

Other

Tech Skill Day – John Papa

The less you write the less code can break.

Tech Skill Day – Deborah Kurata

Every one of us has a superpower which might be different from others. You can improve yourself with things you aren't good at with skills, but your real strengths are where your superpower lie.

All Hands on Tech: A Conversation with Jason Alba

If you do well in your current company, you can still be get fired. So, always be prepared for worst case scenario.

We need to better in communication it can be written, one on one and group. By we can grow in carrier a lot faster.

All Hands on Tech: A conversation with Dan Wahlin

Don't say don't use this technology, because in reality for your team this might be the best technology based on their skill sets.

We should use Kubernetes for containerization.

If you bored in the tech world, you are just not trying hard enough. Everyday there is a new thing.

Fluid is a free and open source computer platform for real-time collaboration across applications.

All Hands on Tech: A Conversation with Filip Ekberg

Microsoft is working on dot net one.

Instead of watching the Netflix code just go read other people code repository or courses from pluralsight.

We should write blogs, upload videos on youtube.com or on twitch.

All Hands on Tech: A Conversation with Julie Lerman

Domain driven design is to design the application based on the business or domain.

If something doesn't work there is two ways have something just to tell you how to do it right or go and understand what is going on, why it is not working and the way it should be working, what is the difference.

All Hands on Tech: A Conversation with Janani Ravi

Now a days to build a machine learning model we don't need coding experience, it has become so simple to create that we just need to do drag and drop.

We should read technology documentation, they might be dry and confusing, but when we learn how to read documentation properly, we will find all the answers, we just have to interpret them.

All Hands on Tech: A Conversation with Katherine McNamara

To learn new technology, we can do combination like videos, reading like training books and labing. Labing is the most important and taking the notes.

All Hands on Tech: A Conversation with Ned Bellavance

We should attend various seminars or conferences to meet like-minded people.

All Hands on Tech: A Conversation with Ross Bargurdes

Making mistakes helps us to learn what you should never do and what you should always do.

If this pandemic has happened 10 years ago then business will get worst hit, because of advancement of technology it didn't impact that much and people has been shifted to WFH very easily.

May things like MAC address, Ethernet frames haven't been changed much even after 50 years.

In IT, it often feels like somebody else always knows more than you, but it is 100% okay to tell it's okay, I never heard of that topic. Just asked them the details about this new technology.

Engineers used to give complex names for the technology but under the hood it is very simple and have same base concept.

Use Wireshark to know how a protocol is behaving internally.

All Hands on Tech: A Conversation with Shelley Benhoff

Many people got into programming or developer from Help Desk by showing that they are interested in programming. Communication is very important.

All Hands on Tech: A Conversation with David Gotrik

To keep updated with technology updates we should read newsletters and journal articles.

All Hands on Tech: A Conversation with Jeffrey Hicks

PowerShell is very useful while working with Cloud. It is core technology that people need to know.

Version 6 of PowerShell has gone cross-platform! That means you can run PowerShell on both Windows and Linux.

All Hands on Tech: A Conversation with Jordan Morrow

Data literacy is an ability to be comfortable and confident in using data. It is an ability to read, work with analyzing, communicate with data.

The end goal of data literacy is to help people be empowered to make a decision using data.

Now a day, the technology and hardware are not the bottlenecks but the people skillsets.

All Hands on Tech: A Conversation with Xavier Morera

Machine learning is one of the things that has come to change how we solve a problem.

Interview with Troy Hunt

SQL injection, configuration misconfiguration are main security problems.

Interview with John Papa

People of who are very organized want a structured path kind of laid out before them with before them with everything available as the go through, angular really appeals to them.

Skill transfer is very easy in angular, as most of the angular projects looks the same.

When we choose a topic what we want to teach, choose one thing and teach that one thing really well. As opposed to teaching 20 features.

Interview with Deborah Kurata

When you use angular, you are getting everything like route, forms package, http package, there is a kind of prescribed way to code with angular but products like react are much more pick and choose, so you can choose which router you want, which http package you want to use.

Windows 11 First Look

The corners are more curvy.

Widgets are also back from windows 7.

New snap layout groups feature.

Teams 101

When you create a team it will also create a SharePoint page and folders for each channels.

We can directly create the meeting from Teams instead of just from outlook.

Chats is different from Posts, posts if one to many but chat is more of one to one.

We can filter chats by unread, during meeting or muted.

The Low-down on Low- and No-Code Development

By low-code we can deliver better software faster. One way is to abstracting automating things so professional developers can go faster. It also takes the constraint away from people who can only build that and it open an ability to adapt to this new world in real time with people who has no traditional skills in software development. It provides speed, simplicity, no huge army of engineers and get live quickly and save tons of the cost to the business, anybody can develop software, business and IT will be on same page by providing common language, has guidance and guard rails, no long red tape of processes.

Power to the Programmers

The team velocity gets increase for technologist if they stay longer in the company because of better communication and understanding of process activity.

Earlier the tech leaders or managers used to be commanders but now things has changed it become more empathic and open where you can share your personal and health issues more freely.

[Book] The Software Craftsman by Sandro Mancuso

It is the pain of feeling constantly constrained to do a poor job. It is the pain of feeling trapped in an unprofession. It is the pain of wanting to do better and not knowing how.

Can you imagine how hard it would be to understand the code if everyone decided to show off how smart he or she is?

How it is done is as important as getting it done.

Many developers still behave like unskilled, unmotivated workers, providing very poor service to their employers and clients.

I promised to myself that I would only choose jobs and positions that would make me excited to wake up in the morning and go to work.

Companies are demanding professional software developers and not just cheap coders who do whatever they are told to do.

The faster we demo a screen or any other feature to a user, the earlier we will get feedback on it.

Agile transformations were focused on process but no on technical disciplines, which means that they did very little to make developers better.

It is never too late to make things better – it just becomes far more painful when you delay it.

We need developers who can deliver software that is fully tested and that can be easily changed.

Software development should be qualified as art, craft, trade, engineering, or science.

Well-crafted software means that , regardless of how old the application is, developers can understand it easily. The side effects are well known and controlled. It has high and reliable test coverage, clear and simple design, and business language well expressed in the code.

We should always leave the code cleaner that we found it.

Writing blogs, contributing to open source projects, making our code publicly available, becoming part of our local communicates, and pairing with other developers are some of the ways we can contribute to the greater good our industry.

Developers that say it is not their job if the task is not code-related are not real software craftsmen.

Knowing how to select clients (or employers) is also an essential skill for a craftsman.

Software craftsmanship is a mindset – a lifestyle that many professional developers adopt. Software craftsmen live and breathe software. They see software as a craft and are committed to do whatever they can to master their craft.

If we think that a piece of code we wrote some time in the past is still good enough today, it means we didn't learn anything since.

Sharing our knowledge with less-experienced software craftsmen is our moral obligation.

We should use own time and money to get better at what we do. Developers who rely only on their companies to provide them knowledge are not professional software developers. They are just factory workers in disguise. On the other hand, factory workers, for example, rely on training. Factories need to train their employees to use new machines so they can do their mechanical repetitive work well.

All software developers should have their own blogs, regardless of how much experience they have. We should treat our blog as a record of our own learning and progression.

When practicing, we need to focus on writing the best code we could possibly write.

The great thing about passionate developers is that they are constantly learning and are very happy to share what they know.

Lunchtime is another great opportunity we have to practice and learn something new. Once or twice a week just grab a sandwich and your laptop, and go somewhere quiet.

We should do whatever we can to finish the pomodoro with no interruptions but in case it needs to be interrupted, then the pomodoro must be terminated and not paused.

When we say yes, people take that into account and make plans based on it. Our bosses will make promises to their bosses, other teams, customers, or partners based on what we said. Not being honest and transparent may cause huge damage to the entire team company.

Always saying no is also not a professional attitude. Every no, ideally, should be followed by a list of alternatives. Remember the story of getting orange by two different persons with different approach.

Managers should be harmony ambassadors and it is their job to keep the team healthy, happy and united.

Craftsmen are gardeners. They are constantly looking after the code base, quickly refactoring it without fear – they are backed by a good battery of automated tests that can test the entire application in just few minutes.

We should reduce the technical debt as we go along and not create more.

We should make sure that whatever code we write can be easily tested and deployed.

Having five years of experience is different from having five times one year of experience.

For a great developers, their job is not just a job; it is also their hobby and their passion.

When a software craftsman makes recommendation, it is implied that the recommended developer is also a software craftsman, and that he or she shares the same passion, values, principles, and dedication.

Supporting user groups and technical communities is a great way to attract and approach great developers.

Good developers are also filtering bad companies out and looking for the best ones.

If the interviewee asks loads of questions, we can assume that she really cares about her career and finding the right company.

A good interview is like a good and informal chat between passionate developers. It is an exchange of information: a good debate about techniques, tools, challenges, and approaches to software development.

Asking candidates to submit some code before the face-to-face interview is a great way to preselect candidates. I prefer to give candidates an interval of time that contains at least two weekends.

Good developers don't hire bad developers. They will try to find developers who are even better than they are.

Don't try to look smart by asking tricky and irrelevant questions.

Brainteasers are a total waste of time.

Don't conduct phone interviews, always favor a face-to-face interview.

When things are not going well, the best developers are always the first ones to go.

We started sharing blog post links, videos and books.

The best person to motivate a developer is another developer.

Encourage pet-project time – passionate developers love to have a playground where they can try many new ideas, technologies and techniques.

Just keep a healthy balance between your learnings activities and the things you would deliver. Be sensible.

The ability to communicate well is key if you want to have your ideas accepted.

As a developer, the best way to build trust is by consistently delivering quality software.

People are far more inclined to follow someone who is passionate about something that someone who doesn't exhibit any passion.

A one-man army won't win a war, so choose your battles carefully, one at a time.

Only incompetent people fear losing their jobs.

It is easier to ask for forgiveness than to get permission.

You cannot say that a coding task is done if you still have a pending "unit test" for that code.

If you want to bring the boss to your side, speak her language. Don't bring up developers problem. Raise the conversion to her level. Improving productivity, reducing cost, increasing revenues, meeting deadlines, reducing the number of bugs, keeping a steady and predictable velocity, and satisfying business requirements are the things the boss will care about.

If you want to be responsible, be prepared to be accountable.

For a company that usually aims for average code, more often than not, average means very poor.

Clients may decide not to pay for quality and ask for a quick and cheap solution, but deep inside, they will always be expected quality and won't be happy if they don't get it.

Passionate developers love writing code. They enjoy creating complex solutions in order to keep themselves interested in the job.

Well-crafted code is simple, small, testable, easy to understand, and most important, does the job.

