

Chainable Data Sources

or how I stopped worrying and
started abusing table view updates



Disclaimer

@ * [:] ;



```
UITableView *tableView;
```



```
- (UITableViewCell*) tableView:(UITableView *)tableView
    cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell* cell = [tableView
        dequeueReusableCellWithIdentifier:@"fruit-cell"
        forIndexPath:indexPath];

    Fruit* fruit = self.fruits[0];
    cell.titleLabel.text = fruit.name;
    return cell;
}
```

Animations 🥰

```
[tableView beginUpdates];  
[self.fruits removeObjectAtIndex:indexPath.row];  
[tableView deleteRowsAtIndexPaths:@[indexPath]  
    withRowAnimation:UITableViewRowAnimationAutomatic];  
[tableView endUpdates];
```



```
//Controller
- (UITableViewCell*) tableView:(UITableView *)tableView
    cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell* cell = [tableView
        dequeueReusableCellWithIdentifier:@"fruit-cell"
        forIndexPath:indexPath];
    Fruit* fruit = self.fruits[indexPath.row]; //Model
    cell.titleLabel.text = fruit.name; //View
    return cell;
}
```



```
- (UITableViewCell*) tableView:(UITableView *)tableView  
    cellForRowAtIndexPath:(NSIndexPath *)indexPath  
{  
    FruitCell* cell = (FruitCell*)[tableView  
        dequeueReusableCellWithIdentifier:@"fruit-cell"  
        forIndexPath:indexPath];  
    Fruit* fruit = self.fruits[indexPath.row];  
    cell.fruit = fruit;  
    return cell;  
}
```



```
- (UITableViewCell*) tableView:(UITableView *)tableView  
    cellForRowAtIndexPath:(NSIndexPath *)indexPath  
{  
    if (indexPath.section == FRUIT_SECTION) {  
        UITableViewCell* cell = [tableView  
            dequeueReusableCellWithIdentifier:@"fruit-cell"  
                                forIndexPath:indexPath];  
        Fruit* fruit = self.fruits[indexPath.row];  
        cell.titleLabel.text = fruit.name;  
        return cell;  
    } else if (indexPath.section == VEGETABLE_SECTION) {  
        UITableViewCell* cell = [tableView  
            dequeueReusableCellWithIdentifier:@"veg-cell"  
                                forIndexPath:indexPath];  
        ...  
    }  
}
```




```
- (UITableViewCell*) tableView:(UITableView *)tableView  
    cellForRowAtIndexPath:(NSIndexPath *)indexPath  
{  
    switch (indexPath.section) {  
        case FRUIT_SECTION: {  
            UITableViewCell* cell = [tableView  
                dequeueReusableCellWithIdentifier:@"fruit-cell"  
                    forIndexPath:indexPath];  
            Fruit* fruit = self.fruits[indexPath.row];  
            cell.titleLabel.text = fruit.name;  
            return cell;  
        } break;  
        case VEGETABLE_SECTION: {  
            ...  
        }
```



```
- (UITableViewCell*) tableView:(UITableView *)tableView
    cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    if (indexPath.section == AD_SECTION &&
        indexPath.row == AD_ROW) {
        return [tableView
            dequeueReusableCellWithIdentifier:@"ad-cell"
            forIndexPath:indexPath];
    }
    indexPath = [self offsetIndexPath:indexPath
                        ifAfter:AD_INDEX_PATH];
    switch (indexPath.section) {
        case FRUIT_SECTION: {
            UITableViewCell* cell = [tableView
                dequeueReusableCellWithIdentifier:@"fruit-cell"
                forIndexPath:indexPath];
            ...
        }
    }
}
```



```
- (UITableViewCell*) tableView:(UITableView *)tableView
    cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    if (indexPath.section == AD_SECTION &&
        indexPath.row == AD_ROW &&
        self.isAdLoaded) {
        return [tableView
            dequeueReusableCellWithIdentifier:@"ad-cell"
            forIndexPath:indexPath];
    }
    indexPath = [self offsetIndexPath:indexPath
                        ifAfter:AD_INDEX_PATH];
    switch (indexPath.section) {
        case FRUIT_SECTION: {
            UITableViewCell* cell = [tableView
                dequeueReusableCellWithIdentifier:@"fruit-cell"
                forIndexPath:indexPath];
            ...
        }
    }
}
```

Animations 🤯

```
2016-09-06 14:07:23.167 MyBeautifulTableView[89346:883532] *** Terminating app due to
uncaught exception 'NSInternalInconsistencyException', reason: 'Invalid update: invalid
number of rows in section 0. The number of rows contained in an existing section after the
update (1) must be equal to the number of rows contained in that section before the update
(9), plus or minus the number of rows inserted or deleted from that section (0 inserted, 0
deleted) and plus or minus the number of rows moved into or out of that section (0 moved
in, 0 moved out).'
```

```
*** First throw call stack:
```

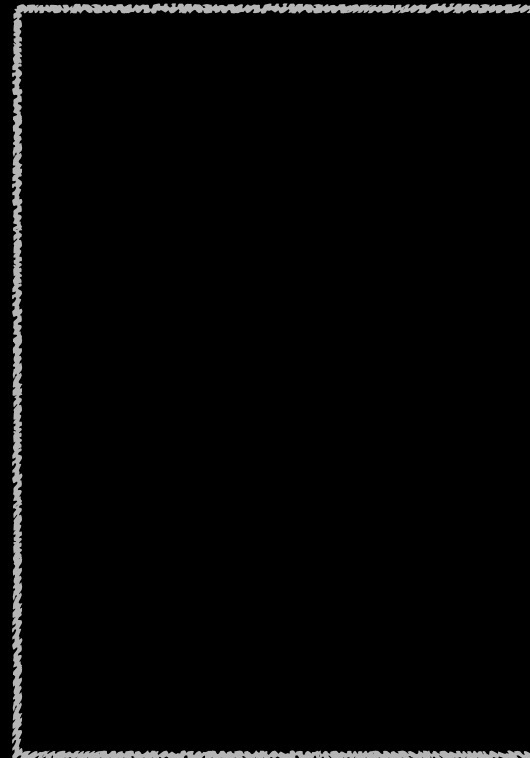
```
(
  0   CoreFoundation          0x000000010eba034b __exceptionPreprocess + 171
  1   libobjc.A.dylib          0x000000010e60121e objc_exception_throw + 48
  2   CoreFoundation          0x000000010eba4442 +[NSException
raise:format:arguments:] + 98
  3   Foundation               0x0000000109620edd -[NSAssertionHandler
handleFailureInMethod:object:file:lineNumber:description:] + 195
  4   UIKit                   0x000000010c2172f4 -[UITableView
_endCellAnimationsWithContext:] + 17558
  ...
```

```
[tableView reloadData];
```

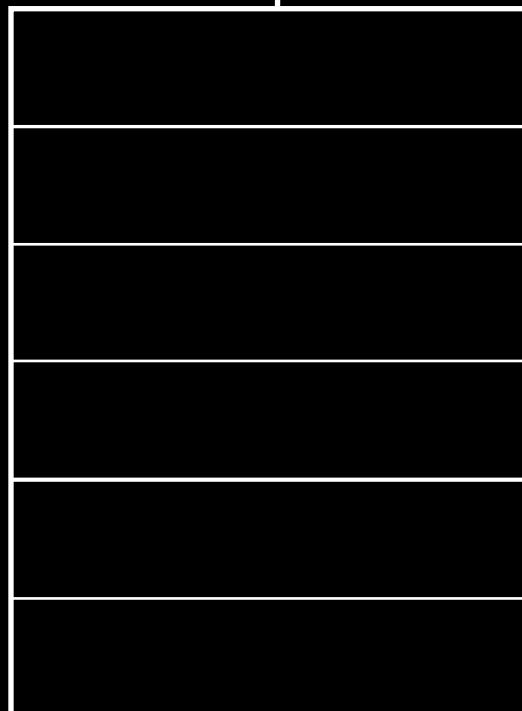


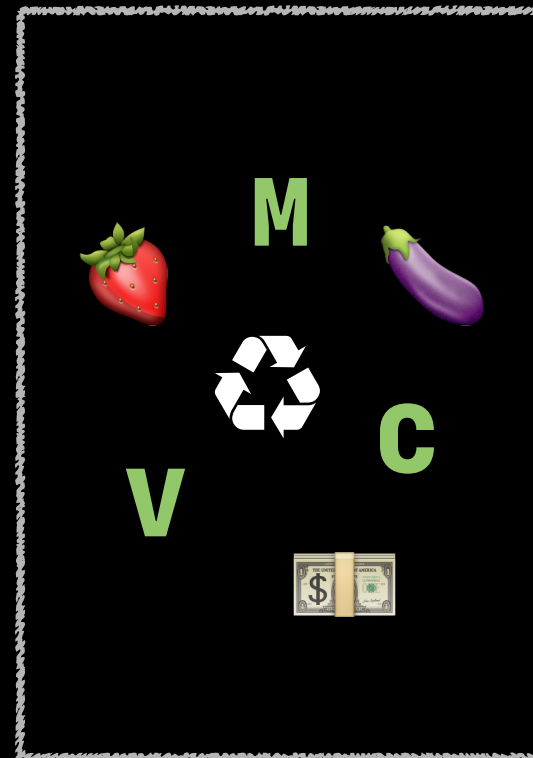
Episode IV
A NEW HOPE

*The evil forces of reloadData and
NSInternalInconsistencyException
are tightening their grip on the
universe of UIKit development.
Unbeknownst to those dark lords
of chaos, a group of rebels still
faithful to the ancient ways of
decoupling put together a secret
weapon. Using the power of
CHAINABLE DATA SOURCES, they
intend to bring back internal
consistency to the galaxy.*

