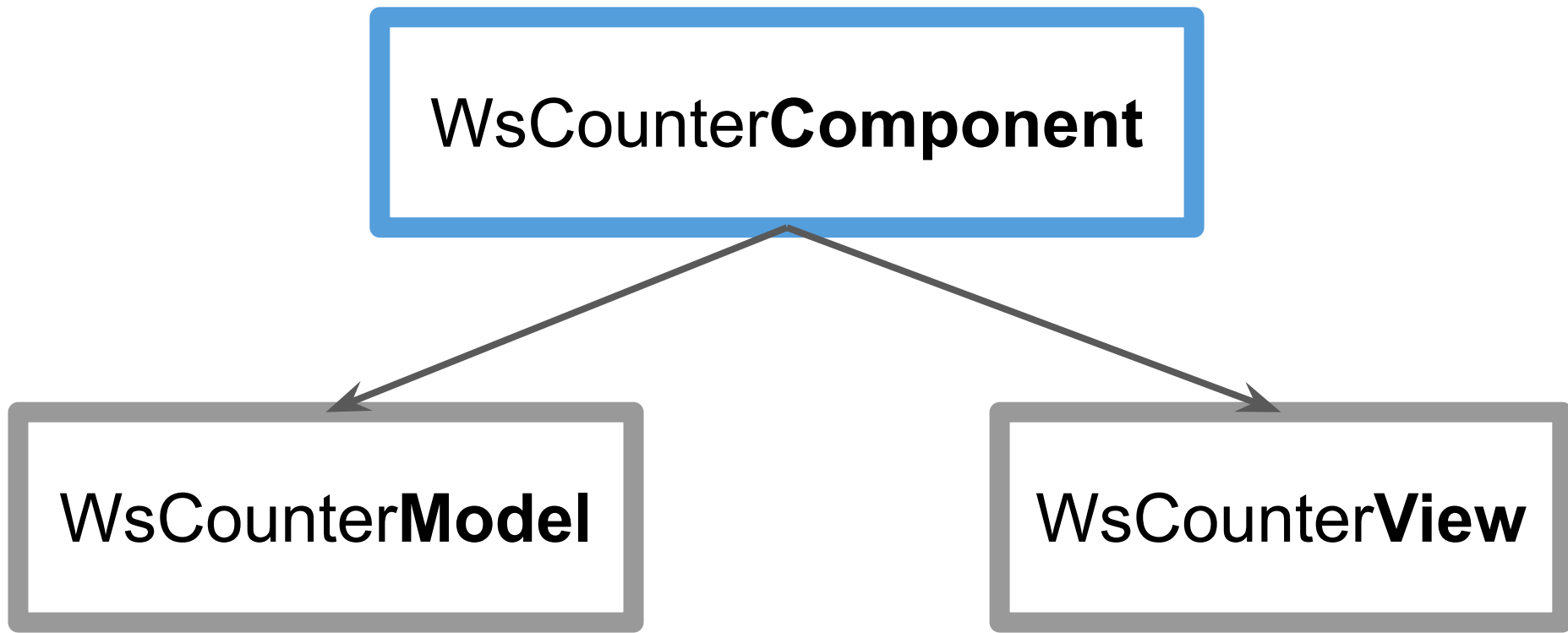
```
countPort notifyWith: newCount
```



Announcer

The diagram illustrates the interaction between the Announcer and the countPort. A blue arrow originates from the Announcer box and points to the countPort object, indicating the direction of the message passing.

