

Scripting guide and reference of Aspen Editor

[개요]

Aspen Editor 는 편집 문서를 Script 로 제어할 수 있는 기능을 제공한다. 이것은 Interaction 편집과 동일한 개념으로, Page 및 Widget 의 각 Event 에 대응하여 실행할 Script 를 작성하는 방식으로 이루어진다. 결과적으로 Event 에 대해서, 편집 Interaction 과 Script code 가 존재할 수 있으며 개념적으로 이들은 동시에 실행된다. (즉 Interaction 과 Script 를 자유롭게 병행할 수 있다.)

Script 는 편집 문서를 제어하기 위한 것이므로 편집 기능에 대응하는 기능을 가지고 있다. 따라서 Script 의 기능을 이해하기 위해서는 Aspen Editor 의 편집 기능을 이해하는 것이 중요하다.

프로그래밍 언어로는 core JavaScript(ECMA-262)를 지원하며 부가적으로 JSON Library 를 지원한다. JavaScript 는 Web Browser 상에서는 DOM API 와 사용이 되어, DOM API 가 JavaScript 자체로 인식되는 경우가 많으나, DOM API 는 JavaScript 가 호출하는 'Client-side API'이다. 마찬가지로 Aspen Editor 도 편집 문서 제어를 위하여 자체적인 'Client-side API'를 제공하며 본 문서는 이것을 기술하기 위한 것이다.

본 문서에서, 'Page'로 기술된 객체는 Aspen Editor 에 따라서 'Slide'로 표기된다. 또한 Event 의 경우에는 Aspen Editor 에 따라서 지원 Event 가 다를 수가 있으므로, 해당 Aspen Editor 에서 보이지 않으면 지원되지 않는 것으로 간주하면 된다.

1. Execution structure

1.1. Event-driven structure

Page 또는 Widget 에 발생하는 Event 에 대응하여 동작을 Script 를 작성하게 된다. Event 에 의해서 Script 가 호출되는 시점에는 다음과 같은 Context 상에서 실행되게 된다.

- Page - 현재 실행 중인 Page
- Widget - (Widget 에서 발생한 Event 인 경우)
- Event Parameter - 현재 발생한 Event 가 1 개의 Parameter 를 가지는 경우가 있음
- 마지막 Pointer 좌표 - Event 발생 시점에서 Pointer 의 마지막 X, Y 좌표

Page 에 발생하는 Event 관련하여, 'Page Create'는 Page 의 생성, 즉 최초 실행 또는 Reset 에 의해서 호출이 되며, 'Page Run'은 매번 Page 가 실행(화면에 표시)될 때마다 호출된다. (Aspen Editor 에 따라서 'Page Create'가 없는 경우도 있다.)

Context 를 알아내기 위해서는 다음과 같은 API 를 사용한다.

- Page 는 기본 실행 Context 로 간주된다. 따라서 현재 Page 를 별도로 참조하지 않아도 모든 API 는 현재 Page 를 Context 로 하여 실행된다.
- 현재 Widget 은 "getWidget();"와 같이 Parameter 없이 호출하면 Widget ID 를 가져올 수 있다.
- Event 가 Parameter 를 가지는 경우, getParam()를 호출하여 알아낼 수 있다.
- Event 가 Pointer 와 연관이 있는 경우, Pointer 의 마지막 좌표는 getPointerX(), getPointerY()를 통해서 알아낼 수 있다.

정리하면 다음과 같은 API 가 사용된다.

■ getWidget(), getParam(), getPointerX(), getPointerY()

1.1.1. Event bubbling from widgets to the page

Script source code 를 하나의 Event handler 에 집중하여 개발할 수 있는 패턴을 지원하기 위한 목적으로 Event Bubbling 을 지원한다. 즉, Widget 에서 발생한 Event 를 Page 에서 감지할 수 있도록 하는 구조이며, Event 발생 Widget 에 Handler 가 있으면 먼저 실행이 되고 그 후 Page 에 Dispatch 가 된다.

