```
8       return seq_dict
9
10
11  Typography of Code
12
13
14  def distance1(s1, s2):
15      kmers_s1 = kmers(s1, k
```

```
if (bool == true) {
    // do something
} else {
    if (x == 1) {
        // do something else
    }
}
```

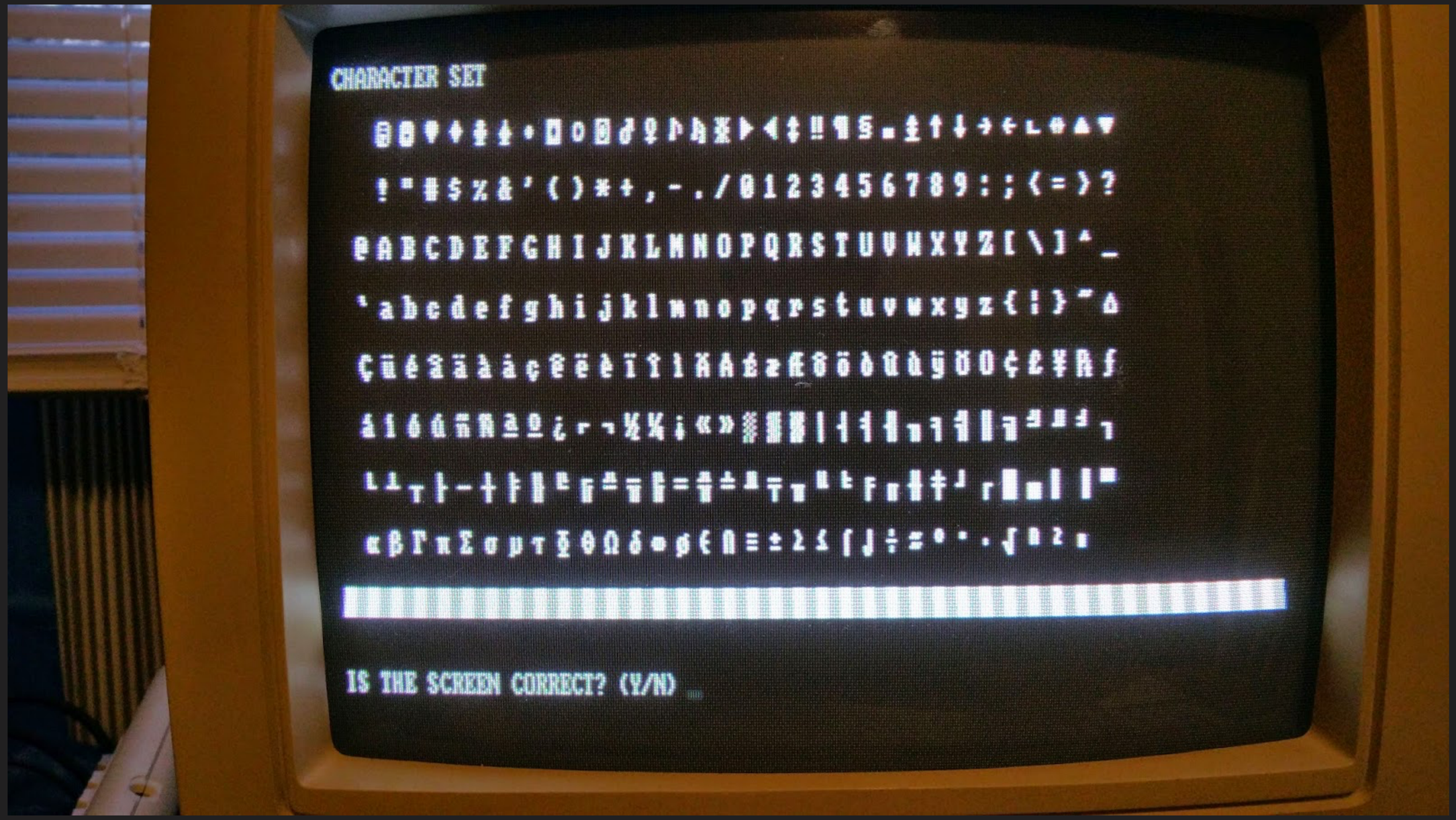
Helvetica Neue
Propotional

```
if (bool == true) {
    // do something
} else {
    if (x == 1) {
        // do something else
    }
}
```

Jetbrains Mono
Monospace

CRT Monitor & Bitmapped Font
Low-res LCD & Anti-aliasing
High-res Monitor

IBM PC 5150 (with CRT Monitor)