Component Links View & Model = MVC



Component Links Model to View

initModel

super initModel.

model -@ #countPort => [:newCount |

self displayCount: newCount].

self updateDisplay

Update View When
Model Changes

updateDisplay

display textContent: model count asString

View Observes the Model

initModel

```
super initModel.
```

```
model -@ #countPort => [ :newCount |
```

```
self displayCount: newCount ].
```

```
self updateDisplay
```

Subscribe to count changes



updateDisplay

```
display textContent: model count asString
```

Component Links View to Model

initView

```
| incrementButton decrementButton resetButton |
```

```
super initView.
```

```
display := view querySel
```

```
incrementButton := view
```



Clicks on View
Increment Model

```
self onClickElement: incrementButton do: [ model increment ].
```

```
decrementButton := ...
```


Navigating the DOM

initView

| incrementButton decrementButton resetButton
super initView.

display := view **querySelector**: '.display'.

incrementButton := view **querySelector**: '.increment'.

self onClickElement: incrementButton do: [model increment].

decrementButton := ...

JavaScript
API !





The
*Phar***JS** force
you must use



Develop
100% Phar **12**

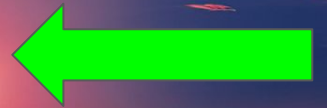
Run 100%
JavaScript

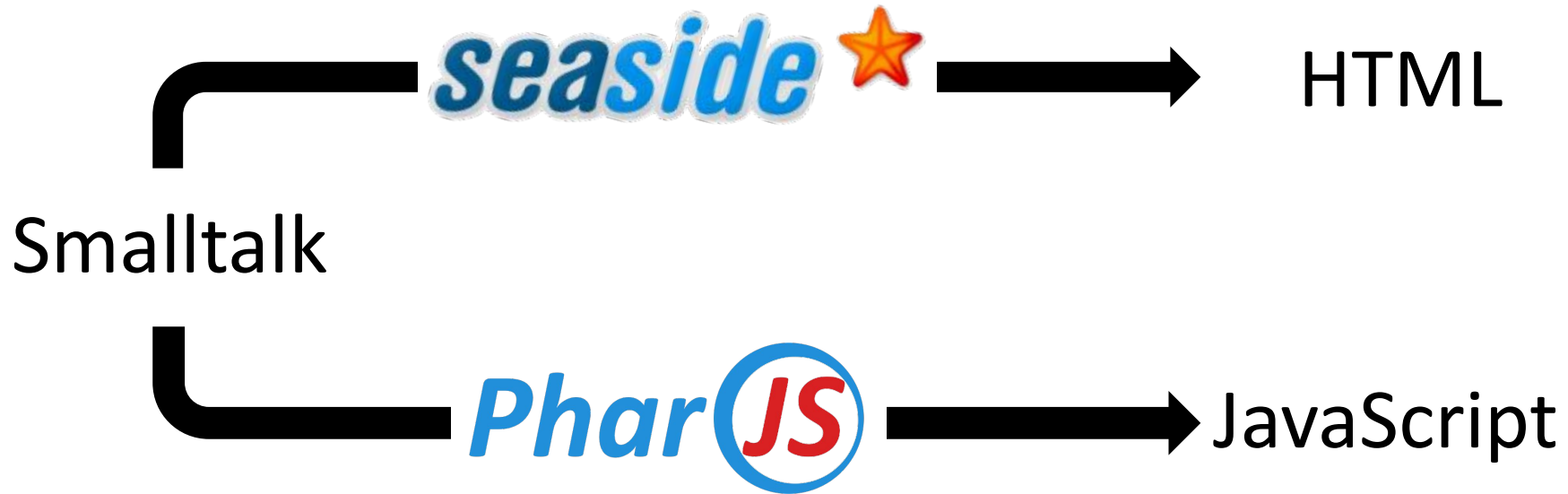




Develop 100% Pharo Run 100% JavaScript

- Generate JavaScript from Smalltalk
- Link JavaScript Third Party Libraries
- Test JavaScript + Web Clients





HTML
JavaScript
CSS ?



Reuse
you must!

HTML
JavaScript
CSS ?



Phar **JS**
The ~~JavaScript~~
force you must use!



Reuse
Candidate



Build fast, responsive sites with Bootstrap

Powerful, extensible, and feature-packed frontend toolkit. Build and customize with Sass, utilize prebuilt grid system and components, and bring projects to life with powerful JavaScript plugins.

