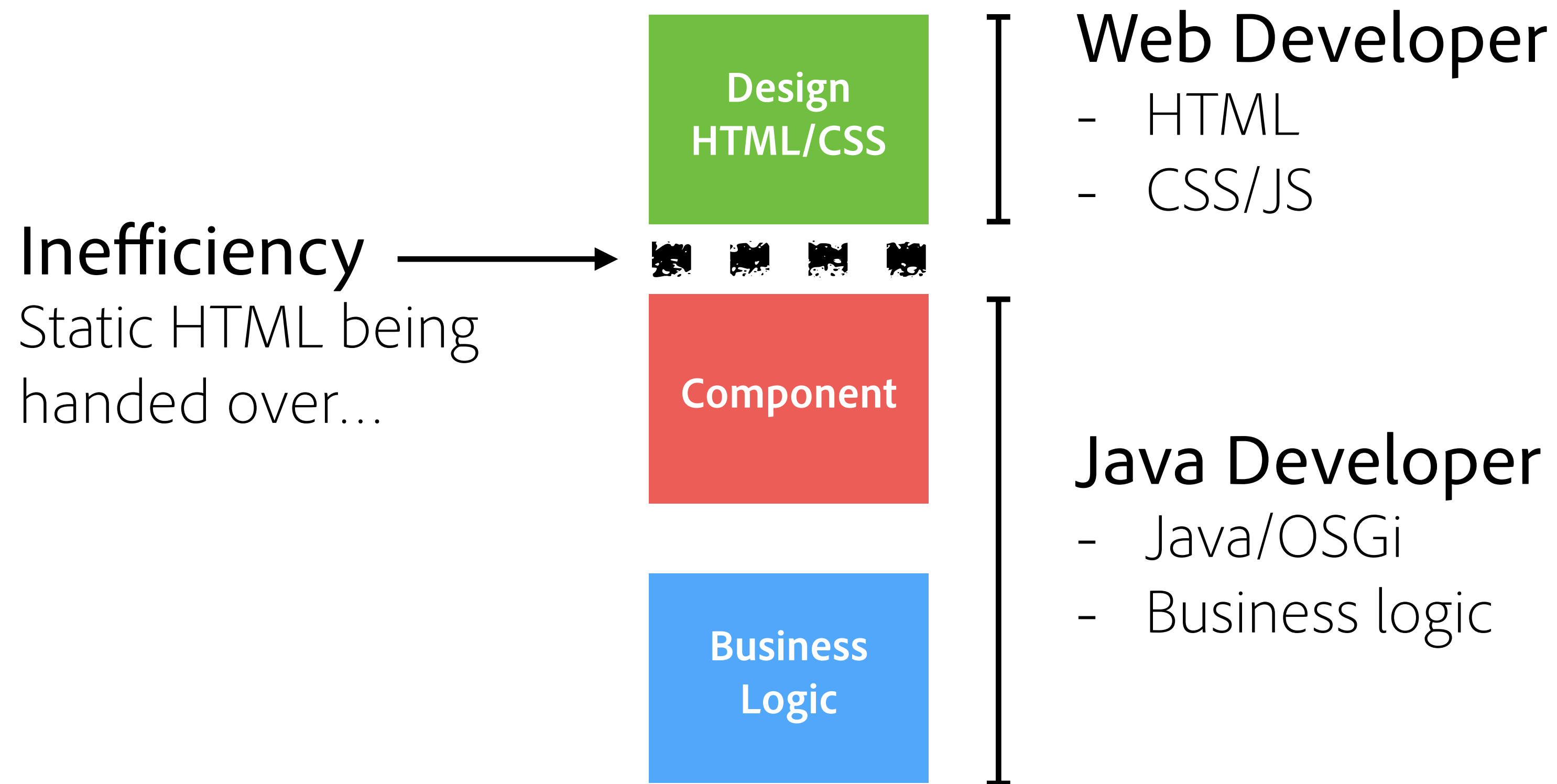


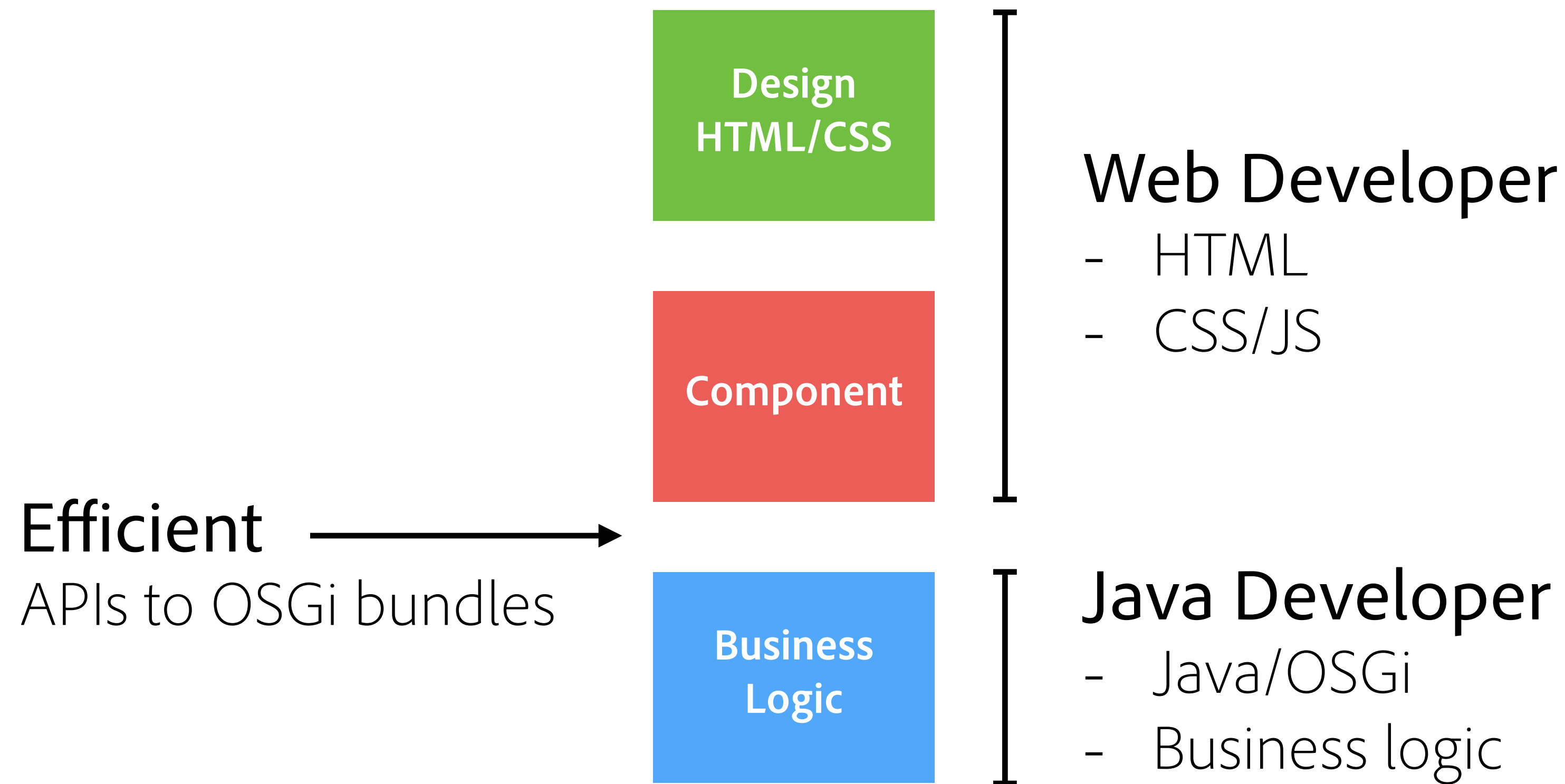
Sightly Component Development

@GabrielWalt, Product Manager
@RaduCotescu, Product Developer

Development Workflow



Development Workflow



Development Workflow

Component

Can be edited by Web Developer:

- × JSP (HTML markup & component logic)
- ✓ **Client Libraries** (CSS & JS)

Development Workflow

Component

Can be edited by Web Developer:

- ✓ **HTML markup** (Sightly template)
- ✓ **Component logic** (server-side JS)
- × ~~JSP (HTML markup & component logic)~~
- ✓ **Client Libraries** (CSS & JS)

sightly
O BEAUTIFUL MARKUP

Sightly

Lightweight

No dependencies, fast and lean

Secure

Automatic contextual XSS protection and URL externalization

Code-less

Enforce separation of concerns between logic and markup

Powerful

Straight-forward API for logic, allowing to do virtually anything

Intuitive

Clear, simple & restricted feature set



Sightly vs JSP

Sightly

```
<a href="{properties.link || '#'}" title="{properties.jcr:title}">
  {properties.jcr:description}
</a>
```