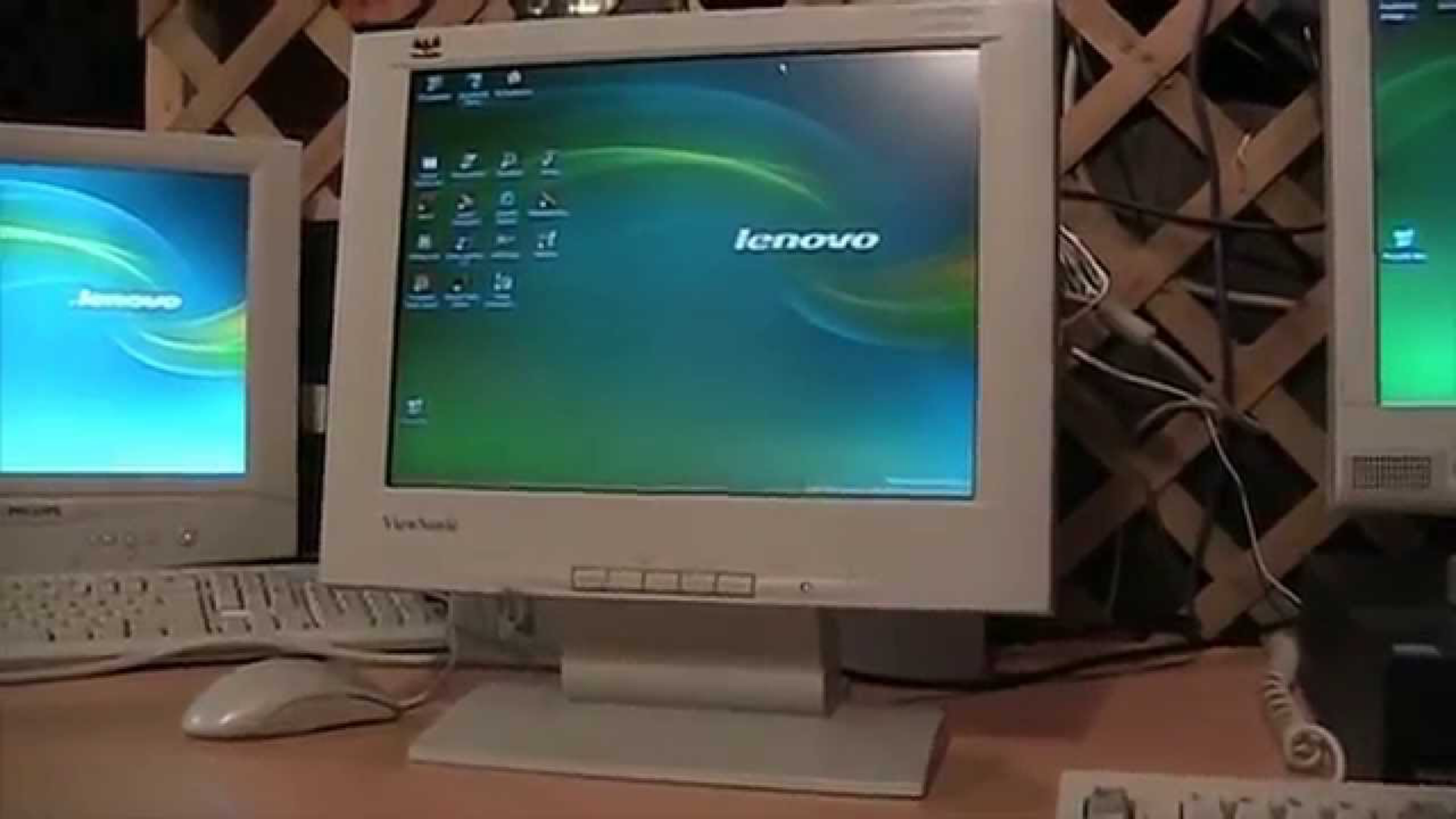

CHARACTER SET
IS THE SCREEN CORRECT? (Y/N)

