피처	설명	예제

네임스페이스	모델 요소를 언급할 때는 접두사를 붙여야 합니다.
	모델 요소에 따른 접두사는 다음과 같습니다.
	dp: 데이터풀 항목、ev: 이벤트、v: 로컬 변수、f: 함수
		dp:x = 100; // **데이터풀 항목 설정**
		fire ev:back(); // **이벤트 발생**
		f:trace_string("hello world"); // **함수 호출**

데이터풀 항목에 액세스	지정의 왼쪽에 배치하여 데이터풀 항목을 씁니다. 식의 모든 곳에서 사용하여 데이터풀 항목을 읽습니다. 리디렉션 참조(=>)는 데이터풀 항목 지정의 특수 형식입니다.	dp:x = 5; // x**에 쓰기**
		dp:x = dp:y + dp:z; // y**와 z 읽기**
		length dp:aList; // **목록 데이터풀 항목의 길이 읽기**
		dp:refX => dp:x; // **리디렉션**

이벤트 전송	**구문:**
	fire ev:<identifier>**(**<parameter-list>**);**
	
	이벤트가 시간 초과 후에 전송될 수 있습니다.
	cancel_fire 식으로 연기된 이벤트를 취소할 수 있습니다.
	
	구문:
	fire_delayed <timeout>**, ev:**<identifier>**(**<parameter-list>**);**
	cancel_fire ev:<identifier>**;**
		fire ev:back();
		fire ev:mouseClick(10, 20);
		
		fire_delayed 3000, ev:back(); // **이벤트** "back" **을** 3 **초 이내에 전송**.
		cancel_fire ev:back; // **이벤트 취소**

이벤트에 반응	**match_event 를 사용하여 이벤트에 반응합니다.**
	match_event 는 if-then-else 문의 특수 형식입니다. If 와 else 분기는 항상 같은 유형이어야 합니다. 지정의 오른쪽에 사용하면 else 분기가 필수입니다.
	
	구문:
	match_event v:<identifier> **= ev:**<identifier>
	in <sequence>
	else <sequence>
		match_event v:event = ev:back in {
		f:trace_string("back event received");
		}
		
		v:this.x = match_event v:event = ev:back in 10 else 0;

이벤트 매개변수에 액세스	**match_event 의 in 식으로 이벤트 매개변수에 액세스할 수 있습니다. 점 표기법을 사용하여 이벤트 매개변수에 액세스합니다.**	match_event v:event = ev:mouseClick in {
		v:this.x = v:event.x;
		v:this.y = v:event.y;
		}

위젯 속성에 액세스	**스크립트는 위젯 작동 명령에 반응하는 입력 처리를 기술하는 위젯의 요소이며, 스크립트로 해당 위젯의 속성에 액세스할 수 있습니다. 현재 위젯을 언급할 때** **v:this 와 같은 특수 로컬 변수를 사용할 수 있습니다. 점 표기법을 사용하여 위젯 속성의 주소를 지정합니다**	v:this.text = "hello world"; v:this.x = 10;,
위젯 트리 이동	**위젯의 요소인 스크립트로 다른 위젯의 로컬 속성에 액세스합니다. 위젯 트리 내비게이션 연산자(->)를 사용합니다.** **식별자(^)를 사용하여 상위 위젯에 액세스합니다.**	v:this->^->caption.text = "Play"; // **상위로 이동, 캡션으로 이동, 속성** text v:this->^.x-= 1; // **상위로 이동, 속성** x
문자열 서식 지정	**+ 연산자가 문자열을 연결합니다. 문자열 변환 함수에 대한 자세한 정보는 문서를 참조하십시오.**	v:this.text = "current speed: " + f:int2string(dp:speed) + "km/h";

피처	설명	예제
상수	문자열 상수는 따옴표를 사용하지 않고 쓸 수 있습니다. 색 상수는 RGBA 4 중 문자입니다.	"hello world" // 문자열 상수 Napoleon // 문자열 상수 5 // 정수 상수 color:0,235,0,255 // EB 초록
산술, 논리 및 지정 연산자	덧셈 및 문자열 연결: +, 뺄셈: -, 곱셈: *, 나눗셈: /, 모듈로: %、큼: >, 작음: <, 크거나 같음: >=, 작거나 같음: <=, 같음: ==, 같지 않음: !=, 그리고: &&, 또는: ||, 부정: !, 지정: =, 지정 증가: +=, 지정 감소: -=	dp:myString = "Hello" + "World"; dp:count += 1; // 증가 1

시퀀싱	시퀀스는 단일 식이거나 중괄호에 묶인 일련의 식입니다. 시퀀스의 마지막 식이 시퀀스 값입니다.	if(dp:something)
		  dp:x = 5; // 단일 식
		if(dp:other) {
		  dp:x = 5; // 중괄호에
		  dp:y = 10; // 묶인 시퀀스
		}

로컬 변수	let 바인딩을 사용하여 로컬 변수를 도입합니다. 초기화되지 않은 변수는 사용할 수 없습니다.
	let 바인딩은 중첩될 수도 있습니다.
	
	구문:
	let v:<identifier> = <expression>**;**
	  v:<identifier2> = <expression>**;**
	  ...
	  in <sequence>
		let v:x = 42;
		  v:text = "hello world";
		in {
		  v:this.x = v:x;
		  v:this.text = v:text;
		}

WHILE 루프	while 루프는 조건 식과 본문 식으로 구성됩니다. 본문 식은 조건 식이 'false'가 될 때까지 반복적으로 평가됩니다.
	
	구문:
	while(<expression> **)** <sequence>
		dp:i = 0;
		while(dp:i <= 10) {
		  dp:sum += i;
		  dp:i += 1;
		}

IF-THEN-ELSE	If-then-else 는 C와 Java의 3 진 조건부 연산자와 비슷하게 작동합니다. 이 조건문을 지정의 오른쪽에 사용하는 경우에는 else 분기가 필수이며 두 분기 모두 같은 유형이어야 합니다.
	
	구문:
	if(<expression> **)** <sequence> **else** <sequence>
		if(dp:buttonClicked) {
		  v:this.x = dp:x;
		}
		else {
		  v:this.x = 0;
		}
		
		v:this.x = if(dp:buttonClicked) dp:x else 0;

주석	C 스타일 블록 주석과 C++ 스타일 줄 주석을 사용합니다.	/* C 스타일 블록 주석임 */ // C++ 스타일 줄 주석임

반환 값	스크립트의 마지막 식이 반환 값입니다.
	void 유형의 반환 값을 도출하려면 unit 또는 {}를 사용합니다.
		dp:x + 2; // 데이터풀 항목 x 더하기 2
		반환