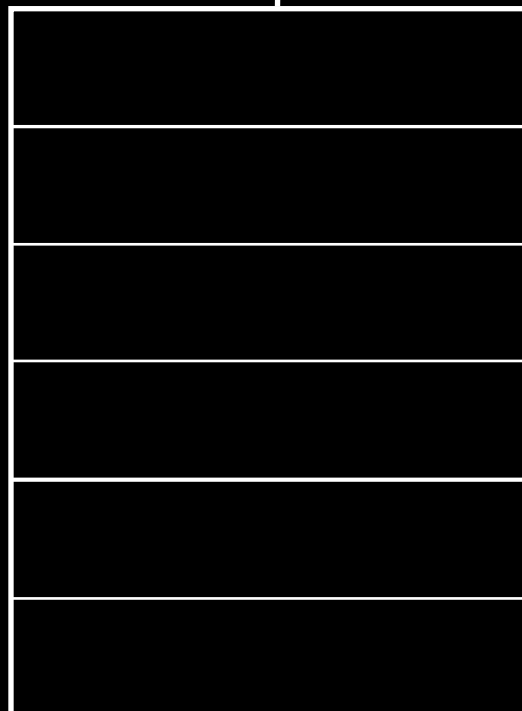


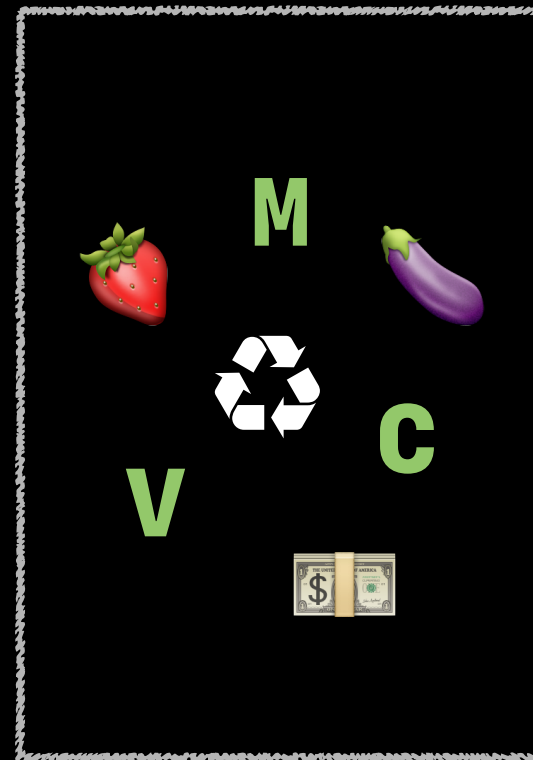
UITableViewDataSource






UITableViewDataSource

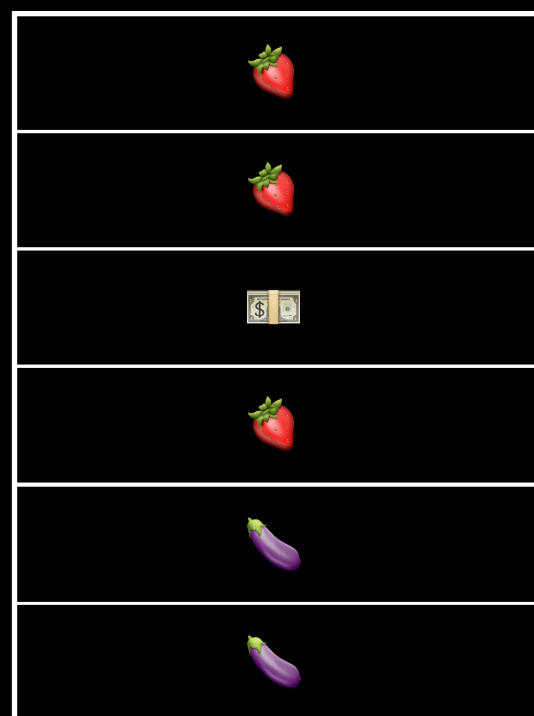




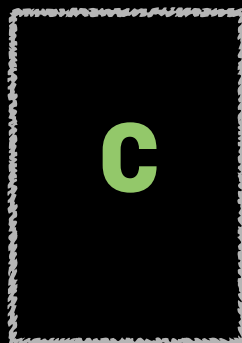
UITableViewDataSource



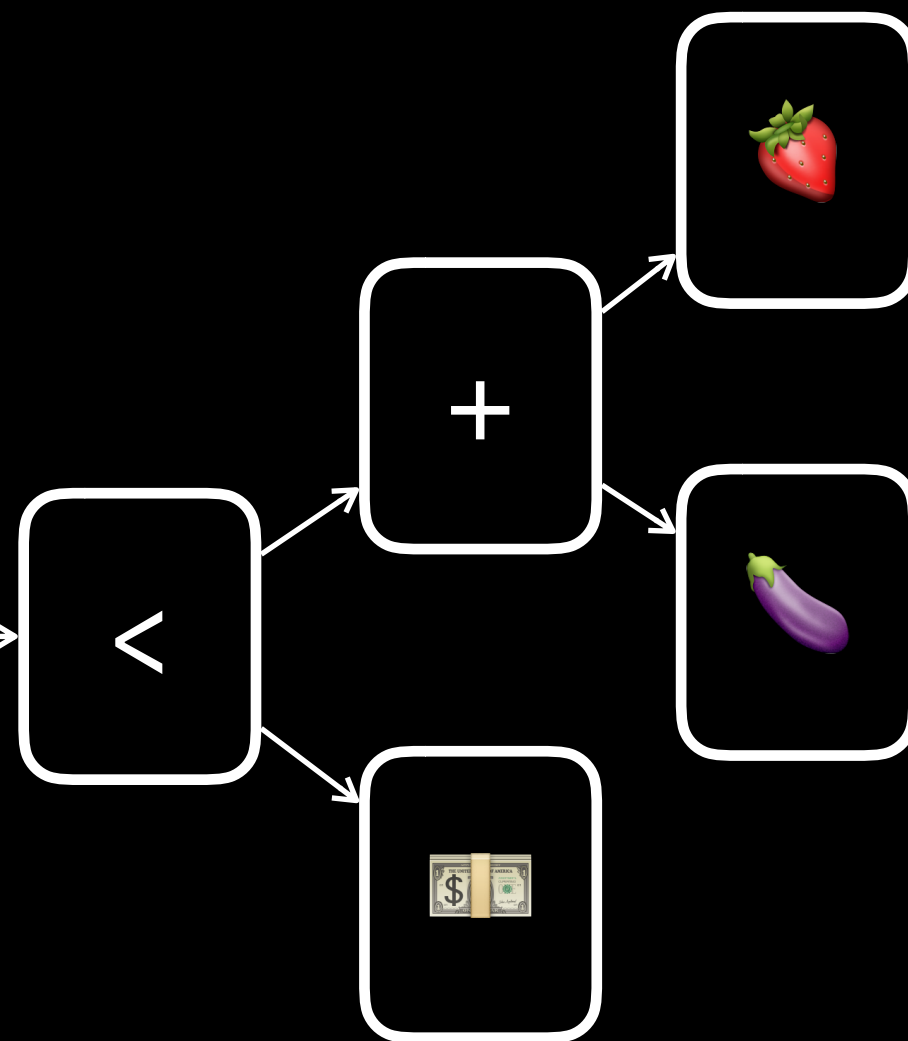
V



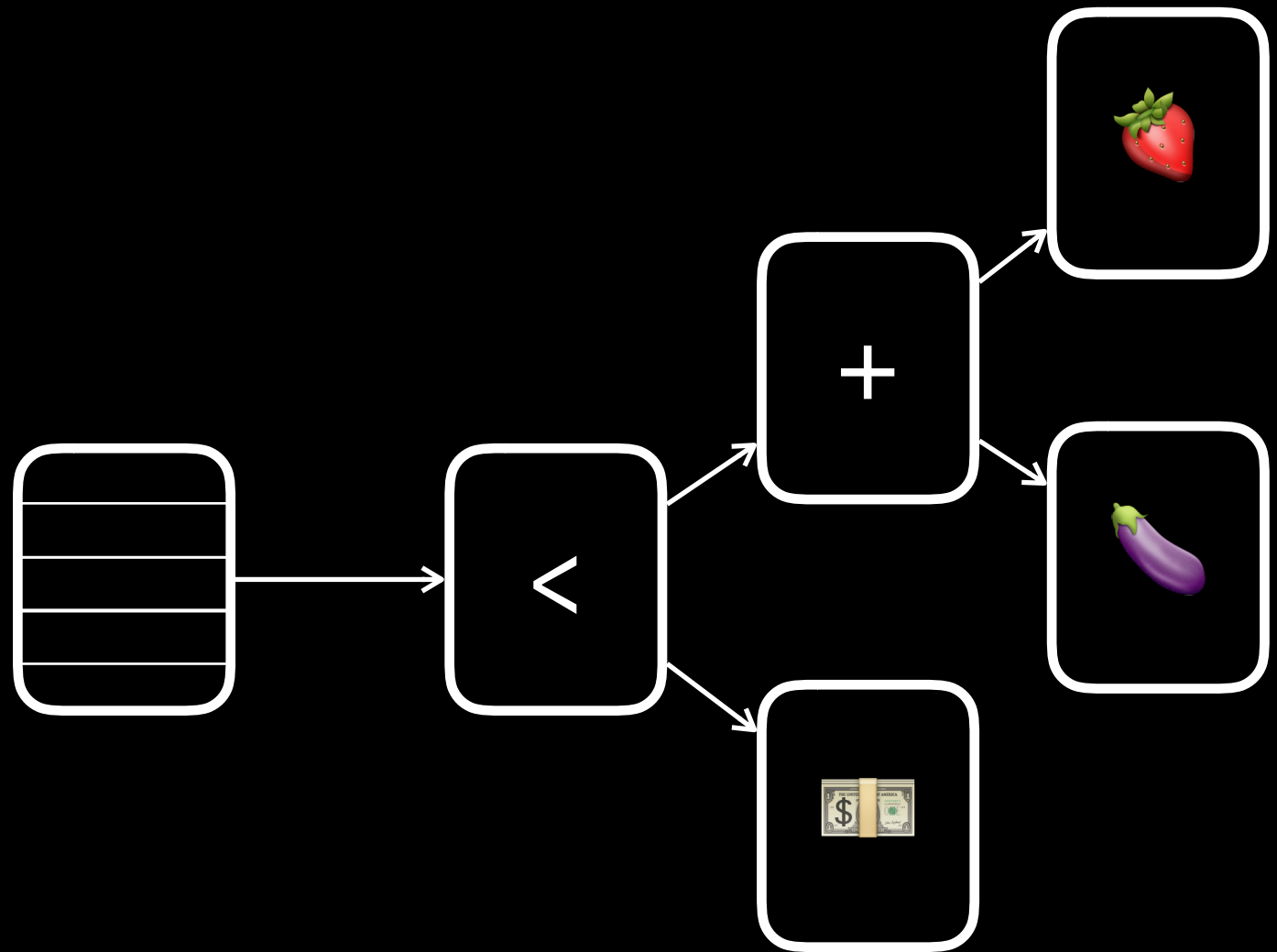
C



M



ChainableDataSource



```
@protocol CDSChainableDataSource <NSObject>

- (NSInteger) cds_numberOfSections;
- (NSInteger) cds_numberOfObjectsSection:(NSInteger)section;
- (id) cds_objectAtIndex:(NSIndexPath*)indexPath;
- (NSString*) cds_nameForSectionAtIndex:(NSInteger)section;

@property (weak) id<CDSUpdateDelegate> cds_updateDelegate;

@end
```

M

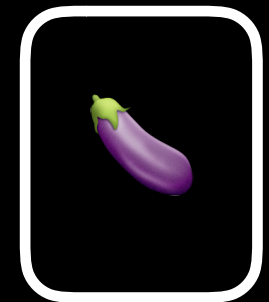
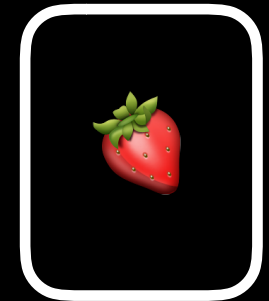