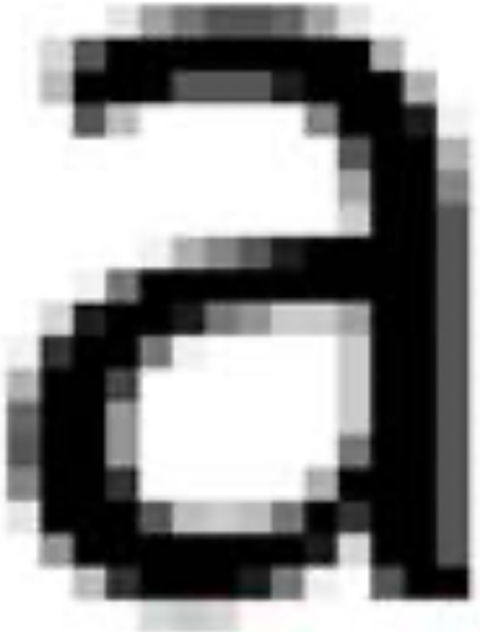
ed (text editor)

CRT Monitor & Bitmapped Font

Low-res LCD & Anti-aliasing

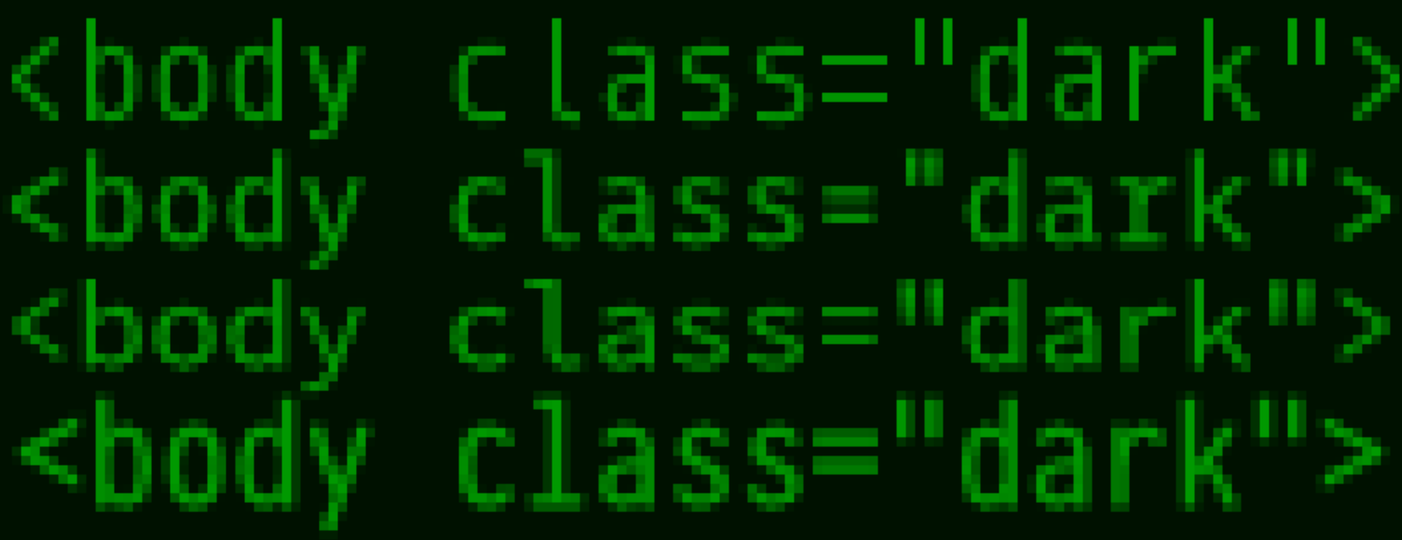
High-res Monitor


lenovo
lenovo



anti-aliased

bitmapped



Monoid, Fira Mono, Source Code Pro, Pragmata Pro (15px)

Designing a Coding Font - Andreas Larsen

Fira Code Ligature

CRT Monitor & Bitmapped Font
Low-res LCD & Anti-aliasing
High-res Monitor

```tsx
import React, {useState} from 'react';
import { SomeType } from './types';

type Props = {
    prop1: string;
    prop2: boolean;
    prop3: SomeType;
}

const SomeComponent: React.FC<Props> = (props) => {
    const [someState, setSomeState] = useState<string>('');
    return (
        <article className='someComponent'>
            <section>
                <h1>Cool new Cascadia Code font Microsoft!</h1>
            </section>
        </article>
    );
};

export default SomeComponent;
```

Dank Mono is about details and aesthetics. It doesn't compromise to optimise rendering at small font sizes. At that point coding can become an endless strain of looking at a lot of text as if it was far away. Instead it focuses on making all glyphs as delightful as possible as sizes you'll actually use.