One of the best ways we have to help business to achieve their goals is to be able to change the code almost as quickly as they change their minds.

When dealing with craftsmen, clients should never pay more for quality.

Software developers are essential for the evolution of the world we live in.

Software craftsmen are humble, always ready to learn from more-experienced developers and eager to help the less experienced.

Every craftsman is a developer but not every developer is a craftsman.

Using the Chrome Developer Tools

Shortcut for developer tools – ctrl + shift + i

Shortcut for JavaScript console – ctrl + shift + j

console.error() method also gives us call trace stack where the error is actually occurred. We can use filters from console tab to view only specific type of log messages. Use clear() method to remove previous log. We can also use console.assert() method to do some sanity check.

Using grouping for logging

```
function FirstStep() {  
  console.group("First Step")  
  console.log("Finding ants");  
  console.log("Building ant farms");  
  console.log("Feeding ants");  
  console.groupEnd();  
}
```

Using formatting using %s, %i, %c, we can even apply styles using console:

```
console.group("%cFinishing part", "font-size: x-large")  
//.....
```

Logging DOM object:

Note: Use `console.dir(document)`, to get the JavaScript notation from HTML DOM view.

Use `console.time()`, `console.timeEnd()` to calculate the time difference.

Console has two parts – console API and command line API. Command line API is like a JS interpreter, it also has a support of selection engine for selecting DOM elements, also helpful monitoring events.

We have three type of selectors in command line API, `$$` gives us collection of all matching elements:

- `$(selection)`
- `$$ (selection)`
- `$x(xpath)`

Practice – using selectors from command line API

We can use inspect method with selectors, which will takes us directly on the element tab on that matching element.

Monitoring events – for this we need to use the `monitorEvents()` method with selectors:

Event type	Corresponding mapped events
mouse	"mousedown", "mouseup", "click", "dblclick", "mousemove", "mouseover", "mouseout", "mousewheel"
key	"keydown", "keyup", "keypress", "textInput"
touch	"touchstart", "touchmove", "touchend", "touchcancel"
control	"resize", "scroll", "zoom", "focus", "blur", "select", "change", "submit", "reset"

For example, the following uses the "key" event type all corresponding key events on an input text field ("msg").

```
monitorEvents($("#msg"), "key");
```

We can modify any parts of DOM elements from element panel.

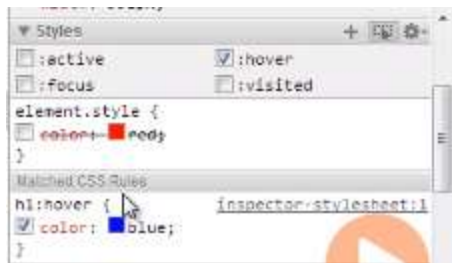
While viewing the elements we can refer the element tree to go from back and forth:



To select a previously selected/inspected element with a console command, we need to use `$0`.

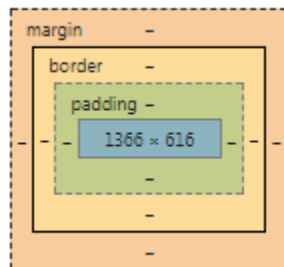
When we disable a style on an element from style section, it just do not disable it for that element but for the elements which have referred that style from that CSS file.

Using pseudostate styles - for this we need to use toggle element state option like below, it adds those styles temporary in inspector-stylesheet:

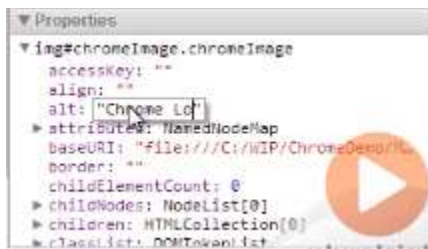


Practice – sing pseudostate styles

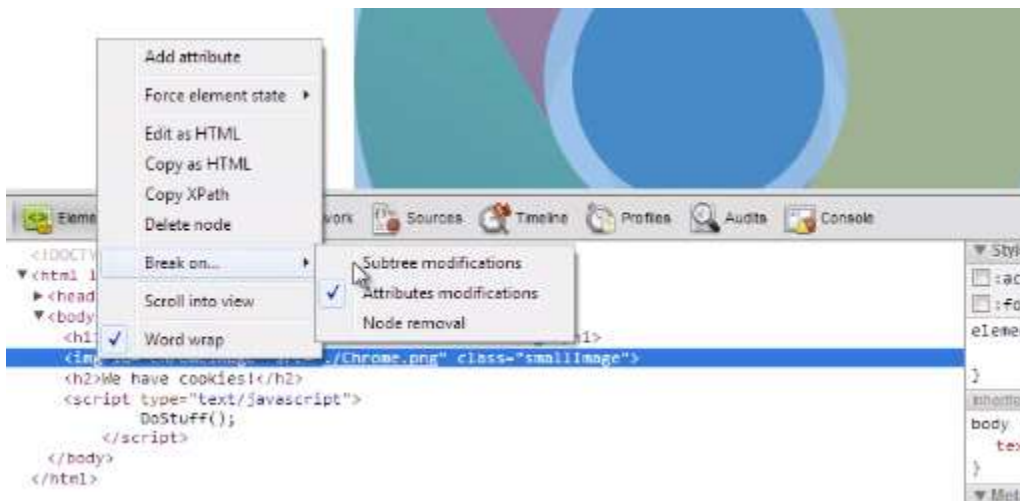
We can directly change the values from the Metrics section:



We can also use the properties section, which shows elements like a JavaScript object properties we can even modify these properties on runtime:



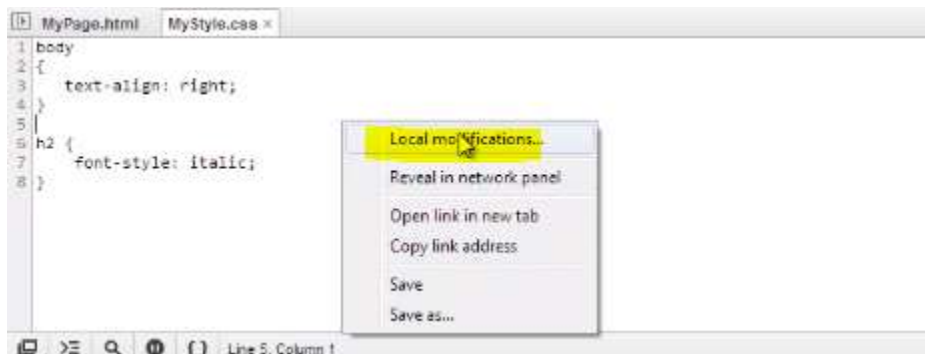
Setting breakpoint in element panel:



Practice - using breakpoint in element panel

We can use event listener section to view the event details.

Saving local modification – we need to do right click and choose Local Modifications option:



Network panel shows how resources like local storage, IndexedDB, HTML pages, images, scripts, style sheets are exactly loaded. Resources tabs shows the data which our application is using.



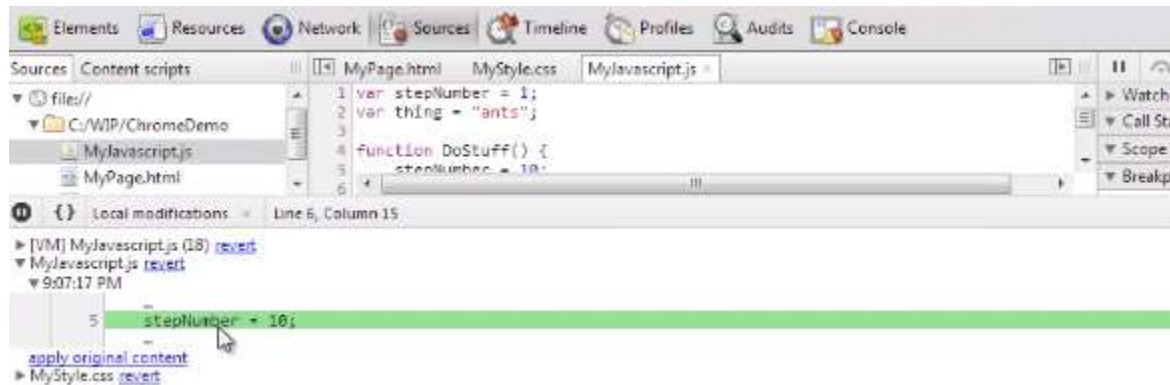
Using application cache we can serve our pages offline as well.

Network panel – Reading the timeline – we can even see the initiator which requested that resource. From size column we can see the size of that resource or whether it is get loaded from cache. And a line when DOM content ready event got fired and then another line for load event fired. We can use different type of sorting on this columns to understand the data more clearly:

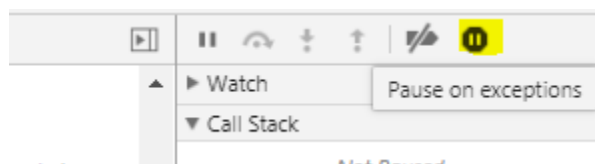


Audit tab is useful for getting some tips for improving the performance of our web pages or correcting other issues.