이 Event 는 Script 화면에서만 Page 에 표시가 되며 이름은 'Event Bubble'이다. Widget 에서 Bubble 되는 Event 종류는 다음과 같다.

'Tap', 'Tap Start', 'Flick Left', 'Flick Right', 'Flick Up', 'Flick Down', 'Drag Start', 'Drag End', 'Drop', 'Drop Result', 'State Change', 'Input Change', 'Widget Event'

Parameter 는 2 개가 추가되어, 3 개가 전송된다.

- getParam() | getParam(1) - Original event 의 Parameter (Event 에 따라서 없는 경우도 많음)
- getParam(2) - Original event 의 Type, 'Tap', 'Tap Start', 등...
- getParam(3) - Original event 가 발생한 Widget ID

Event Bubbling 구조를 사용할 경우에는 다음과 같은 패턴의 개발이 가능해진다.

```
//-----  
// [This Slide] > On Event Bubble  
//-----  
  
if (getParam(2) == 'Tap Start'){ // Tap Start Event 에서  
    if (wgt.getParam(3), 'label').indexOf('A;') != -1){ // Event 발생 Widget 의 Label 이 'A;'를 포함하면  
        wgt.moveBy(getParam(3), 0, -20); // 이동함  
    }  
}
```

1.2. 문서의 분석 및 조작

Script 가 문서에 대하여 부가적 기능을 수행하기 위해서는 먼저 문서의 각 Element 를 찾아서 획득하고 이것을 조작하는 방식으로 동작된다. (Web Browser 의 DOM API 도 유사한 구조임)

문서에서 획득할 수 있는 기본 Element 는 Page 와 Widget 이 있으며 이 Element 는 모두 'Label'로 참조된다. 따라서 Script 를 통해서 참조하고자 하는 Element 에는 편집 화면에서 'Label'을 정의해야 한다.

Page 와 Widget 을 참조하기 위해서는 다음 API 가 사용된다.

- `getPage()`, `getWidget()`, `getWidgets()`

1.3. Non-blocking Wait

편집 Interaction 과 마찬가지로 Non-blocking Wait 를 지원한다. 즉 Wait 가 필요한 경우, Code 의 순차적 실행이 멈추는 것이 아니고, 해당 호출만 Wait 상태로 들어가고 Code 는 계속 실행이 되는 방식이다. Aspen Editor 에서는 별도의 Wait 함수를 제공하지 않고 주요 API 에 Wait parameter 가 모두 존재한다.

UI 구성 및 사용자 동작에 의해서, 실행 대기 중인 Wait 동작을 최소화해야 하는 경우가 발생하게 되는데 이 경우 다음 API 를 사용한다.

- `cancelWait()`

2. Page handling functions

각 Script 가 Event 에 대응하여 개별 동작하는 구조이므로, Script 간 광역변수 개념이 필요하며 Page 에 저장할 수 있다. 이것은 Page 가 Reset 되기 전까지는 유효하다.

■ set(), get()

현재 Page 에서 다른 Page 로 전환하거나 현재 Page 를 Reset 하기 위해서는 다음과 같은 API 가 사용된다. (Page 를 Reset 하면, Page 를 처음 생성한 상태로 돌아가고 Create 와 동일한 Event 가 발생하게 된다.)

■ reset(), linkTo(), linkToBack(), linkToHome()

그 외 부가적 기능은 다음과 같다. 편집 Interaction 에서, Drag 결과의 통보를 위해 사용되는 'Send Drop Result' Action 과 동일한 기능을 제공하는 API 는 다음과 같다.

■ sendDropResult()

Web page 를 표시하기 위한 API 는 다음과 같다. (이것을 대체하기 위해서 IFRAME Widget 을 사용하는 것도 가능하다.)

