§§ 1 Key pattern

Key principle: files should be named appropriately and within a standard:

--- organization is an investment of time but is amazing in efficiency gains)
human-readable and ordinally significant string, such as, all files are names(d) (actively/significantly) in consistent manner, identifying/dentifies changes and abnormalities passively.
Ordinal in the sense that the list is naturally sorting from oldest document to newest document, thus, even, though (in directory or file lists) editing a document does not entail changing its name, renaming does become a significant event, such and signifying a new step in a process, list (scored) alphabetically or in any other way which lack, alone, the assertiveness of control achieved by modern record keeping and version control systems.

4 rules

1. **Always Year first**
2. **Always lowercase see 1:1 (section below)**
3. **Never spaces** in filenames, never -- open spaces are replaced systematically with %20 or other entities and are a headache for developers and human readability generally, think urls: localhost:3000/page%20one
4. **Always separate items with - (single not --) - an ndash**
   2017-6-15-file-description.extension

§ 1.1 USE LOWERCASE for file names

Some computer systems operate case specific journaling, such as that, occasionally, a system exists, which, has case specificity extending to its UI, although many languages are-not case specific, such as email systems. Such as thus, it is never appropriate to use any uppercase letter for file names… Files should always be names lowercase : this removes any discrepancy between files on similar names, singles out outliers, and importantly, is a common feature signifying thinking and support, that is, is the first step in using modern, comprehensive, forward, and future looking style systems.

§ 1.2 File names:

Document names should start with dates, the exception is technical files, whose name are /become programmatically significant in a way that overrides the obvious utility of this naming convention: Files named correctly sort themselves meaningfully, in a way no other can provide (alphabetic, size, creation date [cannot account for true user documentation - **and quickly becomes irrelevant when uncoupled with a secondary feature in a jumble of documents, file type/extension.)** ← **explain?**

Alphanumeric alone is useless
(items can be sorted in alphabetically or ordinally- but not in reverse ordinary?)

§ 1.3 LIST ALL DATES YEAR MONTH DAY

There are 9 ways to list dates (combinatorial of 3 values)
3 accepted standards

Day month year small to big
year month Day big to small ← preferred style
Month day year ← strange American style

§ 1.3.1 General rules:

§ 1.3.1.0 Or, Use → . (single not ..) dots/period between items 2017.6.15.file.description ---> is a 100% valid and readable file name in a system. Dots can often signify important events, such as launching a rendering device for a .png or .svg or an application for a .psd (Adobe Photoshop)
Oddly, dots are acceptable in the file naming schema, where the last .object is always a type of extensions, exactly describing the 'action' for a system to take -- this can inform the user in some cases as well like .txt .README .YAML (ain't you markup language[developed as a specifically human) .pdf .doc .txt .png

§ 1.3.1.1 Change any .jpeg to .jpg by renaming the file (and wait for .jpg 2.0 which promises animation in this simple and progressively optimized visual formats.)

§ 1.3.1.2 Change any .PDF to .pdf (simply rename the file)

Remove any nonsense characters besides 0-9 ordinals, periods dash or underscore _

Remove any parentheses ( ) question marks ? or ! (even if the authoritative source for the document contains a non acceptable character such as.

Non acceptable characters break programmatic operations, are difficult to parse, or odd and difficult to read in rules, do not conform to major naming conventions, do not sort naturally

§ 1.3.2

Where's Waldo?
by Martin Handford ←--*becomes :*
2007-4-10-wheres-waldo-martin-handford

Who's Afraid of Virginia Woolf?
by Edward Albee

2006-8-1-whos-afraid-of-virginia-woolf-edward-albee

Do Androids Dream of Electric Sheep? #1
by Philip K. Dick

1996-5-28-do-androids-dream-of-electric-sheep-philip-k-dick

"Crazy" Therapies: What Are They? Do They Work?
by Margaret Thaler Singer

1996-9-27-crazy-therapies-what-are-they-do-they-work-margaret-thaler-singer

---

1996-5-28-do-androids-dream-of-electric-sheep-philip-k-dick
1996-9-27-crazy-therapies-what-are-they-do-they-work-margaret-thaler-singer
2006-8-1-whos-afraid-of-virginia-woolf-edward-albee
2007-4-10-wheres-waldo-martin-handford

