tf.foldl

```
foldl(
    fn,
    elems,
    initializer=None,
    parallel_iterations=10,
    back_prop=True,
    swap_memory=False,
    name=None
)
```

Defined in `tensorflow/python/ops/functional_ops.py`.

See the guide: Higher Order Functions > Higher Order Operators

foldl on the list of tensors unpacked from `elems` on dimension 0.

This foldl operator repeatedly applies the callable `fn` to a sequence of elements from first to last. The elements are made of the tensors unpacked from `elems` on dimension 0. The callable fn takes two tensors as arguments. The first argument is the accumulated value computed from the preceding invocation of fn. If `initializer` is None, `elems` must contain at least one element, and its first element is used as the initializer.

Suppose that `elems` is unpacked into `values`, a list of tensors. The shape of the result tensor is fn(initializer, values[0]).shape`.

Args:

- `fn`: The callable to be performed.
- `elems`: A tensor to be unpacked on dimension 0.
- `initializer`: (optional) The initial value for the accumulator.
- `parallel_iterations`: (optional) The number of iterations allowed to run in parallel.
- `back_prop`: (optional) True enables support for back propagation.
- `swap_memory`: (optional) True enables GPU-CPU memory swapping.
- `name`: (optional) Name prefix for the returned tensors.

Returns:

A tensor resulting from applying `fn` consecutively to the list of tensors unpacked from `elems`, from first to last.

Raises:

- `TypeError`: if `fn` is not callable.

Example:

```
elems = [1, 2, 3, 4, 5, 6]
sum = foldl(lambda a, x: a + x, elems)
# sum == 21
```


Last updated November 2, 2017.