JSP

```
<a href="{<%= xssAPI.getValidHref(properties.get("link", "#")) %>}" <%
  String title = properties.get("jcr:title", "");
  if (title.length() > 0) {
    %>title="{<%= xssAPI.encodeForHTMLAttr(title) %>}"<%
  } %>>
  <%= xssAPI.encodeForHTML(properties.get("jcr:description", "")) %>
</a>
```


Building Blocks

Expression Language

`${properties.myProperty}`

Block Statements

`<p data-sly-test="${isVisible}">${text}</p>`

Use-API

`<p data-sly-use.obj="script.js">${obj.text}</p>`

Expressions

Literals

`${42}`

`${true}`

`${'Hello World'}`

`${[1, 2, 3]}`

Variables

`${myVar}`

`${properties.propName}`

`${properties.jcr:title}`

`${properties['my property']}`

`${properties[myVar]}`

Expression Operators

Logical operations

`${!myVar}`

`${conditionOne || conditionTwo}`

`${conditionOne && conditionTwo}`

`${properties.jcr:title || conditionTwo}`

Equality / Inequality (only for same types)

`${varOne == varTwo}` `${varOne != varTwo}`

Comparison (only for integers)

`${varOne < varTwo}` `${varOne > varTwo}`

`${varOne <= varTwo}` `${varOne >= varTwo}`

Expression Operators

Conditional

`${myChoice ? varOne : varTwo}`

Grouping

`${varOne && (varTwo || varThree)}`

Expression Options

Everything after the @ is an option

```
${myVar @ optOne, optTwo}
```

```
${myVar @ optOne='value', optTwo=[1, 2, 3]}
```

Parametric expression, containing only options

```
${@ optOne='value', optTwo=[1, 2, 3]}
```

Expression Options

String formatting

```
${'Page {0} of {1}' @ format=[current, total]}
```

Internationalization

```
${'Page' @ i18n}
```

```
${'Page' @ i18n, hint='Translation Hint'}
```

```
${'Page' @ i18n, source='user'}
```

```
${'Page' @ i18n, locale='en-US'}
```

Array Join

```
${['one', 'two'] @ join='; '}
```

Display Context Option

Offers control over escaping and XSS protection

Allowing some HTML markup (will apply XSSAPI.filterHTML)

```
<div>${properties.jcr:description @ context='html'}</div>
```

Adding URI XSS protection to other fields than src or href

```
<p data-link="${link @ context='uri'}">text</p>
```

In script or style contexts, the context option is mandatory

```
<script>trackId="${id @ context='scriptString'}";</script>
```

Use Statement

Initializes a helper object

```
<div data-sly-use.nav="navigation.js">${nav.foo}</div>
```

Output

```
<div>Hello World</div>
```


Server-side JavaScript logic

```
<!--/* template.html */-->
```

```
<div data-sly-use.nav="navigation.js">${nav.foo}</div>
```

```
<!--/* navigation.js */-->
```

```
use(function () {  
    return {  
        foo: "Hello World"  
    };  
});
```

```
<!--/* template.html */-->
```

```
<div data-sly-use.nav="Navigation">${nav.foo}</div>
```

```
<!--/* Navigation.java */-->
```

```
package apps.site_name.component_name;
```

```
import com.adobe.cq.sightly.WCMUse;
```

```
public class Component extends WCMUse {  
    private String foo;
```

```
    @Override
```

```
    public void activate() throws Exception {  
        foo = "Hello World";
```

```
    }
```

```
    public String getFoo() {  
        return foo;
```

```
    }
```

```
}
```

Server-side Java-Script vs Java

Server-side JavaScript

- Can easily be edited by web developers

Java

- Fast
 - Easy to debug
 - Can be located in the component folder, or in an OSGi bundle
- We are working on improving that for server-side JavaScript**

Unwrap Statement

Removes the host element while retaining its content

```
<div data-sly-use.nav="navigation.js" data-sly-unwrap>  
  ${nav.foo}</div>
```

Output

Hello World

Use unwrap with care, the template should correspond as much as possible to the output.

Text, Attr & Elem Statements

Replaces the content, attribute or element name

```
<div class="active" title="Lorem ipsum"  
  data-sly-element="{elementName}"  
  data-sly-attribute.title="{className}"  
  data-sly-text="{content}">Lorem ipsum</div>
```

Output

```
<span class="active" title="Hi!">Hello World</span>
```

Use element with care, it can make the template confusing when the element name gets changed.