M

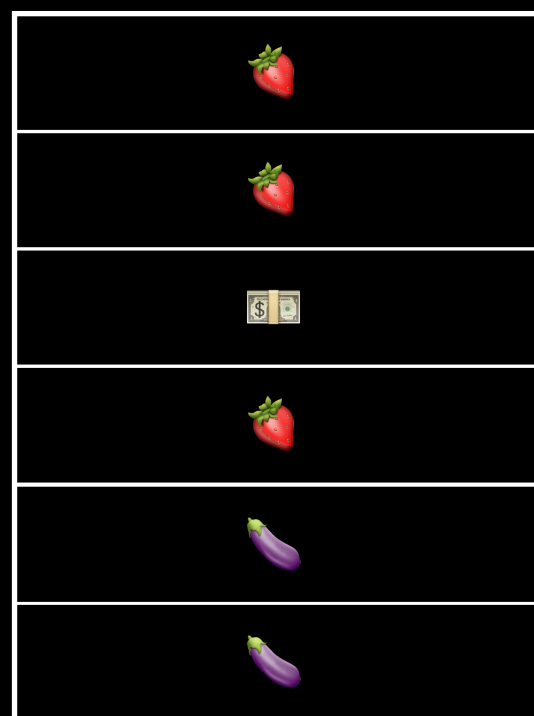
NSArray <CDSChainableDataSource>

CDSFetchWrapper

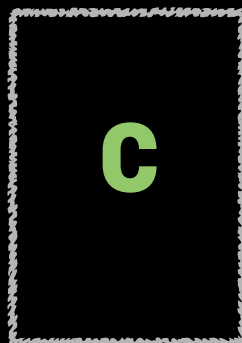
<#YourChainableDataSource#>



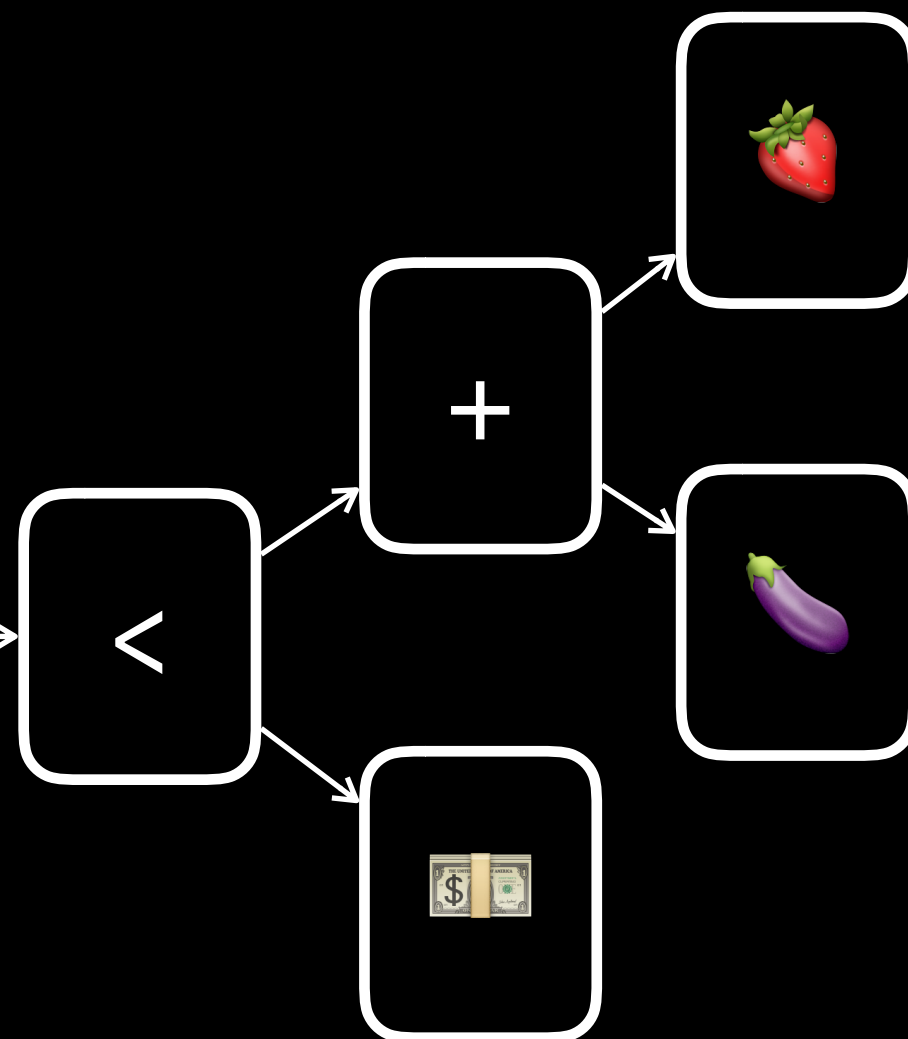
V

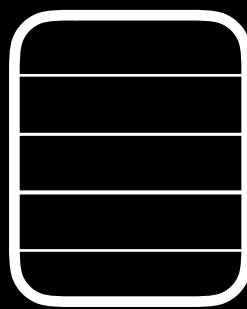


C

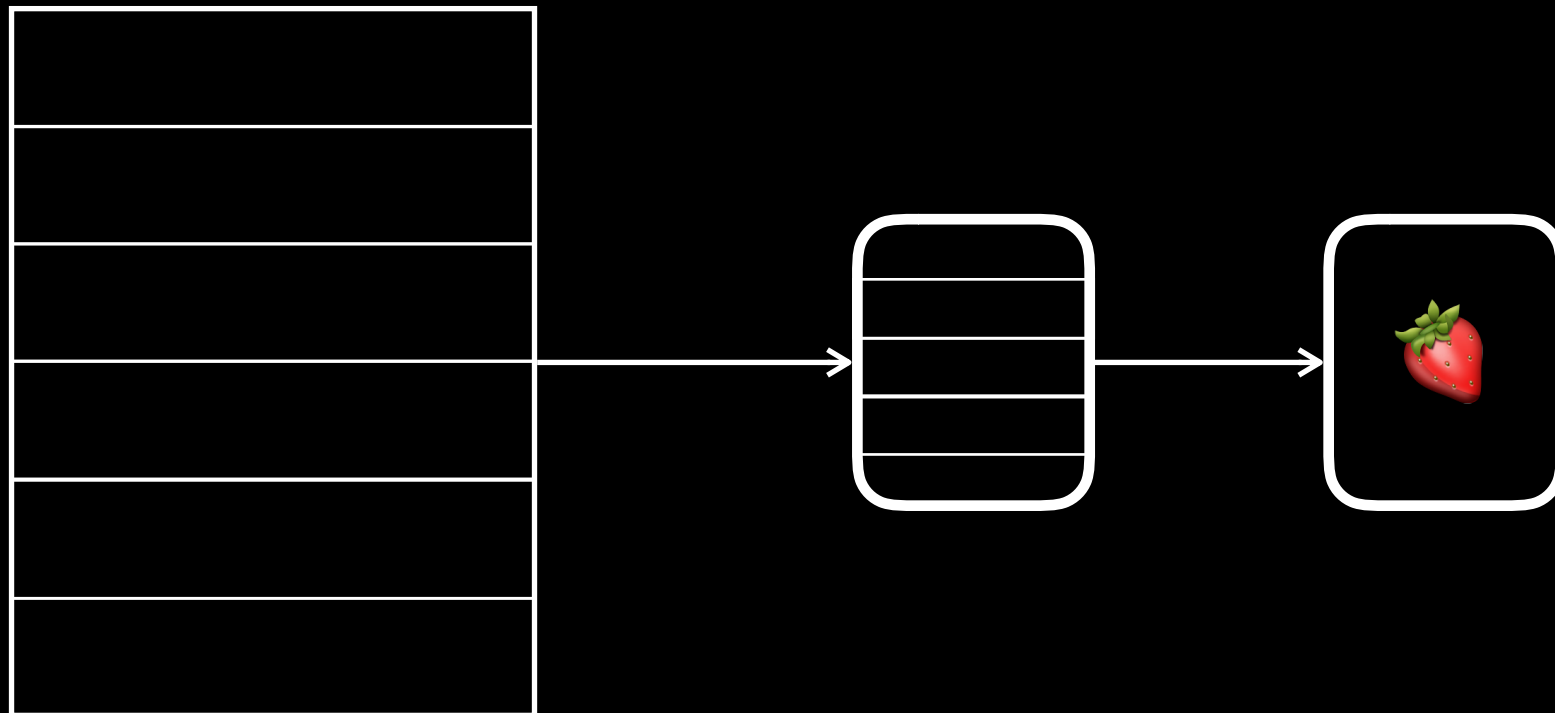


M



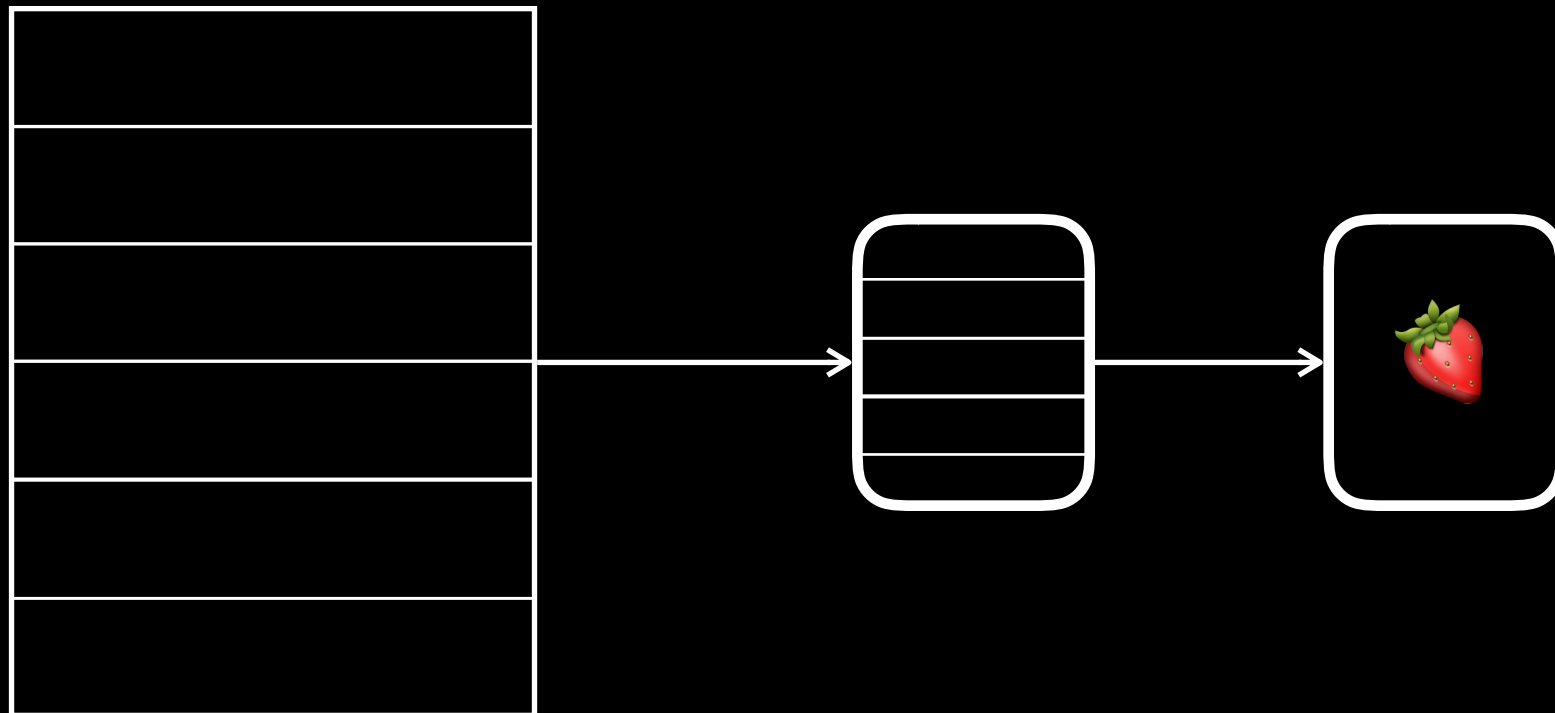


CellDataSource



CellDataSource

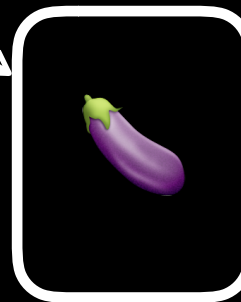
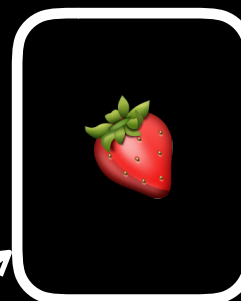
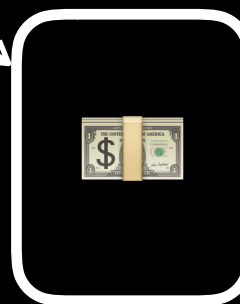
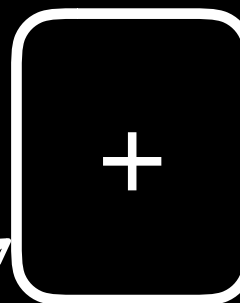
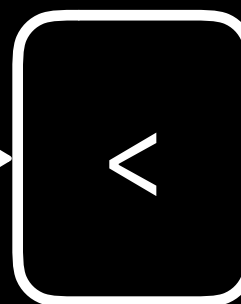
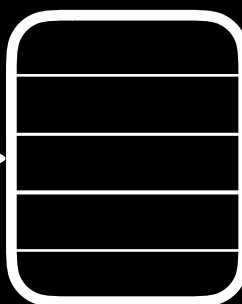
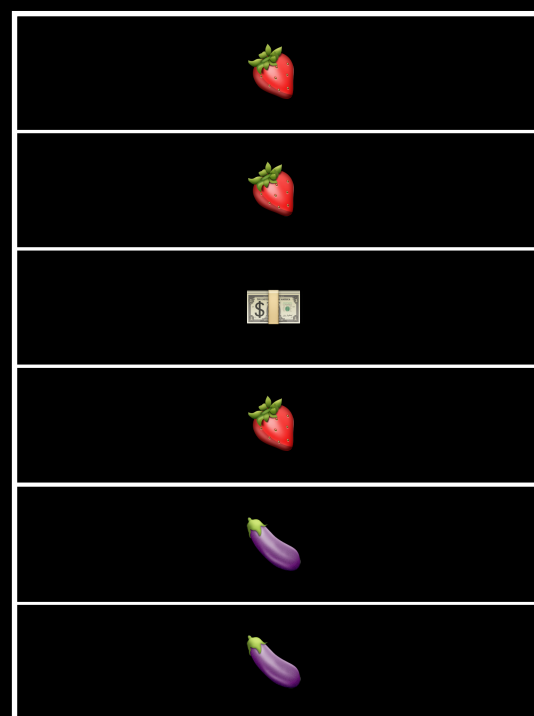
```
@interface CDSCellDataSource : NSObject
<CDSCChainableDataSource, UITableViewDataSource>
- (NSString*) cellIdentifierForObject:(id)object;
- (void) configureCell:(UIView<GenericCell>*)cell withObject:(id)object;
```

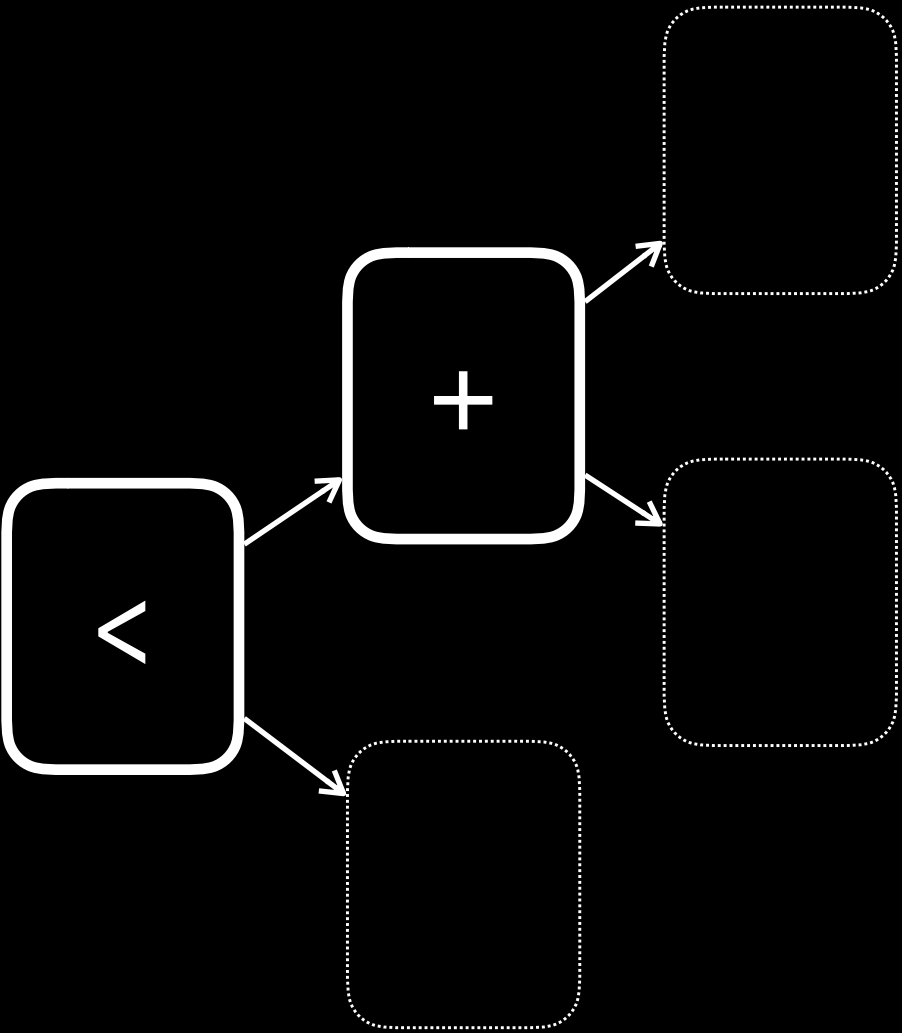


V

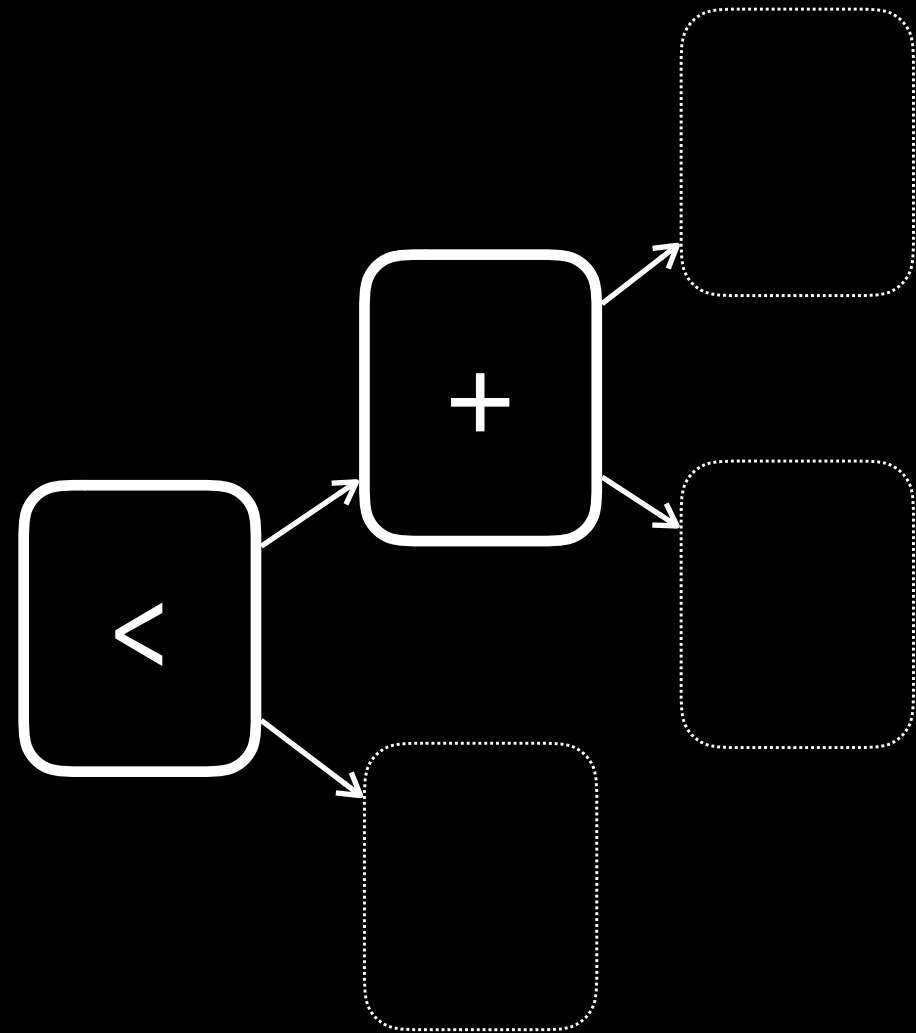
C

M

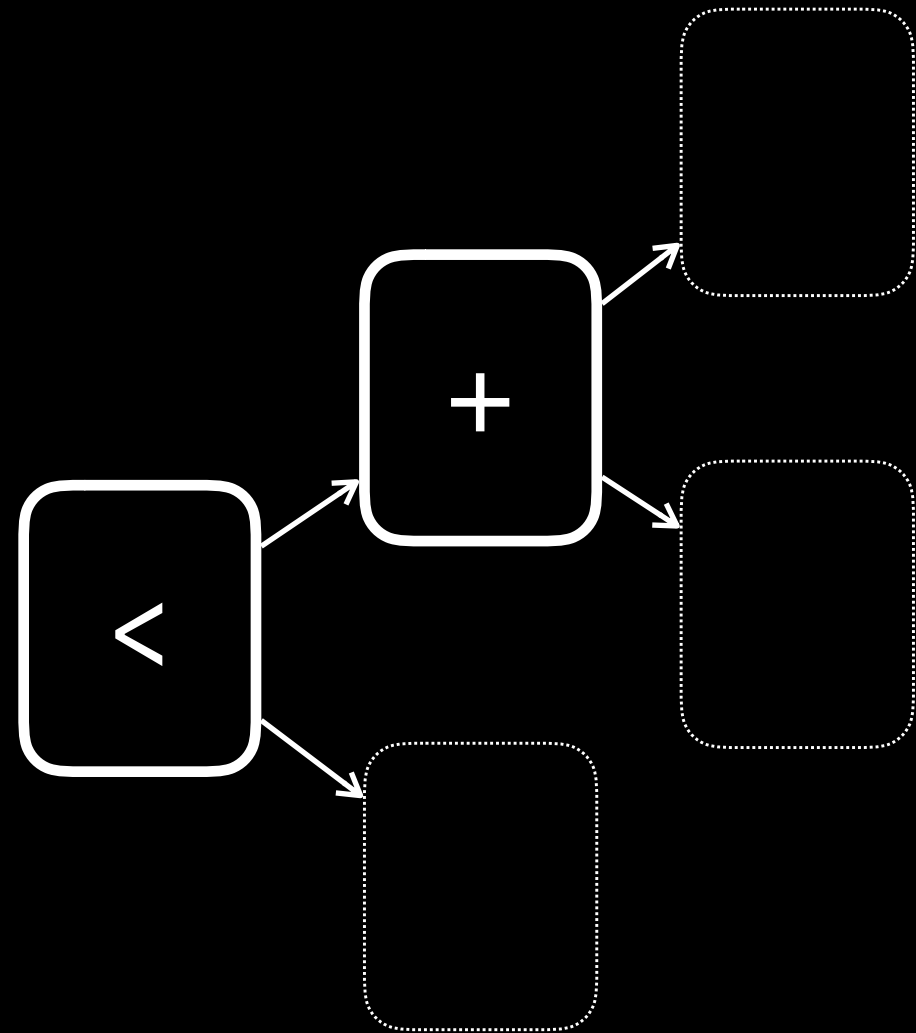




<#YourChainableDataSource#>



<#YourChainableDataSource#>
CDSTransform



```
@interface CDSTransform : NSObject <CDSChainableDataSource>

@property (nonatomic, copy) NSArray<id<CDSChainableDataSource>>* dataSources;

- (NSIndexPath*) sourceIndexPathForIndexPath:(NSIndexPath*)indexPath;

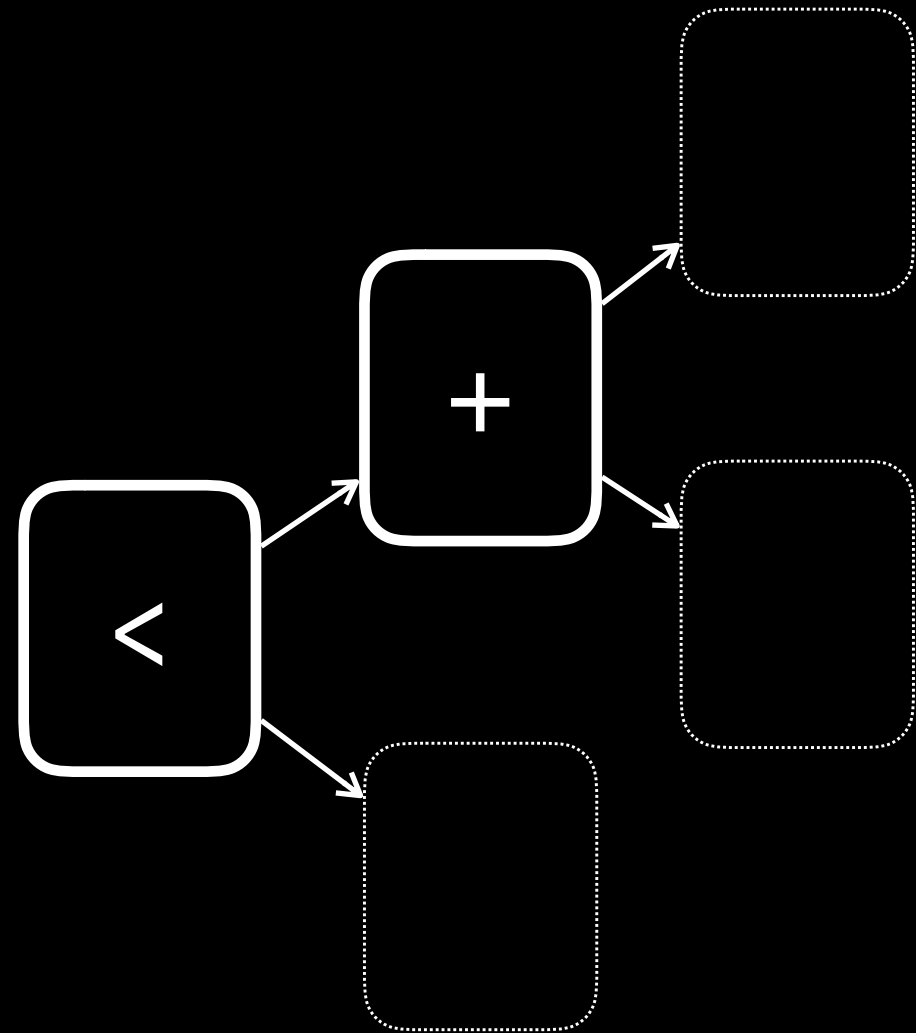
- (NSIndexPath*) indexPathForSourceIndexPath:(NSIndexPath*)sourceIndexPath
  inDataSource:(id<CDSChainableDataSource>)sourceDataSource;

- (NSInteger) sectionIndexForSourceSectionIndex:(NSInteger)sourceSection
  inDataSource:(id<CDSChainableDataSource>)sourceDataSource;

- (NSIndexPath*) sourceSectionIndexPathForSectionIndex:(NSInteger)section;

@end
```