What sets Dank Mono apart? - Phil Plückthun

IBM Plex Mono

```
# This is bogus code by Doug Wilson (from Toshi (
Typeface:IBM Plex Mono()
    except Exception as e:
        print("Export.py Error: %s" % e)
class ExportInDesignTaggedText( object ):
    def __init__( self ):
    windowWidthResize  = 100 1/2 ½ ¾ # user can 
        self.w = findfloat.FloatingWindow(
            "Export InDesign", # window title
# UI elements:
self.w.runButton = vanilla.Button((sp, sp*3+edY*4
    if sender.get():
        self.w.breakCheck.set(True)
```

Cartograph

```
# This is bogus code by Doug Wilson (from Toshi
Typeface:Cartograph()
    except Exception as e:
        print("Export.py Error: %s" % e)
class ExportInDesignTaggedText( object ):
    def __init__( self ):
    windowWidthResize  = 100 1/2 ½ ¾ # user can
        self.w = findfloat.FloatingWindow(
            "Export InDesign", # window title
# UI elements:
self.w.runButton = vanilla.Button((sp, sp*3+edY*
    if sender.get():
        self.w.breakCheck.set(True)
```

```
# This is bogus code by Doug Wilson (from Toshi (
Typeface:IBM Plex Mono()
    except Exception as e:
        print("Export.py Error: %s" % e)
class ExportInDesignTaggedText( object ):
    def __init__( self ):
    windowWidthResize  = 100 1/2 ½ ¾ # user can 
        self.w = findfloat.FloatingWindow(
            "Export InDesign", # window title
# UI elements:
self.w.runButton = vanilla.Button((sp, sp*3+edY*4
    if sender.get():
        self.w.breakCheck.set(True)
```

```
# This is bogus code by Doug Wilson (from Toshi
Typeface:Cartograph()
    except Exception as e:
        print("Export.py Error: %s" % e)
class ExportInDesignTaggedText( object ):
    def __init__( self ):
    windowWidthResize  = 100 1/2 ½ ¾ # user can
        self.w = findfloat.FloatingWindow(
            "Export InDesign", # window title
# UI elements:
self.w.runButton = vanilla.Button((sp, sp*3+edY
    if sender.get():
        self.w.breakCheck.set(True)
```

```
# This is bogus code by Doug Wilson (from Toshi
Typeface:Comic Code()
    except Exception as e:
        print("Export.py Error: %s" % e)
class ExportInDesignTaggedText( object ):
    def __init__( self ):
    windowWidthResize  = 100 1/2 ½ ¾ ⅝ # user c
        self.w = findfloat.FloatingWindow(
            "Export InDesign", # window title
# UI elements:
self.w.runButton = vanilla.Button((sp, sp*3+edY
    if sender.get():
        self.w.breakCheck.set(True)
```

```
# This is bogus code by Doug Wilson (from Toshi (
Typeface:IBM Plex Mono()
    except Exception as e:
        print("Export.py Error: %s" % e)
class ExportInDesignTaggedText( object ):
    def __init__( self ):
    windowWidthResize  = 100 1/2 ½ ¾ # user can 
        self.w = findfloat.FloatingWindow(
            "Export InDesign", # window title
# UI elements:
self.w.runButton = vanilla.Button((sp, sp*3+edY*4
    if sender.get():
        self.w.breakCheck.set(True)
```

```
# This is bogus code by Doug Wilson (from Toshi
Typeface:Comic Code()
    except Exception as e:
        print("Export.py Error: %s" % e)
class ExportInDesignTaggedText( object ):
    def __init__( self ):
    windowWidthResize  = 100 1/2 ½ ¾ ⅝ # user c
        self.w = findfloat.FloatingWindow(
            "Export InDesign", # window title
# UI elements:
self.w.runButton = vanilla.Button((sp, sp*3+edY
    if sender.get():
        self.w.breakCheck.set(True)
```

```
# This is bogus code by Doug Wilson (from Toshi
Typeface:Cartograph()
    except Exception as e:
        print("Export.py Error: %s" % e)
class ExportInDesignTaggedText( object ):
    def __init__( self ):
    windowWidthResize  = 100 1/2 ½ ¾ # user can
        self.w = findfloat.FloatingWindow(
            "Export InDesign", # window title
# UI elements:
self.w.runButton = vanilla.Button((sp, sp*3+edY
    if sender.get():
        self.w.breakCheck.set(True)
```

Why do you hate me?:-(

Recursive

By Arrow Type

Painter unknown; Brooklyn,
NY, near Myrtle & Wyckoff Aves

Recursive Casual Mono
Early sketches

```
8       return seq_dict
9
10
11  Typography of Code
12
13
14  def distance1(s1, s2):
15      kmers_s1 = kmers(s1, k
```