■ linkToURL

3. Widget handling functions

Widget 을 조작하기 위해서는 Widget 의 기본 속성 또는 부가 Data 를 수정하는 방식을 가능하다. 기본 속성은 wgt.get()로 읽을 수 있으며 'text', 'textColor', 'visibility', 'x', 'y', 'w', 'h', 'angle', 'opacity', 'state', 'dragX', 'dragY', 'media', 'label', 'value', 'strokeStyle', 'fillStyle'가 지원된다.

- wgt.get()

Widget 의 속성을 수정하기 위해서는, wgt.set()와 함께 각 Property 별로 부가 API 가 존재한다. 이들 API 는 Effect, Timing 등을 추가로 제어하기 위한 목적이다.

- wgt.set()
- wgt.moveTo(), wgt.moveBy(), wgt.sizeTo(), wgt.sizeBy(), wgt.rotateTo(), wgt.opacityTo(), wgt.changeState()
- wgt.lineTo()
- wgt.zoomTo() (이 API 는 Widget 속성을 바꾸는 API 는 아니지만 같은 동작 방식을 가진다.)

개별 Widget 이 부가적으로 Data 를 가질 수 있으며, 이 Data 를 조작하여 해당 Widget 의 동작을 제어할 수 있게 된다. (Widget 이 지원하는 Data 가 있다면 해당 Widget 의 규격을 참조해야 한다.)

- wgt.setData(), wgt.getData()

Script code 는 각 Page, Widget 에 대하여 작성이 되므로 반복되는 다수의 Widget 에 대한 Scripting 편의를 위해서 다음과 같은 API 를 지원한다.

- wgt.clone() - 문서에 있는 Widget 을 복제하여 생성함
- cloneScriptOf() - 문서에 있는 다른 Widget 의 Code 를 그대로 실행함

4. Development support functions

개발지원을 위해서 다음 API 를 지원한다.

- `console.log()` - HTML5 `console.log()`와 동일. 브라우저 Console 로 출력됨 (`console.error()` 등도 사용 가능함)
- `consoleLog()` - Aspen Editor 의 Preview Mode 의 Console 로 출력됨

API References

API 의 효과적인 기술을 위하여, 다음과 같은 Primitive value 를 미리 정의하고 기술한다.

- **WidgetID** - Widget 을 참조하기 위한 ID 이며 `getWidget()`, `getParam()` 등에 의해서 Return 됨

`wgt.set()` 등과 같이 이 Parameter 를 사용하는 'wgt' Domain 의 함수에 대해서 이 값을 null, undefined 를 주면 현재 Event 를 처리 중인 Widget 으로 자동 인식한다.

- **PageID** - Page 를 참조하기 위한 ID 이며 `getPage()`에 의해서 Return 됨

- **Effect** - Transition Effect 이며 다음과 같은 값을 가질 수 있음

'cardUpLeft', 'cardUpRight', 'cardUpBottom', 'cardUpTop',
'dissolve', 'zoomIn', 'zoomOut', 'fall', 'newspaper',
'moveLeft', 'moveRight', 'moveBottom', 'moveTop',
'slideUpLeft', 'slideUpRight', 'slideUpBottom', 'slideUpTop',
'glueLeft', 'glueRight', 'glueBottom', 'glueTop',
'timeLagLeft', 'timeLagRight', 'timeLagBottom', 'timeLagTop',
'cubeLeft', 'cubeRight', 'cubeBottom', 'cubeTop',
'flipLeft', 'flipRight', 'flipBottom', 'flipTop'

- **Timing** - Effect 나 Motion 에 적용될 Timing 을 정의하기 위한 것으로, Timing 함수, Duration 및 Roundtrip 속성을 조합하여 정의한다. 예를 들면, "ease 500ms roundTrip", "linear 700ms" 등과 같다. Timing 함수는 편집기에서 보이는 항목과 같으며, Script 에서는 'none', 'linear', 'ease', 'ease-in', 'ease-out', 'ease-in-out', 'ease-out-in'로 정의된다.
- **TracePath** - Move 와 같은 동작에서 이동 경로를 변형하기 위한 기능이다. 현재 편집기는 'arcCW'(시계 방향 호), 'arcCCW'(반시계 방향 호)를 지원하고 있다. 이들은 이동 거리를 호의 지름으로 하며 Ratio 를 지정할 수 있도록 되어 있다. 예를 들면, Script 에서는 다음과 같이 지정할 수 있다. "arcCW 2.5", "arcCCW 0.2"

- **Wait** - Wait 시간을 나타내며, 기본적으로 Millisecond 단위의 Number 이다. 그러나 cancelWait()을 통하여 제어하고자 할 경우에는 'Number;group'형태를 사용한다. 예를 들면, "1000;group1"과 같은 형태이다.