<--now i can see commonalities about these 'files' without looking at them

human-readable and ordinally significant string, such as, all files are cnames (actively/significantly)in consistent manner, identifying changes and abnormalities passively. Ordinal in the sense that the list is naturally sorting from oldest document to newest document, thus, even, though (in directory or file lists) editing a document does not entail changing its name, renaming does become a significant event, such and signifying a new step in a process, justifying a new document version or simple moving the 'date via file name' to a new place in time, which is perfectly appropriate, what is inappropriate is scoring list alphabetically or in any other way which lack the assertiveness of control achieved by modern record keeping and version control systems,

(so knowing these files are likely hopefully, subject to versioning which renders this obsolete? no! files should be named appropriately and within a standard -- regardless of a degree of fault tolerance simpla a backup is not, organization is an investment of time but is amazing in efficiency gains)

productivity is valuable ---> so the focus is human readability not versioning, requires human thinking, thus raising the level of competence, practice, as promotes 'humans' not brute force productivity --> such as dealing with machine generated names.

§ Human readability, a personal style guide 2017-6-15

§ DOM Structure:

Many systems, including pollen.computer uses .xml
"XML is a software- and hardware-independent tool for storing and transporting data."
XML, the eXtensible Markup Language, is not HTML! (XML is self-descriptive)
XML is recommended by the W3C (quoted above)

The DOM structure includes this .xml reference material to primary APIs, such as JavaScript, the Browser, and the DOM.

§§ SECTION 2 api (application program interface)

Guide: Distributed Services Across Top-Level Entities as Value and Key pair(d) Methods

This project provides a derivative syntax built inherently on dot notation and other standard language structures, ordinal sorting, alpha-numeracy, journaling protocols, and is a unification of certain of those language structures and is a simplification of those, a reduction in syntax, for the purposes of human readability.

Since this (as a) pattern is not only predictable, the syntax (variation) arbitrary, we can/could represent this via any simple constructor, like grammar syntax:

object : reference : action:

Such as that, the pattern that objects exist prior (as in notationally a representation semantically) is a simple replicable structure is only as unique as its values.
The availability of the . name value is extremely limited, only when correspondence within a system, of meaning of key pair values (where) are likewise dot notation object locations; and
The repetition of the . ([dot] value is distributed across nodes, where individual nodes such as the root node:contain individual / unique values.) And likewise there are common variables (distributed similarity between unique variable objects) and a unique distribution of expressions between values and variables as similar variables across a likewise (or, unlike) set of values.

(As far as I can tell this is a completely unique and un used method, the availability of which corresponds solely on the proximity to certain top-level domain objects.) An API to use these to create sophisticated methods, because is is easy for the user to understand the .computer or .email value over the mostly 100% arbitrary .com value.

I will show distributed work can be done across a unique system of urls (uniform resource locator) I will describe:
And I will **argue *the efficacy of *any system is only as good or as usable as its key pair parameters makes sense to the user and constitute a legible and productive pattern of objects,* methods, attributes, values and actions. And is, in this sense, extendable due to the robustness of the key terms whereas (also) the pattern has high fecundability: the meaning, via the obvious and usability of the pattern is not: that (which) makes this API unique as well as writeable in a more meaningful way: (accounting for the rarity of the for which no more available meaningful patterns exist, such as for the availability of the . name value, e.g., .glass value. For-and (all) this makes our code more readable, more obvious and usable, and I believe (is) conforms to the spec, why else is their object specified in the available top level domain objects if it is not to increase the productivity of our protocols and make code more readable.**

EXAMPLE: So if .bicycle is an available top-level domain object, and John creates john.bicycle, an app tracking data on his bicycle, he can… Thanks to available domain objects and a little thinking.

