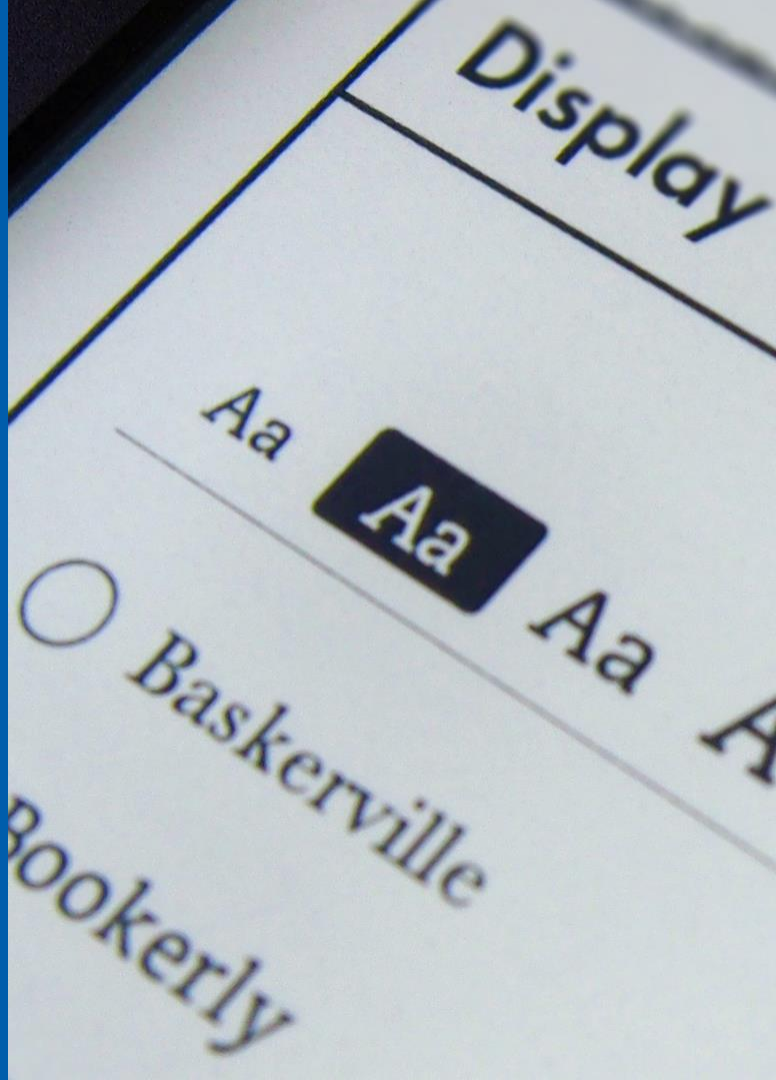