Function parameter 를 기술하는 부분에서, undefined 로 표시가 된 경우는 Optional 이라는 뜻이다. 참고로 JavaScript 에서는, 함수 호출에서 있어서 후순위에 있는 Parameter 들이 undefined 인 경우에는 해당 Parameter 를 제외하고 호출해도 된다.

Aspen Editor 에서 지원하는 API 는 다음과 같다.

■ getParam()

현재 실행 Context 의 Event 에 전달된 Parameter 가 존재하면 Return 한다. (Event 에는 최대 1 개의 Parameter 를 가질 수 있다.)

[Return] String | WidgetID | "

■ getPage(/*String*/label)

'label'에 해당하는 Page 을 찾아 ID 를 Return 한다.

[Return] PageID | "

■ getWidget(/*undefined|String*/label, /*undefined|WidgetID*/parentWgtID)

'label'이 없는 경우, 현재 Context 의 Widget ID 를 Return 한다. 있는 경우, 해당 Label 을 가진 Widget 을 찾아서 Return 한다.

parentWgtID - 이 Widget 에 포함된 Widget 에 대해서만 검색한다.

[Return] WidgetID | "

- `getWidgets(/*undefined|String*/label, /*undefined|WidgetID*/parentWgtID, /*Boolean|undefined*/like)`

`getWidget()`과 동일하지만, 다수의 Widget 을 검색한다. 다수가 있을 경우에는 'WidgetID,WidgetID,...' 형태로 Return 된다. (즉, `String.split(',')`를 사용하여 바로 Array 로 변환할 수 있는 형태이다.)

like - 'LIKE'방식으로 검색함. 즉 'label' 값을 포함하는 Label 을 가진 Widget 들을 모두 검색함

[Return] WidgetID | 'WidgetID,WidgetID,...' | ''

- `set(/*String*/key, /*String|Number|Boolean*/value, /*Wait|undefined*/wait)`

현재 Page 에 'key'에 해당하는 값을 저장한다. (일종의 Page domain 변수, 또는 Global 변수와 같은 개념으로 사용이 가능하다.)

- `get(/*String*/key)`

현재 Page 에서 'key'에 해당된 저장 값을 Return 한다.

[Return] String | Number | Boolean | ''

- `getPointerX(), getPointerY()`

현재 Event 처리 시점에서의 Pointer 의 X, Y 좌표를 각각 Return 한다.

[Return] Number

- `reset(/*Wait|undefined*/wait)`

현재 Page 의 초기화하고 다시 시작한다. 이때 `set()`에 의해서 지정된 값도 모두 삭제된다.

Page 가 Reset 이 되면 최초 생성과 동일한 Process 가 진행된다. 따라서 'Page Create'(지원되는 Aspen 경우), 'Page Run' Event 가 차례로 발생하여 지정된 초기화 동작이 그대로 실행된다. (이 기능은 Game 후 Reset 과 같은 목적으로 많이 사용된다.) `set()`으로 지정된 값도 모두 Reset 되므로 Global 하게 값을 유지하고 싶은 경우에는 `file.set()`을 사용한다.

- `linkTo(/*PageID*/pageId, /*Effect|undefined*/effect, /*Timing|undefined*/timing, /*Boolean|undefined*/reset, /*Wait|undefined*/wait)`

'pageId'에 해당하는 Page 로 전환한다. 이때 'effect'와 'timing'에 해당하는 Transition Effect 로 전환하게 된다. 'reset'이 true 이며 'pageId'의 Page 가 Reset 된다. 'wait'가 있으면 그 시간만큼 대기한 후에 기능이 실행된다. (Wait 동작은 모든 API 에서 동일하다.)

