Examples

This page contains longer scripts that you can copy into a Script block and adapt. The examples that do not use page components can be pasted as they are; the result appears on the block's **output** pin.

Lat-Long Operations

Functions for distance (in km, haversine formula), initial bearing (in degrees from north), and speed (km/h) between two latitude/longitude points. Math functions use radians, so degrees are converted first.

```
function toRad(num) { return num * Math.PI / 180; }
function toDeg(num) { return num * 180 / Math.PI; }

function calcDistance(lat1, lon1, lat2, lon2) {
  var R = 6371; // radius of the earth in km
  var dLat = toRad(lat2 - lat1);
  var dLon = toRad(lon2 - lon1);
  var a = 0.5 - Math.cos(dLat) / 2 +
    Math.cos(toRad(lat1)) * Math.cos(toRad(lat2)) * (1 - Math.cos(dLon)) / 2;
  return R * 2 * Math.asin(Math.sqrt(a));
}

function calcBearing(lat1, lon1, lat2, lon2) {
  var p1 = toRad(lat1);
  var p2 = toRad(lat2);
  var dLon = toRad(lon2 - lon1);
  var y = Math.sin(dLon) * Math.cos(p2);
  var x = Math.cos(p1) * Math.sin(p2) - Math.sin(p1) * Math.cos(p2) *
Math.cos(dLon);
  return (toDeg(Math.atan2(y, x)) + 360) % 360;
}

function calcSpeed(lat1, lon1, lat2, lon2, time1, time2) {
  var d = calcDistance(lat1, lon1, lat2, lon2);
  var ms = dateParse(time2) - dateParse(time1);
  return d / (ms / 1000 / 60 / 60);
}

// Warsaw to Rome
print(Math.round(calcBearing(52.2297, 21.0122, 41.8919, 12.5113)));
print(" ");
print(Math.round(calcSpeed(52.2297, 21.0122, 41.8919, 12.5113,
  "2026-01-01T10:00:00", "2026-01-01T12:00:00")));
```

```
return Math.round(calcDistance(52.2297, 21.0122, 41.8919, 12.5113));
```

Print: 212 658, Output: 1316

Custom Easing Functions

Easing functions map time to a value. Parameters: t current time, b start value, c change in value, d duration, s optional overshoot for the "Back" functions.

```
var easing = {
  swing: function (t, b, c, d) {
    return easing.easeOutQuad(t, b, c, d);
  },
  easeInQuad: function (t, b, c, d) {
    return c*(t/=d)*t + b;
  },
  easeOutQuad: function (t, b, c, d) {
    return -c*(t/=d)*(t-2) + b;
  },
  easeInOutQuad: function (t, b, c, d) {
    if ((t/=d/2) < 1) return c/2*t*t + b;
    return -c/2 * ((--t)*(t-2) - 1) + b;
  },
  easeInCubic: function (t, b, c, d) {
    return c*(t/=d)*t*t + b;
  },
  easeOutCubic: function (t, b, c, d) {
    return c*((t=t/d-1)*t*t + 1) + b;
  },
  easeInOutCubic: function (t, b, c, d) {
    if ((t/=d/2) < 1) return c/2*t*t*t + b;
    return c/2*((t-=2)*t*t + 2) + b;
  },
  easeInQuart: function (t, b, c, d) {
    return c*(t/=d)*t*t*t + b;
  },
  easeOutQuart: function (t, b, c, d) {
    return -c * ((t=t/d-1)*t*t*t - 1) + b;
  },
  easeInOutQuart: function (t, b, c, d) {
    if ((t/=d/2) < 1) return c/2*t*t*t*t + b;
    return -c/2 * ((t-=2)*t*t*t - 2) + b;
  },
  easeInQuint: function (t, b, c, d) {
    return c*(t/=d)*t*t*t*t + b;
  },
```

```
  easeOutQuint: function (t, b, c, d) {
    return c*((t=t/d-1)*t*t*t*t + 1) + b;
  },
  easeInOutQuint: function (t, b, c, d) {
    if ((t/=d/2) < 1) return c/2*t*t*t*t*t + b;
    return c/2*((t-=2)*t*t*t*t + 2) + b;
  },
  easeInSine: function (t, b, c, d) {
    return -c * Math.cos(t/d * (Math.PI/2)) + c + b;
  },
  easeOutSine: function (t, b, c, d) {
    return c * Math.sin(t/d * (Math.PI/2)) + b;
  },
  easeInOutSine: function (t, b, c, d) {
    return -c/2 * (Math.cos(Math.PI*t/d) - 1) + b;
  },
  easeInExpo: function (t, b, c, d) {
    return (t==0) ? b : c * Math.pow(2, 10 * (t/d - 1)) + b;
  },
  easeOutExpo: function (t, b, c, d) {
    return (t==d) ? b+c : c * (-Math.pow(2, -10 * t/d) + 1) + b;
  },
  easeInOutExpo: function (t, b, c, d) {
    if (t==0) return b;
    if (t==d) return b+c;
    if ((t/=d/2) < 1) return c/2 * Math.pow(2, 10 * (t - 1)) + b;
    return c/2 * (-Math.pow(2, -10 * --t) + 2) + b;
  },
  easeInCirc: function (t, b, c, d) {
    return -c * (Math.sqrt(1 - (t/=d)*t) - 1) + b;
  },
  easeOutCirc: function (t, b, c, d) {
    return c * Math.sqrt(1 - (t=t/d-1)*t) + b;
  },
  easeInOutCirc: function (t, b, c, d) {
    if ((t/=d/2) < 1) return -c/2 * (Math.sqrt(1 - t*t) - 1) + b;
    return c/2 * (Math.sqrt(1 - (t-=2)*t) + 1) + b;
  },
  easeInBack: function (t, b, c, d, s) {
    if (s == null) s = 1.70158;
    return c*(t/=d)*t*((s+1)*t - s) + b;
  },
  easeOutBack: function (t, b, c, d, s) {
    if (s == null) s = 1.70158;
    return c*((t=t/d-1)*t*((s+1)*t + s) + 1) + b;
  },
  easeInOutBack: function (t, b, c, d, s) {
    if (s == null) s = 1.70158;
```

```
    if ((t/=d/2) < 1) return c/2*(t*t*(((s*=(1.525))+1)*t - s)) + b;
    return c/2*((t-=2)*t*(((s*=(1.525))+1)*t + s) + 2) + b;
  },
  easeInBounce: function (t, b, c, d) {
    return c - easing.easeOutBounce(d-t, 0, c, d) + b;
  },
  easeOutBounce: function (t, b, c, d) {
    if ((t/=d) < (1/2.75)) {
      return c*(7.5625*t*t) + b;
    } else if (t < (2/2.75)) {
      return c*(7.5625*(t-=(1.5/2.75))*t + .75) + b;
    } else if (t < (2.5/2.75)) {
      return c*(7.5625*(t-=(2.25/2.75))*t + .9375) + b;
    } else {
      return c*(7.5625*(t-=(2.625/2.75))*t + .984375) + b;
    }
  },
  easeInOutBounce: function (t, b, c, d) {
    if (t < d/2) return easing.easeInBounce(t*2, 0, c, d) * .5 + b;
    return easing.easeOutBounce(t*2-d, 0, c, d) * .5 + c*.5 + b;
  }
};
return easing["easeInOutQuad"](0.25, 0, 1, 1);
```