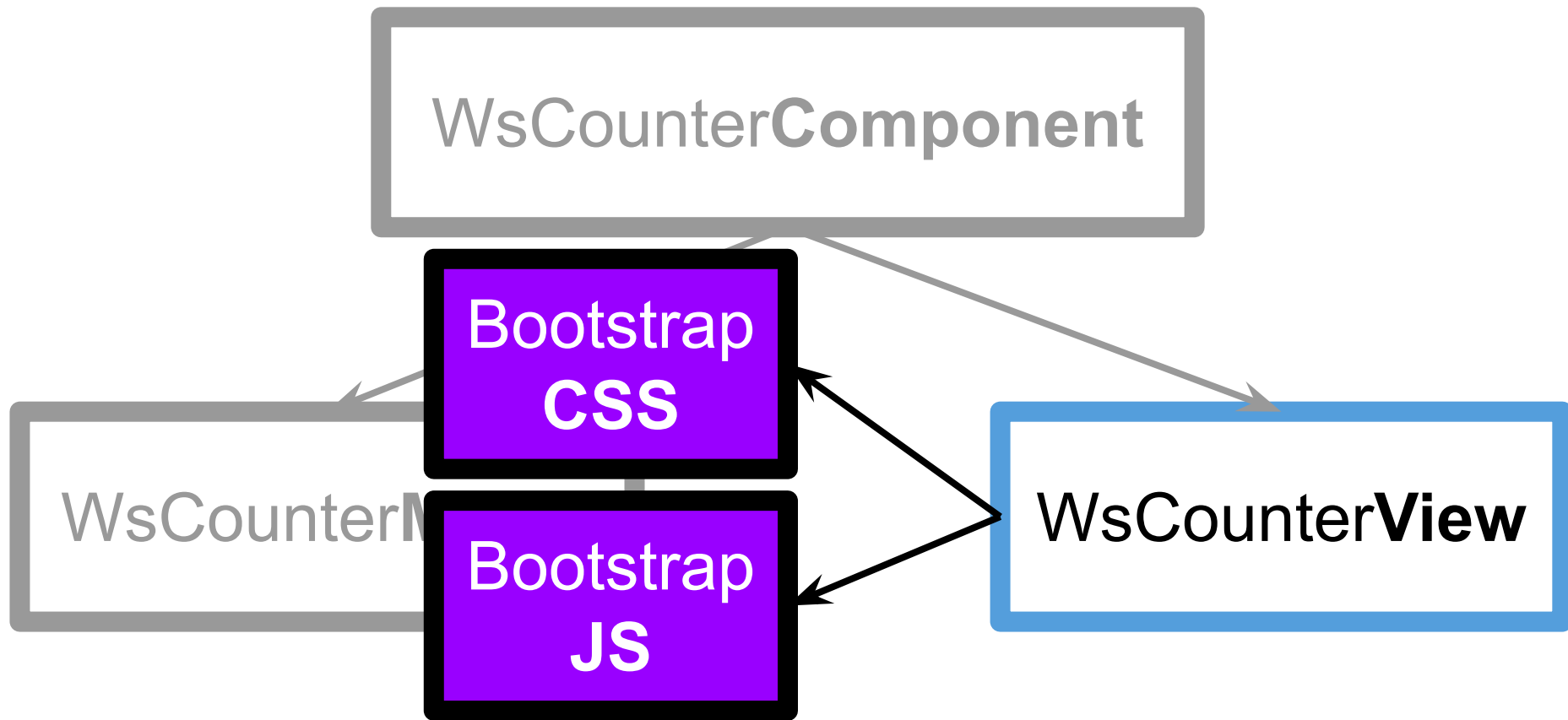
```
$ npm i bootstrap@5.3.3
```



 [Read the docs](#)

Currently **v5.3.3** · [Download](#) · [All releases](#)

