Widget API-overzicht

Achtergrond

De Widget API is ontworpen met de volgende doelen in gedachten:

- om de complexiteit van de applicatie te verbergen voor de widgets
- om een kleine en consistente interface te presenteren voor de widgets
- om een stabiele interface te onderhouden voor de widgets

Overzicht

De Widget API is opgesplitst in twee grote subsystemen: het eventsysteem en de interfaces. Het eventsysteem wordt gebruikt om verschillende berichten van de services naar de widgets te zenden en voor widget-naar-widget communicatie. De interfaces zijn een façade naar de services en de Core-functionaliteit.

Widgetberichten

Zoals eerder vermeld, worden de externe widgets over het algemeen gehost op een ander domein en hebben ze geen directe toegang tot de Widget-API. Om de domeinbarrière te omzeilen zal de Core van de applicatie een berichtenkanaal openen met de Messaging-API van de browser.

De onderstaande code demonstreert één manier om in te schrijven op en berichten af te handelen van de API:

```
const origin = 'the origin of the agent application'; let port; window.addEventListener('message', message => { // Make sure that the channel comes from the correct source: if (message.origin !== origin) return; // Setup the communication channel: if (!port) { port = message.ports[0]; port.onmessage = receiver; } }); function receiver(message) { //code here }
```

De payload van de berichten is vervat inmessage.data. De Core zal ook eenessage.data.type tool aan alle berichten toevoegen.

Verzoeken naar de Widget-API kunnen alleen via de opgegeven poort worden verzonden:

```
port.postMessage(message);
```

Interfaces

Om **een eigenschap op te vragen** of **een methode** van de Widget API aan te **roepen** moet het {call, args} berichtformaat worden gebruikt, waarbij call het pad is naar de methode (of eigenschap) in de API.

In het geval van een methode-aanroep, is args een array van alle vereiste argumenten voor de methode-aanroep. Voorbeeld:

```
port.postMessage({ call: 'tab.setTitle', args: ['new title'] });
```

Als de methode een resultaat terugzendt, zal dit door de port.onmessage handler naar de externe widget worden gestuurd in het formaat{name, value, type}, waarbij name de naam is van de gevraagde eigenschap of methode, value de waarde van de eigenschap of het resultaat van de aanroep en type de string 'resultaat' zal zijn.

Voorbeeldreactie op een *getOption*- aanroep:

```
{ name: 'widget.getOption', value: 'https://demo.puzzel.com/dev/widgets/external/demo/', type: 'result' }
```

Let op dat, als gevolg van de manier waarop de Messaging-API werkt, de payload van het bericht zich in de eigenschap themessage.data bevindt.

Als de aangeroepen methode geen resultaat teruggeeft, wordt er geen bericht door de Core verzonden. Als de aangeroepen methode een belofte teruggeeft, wordt het bericht verzonden als de belofte is opgelost of afgewezen. Als de

belofte wordt opgelost, wordt door de Core een standaardbericht over het resultaat verzonden, waarin de waarde de waarde van de belofte bevat. Als de belofte wordt afgewezen wordt een 'error' bericht verstuurd:

```
{ name: 'widget.setOption', value: 'Unexpected end of JSON input', type: 'error' }
```

Indien het nodig is een resultaat te koppelen aan een oproep, kan het optionele ID aan het verzoek worden toegevoegd. Het zal worden teruggestuurd:

```
{ call: 'tab.getOption', args: ['option name'], id: '0123456789' }
```

Resultaat:

```
{ name: 'tab.getOption', value: 'option value', id: '0123456789' }
```

De widget kan ook **een eigenschap** observeren op veranderingen door een{watch}bericht te sturen. Het watchveld moet het pad naar de eigenschap in de Widget-API bevatten.

Als de waarde van die eigenschap verandert, zal de Core een{naam, oud, nieuw, type} bericht sturen, waarbij de naam hetzelfde eigenschapspad is, oud de waarde van die eigenschap voor de verandering, nieuw na de verandering, en type de string "veranderd".

Evenementen

De externe widgets kunnen **zich abonneren op gebeurtenissen** door een {subcribe, options: {once, address}} bericht naar de Core te sturen. Het subscribe veld moet de naam van het event bevatten. Het hele veld opties is optioneel, net als zijn eigenschappen: de booleanonce en de adres string. Het adres heeft dezelfde betekenis als in de methoden van de “Extended Event”-Aggregator. De instelling once betekent dat de subscribeOnce methode zal worden gebruikt, d.w.z. dat de externe widget slechts een enkele gebeurtenis zal ontvangen voordat het abonnement zelf eindigt.

De gebeurtenissen zullen worden ontvangen met een{naam, waarde, type} bericht, waarbij naam de naam van de gebeurtenis is, waarde de payload en type "gebeurtenis".

```
{ name: 'userStatusChanged', value: 'System', type: 'event' }
```

De volledige API-referentie wordt hier beschikbaar gesteld