- `linkToBack(/*Effect|undefined*/effect, /*Timing|undefined*/timing, /*Boolean|undefined*/reset, /*Wait|undefined*/wait)`

`linkTo()`와 동일하지만 실행 경로상 이전 Page 로 전환한다. 단, 책(Book) 형태의 실행기는 Page 순서상 이전 Page 로 간다.

- `linkToHome(/*Effect|undefined*/effect, /*Timing|undefined*/timing, /*Boolean|undefined*/reset, /*Wait|undefined*/wait)`

`linkTo()`와 동일하지만 Flow 상 시작 Page 로 전환한다. 단, 책(Book) 형태의 실행기는 Page 순서상 첫번째 Page 로 간다.

- `linkToURL(/*TargetFrame*/target, /*String*/url, /*Wait|undefined*/wait)`

'target'에 해당하는 Frame 에 URL 의 사이트를 Open 한다.

target - 'newTab'|'currentFrame'|'currentTab'

- `sendDropResult(/*Boolean*/accepted, /*Wait|undefined*/wait)`

Drop Event 에서 사용되며 Drop 결과를 송출하기 위한 것이다. 이 함수가 호출되면 현재 Drop 된 Widget 에 Drop Result Event 가 발생하게 된다.

■ `cancelWait(/*String*/group)`

Wait 상태에 있는 함수 호출의 실행을 취소하기 위한 것이다. 'group'은 해당 함수를 호출할 때 Wait parameter 에 개발자가 할당한 값이며 같은 값을 가진 모든 호출이 취소된다.

■ `cloneScriptOf(/*WidgetID*/wgtId)`

현재 실행 Context 의 Event 에 대하여, 'wgtId'에 해당하는 Widget 의 기능을 그대로 실행한다. 문서에 동일한 기능을 가진 Widget 이 다수 존재할 경우, 하나의 Widget 에만 Code 를 작성하고 나머지 Widget 에서는 그대로 호출하기 위한 목적이다.

■ `wgt.clone(/*WidgetID*/wgtId, /*WidgetID|undefined*/parentId, /*String|undefined*/parentLayerTitle)`

'wgtId'에 해당하는 Widget 을 그대로 복제하여 새로운 Widget 을 생성한다. (Scene Container 기반의 Widget 에 대한 복제는 지원되지 않는다.)

parentId - 복제된 Widget 을 Container 에 넣고자 할 경우, Parent 가 될 Container WidgetID

parentLayerTitle - 'parentId'에 해당하는 Container 에서 복제된 Widget 을 넣을 Layer 의 이름. 이 값이 없으면 첫 번째 Layer 에 추가됨

[Return] WidgetID | "

Widget 은 Container 안에 있을 경우에도 항상 Page 전체 좌표를 기준으로 절대좌표를 유지함. 따라서 Container 영역 밖에 있는 Widget 을 복제하여 Container 안에 넣을 경우, Container 의 좌표를 기준으로 Widget 을 Container 안으로 Move 해야 안에서 표시가 됨.

■ `wgt.get(/*WidgetID*/wgtId, /*String*/key)`

Widget 의 기본 속성을 Return 한다. Widget 이 해당 속성을 지원하지 않는 경우에는 오류가 발생한다.

text	String	Text
textColor	Color	Text 색상
strokeStyle fillStyle	Color	테두리 색상, 배경 색상
visibility		'visible' - 보이는 상태