A View can Link Resources



seaside to Render CSS Resources

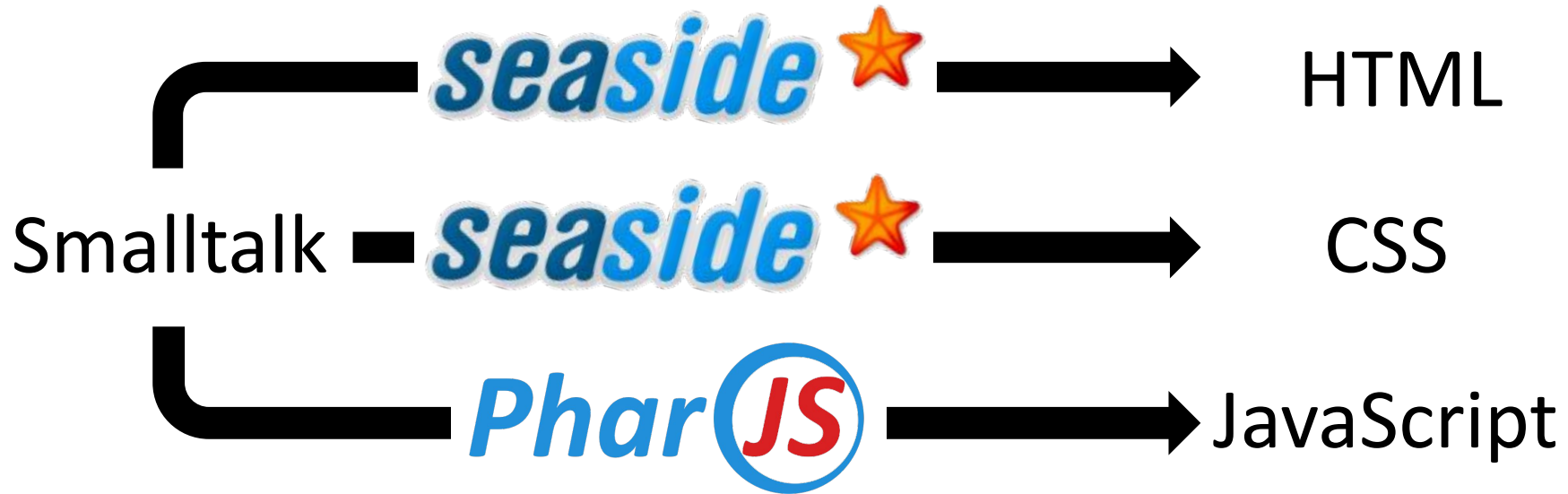
renderTagOn: html

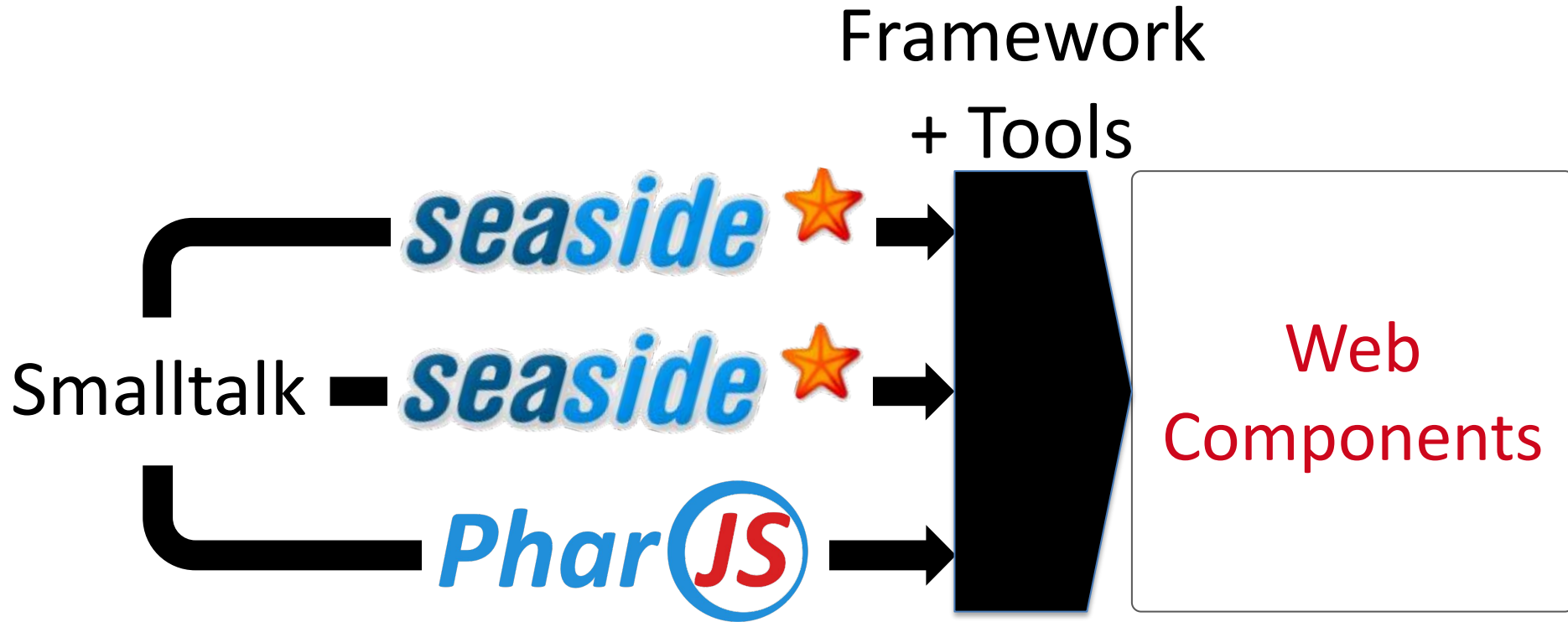
^ html **link**
beStylesheet;
url: self url;
yourself

seaside to Render JavaScript Resources

renderTagOn: html

```
^ html script
  defer;
  type: 'text/javascript';
  url: self url;
  yourself
```









<i>Game Score</i>					
Player A			Player B		
18			21		
+	-	x	+	-	x

How to Make
Complex
Components?



Composite Design
Pattern
You must use!

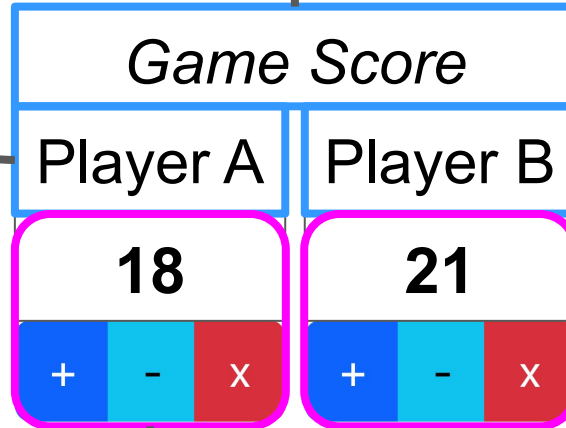


<i>Game Score</i>					
Player A			Player B		
18			21		
+	-	x	+	-	x

How to make
complex
components?