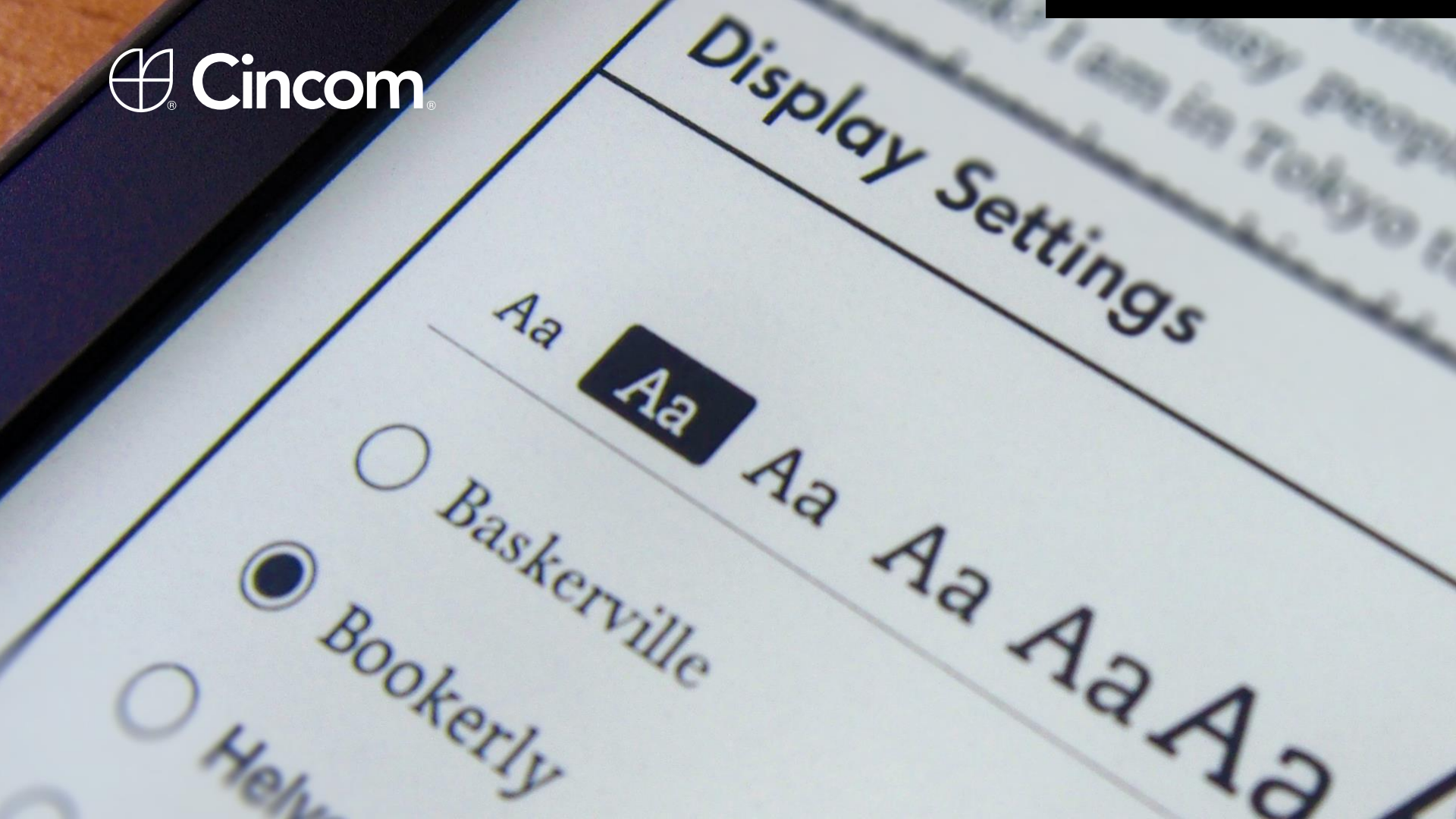
Cincom