So it (this project) is not a computer that is distributed over the internet, but it is a computer that *runs on the internet via protocols between languages rather than a computer that is distributed across the internet, and using available languages: No it uses available standards, which are protocols distinct between the operation of languages, other languages and often other protocols, thus this constitutes a set pattern of such and a specificity of protocols (as object literals).*

§ Dot Notation ~ highly open parameters accept

Standard dot notation uses the space before the dot as a location/space parameter.
Simple rule: for X1 .x2 [X1] is always a variable
X1.x2 [x2] is always a value

[.] is referred to "'a' contextually" to as 'object dot' 'dot to' and is a location provider forming the basic and universal and non unique or generally arbitrary pattern of thisTing.js [<--- ([here]the naming convention is pretty standard but the implementation is

arbitrary, as no protocols exist/s (knowingly) or could exist until today where the object dot notation expresses a variable like the standard pattern: Property:value or key:value references up and down a hierarchy of meaning) --->]

§ Overview of API Structure:

A hierarchy of nodes exists so (as) object literals
as is an ordinance of methods + objects and aliases
is the cache entity validation with permissions to run protocols providing synchronization services and data integration (across the suite), also handles the validations and protocols of the intention. methods, objects and actions.

The names of the API correspond to a .reference or a reference dot notation that is conditionally in a system of methods and objects, a pattern, and as system themself, each a pattern or specifically a set of libraries at the host level, available as a service via rendered as a secured unsecured local service service (browser) or as a secured service (API [application program interface]) .

It's not a new language but it's an API, with a set of methods and objects, which correspond to a pattern which are top level domain entities, whose names are function carriers distributed across that pattern of objects and methods, constituting their organization, functional, programmatically and importantly, the peer wise and hierarchical locations of each API into a body of documents and protocols, all accessible via a standard pattern of
object (use (and +) this API methods and a .object or method. notation), is
(chainable syntax). {depending on the operator's set parameters}
.object or object.
.method or method. expression syntax (can observe component crossover properties such as data migrations or asynchronous operations. These properties, are namesake: (they) restructuring enriching web protocols, open source is our end goal, completely. *Organization is our specialty.*
Pollen programme will:
Where possible I simplify by concatenating into a single reference entity appendable to any DOM as a JavaScript file which will extend into the necessary structure by appending static or dynamic resources like stylesheets and additional .js or .xml resources, creating a regular pattern.
note:
It is important to keep some domains open, rather than excluding them from the foundation methods, various other objects with various values representing sets of attributes

§ Argument for a year | month day |<u>not</u>| day | month year record:

And a year.month record. technically and usably, (year.month records are the most readable, understandable and maintainable, because crucially…. Discrepancies become more obvious to a user, they conform with alpha numeracy (as a principal sorting among and between files, databases and hierarchies of directories. [conforming to best practices]).
The method is Indiscriminate vs discriminate: So it can always be obvious via a collection of records: and mistakable on instance, with discriminate records and either method can be determined via a collection of work so seeing 23 appeal is the second term indicates it is not a month term: but the specificity of year to month over year to day is high. So, year.month notation is more highly specified than year.day notation via ordinance, thus the (a) list automatically structures in an a-z pattern, or, a name based pattern where a base ten to alpha notation is apparent ordinally:

1 > a | and | 1 > 2 . Thus, 2017 < 2018 in ordinance and this or
Or,.e.g., 2017.12.9.info.txt is always < 2018.12.9 or 2018.1.info.text or 2018-1-1.info.txt.

Thus this programme is future proof/ oriented: new patterns and methods will exist and will be compatible, either reinforced by or apparent to this numarancy. (A basic record keeping method relying on named entities not other records, this improves readability, usability, eases identifying outliers (,and is) obvious compared to other methods where there is more and higher (in) indeterminacy [within a set of given objects].
Between a set [of given objects], numerated year.month, not year.day, will be more ordered. (for objective comparison with a tertiary) such as that one set of records ordered by one (pretend unknown method like ordered alphanumerically name alone)

Some computer systems operate case specific journaling, such as that, occasionally, a system exists which has case specificity extending to its UI, although many languages are-not case specific, such as email systems. Such as thus, it is never appropriate to use uppercase letter for file names… Files should always be named, and have names in, lowercase : this removes any discrepancy between files on similar names, singles outliers, and importantly, is a common feature signifying thinking and support, that is, is the first step in using modern, comprehensive, forward, and future looking style system. REPETE