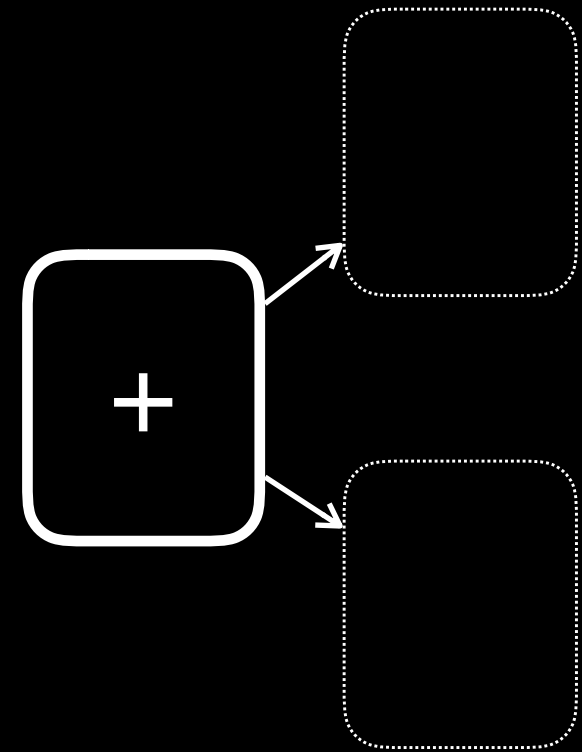
<#YourChainableDataSource#>
CDSTransform



<#YourChainableDataSource#>

CDSDataSource

CDSConcatenatedSections

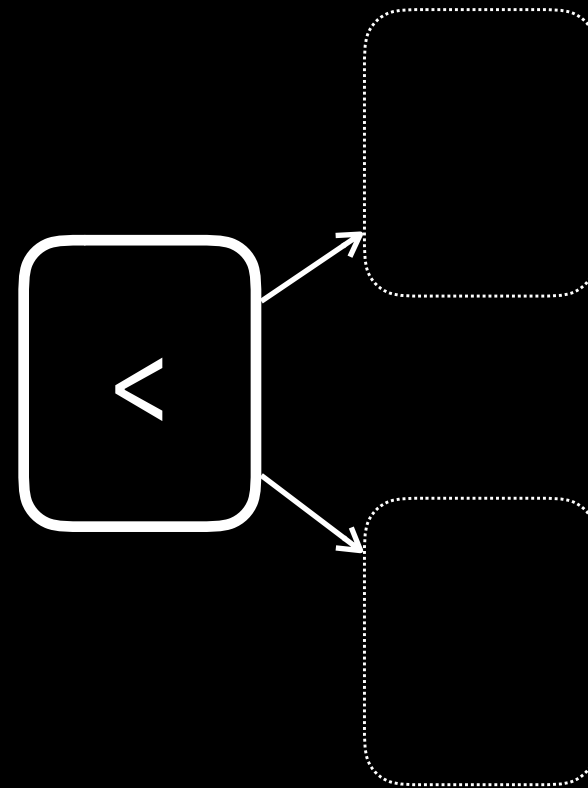


<#YourChainableDataSource#>

CDSTransform

CDSConcatenatedSections

CDSInsert



<#YourChainableDataSource#>

CDSTransform

CDSConcatenatedSections

CDSInsert

CDSSwitch

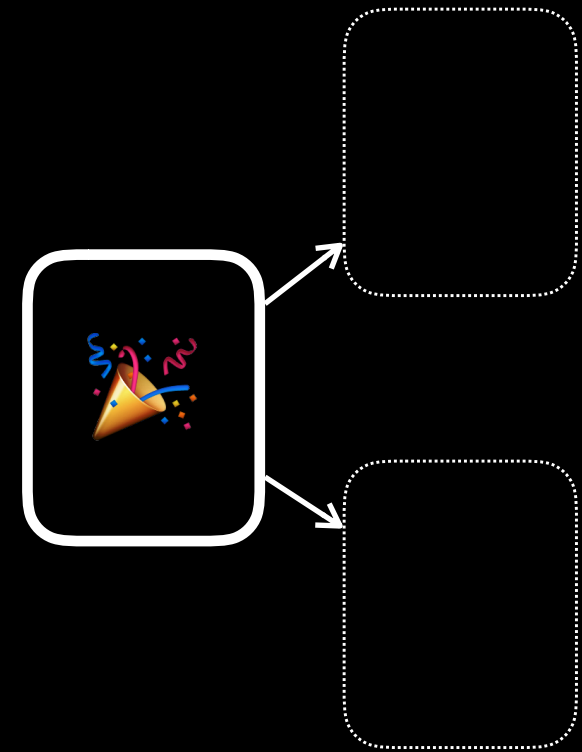
CDSFlattened

CDSPlaceholder

CDSFilter

CDSEmptySectionFilter

<#YourTransformDataSource#>

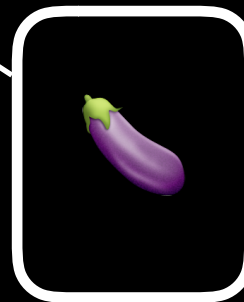
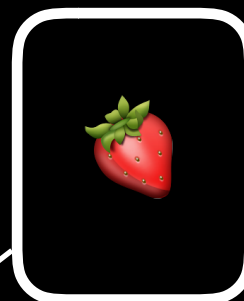
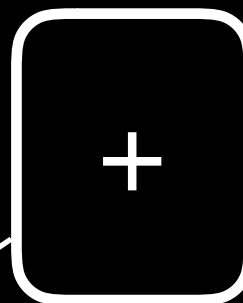
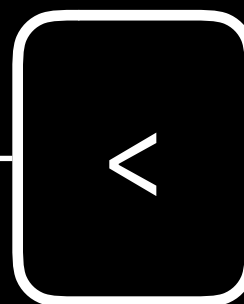
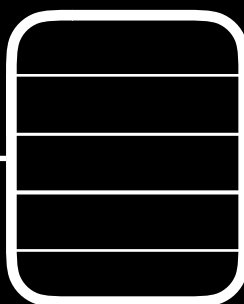
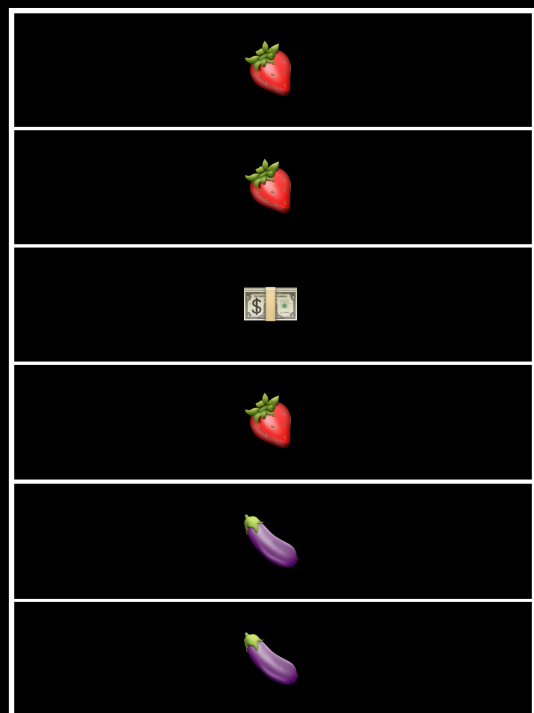


Updates

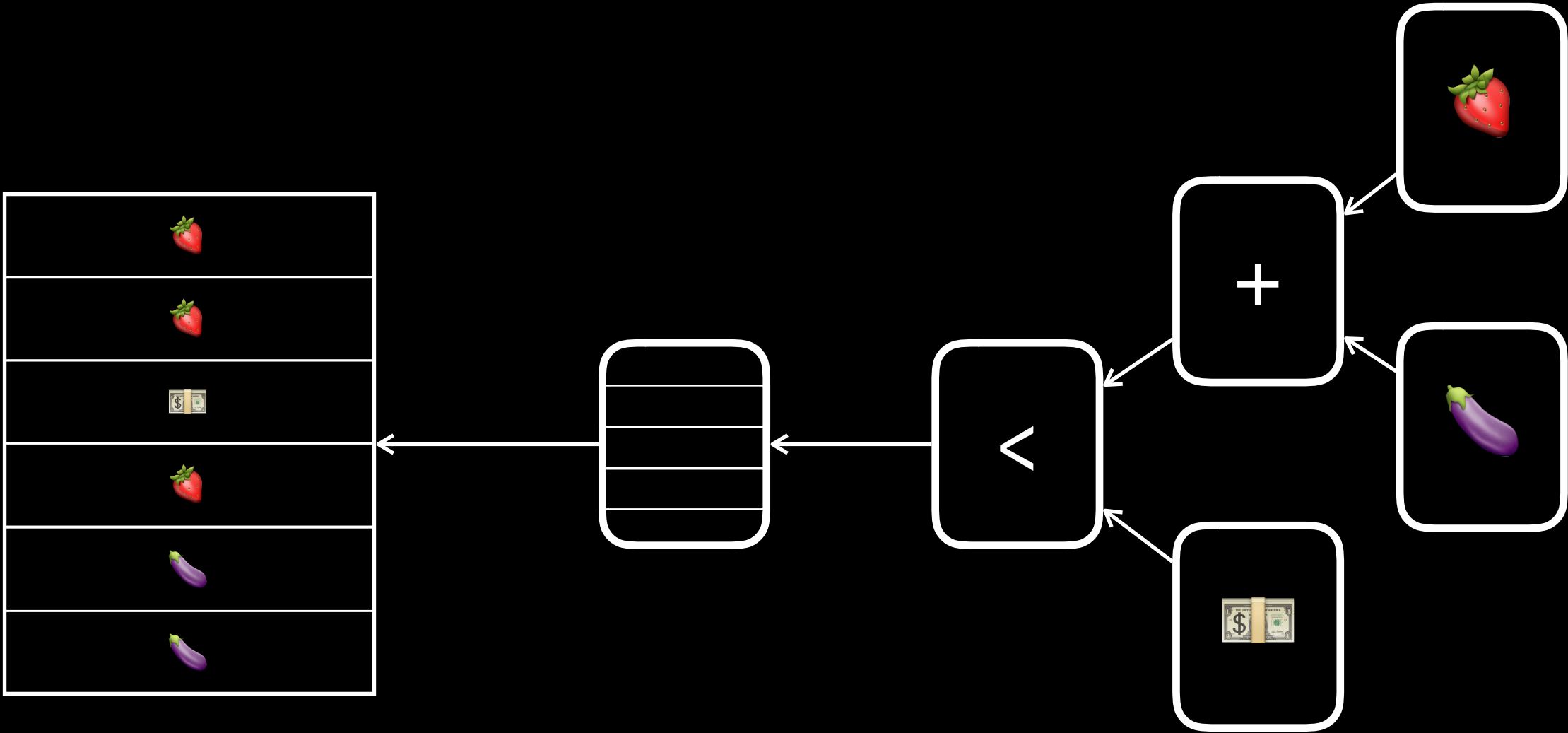
V

C

M



UpdateDelegate



```
@protocol CDSChainableDataSourceDelegate <NSObject>
```

```
- (void) cds_dataSourceDidReload:(id<ChainableDataSource>)dataSource;
- (void) cds_dataSourceWillUpdate:(id<ChainableDataSource>)dataSource;
- (void) cds_dataSourceDidUpdate:(id<ChainableDataSource>)dataSource;

- (void) cds_dataSource:(id<ChainableDataSource>)dataSource
    didDeleteSectionsAtIndexes:(NSIndexSet*)sectionIndexes;
- (void) cds_dataSource:(id<ChainableDataSource>)dataSource
    didInsertSectionsAtIndexes:(NSIndexSet*)sectionIndexes;
- (void) cds_dataSource:(id<ChainableDataSource>)dataSource
    didDeleteObjectsAtIndexPaths:(NSArray<NSIndexPath*>*)indexPath;
- (void) cds_dataSource:(id<ChainableDataSource>)dataSource
    didInsertObjectsAtIndexPaths:(NSArray<NSIndexPath*>*)indexPath;
- (void) cds_dataSource:(id<ChainableDataSource>)dataSource
    didUpdateObjectsAtIndexPaths:(NSArray<NSIndexPath*>*)indexPath;
```

```
@end
```

The truth about updates

A
B
C
D
E
F

A
C
G
D
H

The truth about updates

delete: 1, 4, 5

A
B
C
D
E
F

insert: 2, 4

A
C
G
D
H

The truth about updates

update: 1

A
B
C
D
E
F

A
C
G
D
H

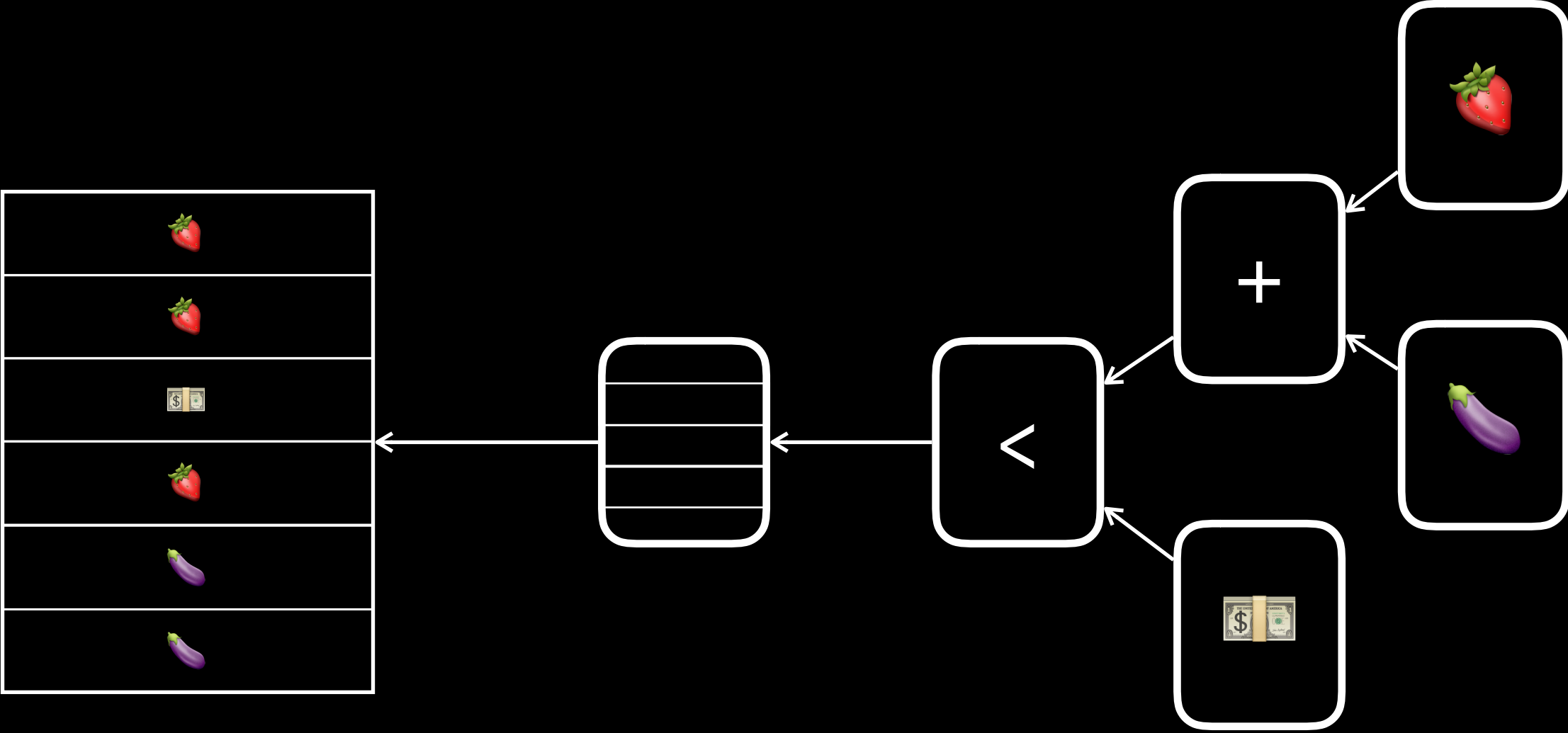
The truth about updates

```
[tableView beginUpdates];  
[self.fruits removeObjectAtIndex:indexPath.row];  
[tableView deleteRowsAtIndexPaths:@[indexPath]  
                withRowAnimation:UITableViewRowAnimationAutomatic];  
[tableView endUpdates];  
//OK
```