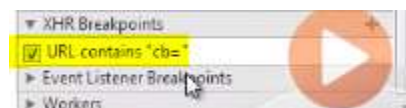
We can update the JavaScript on the fly from the Source panel, and see the local changes which we have done:



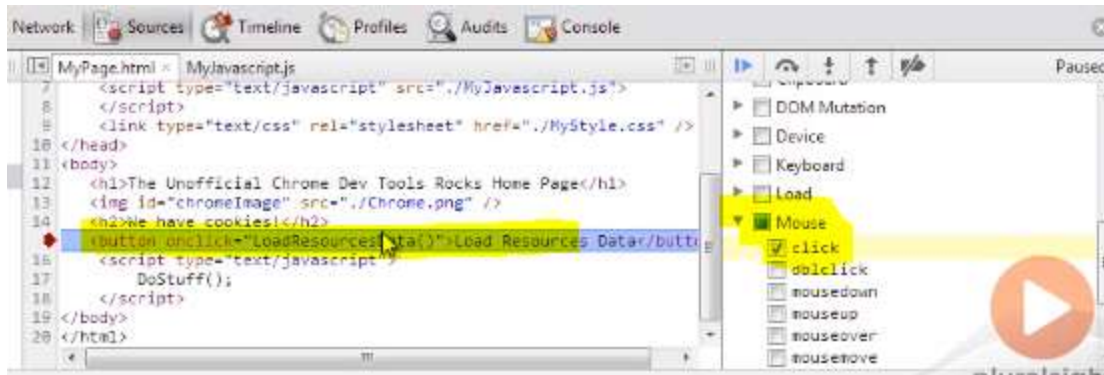
Note: we can use “pause on exception” selection to break the code when any exception does happen, if we click it one more time again then it will only break on uncaught exceptions:



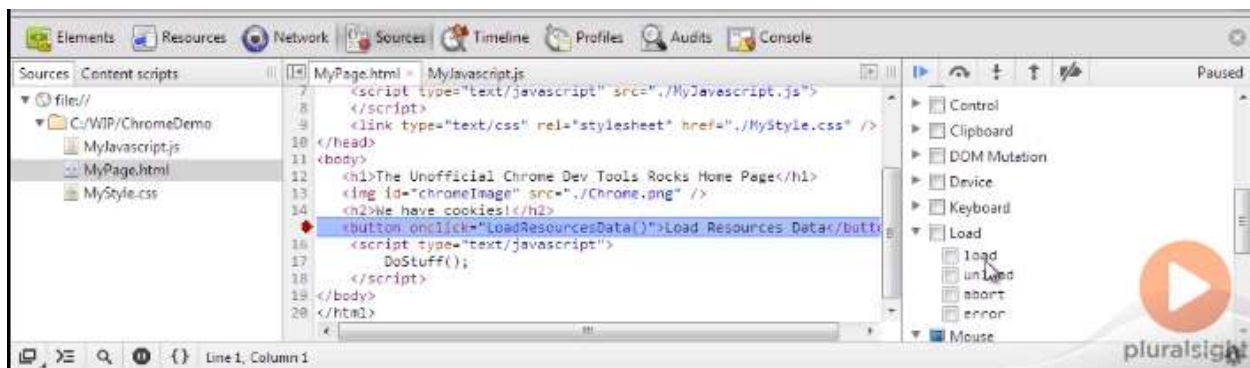
Note: we can use XHR breakpoint section to apply breakpoint on a XHR request as a contain condition automatically:



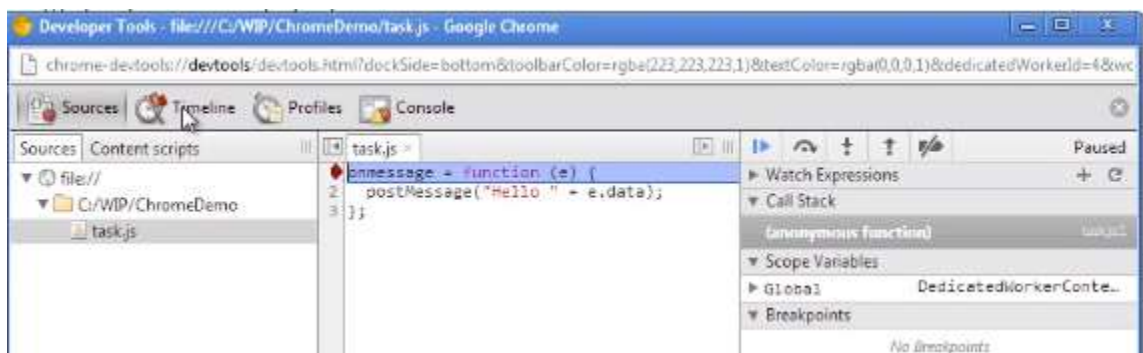
Event listener breakpoints



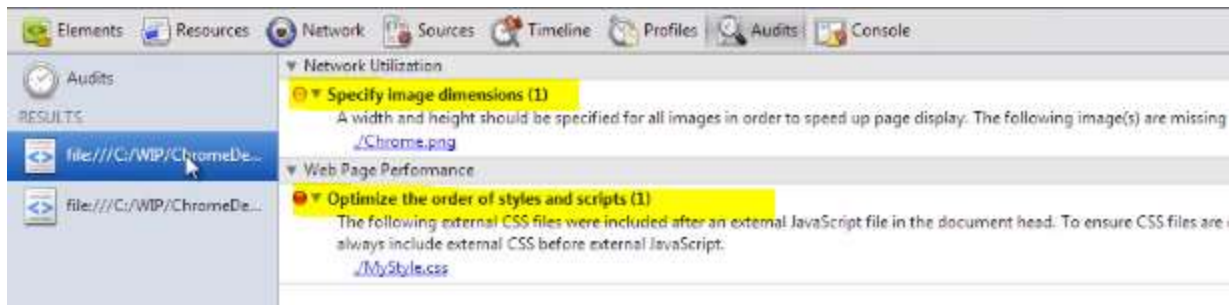
Event listener breakpoints - by using this we can break page as a breakpoint on different events like loading or unloading of a page.



Breaking when web worker get started, it will open a new thread window instance for debug:



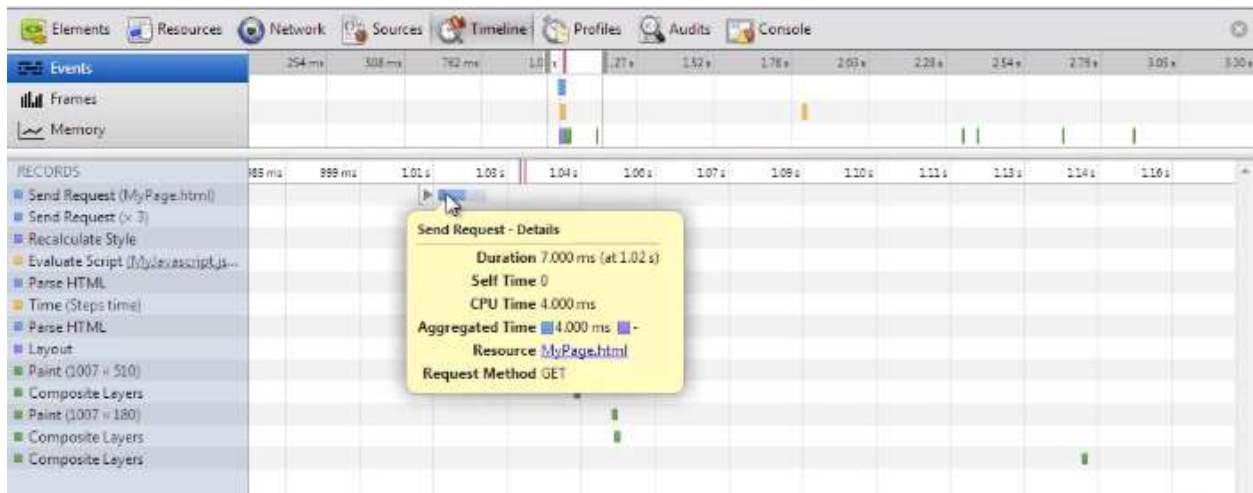
We should load our styles before JavaScript.



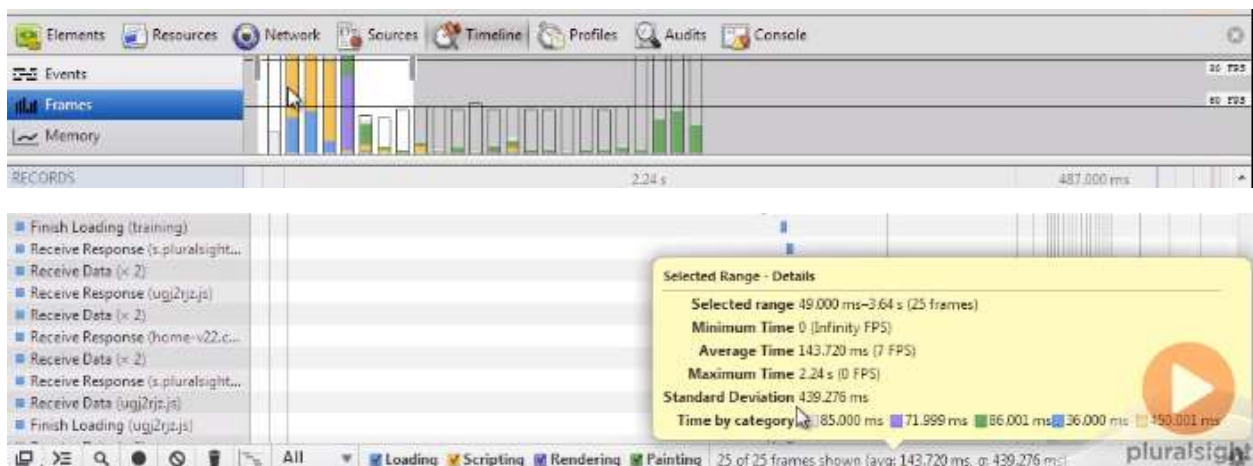
Timeline panel has three views – first is event to see when event took place and long it took, second is frame view to see how long it is taking to render each page in the browser i.e. to display a update in the browser, third is memory view to see how much memory was being used at that time.

The dark section in below is how long the actual event took, then next lighter section is how long the CPU took, lightest bar is total beginning to end.

Event view:

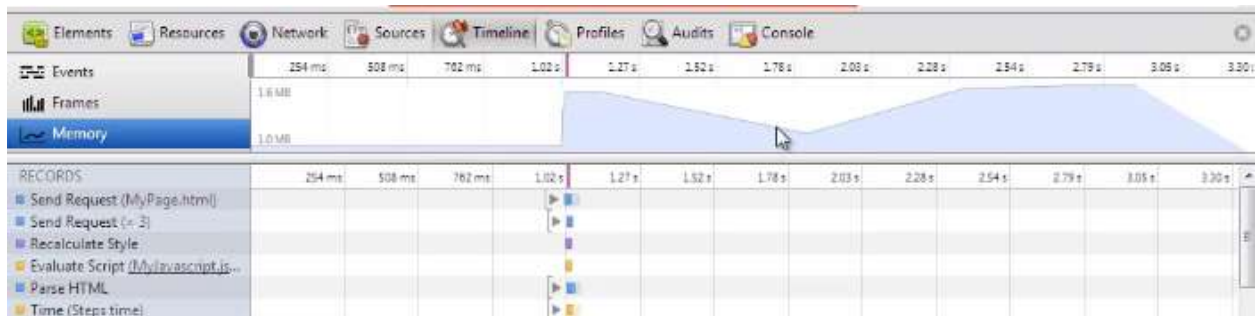


Frame view:

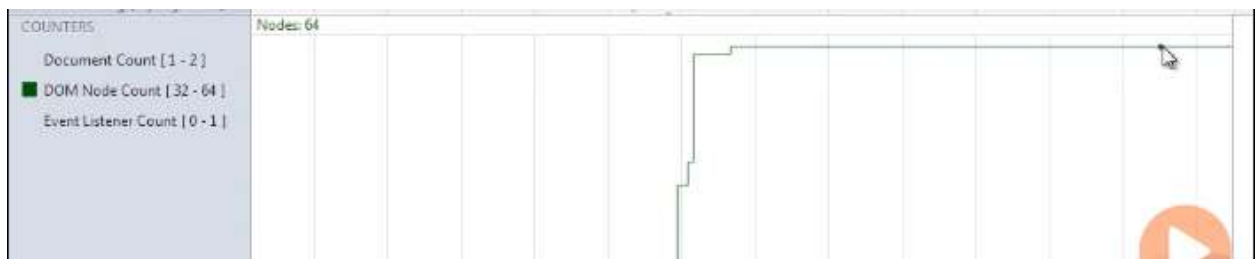


Chrome is able to refresh 60 frames per second.