File names:

Document names should always start with dates, the exception is technical files, whose name became programmatically significant in a way that overrides the obvious utility of this naming convention: Files named correctly sort themselves meaningfully, in a way no other can provide (alphabetic, size, creation date [cannot account for true user documentation - and quickly becomes irrelevant when uncoupled with a secondary feature in a jumble of documents, file type/extension,) alphanumeric alone is useless
(items can be sorted in alphabetically or ordinally- but not in reverse ordinary?)

There are 9 ways to list dates (combinatorial of 3 values)
3 accepted standards

Day month year small to big
year month Day big to small ← preferred style

Month day year ← strange American style

Never spaces in filenames, never -- open spaces are replaced systematically with %20 or other entities and are a headache for developers and human readability generally, (or machine learning think urls,

Always lowercase

Always Year first

Always separate items with - (single not --) - an ndash 2017-6-15-file-description.extension

Or, Use → . **(single not ..) dots/period between items** 2017.6.15.file.description is a 100% valid and readable file name in a system. Dots can often signify important events, such as launching a rendering device for a .png or .svg or an application for a .psd (Adobe Photoshop)

Oddly,
Dots are acceptable in the file naming schema, where the last .object is always a type of extensions, exactly describing the 'action' for a system to take -- this can inform the user in some cases as well like .txt .README .YAML (ain't you markup language[developed as a specifically human) .pdf .doc .txt .png

Change any .jpeg to .jpg by renaming the file and wait for .jpg 2.0 which promises animation in this simple and progressively optimized vis formats.

Change any .PDF to .pdf (simply rename the file)

Remove any nonsense characters besides 0-9 ordinals, periods dash or underscore _

Remove any parentheses ( ) question marks ? or any punctuation besides dots! (even if the authoritative source for the document contains the character such as

And a year.month record. technically and usably, (year.month records are the most readable, understandable and maintainable, because crucially…. Discrepancies become more obvious to a user, they conform with alpha numeracy (as a principal sorting among and between files, databases and hierarchies of files. [conforming to best practices]).

---------- ---------- ---------- ----------

Best practices are not just some arbitrary thing for coding, no they are shorthand for usable and governed by discrete specification under which and for usability works and is is meant to work/ be used. And this (all , above) specification conform still to implementation. Thus best practices are **the good conflation of usability, principles, and implementation/support.**

This is principally not governed by technology and humans but is for the terms: usability and support: Human readable and ordinally significant strings, such as, all (new) files are name(s/d) (actively/significantly) in consistent manner, identifying changes and abnormalities passively.

(this type is,)
Ordinal in the sense that the list is naturally sorting from oldest document to newest document, thus, even, though editing a document does not entail changing its name, renaming does become a significant event, such and signifying a new step in a process, justifying a new document version or simple moving the 'date via file name' to a new place in time, which is perfectly appropriate; what is inappropriate is scoring list alphabetically or in any other way which lack the assertiveness of control achieved by modern record keeping and version control systems. (so knowing these files are likely hopefully, subject to versioning which renders this obsolete? No! ~Files should be named appropriately and within a standard -- regardless of a degree of fault tolerance simply a backup is not; organization is an investment of time but is amazing in efficiency gains). Practice (any of) simple things:

EXAMPLE (for instance): turn on your cell phone and point, (close eyes)take a picture with touch (alone), and ask yourself how intuitive that is, practice 50 times till done methodically. Conditioning generally delivers with API but critical knowledge is building or using.relying on intuition for the novice. But is rediscovering for the expert or master; and is rediscovery always for the creator.

productivity is valuable ---> so the focus is human readability not versioning, requires human thinking, thus raising the level of competence, practice, as promotes 'humans' not brute force productivity --> such as dealing with **machine generated names**