The truth about updates

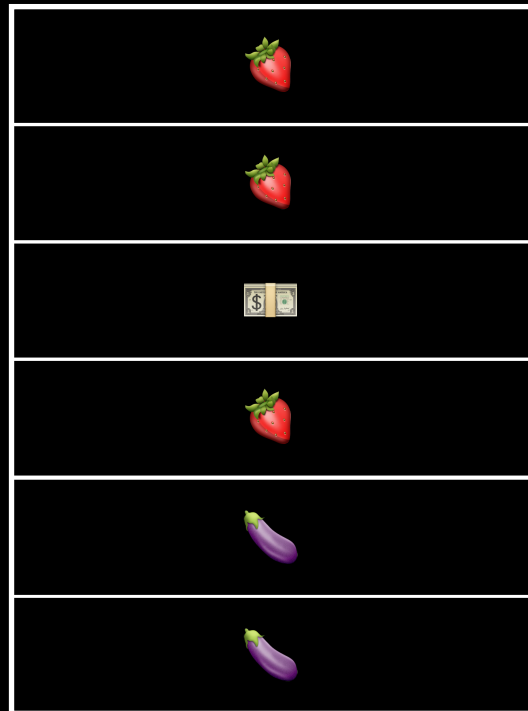
```
[tableView beginUpdates];
[self.fruits removeObjectAtIndex:indexPath.row];
[tableView deleteRowsAtIndexPaths:@[indexPath]
      withRowAnimation:UITableViewRowAnimationAutomatic];
[tableView deleteRowsAtIndexPaths:@[indexPath]
      withRowAnimation:UITableViewRowAnimationAutomatic];
[tableView deleteRowsAtIndexPaths:@[indexPath]
      withRowAnimation:UITableViewRowAnimationAutomatic];
[tableView deleteRowsAtIndexPaths:@[indexPath]
      withRowAnimation:UITableViewRowAnimationAutomatic];
[tableView endUpdates];
//Still OK
```

UpdateDelegate

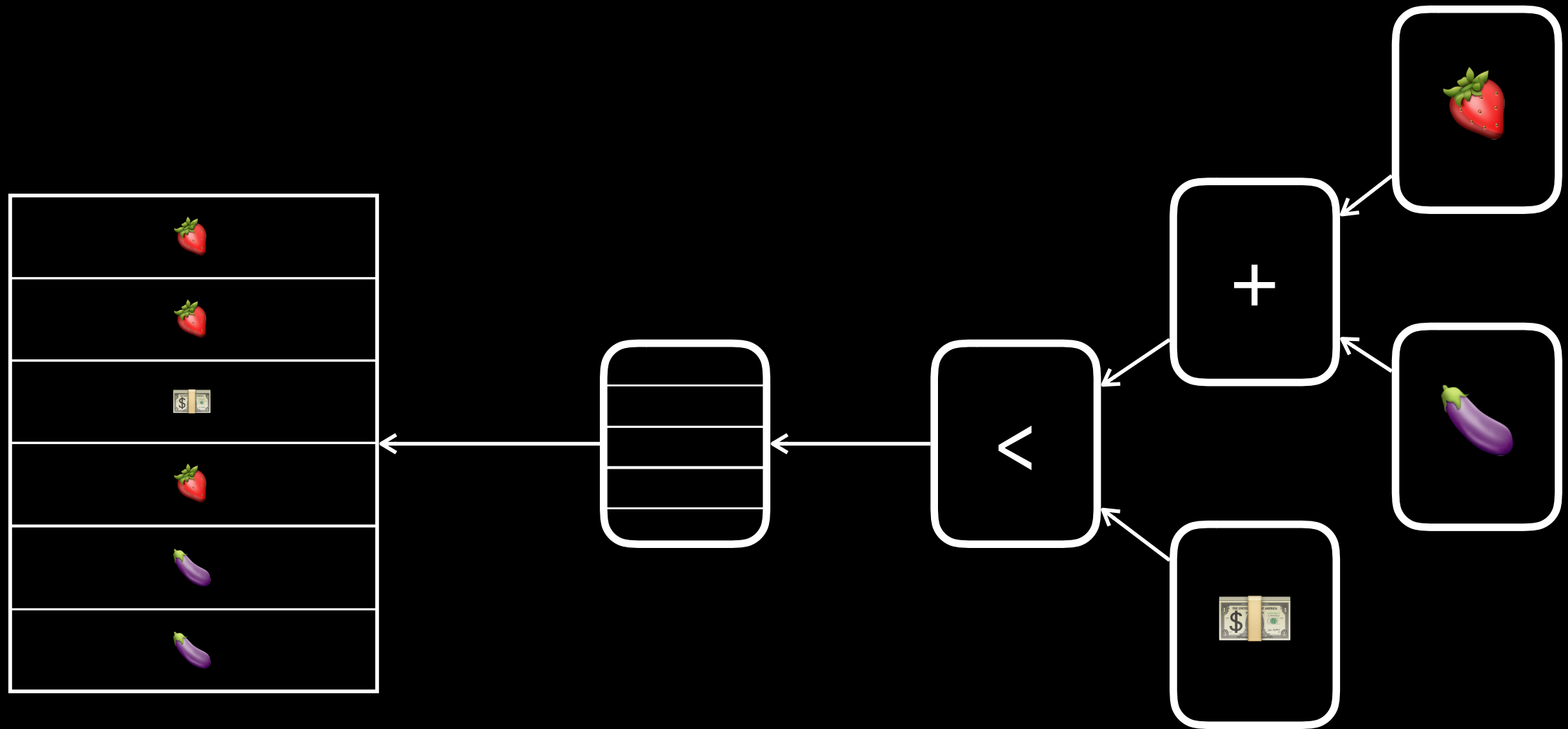


UITableView <CDSUpdateDelegate>

UICollectionView <CDSUpdateDelegate>

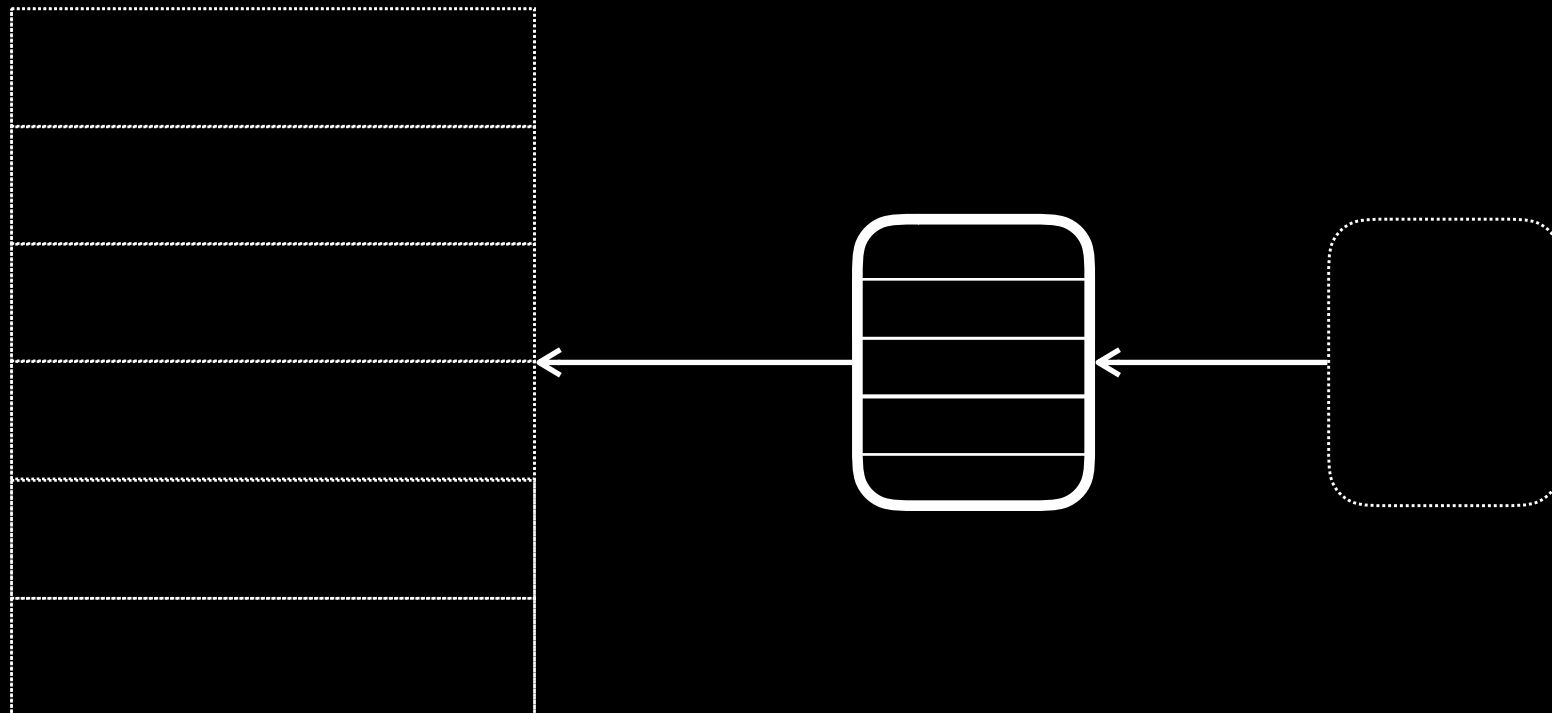


UpdateDelegate

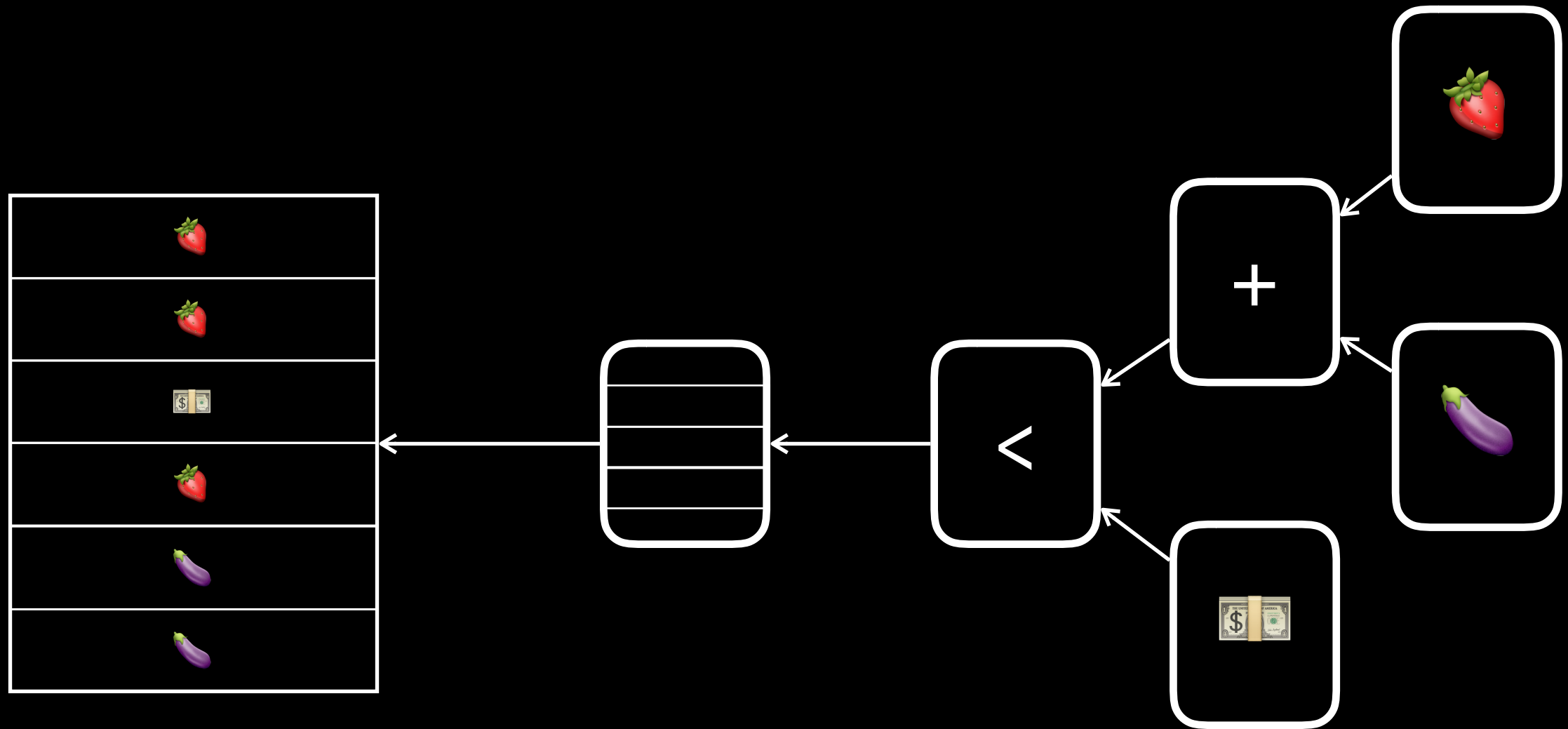


CellDataSource

didUpdateObjectsAtIndexPaths -> configureCell:withObject:
everything else -> forward

