Brian’s Wolfram Language Cheat Sheet

A Wolfram Language notebook containing a compilation of fundamental, low-level syntax and functions (such as @@, @@@, /@ ./, Table, Array, Module, etc.)

All of these are covered in EIWL3, but it is relatively slow to flip through a 48-section survey instead of a focused cheat sheet.

Fundamental Functions and Syntax

These are functions and syntax that relate directly to the application of functions to symbols or lists.

Apply — Another way of Applying a Function to a List of Arguments

```
In[∘]:= Apply[f, {x, y}]
Out[∘]=
        f[x, y]
```

@ — Yet Another way to Apply a Function — Only Single Argument Functions Though

```
In[∘]:= f@x (* is completely equivalent to f[x] *)
Out[∘]=
        f[x]
```

// — Apply as an Afterthought

```
In[∘]:= Array[Plus, {10, 10}] // Grid
Out[∘]=
         2  3  4  5  6  7  8  9 10 11
         3  4  5  6  7  8  9 10 11 12
         4  5  6  7  8  9 10 11 12 13
         5  6  7  8  9 10 11 12 13 14
         6  7  8  9 10 11 12 13 14 15
         7  8  9 10 11 12 13 14 15 16
         8  9 10 11 12 13 14 15 16 17
         9 10 11 12 13 14 15 16 17 18
        10 11 12 13 14 15 16 17 18 19
        11 12 13 14 15 16 17 18 19 20
```

Map — Make a New List by Applying a Function to Each Element in a List

```
In[∘]:= Map[f, {x, y, z}]
Out[∘]=
        {f[x], f[y], f[z]}
```

Map and /@ are Not Needed for Functions that Are Already Listable

```
In[∘]:= Map[Sin, {x, y, z}]
Out[∘]=
        {Sin[x], Sin[y], Sin[z]}

In[∘]:= Sin /@ {x, y, z}
Out[∘]=
        {Sin[x], Sin[y], Sin[z]}

In[∘]:= {x, y, z} // Sin
Out[∘]=
        {Sin[x], Sin[y], Sin[z]}
```

Since Sin is listable, just use:

```
In[∘]:= Sin[{x, y, z}]
Out[∘]=
        {Sin[x], Sin[y], Sin[z]}

In[∘]:= Sin@{x, y, z}
Out[∘]=
        {Sin[x], Sin[y], Sin[z]}
```

But interestingly, even though Sin is listable, you cannot use:

```
In[∘]:= Apply[Sin, {x, y, z}]
        ••• Sin: Sin called with 3 arguments; 1 argument is expected. ⓘ
Out[∘]=
        Sin[x, y, z]
```

which means that Apply and @ are not identical.

/@ — A Shorthand for Map

```
In[∘]:= f /@ {x, y, z}
Out[∘]=
        {f[x], f[y], f[z]}
```

MapApply

```
In[•]:= MapApply[f, {{x, y}, {z}, {a, b, c}}]
Out[•]=
        {f[x, y], f[z], f[a, b, c]}
```

@@@ — A Shorthand for MapApply

```
In[•]:= f @@@ {{x, y}, {z}, {a, b, c}}
Out[•]=
        {f[x, y], f[z], f[a, b, c]}
```