Test Statement

Conditionally removes the element and it's content

```
<p data-sly-test="{properties.showText}">text</p>
```

```
<p data-sly-test.abc="{a || b || c}">is true</p>
```

```
<p data-sly-test="{!abc}">or not</p>
```

Output

```
<p>text</p>
```

```
<p>is true</p>
```

List Statement

Repeats the content for each enumerable property

```
<ol data-sly-list="{currentPage.listChildren}">  
    <li>{item.title}</li>  
</ol>
```

Output

```
<ol>  
    <li>Triangle Page</li>  
    <li>Square Page</li>  
</ol>
```

Resource Statement

Includes the result of the indicated resource

```
<article data-sly-resource="path/to/resource"></article>
```

Output

```
<article><!-- Result of the rendered resource --></article>
```


Resource Statement Options

Manipulating selectors (selectors, addSelectors, removeSelectors)

```
<article data-sly-resource='${path/to/resource' @  
selectors='mobile'}"></article>
```

Overriding the resourceType

```
<article data-sly-resource='${path/to/resource' @  
resourceType='my/resource/type'}"></article>
```

Changing WCM mode

```
<article data-sly-resource='${path/to/resource' @  
wcmmode='disabled'}"></article>
```

Include Statement

Includes the rendering of the indicated template (Sightly, JSP, ESP, etc.)

```
<section data-sly-include="path/to/template.html"></section>
```

Output

```
<section><!-- Result of the rendered resource --></section>
```

Template & Call Statement

```
<!--/* template.html */-->
```

```
<template data-sly-template.one="{@ class, text}">  
  <span class="{class}">{text}</span>  
</template>
```

```
<!--/* other-file.html */-->
```

```
<div data-sly-use.tmpl="template.html"  
  data-sly-call="{tmpl.one @ class='example',  
                                     text='Hi!'}"></div>
```

Output

```
<div><span class="example">Hi!</span></div>
```

Sightly FAQ

Why didn't you choose an existing template language?

- We want it to be coding language agnostic (also client/server agnostic)
- We want it to have just the right feature set (with a strong focus on security)
- We want it to naturally lead developers towards best practice

Do I need to migrate my components to Sightly?

- No, we want Sightly to help you to be more efficient on the new components you build, not loose time on migrating components that already work.

Can I still use JSP, or will JSP get deprecated?

- Sightly and JSP can very well be mixed, even within the same component
- Today, we have no plan to deprecate JSP-based components



Development Workflow

Two developer roles

Component

Web Developer

Business
Logic

Java Developer



Development Workflow

An IDE plugin for each developer role



Brackets plugin

- Slightly code completion & syntax highlighting
- Content nodes & properties manipulation