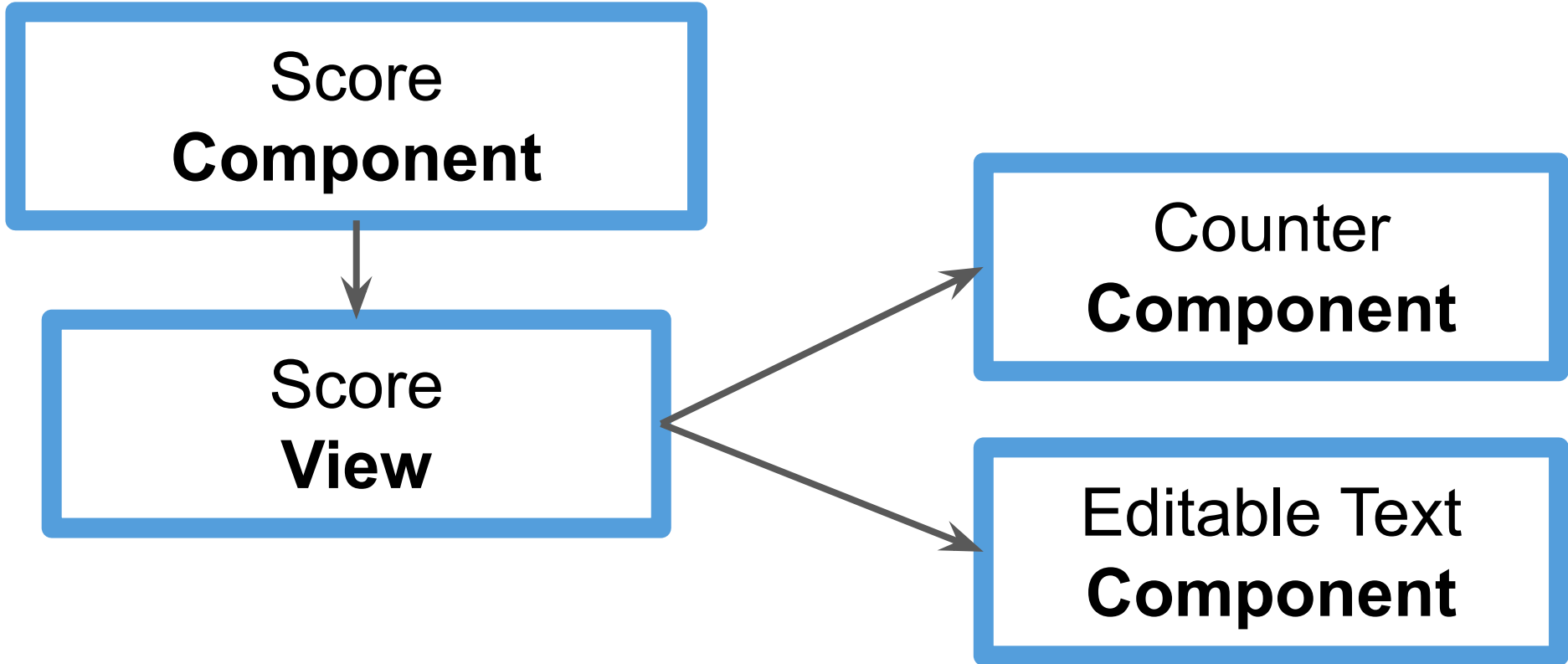


***Editable Text
Component***



***Counter
Component***

View Rendering Sub-Components



View Rendering Sub-Components

html

Smalltalk

```
render: EditableTextComponent  
dataAttributes: { #text -> 'Game Score' }.
```

Generated HTML

```
<ws-editable-text data-text="Game Score">  
</ws-editable-text>
```

I want
more!

I want to develop
100% in Smalltalk
ALL THE TIME!





Export Full Web
Clients + Server
You Can



Wait!

I want
Test-Driven Development
too!





The
*Phar***JS**
TDD force
you must use

Test Client-Server
Interaction from
Both Sides!



Pharo *100%*



1. Write tests
2. Pass the tests
3. Export

Server + Clients 100%
HTML + JavaScript

Final Thoughts



WebST



Development



Production

seaside 

Phar  ⁵⁸



seaside 

Phar  ⁵⁹



Willow



Egg
Smalltalk



WebST



Code
Paradise

`[:web | side]`



[GitHub.com/bouraqadi/WebST](https://github.com/bouraqadi/WebST)

WebST

Web Components
& Smalltalk