§ Human readability, a personal style guide 2017-6-15

Oddly,

Dots are acceptable in the file naming schema, where the last [dot]t .object is always a *type* of extension, exactly describing the ‘action’ for a system to take -- this can inform the user in some cases as well like .txt .README .YAML (*ain't you markup language* [developed as a specifically human readable format) otherwise, e.g., .pdf .doc .txt .png

~~~~~~~~

Change any .jpeg to .jpg by renaming the file
Change any .PDF to .pdf (simply rename the file)
Remove any nonsense characters besides 0-9 ordinals, periods dash or underscore _
Remove any parentheses ( ) question marks ? or ! (even if the authoritative source for the document contains the character such as ? # ‘ “ .

Where's Waldo? by Martin Handford ~ write as ~ **2007-4-10-wheres-waldo-martin-handford**

Who's Afraid of Virginia Woolf? by Edward Albee ~ write as ~
2006-8-1-whos-afraid-of-virginia-woolf-edward-albee

Do Androids Dream of Electric Sheep? #1 by Philip K. Dick ~ write as ~
1996-5-28-do-androids-dream-of-electric-sheep-philip-k-dick

"Crazy" Therapies: What Are They? Do They Work? by Margaret Thaler Singer
~ write as ~ **1996-9-27-crazy-therapies-what-are-they-do-they-work-margaret-thaler-singer**

(the same list as above: ordered a-z [below:])
1996-5-28-do-androids-dream-of-electric-sheep-philip-k-dick
1996-9-27-crazy-therapies-what-are-they-do-they-work-margaret-thaler-singer
2006-8-1-whos-afraid-of-virginia-woolf-edward-albee
2007-4-10-wheres-waldo-martin-handford

<--now one can see commonalities about these 'files' **without (thinking)** looking at them
(when a date is unknown ALWAYS use today current date)

§§ SECTION 3 API STRUCTURE AND REZ USE CASES

Top-level tier: is open as a resource, i.e., is a repository of its own documents, it servers secure resources to clients via the https:// protocol (http over ssl)

Mid-range tier: protocols for each API work as token of the services offered by that API and these create its methods by virtue of their own object literals used in expressions. The resolution of each api parameter depends on the use case: the mid rage use is as token operators in a distributed system.
~~~~~~~~

Deep tier: the deep tier API use case is for pollen.computer specific process', such as that require the security.flowers API to function like the composer.computer method in when deep tier use of the composer.computer api method is needed.

Each API is completely open-source, under the MIT license, and free to use.

Users may wish to engage with the project in various ways such as going to the domain locations in the browser (top), reference their static or dynamic content like scripts,(mid) use the set of methods via the object method pattern, use the deep tier of any or all of the APIs to build a new pattern or may create a completely unique arrangement and use. (top)

§ Recap of general rules:

1. **Never spaces** in filenames, never -- open spaces are replaces systematically with %20 or other entities and are a headache for developers and human readability generally, think urls,
2. **Always lowercase**
3. **Always Year first**
4. **Always separate items with - (single not --) - an ndash**
   2017-6-15-file-description.extension
5. Replace word 'the' with 'today' where possible

§ Glossary:

Object = collection of association, (key:value)
Method = key:(function(value))
associations, key to value, are object properties

XML:
"XML is a software- and hardware-independent tool for storing and transporting data."
XML, the eXtensible Markup Language, is not HTML! (XML is self-descriptive)
XML is recommended by the W3C (quoted above)

YAML:

Directory: is another term for a folder containing files or other directories. Synonymous with folder or (indexable) file.

DOM: document object model, is the concept that a program is a collection of documents, arranged semantically and often in a hierarchy, allowing for distributed and pattern distribution and generation across the network of elements and documents that contain elements; DOM modeling also entails a top down approach to readability as functionality such as thus documents and elements become the DOM in the order that they appear ordinally as in the lowest level is the top of the document and the 'highest' or most applicable elements are often lowest in the document, thus are read programmatically by the machine after other elements containing operations are read and sometimes executed , in order first to lst or in oper words, in their apparent order top to bottom. DOM technique is a sophisticated modeling system to achieve better understanding over previous techniques such as MVC (model view controller).

UI : user interface such as a keyboard, visual programme, a book or a faucet is a user interface

Internal citation format:

§ § See **para. 3**