Memory view:

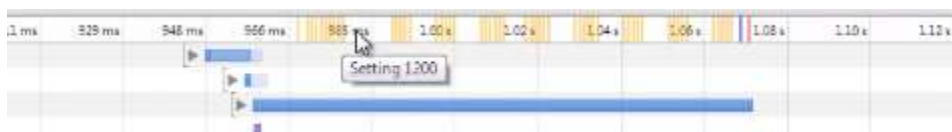


Counter in memory view for Document, DOM Node, and Event Listener:



Practice – troubleshooting slow JavaScript

Marking the timeline using console.timeline()



Profiles panel – using this panel we can find out what parts of our JavaScript are using the most CPU, or time CSS selectors are taking, or to find out more about Chrome's use of memory.



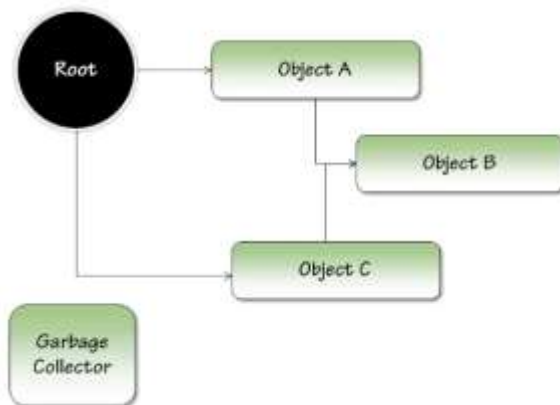
CPU profiles

Self	Total	Function
94.47%	94.47%	(idle)
4.46%	4.46%	(program)
0.03%	0.03%	(garbage collector)
0%	0.01%	▼ v8/18n
0.01%	0.01%	(anonymous function)
0%	1.03%	▼ file:///C:/WIP/ChromeDemo/MyPage.html
0%	1.03%	▼ DoStuff
0.04%	0.04%	group
0%	0.01%	▶ DebuggerScript.getAfterCompileScript
0%	0.01%	▶ InjectedScript.wrapObject
0.96%	0.96%	ThirdStep

CSS selector profile

Selector	Source	Total	Matches
h2	MyStyle.css:6	0	1
:-webkit-any(article,aside,nav,section) :-webkit-any(article,aside,nav,section) h2		0	0
body:-webkit-seamless-document	file:///C:/WIP/ChromeDemo/MyStyle.css	0	0
input[type="button"]:active:disabled, input[type="submit"]:active:disabled, input[type="reset"]:active:disabled		0	0
input, textarea, keygen, select, button, isindex		0	1
body		0	1
html		0	3
:-webkit-any(article,aside,nav,section) :-webkit-any(article,aside,nav,section) :-webkit-any(article,aside,nav,s...		0	0
input, textarea, keygen, select, button, isindex, meter, progress		0	1
button		0	1
:-webkit-any(article,aside,nav,section) :-webkit-any(article,aside,nav,section) :-webkit-any(article,aside,nav,s...		0	0
body	MyStyle.css:2	0	1
title		0	1
h1		0	1
head		0	4
script		0	1
:-webkit-any(article,aside,nav,section) h2		n	0

Memory profile and snapshot – chrome uses GC to memory clean-up automatically whenever required. All objects have to be referenced from some root that must always exist. Every object that needs to be sticking around should be accessible by following some path of objects and references. If an object cannot be reached from a root, then it must not be needed and can be collected.



Re-watch - <https://app.pluralsight.com/player?course=chrome-developer-tools&author=john-sonmez&name=chrome-dev-tools-m5-timeline&clip=14&mode=live>

The screenshot shows the Chrome DevTools Profiles tab with a memory snapshot selected. The left sidebar shows 'Profiles' with 'CPU PROFILES' and 'HEAP SNAPSHOTS'. Under 'HEAP SNAPSHOTS', 'Snapshot 1' (1.3 MB) is selected. The main panel displays a table of objects in the snapshot.

Class filter	Distance	Objects Count	Shallow Size	Retained Size
Constructor	3	2 0%	24 0%	960 0%
String	3	2 0%	32 0%	588 0%
String @23989	3	1 0%	16 0%	572 0%
String @23988	3	1 0%	16 0%	16 0%
Script	4	2 0%	24 0%	568 0%
Arguments	3	6 0%	112 0%	536 0%
ReplaceResultBuilder	4	2 0%	24 0%	528 0%
Navigator	4	2 0%	32 0%	524 0%
SourceLocation	4	2 0%	24 0%	416 0%

Below the table, the 'Object's retaining tree' is shown for the selected String object:

Object	Shallow Size	Retained Size	Distance
prototype in function String() @20463	36 0%	804 0%	2
String in Window //C:/WIP/ChromeDemo/MyPage.html @7533	40 0%	11544 1%	1
SString in @20663	276 0%	31068 2%	2
constructor in String @23989	16 0%	572 0%	3

The shallow size is how much actual memory that particular thing takes itself. The retained size is how much memory it ends up preventing from being collected.

Comparing snapshots – using comparison view

The screenshot shows the Chrome DevTools Profiles tab with the 'Comparison' view selected. The left sidebar shows 'Profiles' with 'CPU PROFILES' and 'HEAP SNAPSHOTS'. Under 'HEAP SNAPSHOTS', 'Snapshot 1' (1.3 MB) and 'Snapshot 2' (1.4 MB) are listed. The main panel displays a table comparing the two snapshots.

Class filter	# New	# Deleted	# Delta	Alloc. Size	Freed Size	Size Delta
Constructor	955	730	+225	113 052	89 464	+23 588
(array)	408	317	+91	77 224	40 928	+36 296
(system)	1 782	1 328	+454	44 148	32 256	+11 892
(closure)	898	841	+57	32 328	30 276	+2 052
HTMLElement	1 001	0	+1 001	20 016	0	+20 016
(string)	284	166	+118	6 916	3 980	+2 936
Object	161	96	+65	3 824	2 912	+912
Array	138	138	0	2 208	2 208	0
system / Context	32	32	0	1 232	1 232	0
(regexp)	9	9	0	324	324	0
Error	6	6	0	168	168	0
(number)	13	0	+13	156	0	+156
Window	3	2	+1	88	48	+40

Practice – comparing memory snapshots

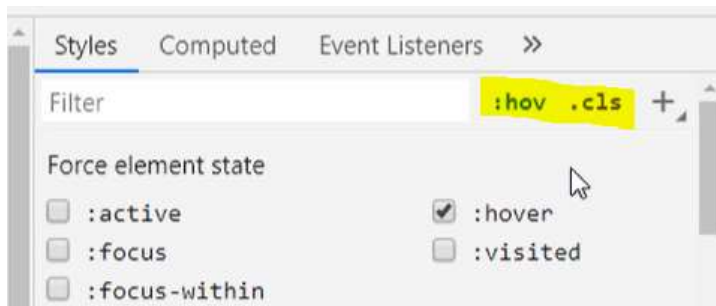
Debugging Sites Using Chrome DevTools

Elements, console, sources tabs are called panel and windows inside each of them are called pane.

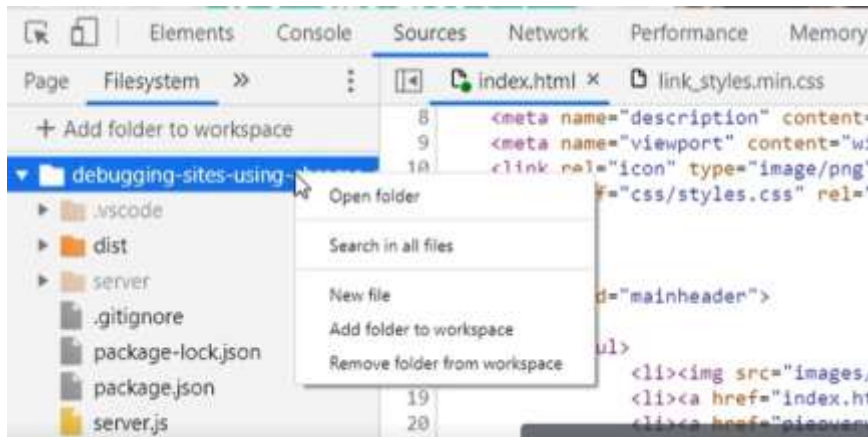
To open the chrome dev tools command menu press ctrl + shift + p.

Use ctrl + shift + c to directly open the 'elements' panel.

We can also view the styles by apply special state on the elements –



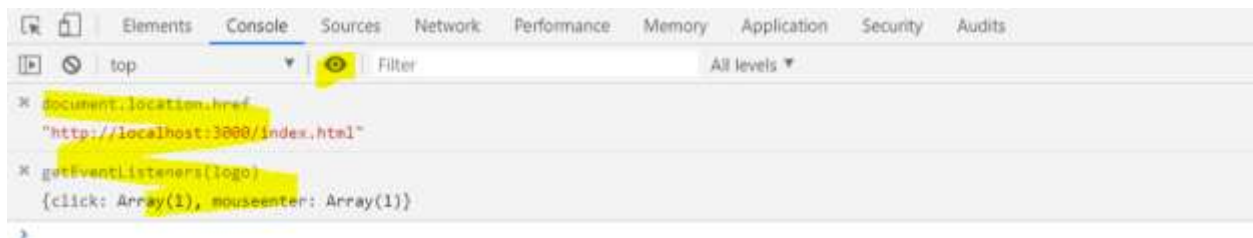
We can map to the actual file system with chrome dev tools so that it can persists the changes with original code-base –



We should most of time use `console.log()`, `console.info()`, `console.warn()` and `console.error()` methods to produce proper output on console. Use `console.dir()` for nice hierarchical JSON representation of data. And use `console.table()` for tabular data.

Use `$0`, `$1`, `$2`, `$3`, `$4` to evaluate currently or previous selected elements from console.

