

Test-Driving a \$200 Coding Font: Operator Mono

2016

2 min read • 423 words

```
models.py — requests
models.py x
255
256 class PreparedRequest(RequestEncodingMixin, RequestHooksMixin):
257     """The fully mutable :class:`PreparedRequest` object,
258     containing the exact bytes that will be sent to the server.
259
260     Generated from either a :class:`Request` object or manually.
261
262     Usage::
263
264     >>> import requests
265     >>> req = requests.Request('GET', 'http://httpbin.org/get')
266     >>> r = req.prepare()
267     <PreparedRequest [GET]>
268
269     >>> s = requests.Session()
270     >>> s.send(r)
271     <Response [200]>
272     """
273
274     def __init__(self):
275         #: HTTP verb to send to the server.
276         self.method = None
277         #: HTTP URL to send the request to.
278         self.url = None
279         #: dictionary of HTTP headers.
280         self.headers = None
281         #: The `CookieJar` used to create the Cookie header will be stored here
282         #: after prepare_cookies is called
283         self._cookies = None
284         #: request body to send to the server.
285         self.body = None
286         #: dictionary of callback hooks, for internal usage.
287         self.hooks = default_hooks()
288
289     def prepare(self, method=None, url=None, headers=None, files=None,
290               data=None, params=None, auth=None, cookies=None, json=None):
291         """Prepares the entire request with the given parameters."""
292
293         self.prepare_method(method)
294         self.prepare_url(url, params)
295         self.prepare_headers(headers)
296         self.prepare_cookies(cookies)
297         self.prepare_body(data, files, json)
298         self.prepare_auth(auth, url)
299
300         # Note that prepare_auth must be last to enable authentication schemes
301         # such as OAuth to work on a fully prepared request.
302
```

Line 274, Column 24

As programmers, typefaces surround us — everything we do, build, manage, and orchestrate is typically encoded with a pleasant and comforting monospace typeface emanating from our console or editor of choice.

Which Typeface to Use?

I've invested a lot of time in optimizing my development environment and workspace — and typefaces are a large part of that.

Over the years, I've cascaded through a number of monospaced typefaces for coding, including— Monaco, MS Consolas, Inconsolas, Inconsolata-dz, and Ubuntu Mono.

[Ubuntu Mono](#) was my favorite monospaced typeface, used longer than any other, until I started using Operator Mono about six months ago.

The Hoefler & Co. type foundry released [Operator Mono](#) around this time—a very thoughtfully handcrafted typeface inspired by the pica-ruled days of typewriters. It is beautiful, and I have found it to be a pleasure to use.

Operator Mono for Programming

I find that different typefaces (as well as [color schemes](#)) both set the mood for my development environment, as well as offer utility in the way that I parse programmable text. While using Operator Mono, I found that I scan entire words as I read code more easily; while in other typefaces, my parsing style is often more letter-by-letter.

This observation could be written off as a subjective one, but I am all about the subjective, above all else — especially when it comes to my environments.

Feature: Italics in Script

The most unique feature of this font is that it includes an italicized typeface that uses monospaced script characters.

While perhaps appearing novel at first, I have found that using this (beautiful) script typeface for code comments has improved both the quality of tone and frequency of my code annotations — something which I highly value.