Output: `0.125`

To drive it from the block, add custom parameters (for example `name`, `t`, `from`, `to`) to the Script block and end the script with:

```
return easing[@.name](@.t, @.from, @.to - @.from, 1);
```

Summarize JSON data

Parse a JSON string, filter it, and compute a total and an average.

```
var text = '[{"name":"Pump 1","kw":12.5,"on":true},' +
           '{"name":"Pump 2","kw":0,"on":false},' +
           '{"name":"Fan","kw":3.5,"on":true}]';
var items = JSON.parse(text);
var total = 0;
var names = [];
for (var i = 0; i < items.length; i++) {
  if (items[i].on) {
    total += items[i].kw;
    names.push(items[i].name);
  }
```

```
}
print(names.join(", "));
return total / names.length;
```

Print: `Pump 1, Fan`, Output: 8

To show the result in a table, pass the rows to `tableSet`; see Tables.

Format a time range

Format two timestamps as a range and a duration. The `true` parameter of `dateFormat` formats the time in UTC so the result does not depend on the viewer's time zone. See Date and Time.

```
function pad(n) { return n < 10 ? "0" + n : "" + n; }
function formatRange(start, end) {
  var mins = Math.round((end - start) / 60000);
  return dateFormat(start, "HH:mm", true) + " - " + dateFormat(end, "HH:mm",
true) +
         " (" + Math.floor(mins / 60) + "h " + pad(mins % 60) + "m)";
}
var start = 9.5 * 3600000;   // 09:30 UTC, 1 Jan 1970
var end = 17 * 3600000 + 5 * 60000;
return formatRange(start, end);
```

Output: `09:30 - 17:05 (7h 35m)`

Call a REST API and fill a table

`fetchHttpUrl` returns a promise; the table is filled when the response arrives. Add a custom output parameter named `table` to the Script block; `tableSet(@.table, …)` writes to it. See HTTP and Utilities.

```
fetchHttpUrl("https://api.example.com/sensors").then(function(resp) {
  if (resp.status != 200) {
    print("HTTP error " + resp.status);
    return;
  }
  var list = JSON.parse(resp.data);
  var rows = [];
  for (var i = 0; i < list.length; i++) {
    rows.push([list[i].id, list[i].name, list[i].value]);
  }
  tableSet(@.table, rows, ["id", "name", "value"]);
```

```
});
```

Show one component and hide the others

Show the component named in the custom parameter `selected` and hide the other components in `@stage.@panels`. See Model Functions.

```
var group = @stage.@panels;
modelIterateChildren(group, function(child) {
  if (modelIsComponent(child)) {
    child.visible = (child.@name == @.selected);
  }
  return false; // do not go deeper
});
```

From:
http://wiki.dglogik.com/ - **DGLogik**

Permanent link:
http://wiki.dglogik.com/dglux5_wiki:dgscript:examples

Last update: **2026/10/22 16:19**