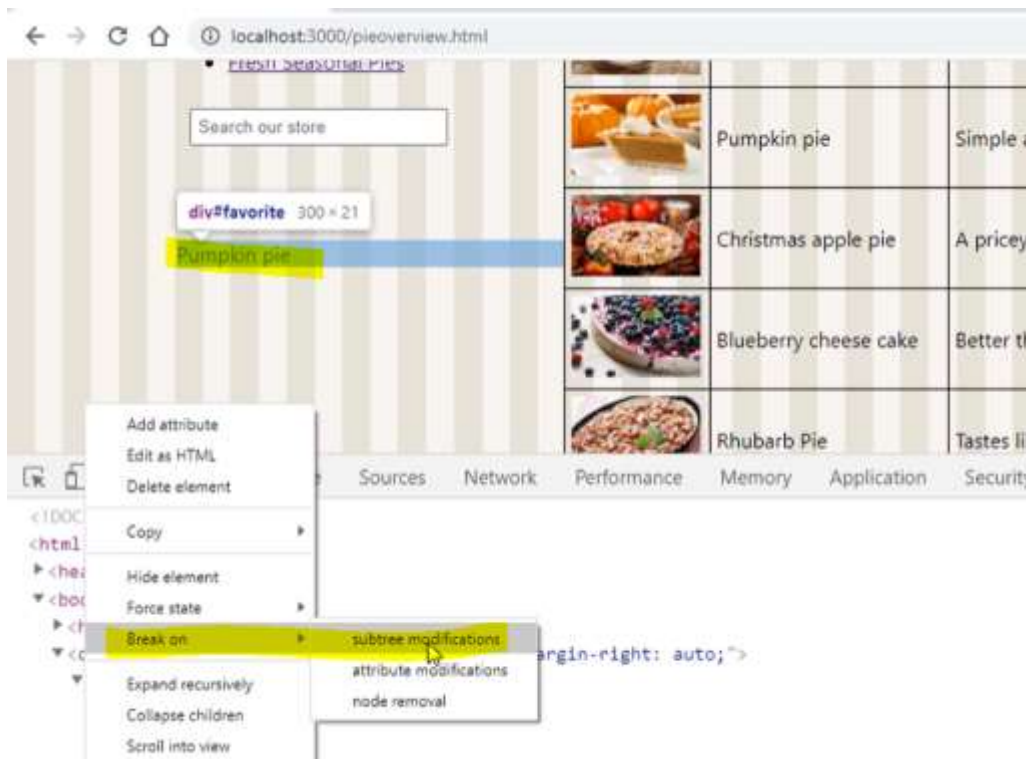
Use live expression to keep eyes on various debug values –



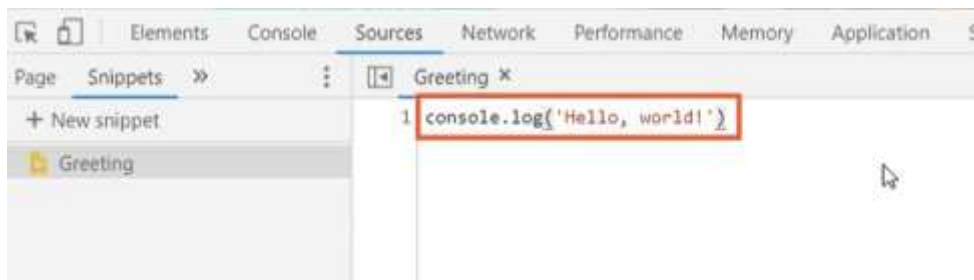
Adding log point, instead of `console.log()` in actual code –



We can also break the code when a specific element on HTML gets changed –



Using snippets in console –



Check the 'disable cache' checkbox on network panel to avoid caching of requests.

Technical – Other Resources

Time begins on 1st jan 1970 with the unix epoch, when we say `Date.now()`, it gives milliseconds from this time.

Design patterns - singleton, builder, prototype, factory, facade, proxy, iterator, observer, mediator, state

IDE's Evaluation - vi -> emacs -> vim -> neo vim -> nano -> notepad++ -> Dreamweaver -> visual studio code -> visual studio -> WebStorm (features looks more reliable and polished than vs code)

Choosing the framework depends whether website mostly have static content or highly interactive, if highly interactive then whether it needs SEO, if SEO and content rarely change like in blogs website then use JAM stack and prerender the content and cache on the CDN. if dynamic content then need full SSR + hydration.

Angular bootstrap process - angular.json (main) -> main.ts (bootstrapModule) -> app.module (bootstrap) -> app.component -> app.component.html -> index.html (<app-root></app-root> selector is used as an element to get the app component) -> The javascript files {runtime.js, polyfill.js, ... etc } are responsible to make our application a single page and they are handled by the browser itself. But, the HTML code should be available in our application itself.

Bun – the new runtime on the block – Node allowed users to run, execute and work with JS outside a browser. Now, a runtime is an environment to execute some code. JS now has two runtime environments. One, inside the browser and the other one outside the browser. Bun, is a JS environment too. But, at its core, is essentially a JS runtime. Just like Node or Deno, it will allow you to run JS outside the browser. JS runtime doesn't mean that it has to be similar in architecture as Node or Deno. Node and Deno both are built on top of Chrome's V8 engine which is an open source JS engine. Bun uses the JavaScriptCore Engine from WebKit. WebKit is Apple's implementation of JS Engine from WebKit. WebKit is Apple's implementation of a JS Engine. Just like chrome uses V8 engine to power with Chrome browser, Apple uses WebKit to power AppStore, Safari, etc.

Bun is written in a low level language called Zig. And since Zig is a low level programming language, it provides much more manual control over memory as compared to other languages. In bun.js, every file is transpiled. TS and JSX just work.

Performance compare – Node was faster than Bun in native JS, if we use TS, Bun completed much faster than using ts-node compiled JS. This just goes to prove that if you're a TS fan then you are going to love Bun!

Bun is designed in such a way that it can work with the existing npm packages too.

What kind of software development shortcuts feel fast but often will slow you down in the long run? - "We'll refactor it some other time", "I can do this on my own, we don't need to pair-program as it's a waste of time", "I don't need to write tests for this", "We can skip testing this part", "We can skip the code review", "It works on my machine, let's ship it", "Yeah, let's use the PoC as the foundation for the next project and not use what we've learned from building the PoC to build a proper foundation", "I don't need to read technical books in my spare time, I just pick things up at work"

You need to challenge your mind constantly. Otherwise it gets fat and lazy too. Don't always go the easy way. Try building stuff, try coding stuff and get your hands dirty with real hardware. Ask questions online, document your journey and talk to friends about it. It is that easy to get executed again about the things that brought you into software engineering in the first place. And while you are at it, try to offer coaching for junior engineers and share your work & wisdom online. Becoming a Mr Miagi or Yoda for a new generation of engineers is not such a bad thing.

Higher order observables - switchMap, mergeMap - switchMap cancels previous HTTP requests that are still in progress, while mergeMap lets all of them finish

Svelte is the most popular than angular and react

"HTML is the body, and JavaScript is the brain of the web application."

Mongo DB word is derived from an another word humongous

Senior programmers can see the future and take action before any problems occur.

We make products at google which will save time - google

Skills senior engineers need, beyond coding - How to give up your baby, that project that you built into something great, so you can do something else, How to communicate project status to stakeholders, How to find interesting work on your own, instead of waiting for someone to bring it to you, How to tell someone they're wrong without making them feel ashamed

In software development going slow is actually going fast. do lots of thinking and research before writing the code.

Admirable attributes of a great technical lead - Having an Opinion Yet Not Being Opinionated On Everything, Filled With Energy Yet Calm In All Situations, Disown Their Team's Successes and Own Their Team's Issues, Can Comment Well and Code Even Better, Appreciate Old Tech and Embrace New Tech, Communicate Fluently and Connect Frequently, Strive to Answer Brilliantly and Dare to Ask Stupidly, Is Firm Yet Can Be Flexible, Execute It With Brain and Do It With Heart

Talk is cheap. Show me the code. — Linus Torvalds

People don't care how much you know until they know how much you care. — Theodore Roosevelt

An incomplete list of skills senior engineers need, beyond coding - How to tell someone they're wrong without making them feel ashamed, How to find interesting work on your own, instead of waiting for someone to bring it to you, How to build software while delivering incremental value in the process.

Format images - Sometimes you lead a website where they have banners at the top. But they forgot to preformat the image. Images of a few MB can cause prolonged loading times, and it is not what you want for your users. Use the correct height, width, and file extension. Be careful with png files. They can be high quality but take up a lot of MBs. JPG: Use this for photos. PNG: Use this for high-quality photos for download purposes and transparent images. Be careful. This comes with costly data. GIF: Use this for images or icons with a meager amount of colors. WebP: This is a newer image type. It can hold very high-quality photos without the downside of png files.

Writing comments - Exceptional "Hard To Read" - We should add comments when the code contains regular expressions or complex algorithms. Those parts are usually hard to read and often greatly benefit from the comments.

If you act without thinking, you are wasting the very edge that you have over machines. You are a candidate to being replaced in the near future (either by a thoughtful human or for a cheaper dumb machine). Always try to understand what's the purpose of the task you've been assigned to. Unless, of course, you want to become a bad developer.

Google searching tips - cut the crap - Forget "what," "how," and other words that serve only a syntactical purpose. Demand answers using meaningful and descriptive verbs example - instead of 'what is algorithmic complexity' use 'define algorithmic complexity'. Order keywords from broad to specific, instead of 'consume an api using typescript with axios' use 'typescript axios consume api'. Use Images for Diagrams and Visualizations. Many times, Images will contain concise and informative graphics that will answer your question much faster than a web page could.

Performance of using backticks are better than single quote and double quotes for string.

Useful http status codes - 100 Information - 100 — Continue; 101 — Switching protocol; 103 — Checkpoints. 200 Successful. 300 Redirection - 301 — Moved Permanently; 302 — Found; 304 — Not Modified; 305 — Use Proxy; 307 — Temporary Redirect. 400 Client Errors - 400: Bad Request; 401: Unauthorized; 403: Forbidden; 404: Not Found; 408: Request Timeout; 410: Gone; 429: Too Many Requests. 500 Server Errors - 500 — Internal Server Error; 502 — Bad Gateway; 503 — Service Unavailable; 504 — Gateway Timeout.

With picture tag, we can easily achieve resolution switching by using multiple source tags inside the picture tag.

While designing these action buttons make sure of the following points. Primary actions should be obvious: Use solid and higher contrast background colors. Secondary actions should be clear: Try not to make them too prominent, outline styles and lower contrast background colours work great in this case. Tertiary actions shouldn't be unobtrusive: Style these actions like links works the best.