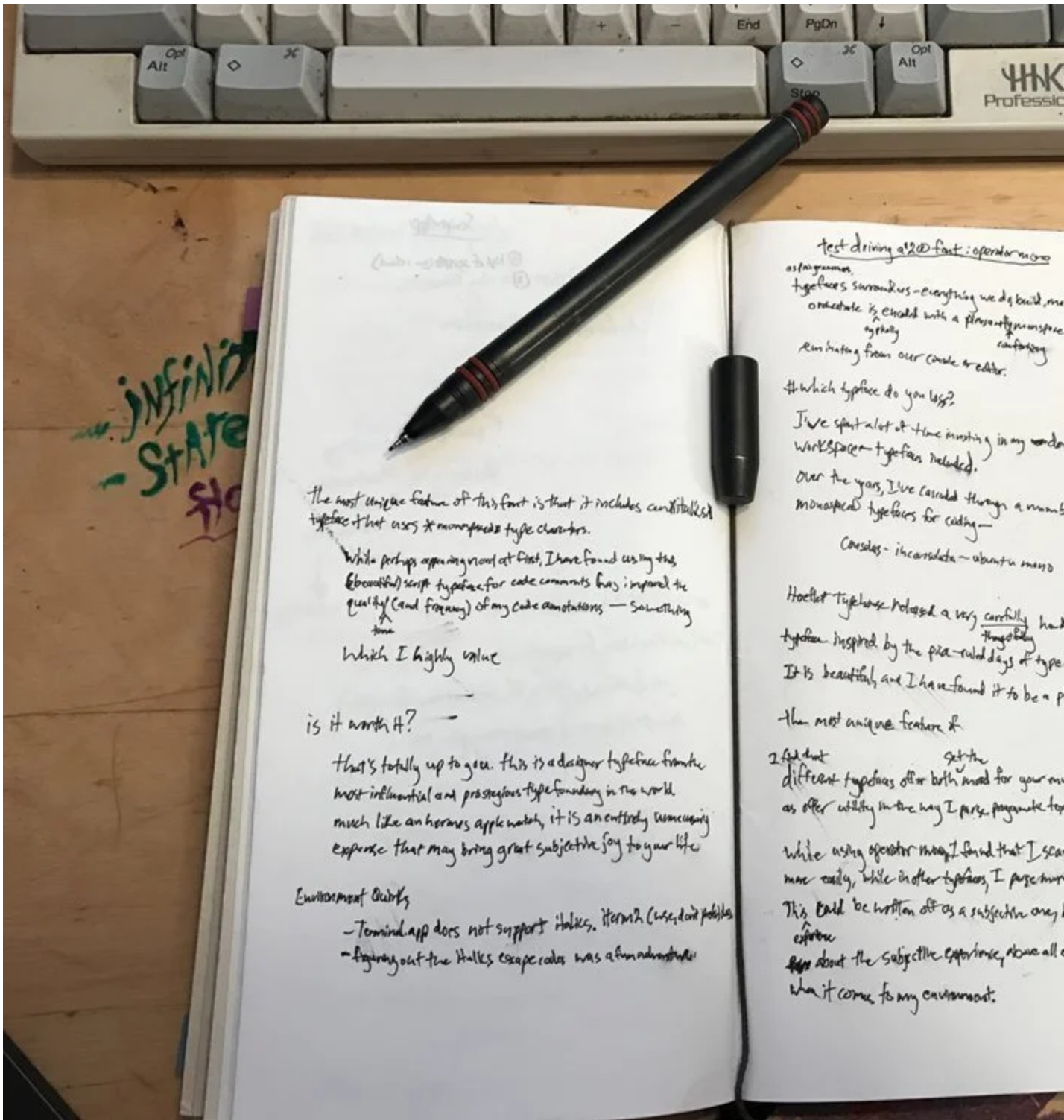
Is it worth it?

That's totally up to you.

This is a designer typeface from the most influential and prestigious type foundry in the world. Much like an [Hermès Edition Apple Watch](#), it is an entirely unnecessary expense that may bring great subjective joy to your life.

To me, that's all that really matters—part of understanding [programming as spiritual practice](#) where the tools and environment we choose shape our consciousness and creative expression.

***P.S.** This blog post was written entirely with pen and ink—surprisingly fun.*



Relevant Links

- [Introducing Operator by Hoefler & Co.](#)

- [Operator ScreenSmart Mono on Typography.com](#) (1 computer license)
- [pep8.org](#) utilizes this typeface throughout (code examples)

Bonus Integration Quirks

- Terminal.app (my favorite console) does not support italics. [iTerm2](#), however, does.
- Figuring out the [italics escape codes for zsh](#) was a fun adventure.

Generated from [kennethreitz.org](#) • 2026