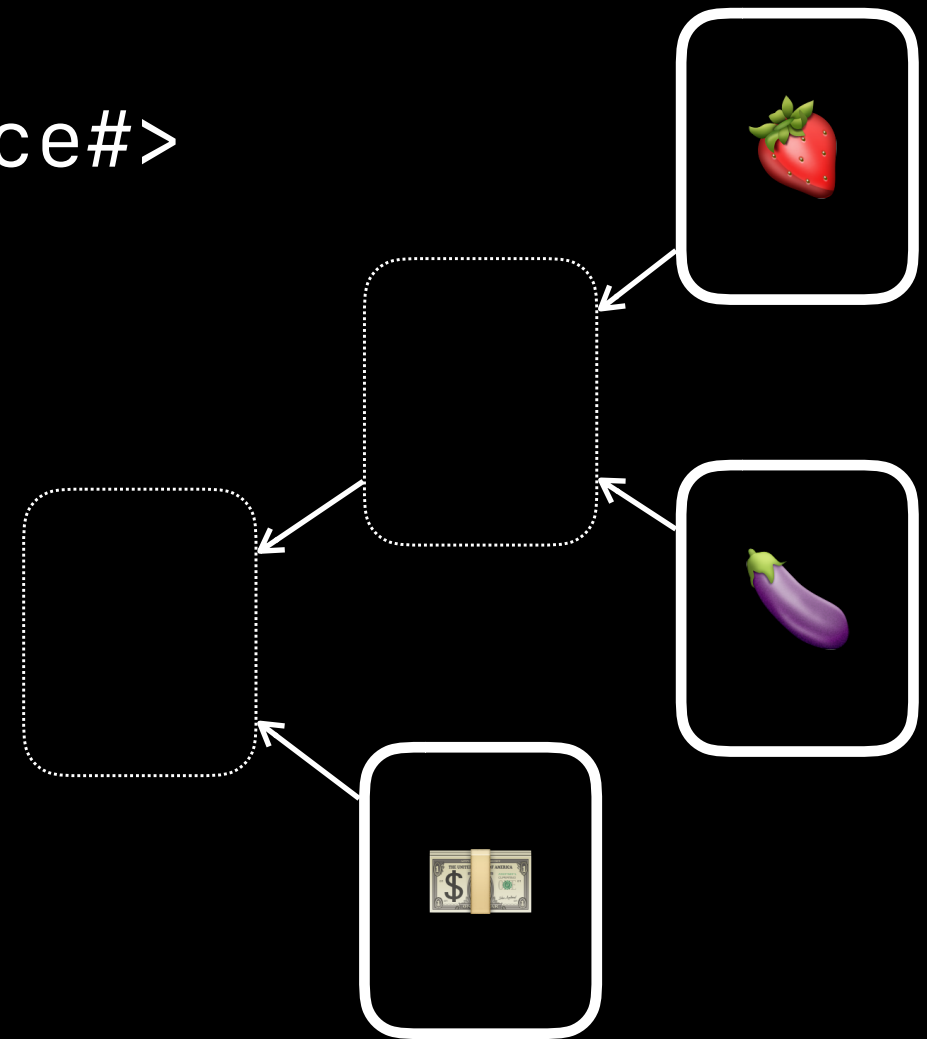


UpdateDelegate

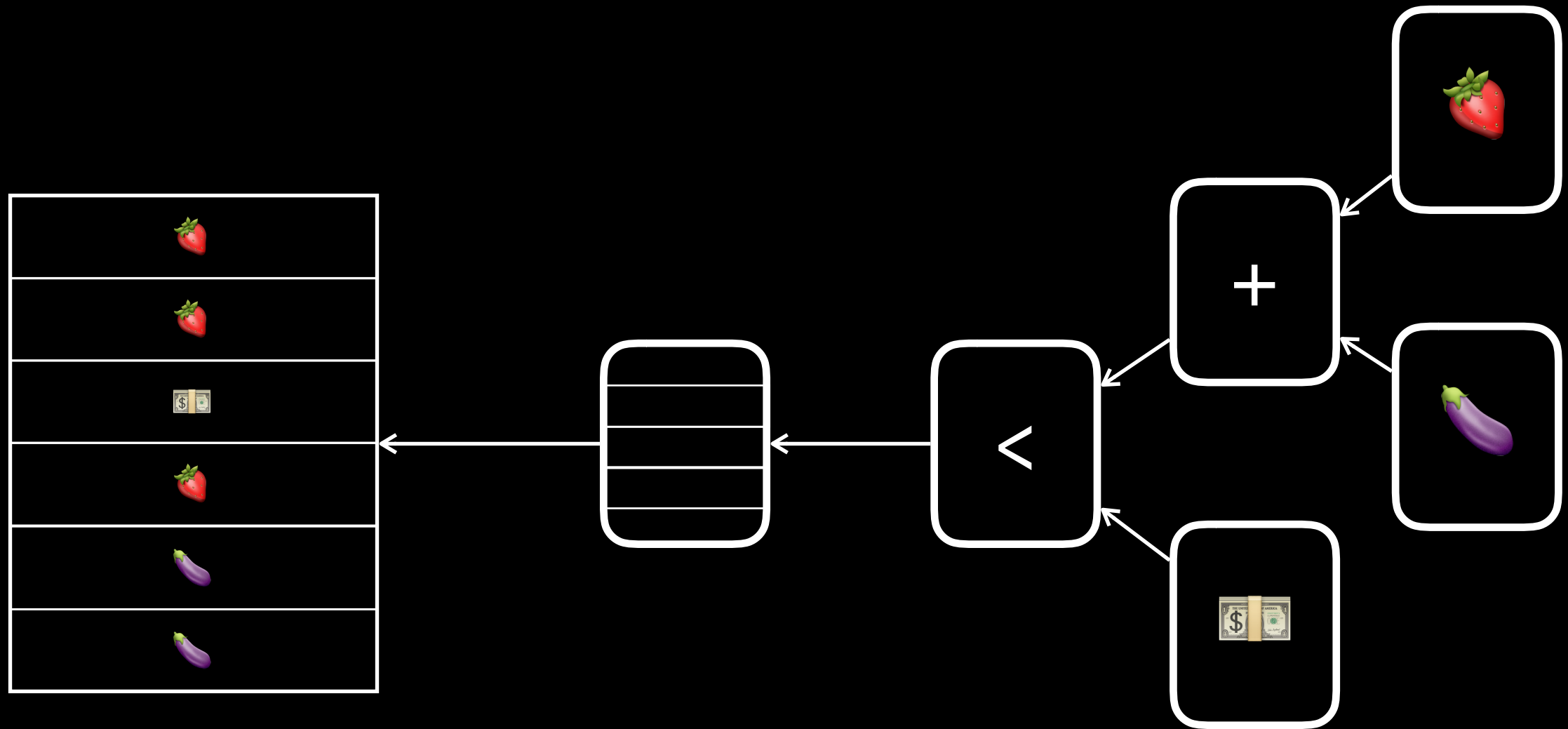


CDSFetchWrapper

<#YourChainableDataSource#>

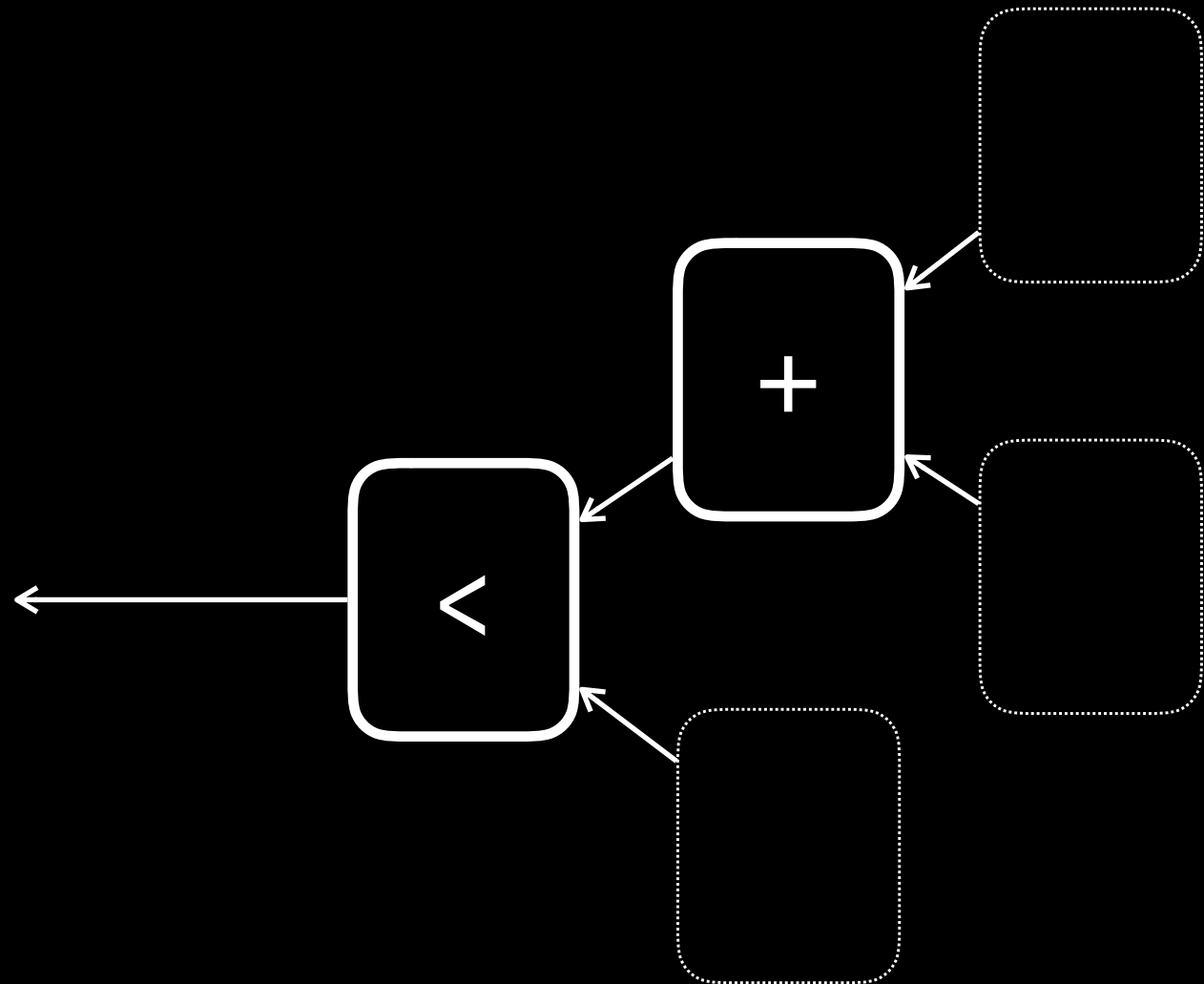


UpdateDelegate



CDSTransform

basic mapping \rightarrow automatic









CDSTransform

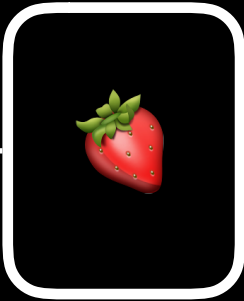
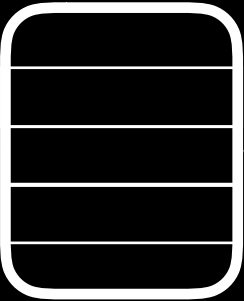
advanced mapping:

tweak update set pre and post upstream updates



- (void) preRefreshTranslateSourceUpdateCache:(UpdateCache*)sourceUpdateCache
fromDataSource:(id<ChainableDataSource>)dataSource
toUpdateCache:(UpdateCache*)updateCache;
- (void) postRefreshTranslateSourceUpdateCache:(UpdateCache*)sourceUpdateCache
fromDataSource:(id<ChainableDataSource>)dataSource
toUpdateCache:(UpdateCache*)updateCache;

cds_dataSourceDidReload



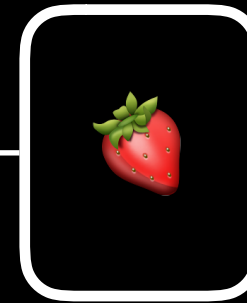
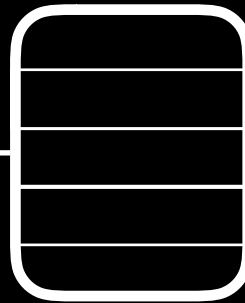


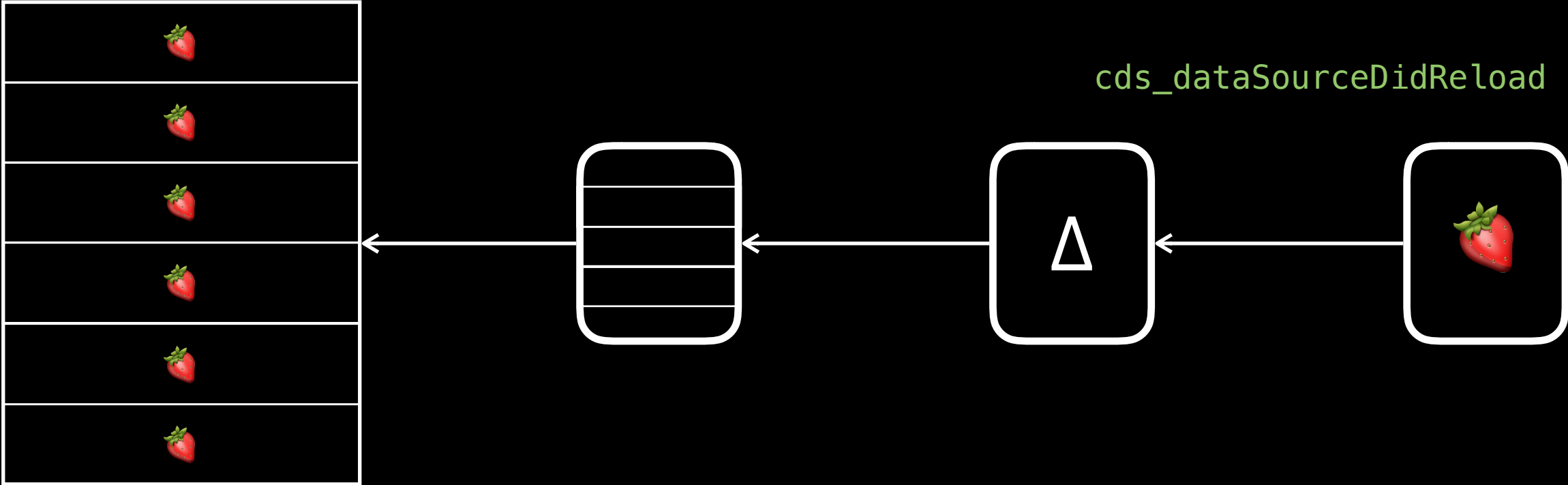


cds_dataSourceDidReload
(reloadData - 🥲)

cds_dataSourceDidReload



CDSDeltaUpdater



A
B
C
D
E
F

A
C
G
D
H

A
B
C
D
E
F

—

A
C
G
D
H

=

?