We should have a home button on the navigation page, don't just rely on the logo

Coding faster will not make you a better programmer. Abstraction, design, and the experience of real-world engineering are what will make you a better software engineer.

As a good rule of thumb, 24 hours is a good time range. As long as my requested reviewers can finish the review within 24 hours, we should expect no reminder. If no review within 24 hours, I can ping the reviewers.

Use NanoID instead of UUID

Redis - key value store, keep data in ram, used in high volume on data like twitter or full text search.

Today's low-code products cover nearly every domain, from email builders, crms, and web designers to workflow and prototyping tools, and even ai. old examples are excel, visual basic, Microsoft frontpage, yahoo pipes, Delphi

Crypto concepts - hash, salt, hmac, symmetric encryption, keypairs, signing, asymmetric encryption

There is the International Software Architecture Qualification Board (ISAQB®) that offers the Certified Professional for Software Architecture (CPSA®) certification scheme. CPSA® certifications are globally recognized.

Even something as simple as setting a breakpoint on a function can be surprisingly complex. The compiler needs to emit information the debugger can use to map source level concepts such as files, lines of code, and functions, into machine level concepts such as addresses, registers, and stack locations. This information is not always correct or well-structured and debuggers (including Pernosco) implement several heuristics to manage the problems.

When we want to switch to a new company, only then practice for competitive programming

Keep yourself healthy, keep yourself fit and keep writing amazing code!!!

Kouch typing is possible because of muscle memory

QWERTY keyboard layout was designed to avoid key jamming issue in era of typewriters.

Developers advocate === developers friend. to improve developers experience while using tools.

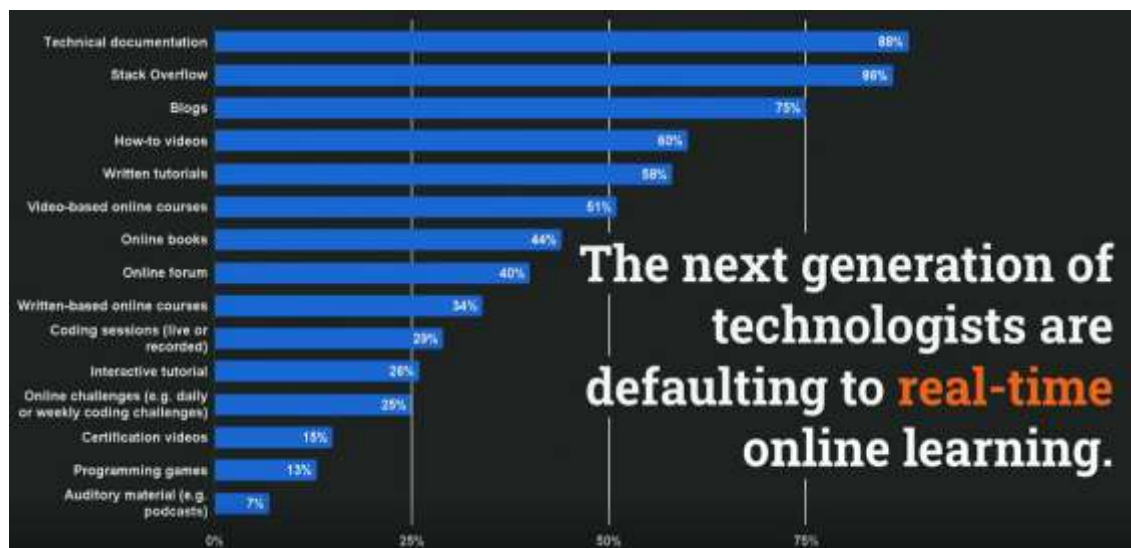
I'm almost 46 and I quote every single word of your article. When I read "think outside the box" I elaborated on the metaphor and thought: "when we are young, it's easy to think outside the box because that box is small if ever exists."

Learn how to research - Programming is problem-solving. In order to solve problems, it is vital to know how to research for solutions properly. Take notes, ask questions, and try to find the right documentation or solution documents.

Get your first freelance gig - Make this the year for a career change and land your first contracting job. Have a look at web platforms that offer project opportunities, contact recruiters, participate in community events and conferences, or see if your employer might want to hire you as a freelancer.

CEO of stack overflow is also an Indian.

Priority – flexibility in working, feeling productive (without having disturbance), tech stack.



Over 70% of developers are learning a new technology at least once a year.

53% of developers want to work for companies that prioritize the developer experience i.e. flexibility, learning and productivity.

Event based learnings (like office training) vs continuous learnings (like from pluralsight).

JWT Fundamentals

JWT has three segments, each separated by dots. If it is a base64-encoded JSON then its first two segments would start from characters "eyJ" because when decode it becomes base64({").

```
eyJ0eXAiOiJhZCtqd3Q1LCJhbGciOiJIUzI1NiIsImtpZCI6IjMxMzZjNWZhYWE0MmIwNz  
hkZTZkNGEwYmQ0GE2ZWmIn8.eyJpc3MiOiJodHRwczovL2lkcc5sb2NhbcCI6ImF1ZCI6  
ImFwaTE1LCJzdWIiOiI1YmU4NjM1OTA3M2M0MzR1YWQyZGEzOTMyMjIyZGF1ZSI6ImNsaW  
VudF9pZCI6ImFwcCI6ImV4cCI6MTY1MzU3NzA1OCw1aWF0IjoxNjUzNTczNDU4LCJqdGki  
OiIwZTc2NTkwY2NiN2RmYzY3ODZiZmU2NDEzYjk1MTg2OSJ9.vIv1L0bafKHNM1_5YY-  
kkTw6gWD791WJGKz2BH0j9EU4MrUeyavhJoWwql1xX4mRTHluqx4ECpziFyAhJjvwg
```

First part it has Header that describes the token itself and how to read & validate the token. It has properties like type, alg and kid.

```
{  
  "typ": "at+jwt",  
  "alg": "ES256",  
  "kid": "3136c5aaaa42b078de6d4a0bd48a6eff"  
}
```

The second part is the payload, it is the content of the token itself. It contains claims about the entity.

```
{  
  "iss": "https://idp.local",  
  "aud": "api1",  
  "sub": "5be86359073c434bad2da393222dabe",  
  "client_id": "app",  
  "exp": 1653577058,  
  "iat": 1653573458,  
  "jti": "0e76590ccb7dfc6786bfe6413b951869"  
}
```

The final part is the signature value, it is created using the header, payload and signing key. Its length varies based on the algorithm and key.

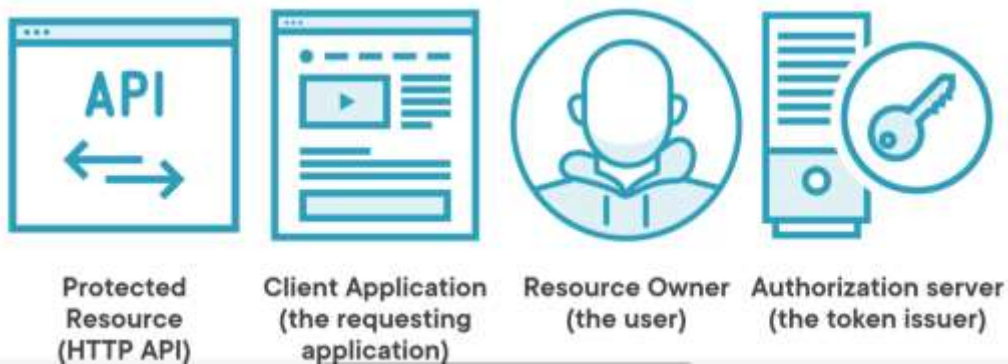
The JWT pronounce as “jot”. It is originally created by the OAuth working group due to demand for JSON representation of claims and to replace SAML assertion.

OAuth 2 delegation –

OAuth 2 Delegation

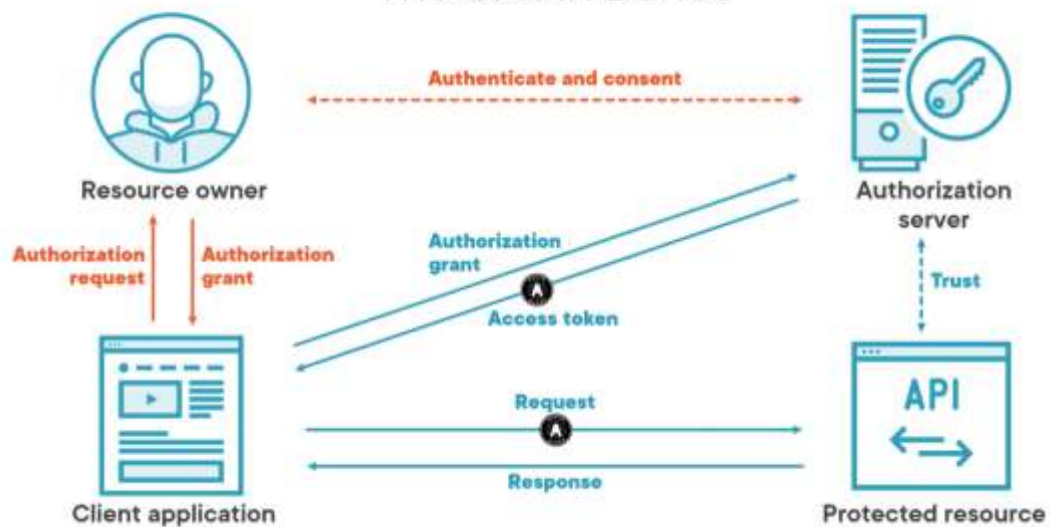


OAuth 2 players –



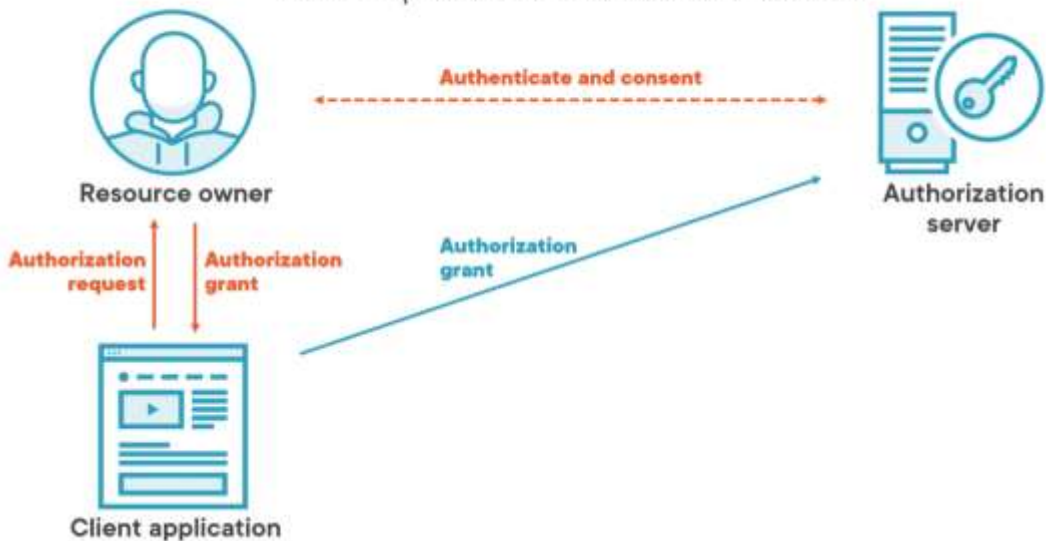
OAuth Dance –

The OAuth Dance



Identity Token

The OpenID Connect Dance



Identity Token Example –

```
{
  "iss": "https://idp.example.com",
  "aud": "https://sp.example.com",
  "nbf": 1602927477,
  "exp": 1602927777,
  "iat": 1602927477,
  "nonce": "e3f1c258a9a047bfb13a78fa6ace004a",
  "sid": "472ae8c4776c4d109a9ed7ddb8780cfa",
  "sub": "994211131",
  "auth_time": 1602927277,
  "amr": [ "pwd" ],
  "given_name": "Scott",
}
```

