



SAP Analytics Cloud Custom Widgets

David Stocker, SAP
March, 2020

PUBLIC

Disclaimer

The information in this presentation is confidential and proprietary to SAP and may not be disclosed without the permission of SAP. Except for your obligation to protect confidential information, this presentation is not subject to your license agreement or any other service or subscription agreement with SAP. SAP has no obligation to pursue any course of business outlined in this presentation or any related document, or to develop or release any functionality mentioned therein.

This presentation, or any related document and SAP's strategy and possible future developments, products and or platforms directions and functionality are all subject to change and may be changed by SAP at any time for any reason without notice. The information in this presentation is not a commitment, promise or legal obligation to deliver any material, code or functionality. This presentation is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. This presentation is for informational purposes and may not be incorporated into a contract. SAP assumes no responsibility for errors or omissions in this presentation, except if such damages were caused by SAP's intentional or gross negligence.

All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.

SAP Analytics Cloud Widget SDK

Provide SDK for custom third-party widgets

Widget SDK

- Allows for custom, third party widgets in stories and apps
- Once installed, are used like SAP delivered widgets
- Built using WebComponents
- Developers can use their favorite JavaScript libraries (e.g. D3)
- Extend the styling panel, for refined widget configuration
- Extend Analytics Designer's scripting API with your own commands
- Create custom script events
- Data driven, with connections to models