diff algorithm

Longest Common Subsequence

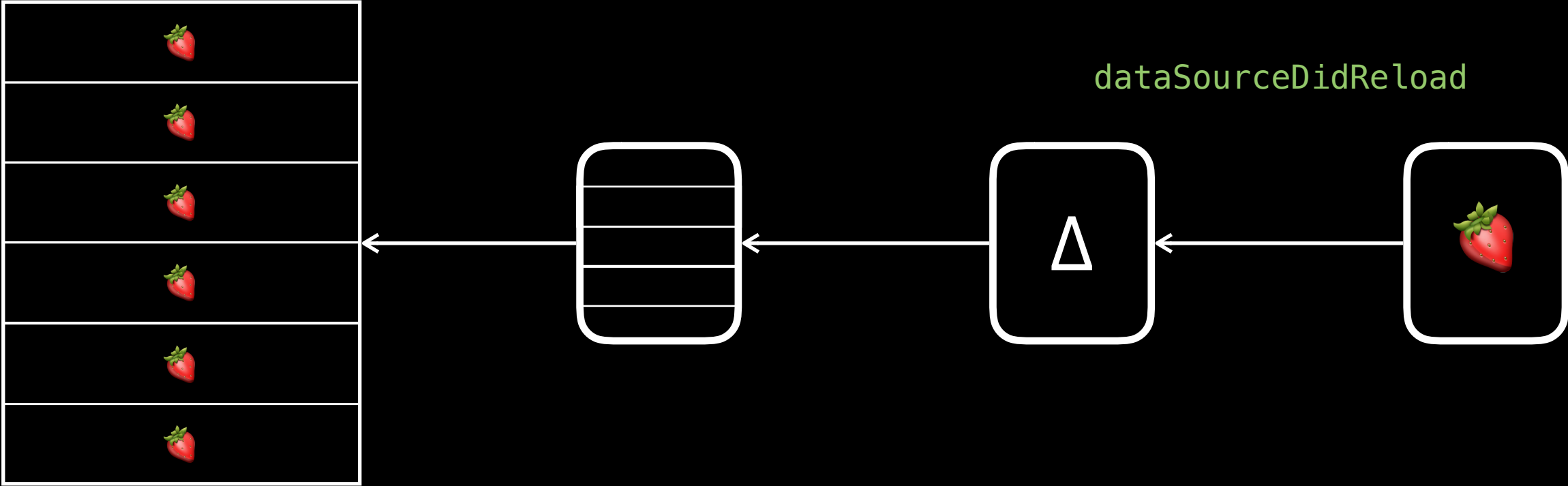
delete: 1, 4, 5

A
B
C
D
E
F

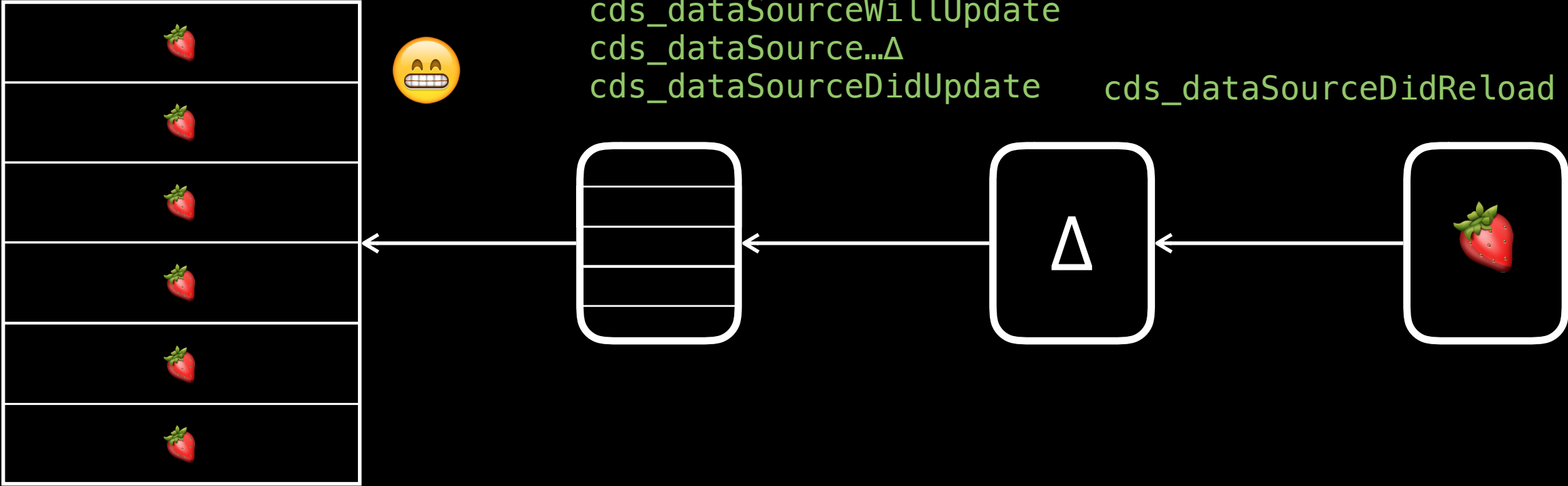
insert: 2, 4

A
C
G
D
H

CDSDeltaUpdater



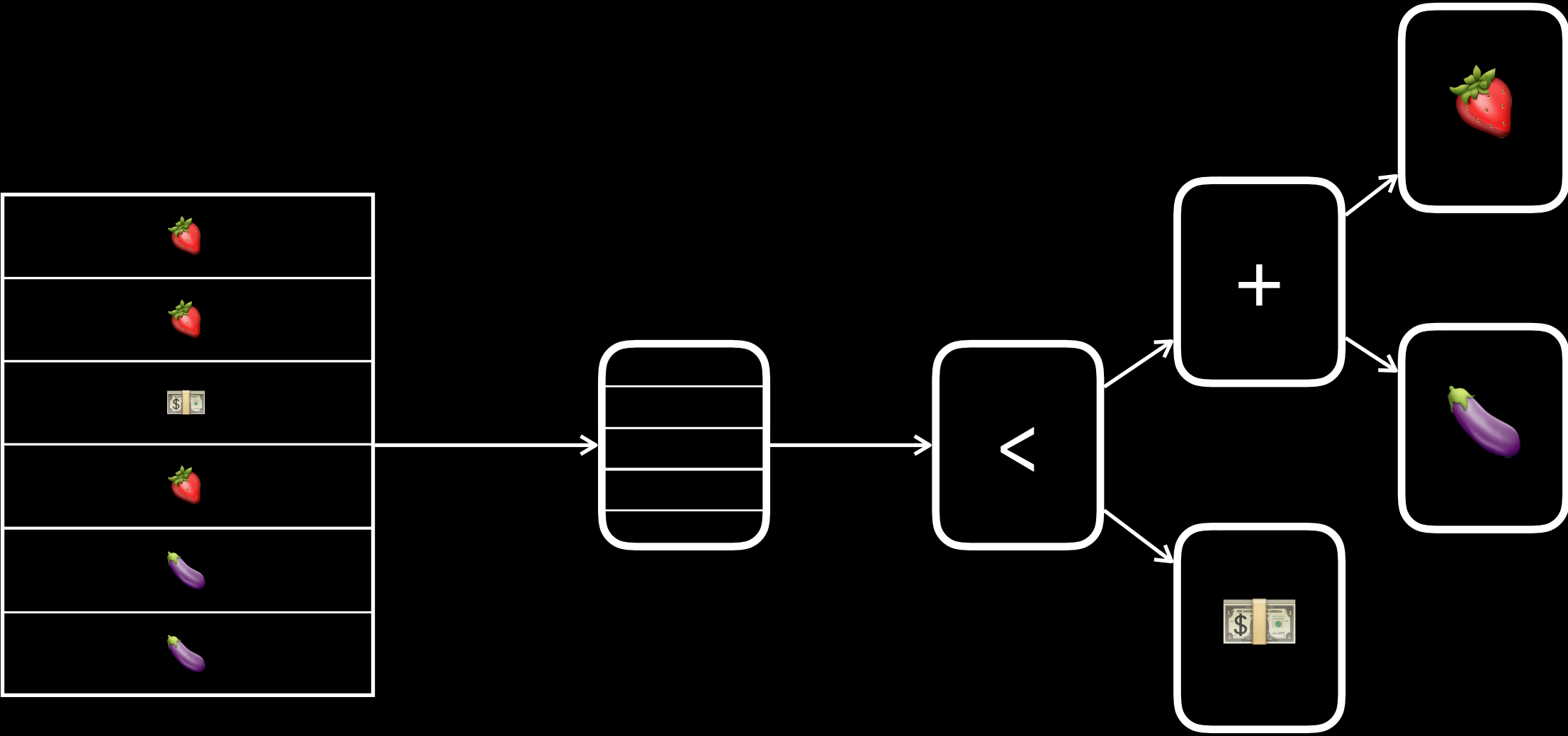
CDSDeltaUpdater



Updates

//TODO: Nothing

CellDataSource

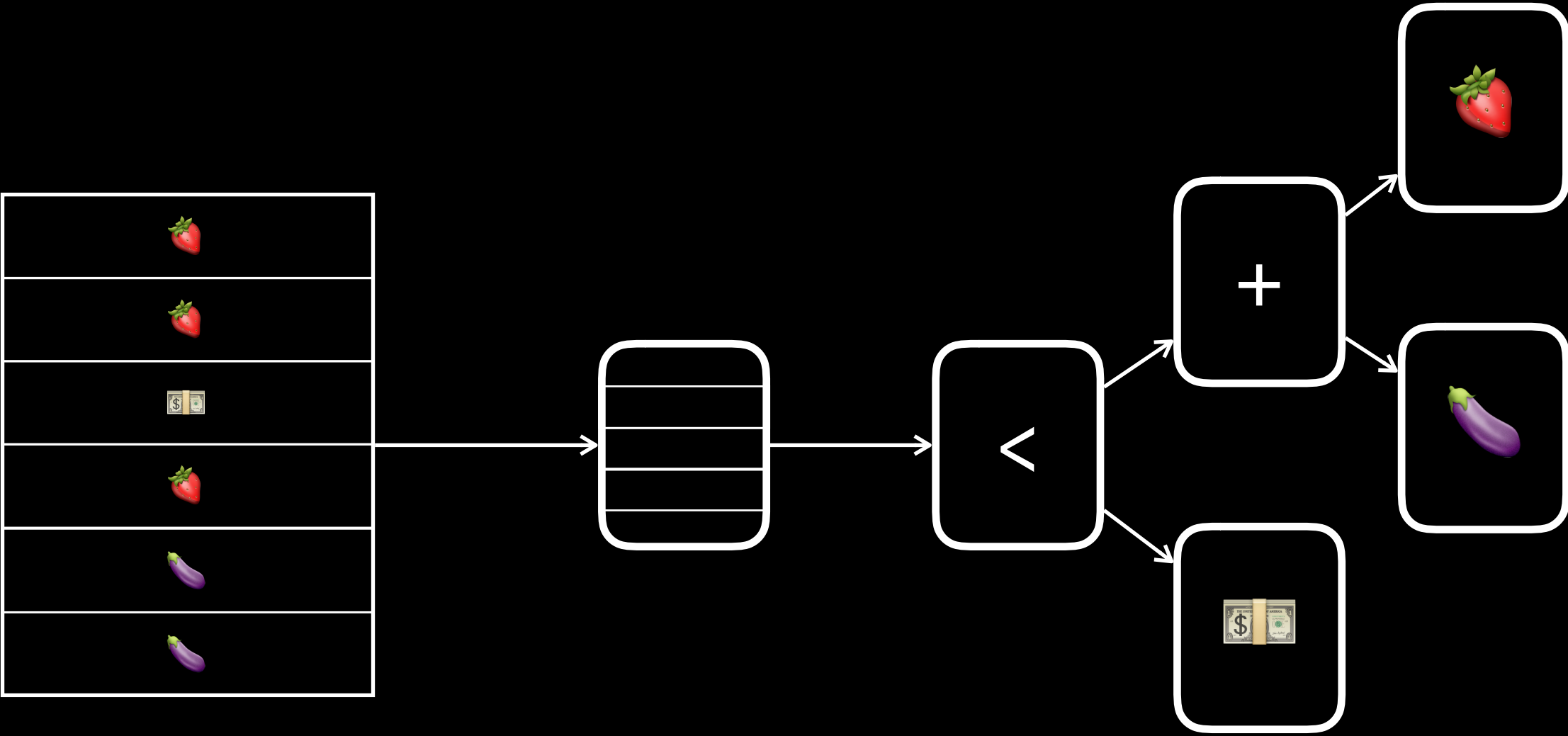


CellDataSource

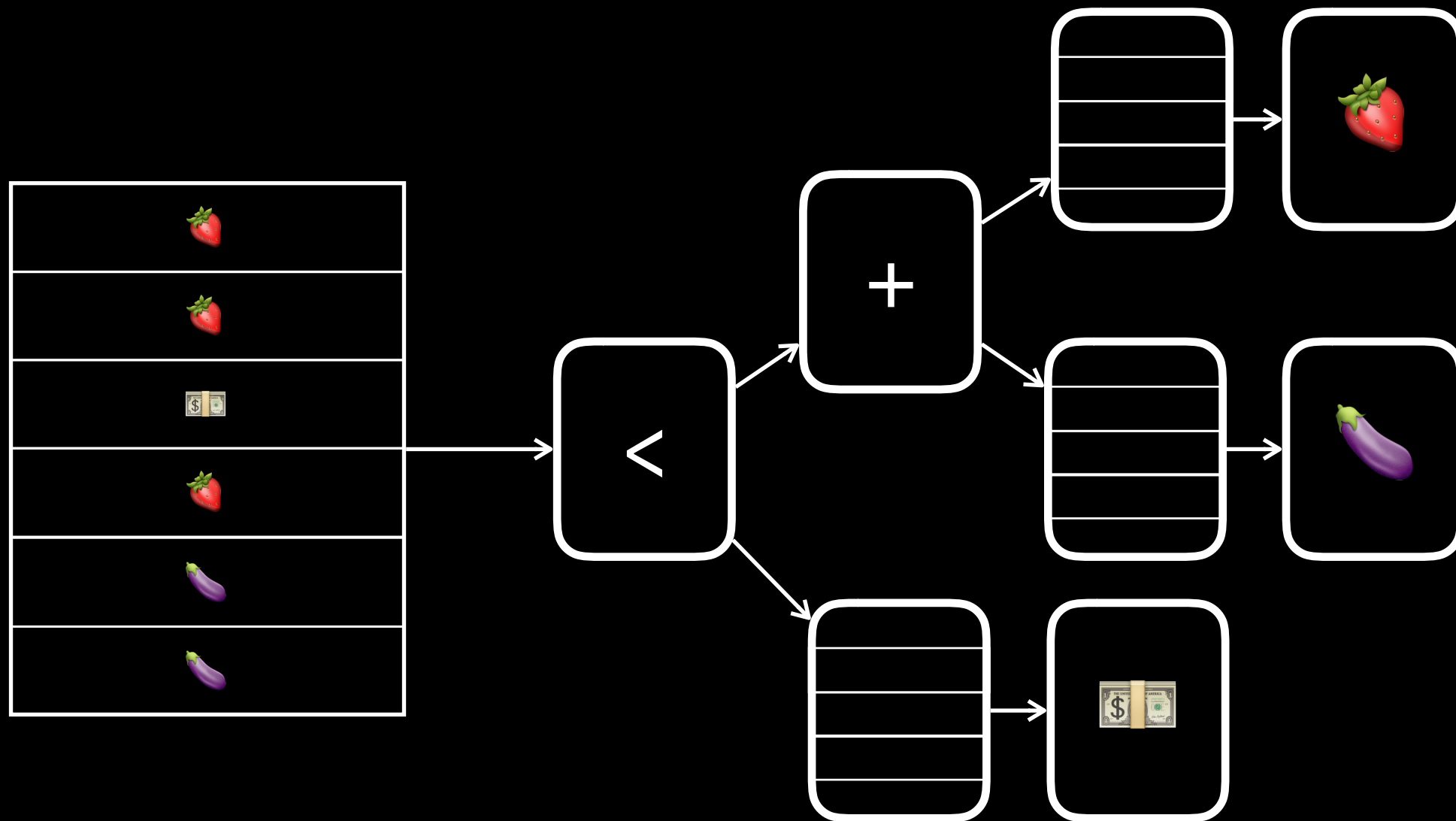
```
- (NSString*) cellIdentifierForObject:(id)object
{
    return NSStringFromClass([object class]);
}

- (void) configureCell:(UITableViewCell*)cell withObject:(id)object
{
    if ([cell.reuseIdentifier isEqual:@"Fruit"]) {
        //...
    } else if ([cell.reuseIdentifier isEqual:@"Vegetable"]) {
        //...
    }
    //😓
}
```