		'visibleEventPass' - 보이지만 Event 를 Pass 하는 상태 'visibleEventBlock' - 보이지만 Event 를 Block 하는 상태 'hidden' - 보이지 않는 상태
x, y, w, h	Number	X 좌표, Y 좌표, 넓이, 높이 (Line widget 은 'w', 'h'를 지원하지 않음)
angle	Number	각도(Degree)
opacity	Number	불투명도, 0~1, 0=투명, 1=불투명
state		Widget 이 지원하는 State
dragX, dragY	Boolean	X, Y 축으로 Drag 가능 여부
media		Image, Audio 등 Asset 의 ID (Media 를 가지는 Widget 의 Data 를 통해서 획득할 수 있음)
label	String	Widget 의 Label. Script 는 이 값으로 Widget 을 참조함
value	String JSON	Input 또는 Data 형 Widget 등이 가지고 있는 값. 값이 Object 형태의 경우 JSON 으로 Return 됨

[Return] String | Number | Boolean | JSON

- `wgt.set(/*WidgetID*/wgtId, /*String*/key, /*String|Number|Boolean*/value, /*Wait|undefined*/wait)`

Widget 의 기본 속성을 수정한다. 각 속성에 해당하는 동작이 발생하게 된다. Timing 제어 등 부가적 제어를 위하여 다음 API 들을 부가적으로 사용할 수 있다.

`wgt.moveTo(wgtId, x, y, wait, /*Timing|undefined*/timing, /*TracePath|undefined*/tracePath)`

`wgt.moveBy(wgtId, deltaX, deltaY, wait, /*Timing|undefined*/timing, /*TracePath|undefined*/tracePath)`

`wgt.sizeTo (wgtId, w, h, wait, /*Timing|undefined*/timing)`

`wgt.sizeBy(wgtId, deltaW, deltaH, wait, /*Timing|undefined*/timing)`

`wgt.rotateTo(wgtId, deg, wait, /*Timing|undefined*/timing)`

`wgt.opacityTo(wgtId, opacity, wait, /*Timing|undefined*/timing)`

`wgt.changeState(wgtId, /*String*/state, /*Effect|undefined*/effect, /*Timing|undefined*/timing)`

(Layer/Scene Container 가 지원하는 Pseudo-state 를 사용하기 위해서는 'state'에 다음 값을 사용한다.
'state.forward', 'state.backward', 'state.forwardRoundUp', 'state.barkwardRoundUp')

- `wgt.zoomTo(/*WidgetID*/wgtId, /*String*/w, /*String*/h, wait, /*Timing|undefined*/timing)`

'w', 'h'에 해당하는 비율로 Widget 을 확대/축소한다. 이것은 Widget 의 Size 속성 자체를 바꾸지 않고 시각적으로만 확대/축소를 한다.

w, h - Percent String. 예들 들면, '55%', '200%'

- `wgt.lineTo(/*WidgetID*/wgtId, sx, sy, ex, ey, /*Wait|undefined*/wait, /*Timing|undefined*/timing)`

Line Widget 에만 적용되며, Line 의 2 개점, (sx,sy), (ex,ey)를 변경하여 Line 을 움직인다

- `wgt.setData(/*WidgetID*/wgtId, /*String*/key, /*String|Number|Boolean*/value, /*Wait|undefined*/wait)`

Widget 의 내부 Data 를 수정한다. 통상적으로 Widget 은 Data 가 수정되면 연관 동작을 수행하게 된다.

Widget 의 지원하는 Data 는 각 Widget 의 사용 규격을 참조해야 한다. (이 API 는 Widget 에 임의의 변수를 저장하기 위한 목적은 아님)

- `wgt.getData(/*WidgetID*/wgtId, /*String*/key)`

Widget 의 내부 Data 를 읽어온다. Object 형태의 복잡한 Data 의 경우, Widget 이 JSON 으로 Return 한다.

[Return] String | Number | Boolean | JSON | "

- `console.log(...)`

HTML5 `console.log()`와 동일한 함수로 Browser 의 Console 에 Log 를 출력한다. Browser 의 Console 을 사용하므로 `console.error()` 등도 사용 가능하다.

- `consoleLog(/*String*/log)`

Aspen Editor 의 'Preview Mode'에 있는 Console 에 Log 를 출력한다.