FONTS!

By Arden Thomas




Cincom
Display Settings
Aa
Aa
Aa
Baskerville
Bookerly

Fonts!

Amazon recently introduced a new font called “Bookerly,” which according to Amazon, is optimized for readability.

Bookerly was designed for easier reading and reduced eyestrain. You may have seen this font recently installed on your Kindle or Kindle app if you use it.

This sounds like it might be a good candidate for Smalltalk code editing.

Monospaced vs Proportional Fonts

Monospaced fonts popular for code editors

Pros:

- Better separation for punctuation and non alpha characters
- Better character identification

Cons:

- Often low resolution
- Harder on eyes
- The uniformity makes it harder to see typos

Monospaced vs Proportional Fonts

A Proportional font is the original Smalltalk font

Pros:

- Less screen space
- Easier on the eyes
- More modern and usually renders better

Cons:

- Vertical lineup
- Punctuation characters get “crowded”

Fonts!

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ˆnewObject
```

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

Fonts!

Source Code Pro semibold

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

Charis sil modifiedlarger

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

Inputmono medium

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

```
add: newObject after: oldObject
	"Add the argument, newObject, as an element of the receiver.  Put it
	in the position just succeeding oldObject.  Answer newObject."

	| index |
	index := self find: oldObject.
	self insert: newObject before: index + 1.
	^newObject
```


```
add: newObject after: oldObject
	"Add the argument, newObject, as an element of the receiver.  Put it
	in the position just succeeding oldObject.  Answer newObject."

	| index |
	index := self find: oldObject.
	self insert: newObject before: index + 1.
	^newObject
```

```
add: newObject after: oldObject
	"Add the argument, newObject, as an element of the receiver.  Put it
	in the position just succeeding oldObject.  Answer newObject."

	| index |
	index := self find: oldObject.
	self insert: newObject before: index + 1.
	^newObject
```

```
add: newObject after: oldObject
	"Add the argument, newObject, as an element of the receiver.  Put it
	in the position just succeeding oldObject.  Answer newObject."

	| index |
	index := self find: oldObject.
	self insert: newObject before: index + 1.
	^newObject
```

```
add: newObject after: oldObject
	"Add the argument, newObject, as an element of the receiver.  Put it
	in the position just succeeding oldObject.  Answer newObject."

	| index |
	index := self find: oldObject.
	self insert: newObject before: index + 1.
	^newObject
```

Arial

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

Century Schoolbook

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

Lucida Sans

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

Lucida Sans – 2

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

Lucida Console

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

```
add: newObject after: oldObject
	"Add the argument, newObject, as an element of the receiver.  Put it
	in the position just succeeding oldObject.  Answer newObject."

	| index |
	index := self find: oldObject.
	self insert: newObject before: index + 1.
	^newObject
```


```
add: newObject after: oldObject
	"Add the argument, newObject, as an element of the receiver.  Put it
	in the position just succeeding oldObject.  Answer newObject."

	| index |
	index := self find: oldObject.
	self insert: newObject before: index + 1.
	^newObject
```

```
add: newObject after: oldObject
	"Add the argument, newObject, as an element of the receiver.  Put it
	in the position just succeeding oldObject.  Answer newObject."

	| index |
	index := self find: oldObject.
	self insert: newObject before: index + 1.
	^newObject
```

```
add: newObject after: oldObject
	"Add the argument, newObject, as an element of the receiver.  Put it
	in the position just succeeding oldObject.  Answer newObject."

	| index |
	index := self find: oldObject.
	self insert: newObject before: index + 1.
	^newObject
```

Verdana

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

Bookerly + 3

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

Inputsans

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

San Francisco

```
add: newObject after: oldObject
    "Add the argument, newObject, as an element of the receiver.  Put it
    in the position just succeeding oldObject.  Answer newObject."

    | index |
    index := self find: oldObject.
    self insert: newObject before: index + 1.
    ^newObject
```

Fonts You Might Enjoy Trying

- **Verdana**
  - Award winning, default in Cincom Smalltalk
- **Bookerly**
  - Crisp, easy to read
- **Input family**
  - Specifically for coding, best of proportional and mono
- **SanFrancisco**
  - El Capitan and iWatch font
- **Arial**
- **Century Schoolbook**

Questions?

Contact Information

Star Team (Smalltalk Strategic Resources)

- **Suzanne Fortman** (sfortman@cincom.com)
  Cincom Smalltalk Program Director
- **Arden Thomas** (athomas@cincom.com)
  Cincom Smalltalk Product Manager
- **Jeremy Jordan** (jjordan@cincom.com)
  Cincom Smalltalk Marketing Manager
- **Suzanne Fortman** (sfortman@cincom.com)
  Cincom Smalltalk Engineering Manager

Try Cincom Smalltalk

Evaluate Cincom Smalltalk:

- **try.cincomsmalltalk.com**

Join our Cincom Smalltalk Developer Program:

- **develop.cincomsmalltalk.com**