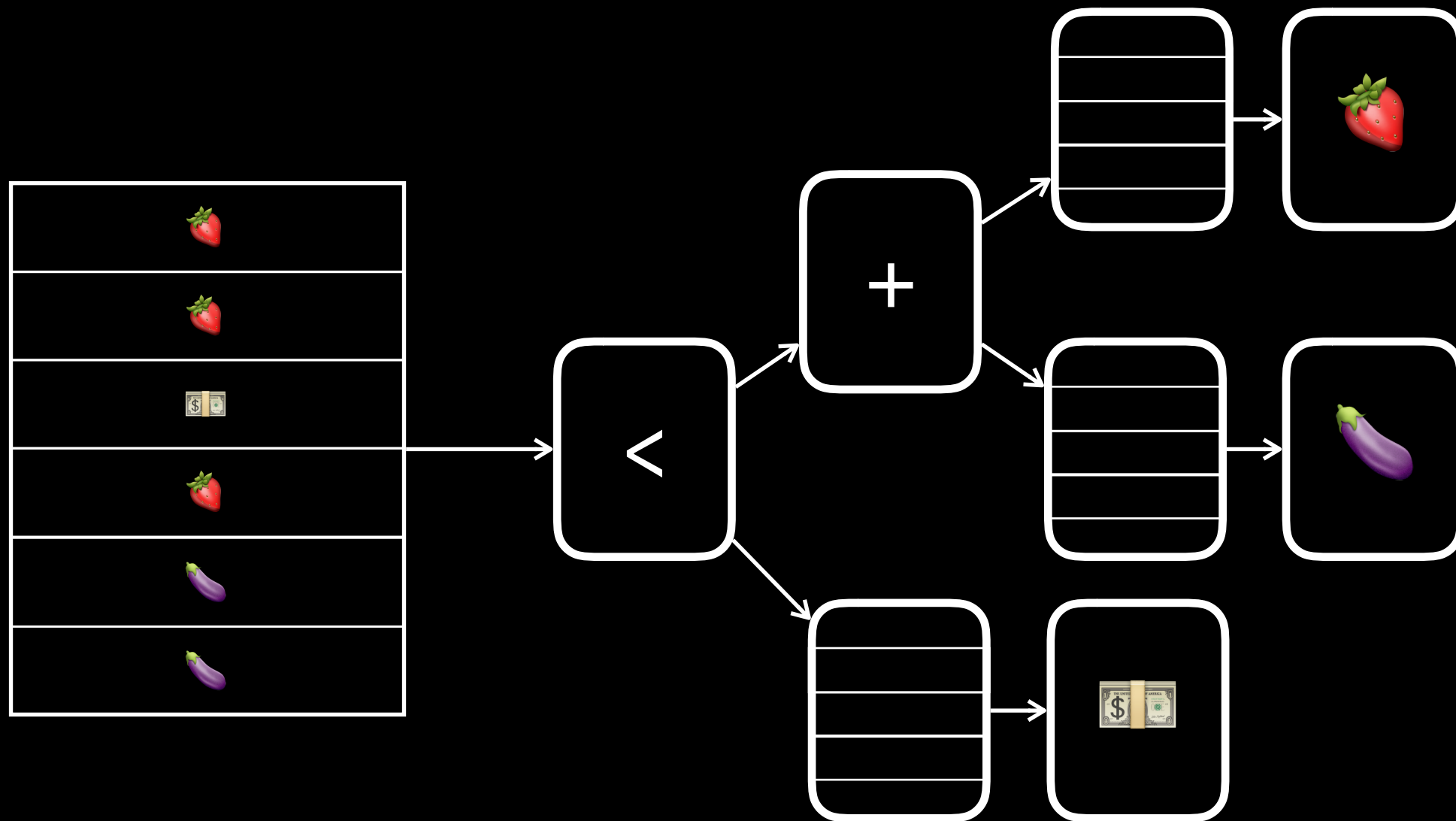

CellDataSource



CellDataSource: alternative approach



CDSTransform <CDSCellDataSource>
forward delegate/data source methods
based on index mapping



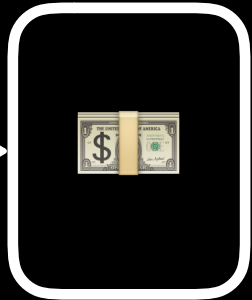
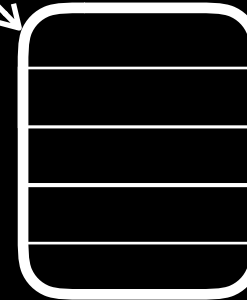
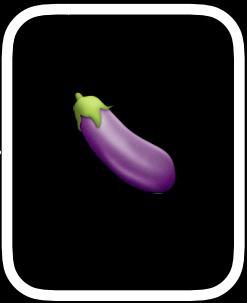
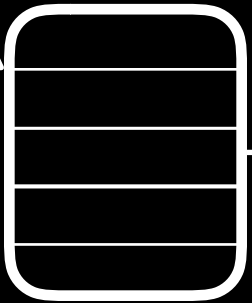
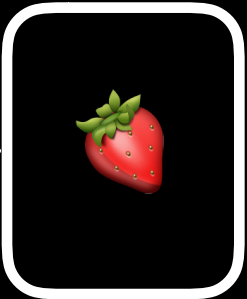
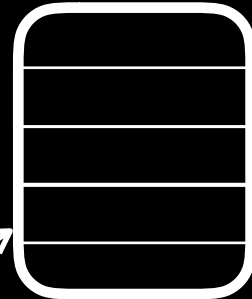
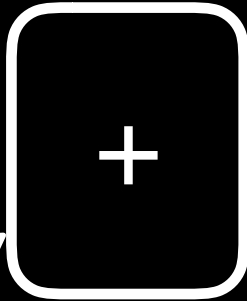
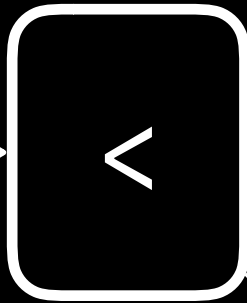
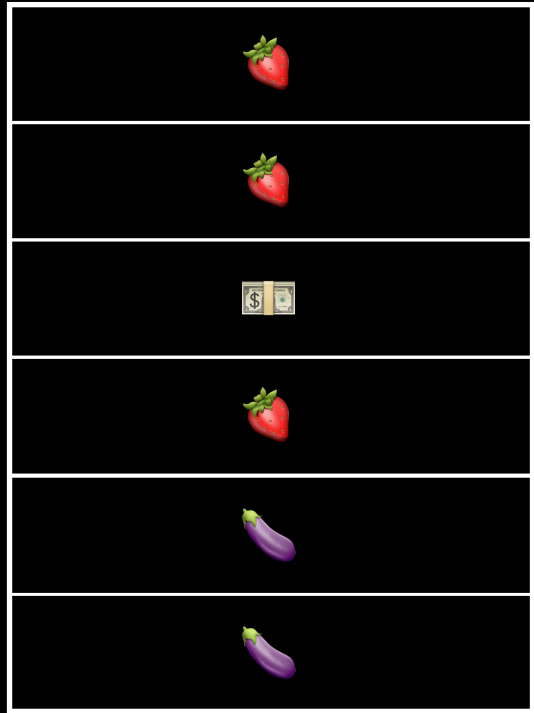
CDSTransform <CDSCellDataSource>
forward delegate/data source methods
based on index mapping

(My dirty secret)

```

- (void) forwardInvocation:(NSInvocation *)anInvocation
{
    if ([self isForwardableDelegateSelector:anInvocation.selector]) {
        NSMethodSignature* signature = anInvocation.methodSignature;
        NSArray* components = [NSStringFromSelector(anInvocation.selector)componentsSeparatedByString:@":"];
        for (NSInteger argIndex = 0; argIndex < signature.numberOfArguments; argIndex++) {
            //skip self and cmd
            if (argIndex < 2) {
                continue;
            }
            //won't work with targetIndexPathForMoveFromRowAtIndexPath, since two indexPath are present
            const char* argType = [signature getArgumentTypeAtIndex:argIndex];
            if (strcmp(argType, @encode(NSIndexPath*)) == 0) {
                __unsafe_unretained id arg;
                [anInvocation getArgument:&arg atIndex:argIndex];
                if ([arg isKindOfClass:[NSIndexPath class]]) {
                    NSIndexPath* indexPath = arg;
                    NSIndexPath* fullIndexPath = [self sourceIndexPathForIndexPath:indexPath];
                    id<ChainableDataSource> dataSource = self.dataSources[[fullIndexPath indexPathAtIndex:0]];
                    NSIndexPath* dsIndexPath = [NSIndexPath indexPathForRow:[fullIndexPath indexPathAtIndex:2] inSection:[fullIndexPath indexPathAtIndex:1]];
                    [anInvocation setArgument:&dsIndexPath atIndex:argIndex];
                    if ([dataSource respondsToSelector:anInvocation.selector]) {
                        [anInvocation invokeWithTarget:dataSource];
                    }
                    return;
                }
            }
            else if (strcmp(argType, @encode(NSInteger)) == 0 && [components[argIndex-2] hasSuffix:@"Section"]) {
                NSInteger section;
                [anInvocation getArgument:&section atIndex:argIndex];
                NSIndexPath* sourceSectionIndexPath = [self sourceSectionIndexPathForSectionIndex:section];
                if (sourceSectionIndexPath) {
                    id<ChainableDataSource> dataSource = self.dataSources[[sourceSectionIndexPath indexPathAtIndex:0]];
                    NSInteger sourceSection = [sourceSectionIndexPath indexPathAtIndex:1];
                    [anInvocation setArgument:&sourceSection atIndex:argIndex];
                    if ([dataSource respondsToSelector:anInvocation.selector]) {
                        [anInvocation invokeWithTarget:dataSource];
                    }
                }
            }
            return;
        }
    }
    return [super forwardInvocation:anInvocation];
}

```



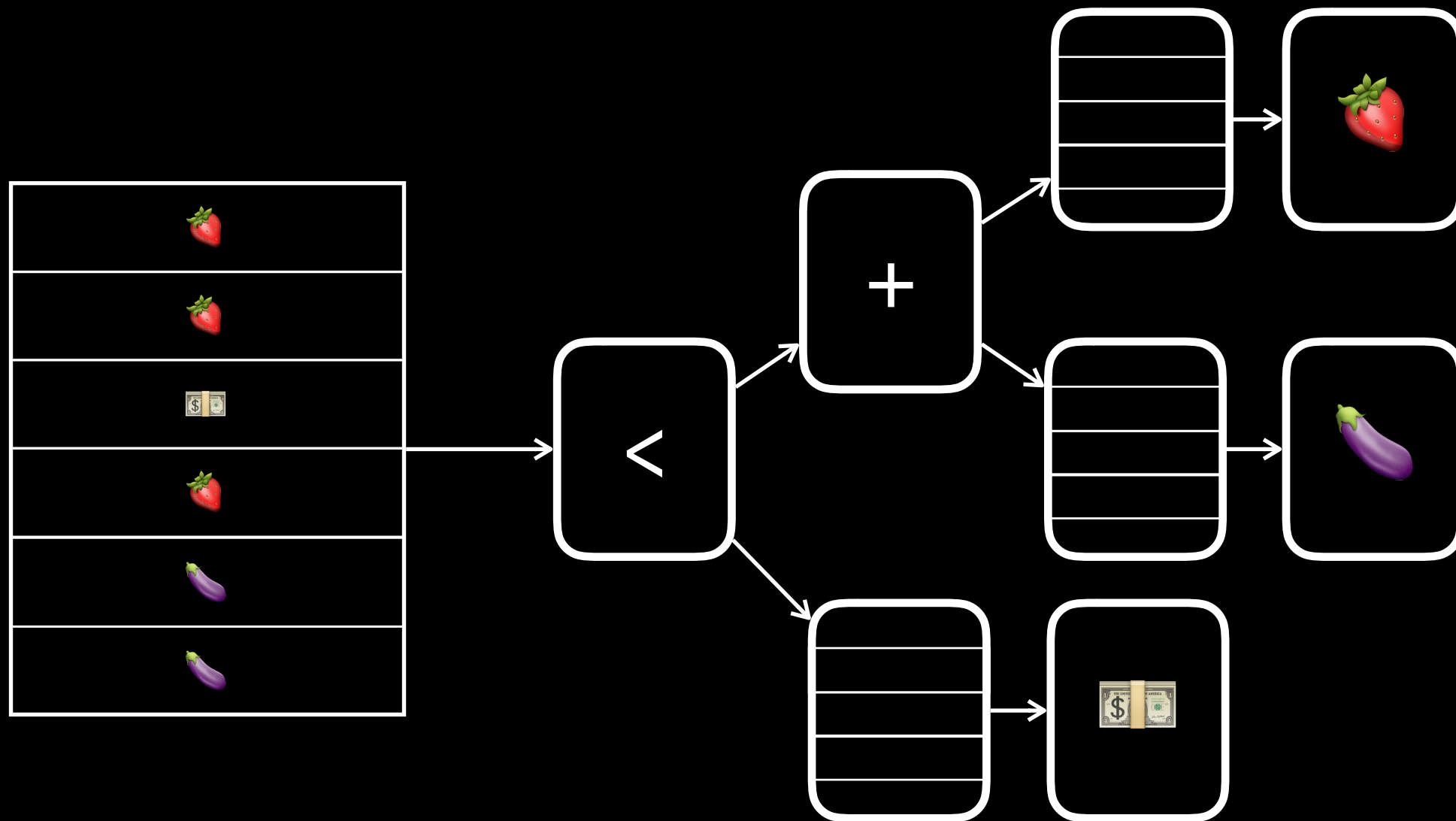
CellDataSource

```
@implementation FruitCellDataSource
- (NSString*) cellIdentifierForObject:(id)object
{
    return @"fruit-cell";
}

- (void) configureCell:(UITableViewController*)cell withObject:(id)object
{
    Fruit* fruit = object;
    cell.titleLabel.text = fruit.name;
}

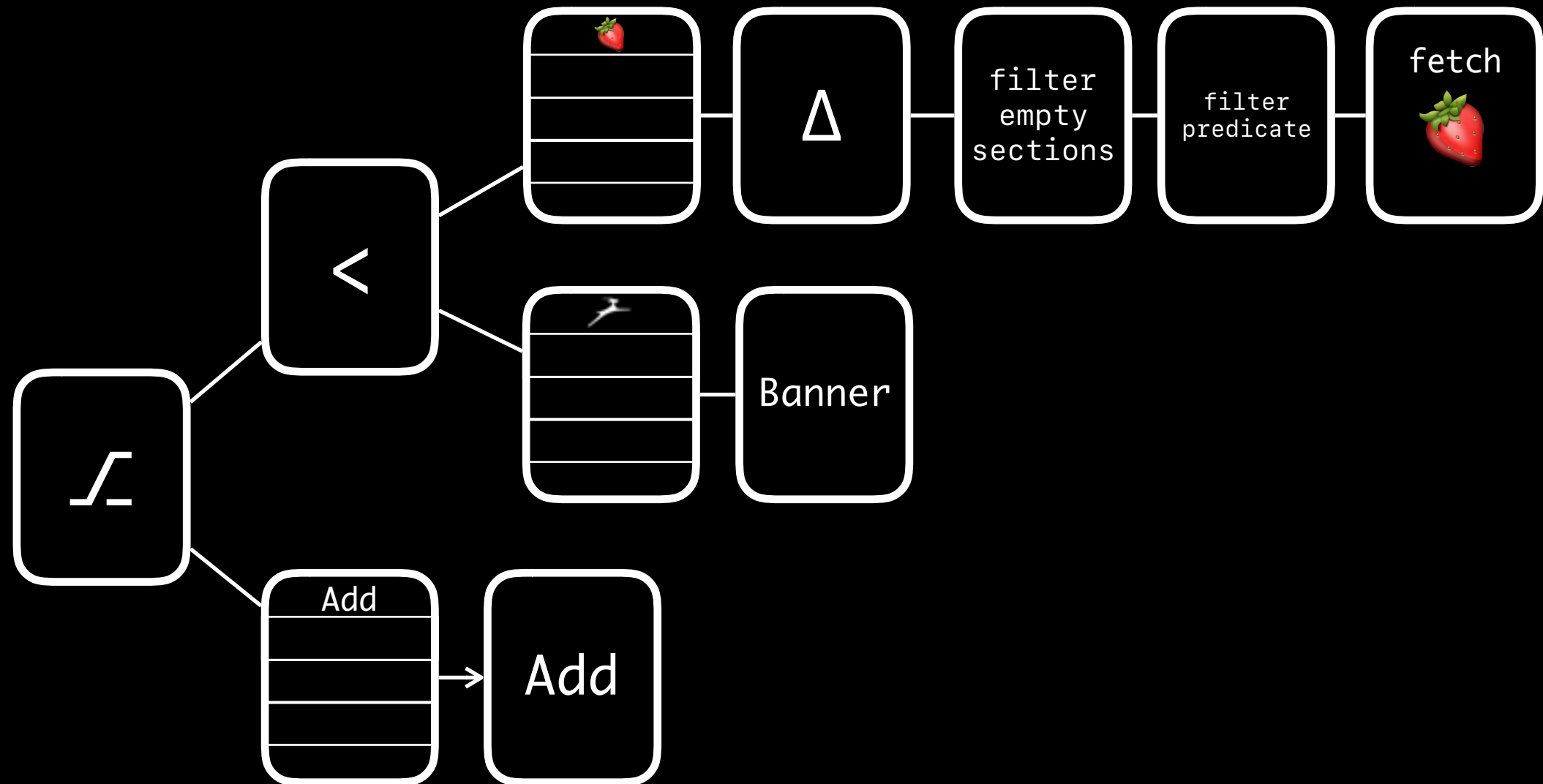
- (void) tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    Fruit* fruit = [self.objectsDataSource
                    dataSourceObjectAtIndexPath:indexPath];
    //...
}
@end
```

It's alive!



Demo

Fruitypedia





Open source