We should use SSL while using JWT, also don't send them using route param or query strings. We should send it in authorization header prepended by authentication scheme like basic, negotiate –

```
Authorization: Bearer
eyJ0eXAiOiJhdCtqd3Q1LCJhbGciOiJFUzI1NiIsImtpZCI6ImxMzZjNWfhYWE0MmIwNzhkZTZkNGEwYmQ0OGE2ZWZmIn0,eyJpc3MiOiJodHRwczovL2lkcS5sb2NhbnB1IiwiaWF0IjE1YmU4NjM1OTA3M2M0MzRlYWQyZGEzOTMyMjIyZGF1ZSI6ImNsaWVudF9pZCI6ImFwcCI6ImV4cCI6MTY1MzU3NzA1OCwiaWF0IjoxNjUzNTczNDU4LCJqdGkiOiIwZTc2NTkwY2N1N2RmYzY3ODZlZmU2NDEzYjk1MTg2OSJ9.vIv1LObafKHNM1_5YY-kkTw6gWD791WJGKz2BH0j9EU4MrUeyavhJoWwqllxX4mRTHluqx4ECpz1FyAhJjvwg
```

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="https://idp.example.com",
  error="invalid_token",
  error_description="The token has expired"
```

When to use JWTs – for API access, for information transfer (identity token), security proofs. We should always use JWTs in combination with something else like OAuth or Open Identity Connect protocols where rules are defined and low risk of misuse.

JWTs are not a replacement for cookies and sessions. Browsers cannot maintain JWT sessions, we have to implement token storage and management. There is no out-of-the-box method to invalidate a single JWT.

We should not store application or permission data as we should keep our JWTs small as it can easily hit header size limits.

JWTs with permission like below will become unmaintainable, we should only keep data into JWT that will be unlikely to change during the JWT lifetime.

JWTs with Permissions

```
{
  "iss": "https://idp.example.com",
  "aud": "https://api.example.com",
  "exp": 1656060314,
  "iat": 1656060014,
  "sub": "b3bfa8d95a17468292a87995e36552dd",
  "name": "Scott Brady",
  "subscriptions": [
    { "id": 1, "account": 123, "type": "premium", "join_date": 1656010314 },
    { "id": 2, "account": 321, "type": "joint", "join_date": 16560003632 }
  ],
  "roles": [
    "admin",
    "super_user",
    "can_read_employee_records",
    "can_write_employee_records"
  ]
}
```

JWTs without Permissions

```
{
  "iss": "https://idp.example.com",
  "aud": "https://api.example.com",
  "exp": 1656060314,
  "iat": 1656060014,
  "sub": "b3bfa8d95a17468292a87995e36552dd",
  "name": "Scott Brady",
  "department": "engineering",
  "role": "engineering_manager"
}
```

JOSE (JavaScript object signing and encryption) standards – JWON Web Tokens, JSON Web Signature, JSON Web Encryption, JSON Web Key, JSON Web Algorithms.

Initial format check of JWT - three sections, two dots, base64url data, valid JSON objects.

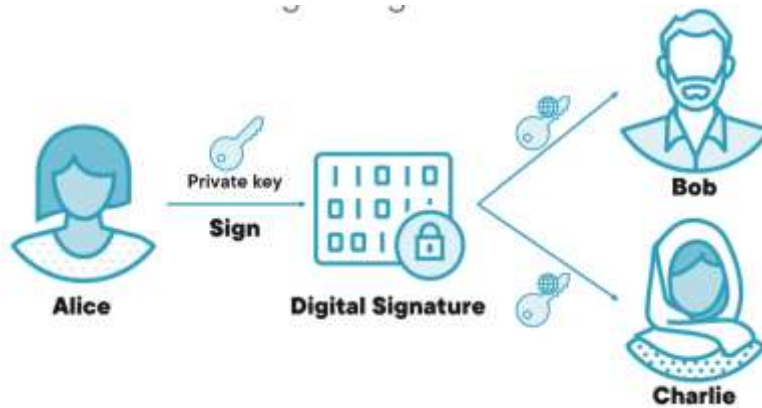
We should first validate the token like checking the issuer, subject value, audience, expiration date, before parsing it.

We can generate JWT token from below website, in real life we should use our internal code to generate it –

<https://www.scottbrady91.com/tools/jwt>

we can use jose npm library to create JWT in javascript, for more details refer this course - <https://app.pluralsight.com/course-player?clipId=d7fe65e9-d4dc-414e-8994-ba14b2d7b65b>

Digital Signatures – it provides integrity, authentication and non-repudiation.



JSON Web Algorithms (JWA) – parts of it –



Signing algorithm types – EdDSA, ES256, PS256, RS256. The EdDSA is the best alternative.

We should use hashing algorithm from the SHA-2 family, never use SHA-1.

We should not use HMAC Security approach as it does not provide non-repudiation as both parties uses the private key and can create token.

Asymmetric encryption is opposite of asymmetric signing where we want many to encrypt the data but only one to decrypt it –

Asymmetric signing -

Initialization vector (IV) – it is a value which is used during symmetric encryption. Must be unique per token. It is a random value.

Ciphertext – the encrypted message that has been encrypted using symmetric cryptography. It has been created using the message, IV, CEK and protected header.

Authentication tag – used to ensure the integrity of the ciphertext.

Nested JWT – A signed JWT (JWS) which must itself be validated. Allows us to prove who created the token.

When to use JWE – if we use PII (personally identifiable information) like names, email address street address, IP address, account number, telephone number etc., if token needs to be passed through multiple systems (including the 3rd party).

For JWE, we should use RSA + AES algorithm for encryption.

Days since unsigned JWT vulnerability - <https://www.howmanydayssinceajwtalnonevuln.com/>

Best practices – we should rely on standardized claims to prevent misuse – issue, audience, subject, lifetime.

Proof of possession – if someone else also possess the token, we can validate the request as-well. To avoid this we can use approach like JWT certificate thumbprint confirmation method (mTLS) and another method is DPoP proof which tells the authorization server what key to bind an access token to. Both of the approaches doesn't work in 100% scenario and have some drawbacks.

Alternatives to JWTs – PASETO (Platform-Agnostic security token), Branca Format token, Cookies (but vulnerable to CSRF).

YAML 1 Fundamentals

It is short for YAML aren't markup language. It is human-readable data serialization language. It can be used to keep and transfer the data. Its most common purposes is the configuration files. It is a true superset of JSON.

White space are unprintable characters.

YAML use cases – cross-language data sharing, configuration files, log files, object persistence, working with language like ruby, python, etc.

It has two style – block (human readable) and flow (less human readable like JSON)



Block style

YAML's format, indentation to give the data structure.



Flow style

JSON style format, using characters as explicit indicators to indicate structure.

Building blocks – sequence (arrays), mapping (key-value) and scalar (string, number, boolean and dates).

We should do indentation with spaces not with tabs.

For list we need to use (-) and for key-value we need to (:).

```
key: value
list:
  - first item
  - second item
  - third item
flowlist: [just, a list, in flow style]
```

Scalar values – with string values we can use quotes or without quotes.

```
tool: yaml
version: 1.2
awesome: true
nothing: null
duplicateKeys: not allowed
```

By using the '#' we can add comment.

One YAML file can contain multiple documents. The documents can be separated by 3 hyphens (---).

Below image shows 3 documents –

```
tool: yaml
version: 1.2
awesome: true
nothing: null
duplicateKeys: not allowed
---
This is a new document
---
One file can contain multiple documents
```


Multi-line strings – folding and chomping



Folding

Dealing with new lines in multi-line strings



Chomping

How to deal with trailing newlines in multi-line strings

Nested sequence (array) –

```
flow_lists:
- [yaml, json]
- [python, javascript]

block_lists:
-
- yaml
- json
-
- python
- javascript
```

Nested mappings –

```
person:
  name: Maaïke
  age: 30
  address:
    streetname: Langstraat
    number: 1
    zipcode: 1234AB
    city: Amsterdam
    country: The Netherlands
```

Combining sequences and mappings –

```
languages:
  - programming:
      - frontend:
          - html
          - css
          - js
      - backend:
          - java
          - python
          - c#
  - data serialization:
      - yaml
      - json
```

Dates scalar –

```
timestamp: 2022-03-22T22:19:56.10+02:00
simple_date: 2022-03-22
not_a_date: !!str 2022-03-22
```

Explicit typing – by using like !!str is an explicit typing which will convert the date into a string type.

Repeated nodes – to avoid code repeat.