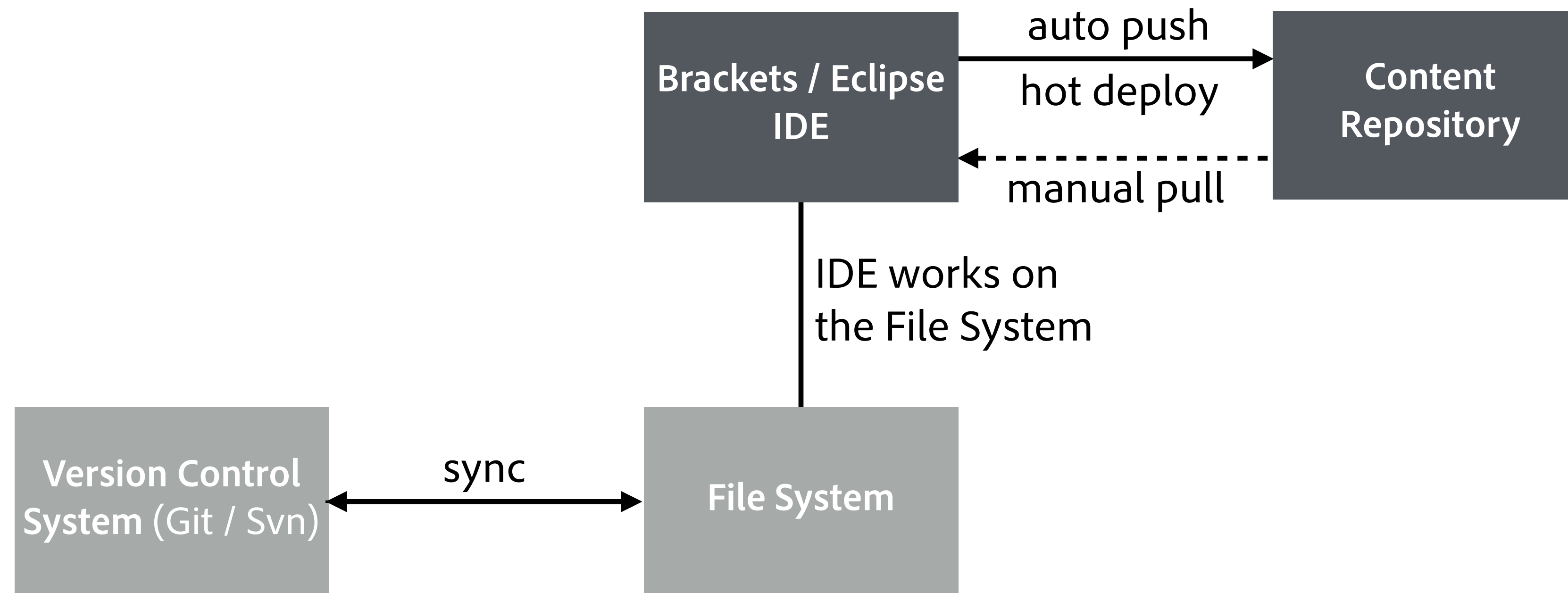


Eclipse plugin

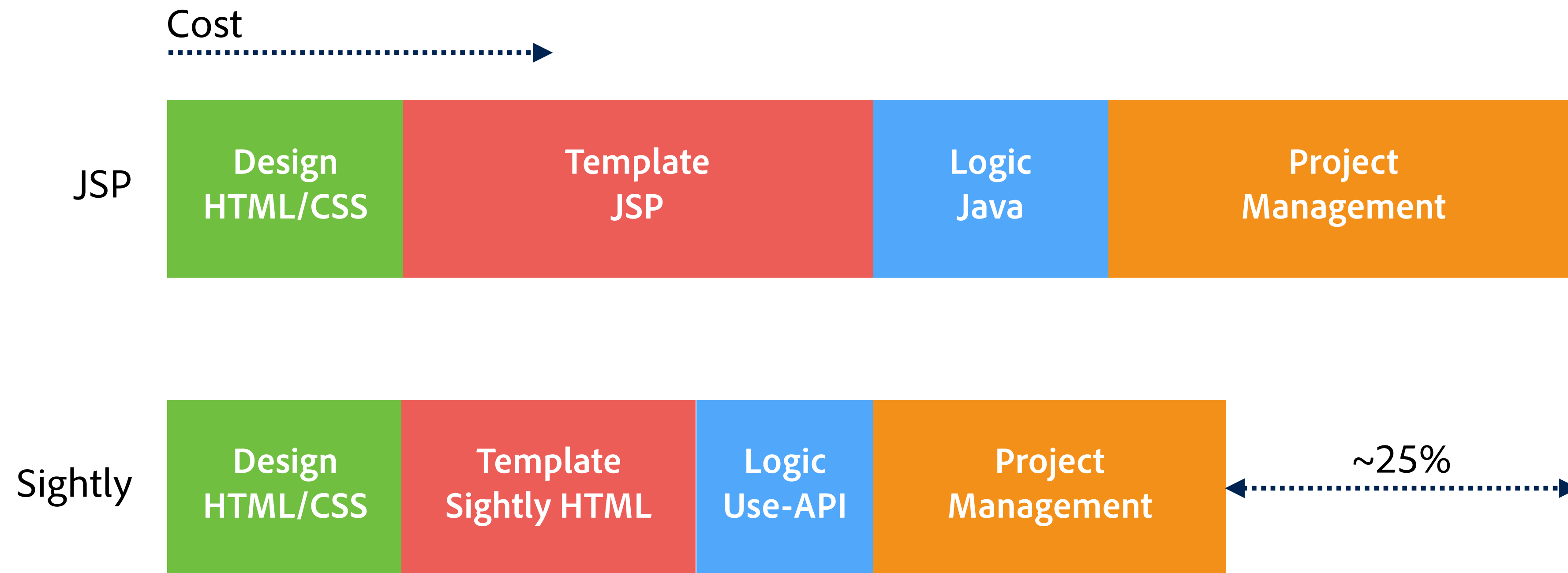
- Content nodes & properties manipulation
- Bundle development & hot deployment

IDE Sync

Work on file system + transparent sync & content editing



Component Development



Resources

Documentation

<http://dev.day.com/content/docs/en/aem/6-0/develop/sightly.html>

Java Docs

<http://docs.adobe.com/content/docs/en/aem/6-0/develop/ref/javadoc/index.html?com/adobe/cq/sightly/WCMUse.html>

Experience Delivers blog posts (<http://experiencedelivers.adobe.com/>)

- [Sightly intro part 1](#)
- [Sightly intro part 2](#)
- [Sightly intro part 3](#)
- [Sightly intro part 4](#)
- [Sightly intro part 5: FAQ](#)





Adobe