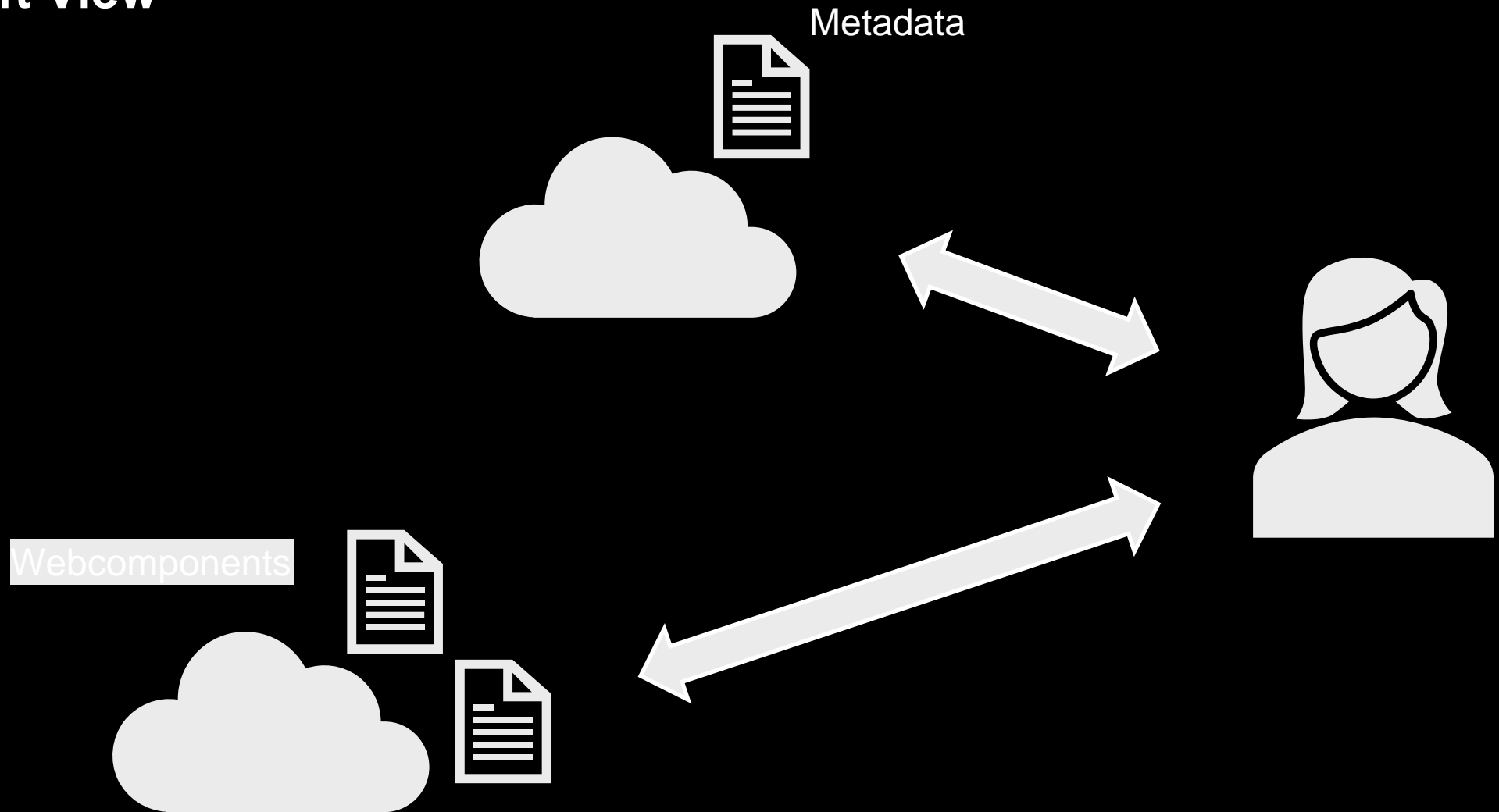
Modern, cloud friendly distribution

- Customers can create their own widgets
- Partners can sell widgets on the SAP Store
- Deployment is handled via the ACN



This is the current state of planning and may be changed by SAP at any time.

10,000 ft View



Metadata

Stored on SAP Analytics Cloud

- Saved in the public folder (.json file)
- Defines the widget, from the perspective of SAP Analytics Cloud
 - Where is the canvas webcomponent hosted?
 - What properties does the widget have?
 - What does its scripting API look like?
 - What events does it support?
 - Does it have a styling sheet footprint? If so, where is that hosted?
- Defines the widget, from the perspective of SAP Analytics Cloud

Web Components

Used for both canvas and styling sheet

- Canvas web component is “main”
- Styling web component is “styling”

Do the actual rendering

Creates a new html tag (custom html tags)

Extends the HTMLElement JavaScript class

```
customElements.define('com-sap-teched-gauge-solution-exe5', class Gauge extends HTMLElement {})
```

Web Component boilerplate structure

External library references

Inner HTML

- Hardcoded HTML and CSS Styling

Constructor

- Runs when the widget is bootstrapped

Getters and Setters

- Run when properties changes

Whatever other methods you might want

Properties

Defined in metadata

- Three attributes in metadata
 - Type (e.g. string or number)
 - Default value
 - Description (for the designer)

```
"properties": {  
  "foo": {  
    "type": "string",  
    "description": "Hi!  I'm a property",  
    "default" : "Hello World"  
  }  
}
```

Have getters and setters in the main web component

- These methods are fired whenever the properties change or are accessed


```
get foo()  
set foo()
```


Methods

Defined in metadata

- Three attributes in metadata
 - Name (e.g. name of the script api call that the designer will use)
 - Parameters
 - Body (script)

```
"methods": {  
  "newFoo": {  
    "description": "Change the value of foo via a method",  
    "parameters": [  
      {  
        "name": "newValue",  
        "type": "string",  
        "description": "The new value of foo"  
      }  
    ],  
    "body": "this.foo = newValue;"  
  }  
},
```



Be careful to only execute scripts involving the current widget (via the keyword ***this***) or objects that you can always expect in the app!

Events

Traditional JavaScript browser events

- Run only within the widget

Scripting events that you define in metadata

- Scripting hooks for your designers
- You call them from within web component

Thank you.

Contact information:

F name L name

Title

Address

Phone number

Partner logo

THE BEST RUN 