```
key: &repeated Hi there
new_key: *repeated

person: &person
  name: Maaïke
  age: 30
  address:
    streetname: Langstraat
    number: 1
    zipcode: 1234AB
    city: Amsterdam
    country: The Netherlands
another_person: *person
```

We can also override the repeated code with the merge key "<<:" – in below example the 'name' property value will be overridden by 'maria'


```

person: &person
  name: Maaïke
  age: 30
  address:
    streetname: Langstraat
    number: 1
    city: Amsterdam
    zipcode: 1234AB
    country: The Netherlands

another_person: *person

yet_another_person:
  <<: *person
  name: maria

```

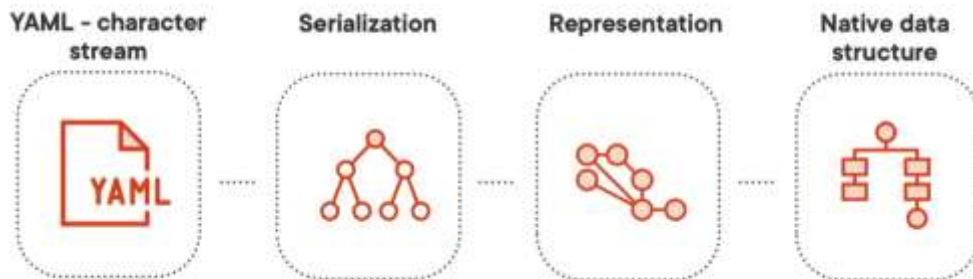
Tags gives a node a type –

```

no_date: !!str 2022-03-21
no_number: !!str 20
unnecessary: !!int 20

```

Processing of YAML –



To avoid common errors – we can use single quotes for string instead of double quotes, wrong indentation, put phone numbers into string to avoid getting converted into unrelated type like octal, duplicate keys are not allowed, accidental list entry, lists as keys in languages that don't allow this.

```

! examples_common_errors.yaml x
! examples_common_errors.yaml > real_phone
1  octal_phone: 03012341234 # converted to 405389980
2  real_phone: "03012341234"

```

YAML vs. JSON – YAML is standard for configuration and JSON is standard for service API.

YAML	JSON
Can be parsed with a YAML parser	Can be parsed with a YAML parser
Comments with a #	Comments not allowed
Objects and lists are denoted with indentation or {} and []	Objects and lists are denoted with {} and []
String quotes are optional, can be double or single	String quotes are mandatory, must be double quotes
Root node can be any valid data type	Root node must be object or list
Standard for configuration	Standard for APIs

YAML vs. XML

YAML	XML
Data serialization language	Markup language
Easier to read	Harder to read
Querying YAML relies on many different external tools	Well established options to query data in XML file
Typically, preferred option for configuration	Decreasing in popularity because of JSON and YAML

Getting Started with Podman

The early 2000s was the era of the virtual machine. We are now firmly in the container era. They have much smaller footprint, quicker to download and a lot less resource needed to run them.

Container image – a file system bundle containing all files, packages, dependencies, and kernel needed to run a service. To the host this is a single process.

Container run isolated from other containers so software and dependencies installed on one container do not affect other containers.

It has small footprint because there is no complete operating system with a container making them are much less resource hungry than running virtual machines.

Container systems –



LXC: The LXC Project is funded by Ubuntu and Canonical and is built around the basics of CGroups and Namespaces and dates back to 2009



Docker: Docker is a project dating back to 2013 that is now managed by Docker Inc. It uses a docker client and a docker daemon process in the background



Podman: Podman dates from 2017 and is the newest and heavily funded by Red Hat. Podman complies with OCI (Open Container Initiative) and does not rely on daemons

It also has rootless containers which means to being able to create and execute containers without the need of elevated privileges is a major improvement in Podman over Docker. This is achieved using user namespaces which are unique environments for each user keeping their processes within that namespace. UID mapping allows the user to carry out tasks as a pseudo-root user within their own namespace.

Containers are created from filesystem bundles called images. By default no images are installed with podman. The ps sub-command is used to list running containers. Use the option -a to list all containers including stopped containers. There are no containers by default either.

```
podman image ls
```

```
podman ps
```

```
podman ps -a
```

```
podman search httpd
```

QWIK Framework.

Here comes Qwik, which adopted a different approach, the focus shift from "doing faster" to "doing less". The last framework that came with a new concept was Astro, which introduced partial hydration, a revolution in javascript frameworks.

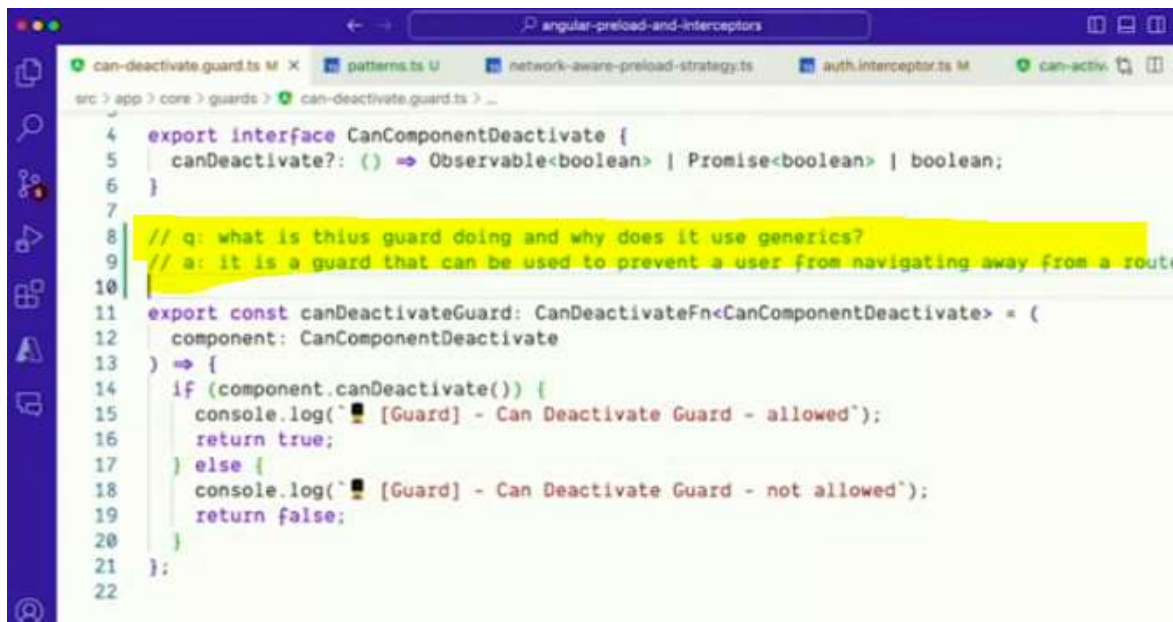
For example, if we have a page that starts with a text that does not need javascript, we load nothing, but if we scroll down to a component that needs javascript for the interaction, we hydrate the page with the needed javascript. Qwik killed the game, with no partial hydration but zero hydration and 0 javascript on the first load.

For instance, if we have a form contact at the bottom of a page, Angular, React, or VueJS will load the javascript needed on the first load, Astro will load it if we scroll down, Qwik will load it only and only if we click on the send button of the form, it loads javascript not when we have the form in front of us but if we interact with it. The framework delays the invocation of the javascript as much as possible and invokes it only what is absolutely needed. The focus is on the first load. Qwik is unique in that it has resumability.

The name of the concept gives us a perception of what we are talking about, “stop/resume,” resume the application where the server left off in place of replaying all of the work. To be precise, Qwik will not download every file only when you click, but after doing the first load to show the page (that is interactive), the framework will start automatically downloading the code necessary for the web app using a service worker, and it will store it in the browser's cache, so we will not send a request to the server every time, but it will grab it instantly from the cache, if the code is not downloaded yet it will download it from the server, and that's what we call prefetching. So Qwik is a mix of server-side rendering and client-side rendering. It takes advantage of the two of them.

Copilot

We can use q & a formatted comment with copilot to explain any code –



```
4 export interface CanComponentDeactivate {
5   canDeactivate?: () => Observable<boolean> | Promise<boolean> | boolean;
6 }
7
8 // q: what is this guard doing and why does it use generics?
9 // a: it is a guard that can be used to prevent a user from navigating away from a route
10
11 export const canDeactivateGuard: CanDeactivateFn<CanComponentDeactivate> = (
12   component: CanComponentDeactivate
13 ) => {
14   if (component.canDeactivate()) {
15     console.log(`[Guard] - Can Deactivate Guard - allowed`);
16     return true;
17   } else {
18     console.log(`[Guard] - Can Deactivate Guard - not allowed`);
19     return false;
20   }
21 };
22
```

Introduction to Microsoft Copilot

ChatGPT is created by an organization called OpenAI.

Large language models (LLMs) applications –

Content Generation	Writing articles, stories, poems, and more
Language Translation	Translating languages with context
Coding Assistance	Generating code, debugging help
Customer Support	Crafting responses, addressing queries
Research Aid	Summarizing articles, finding relevant information

Microsoft Responsible AI –

<u>Fairness</u> AI systems should treat all people fairly	<u>Reliability and safety</u> AI systems should perform reliably and safely	<u>Privacy and security</u> AI systems should be secure and respect privacy
<u>Inclusiveness</u> AI systems should empower everyone and engage people	<u>Transparency</u> AI systems should be understandable	<u>Accountability</u> People should be accountable for AI systems

Copilot suite of products –

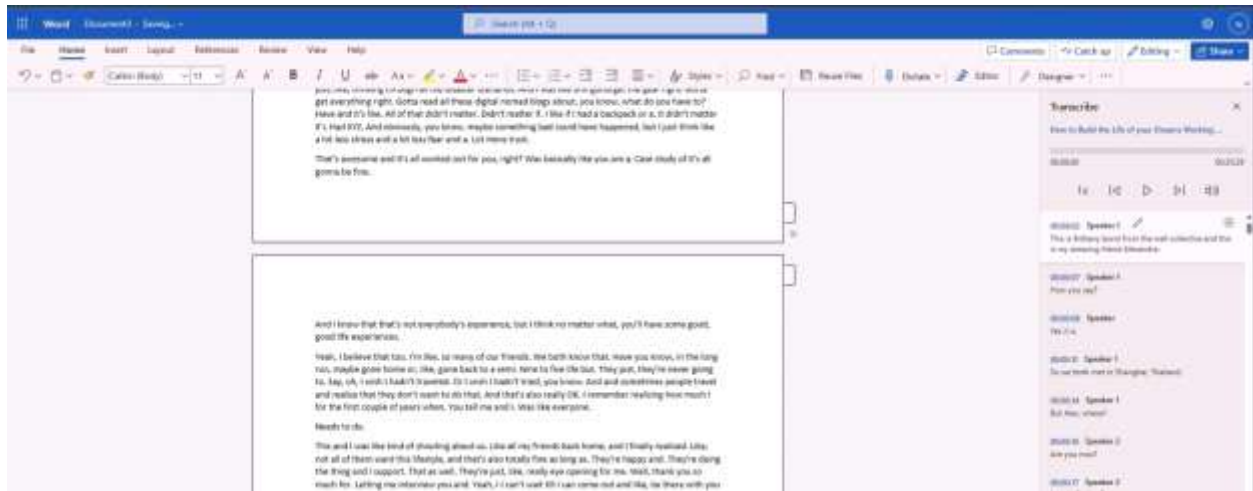
Copilot in Windows	Copilot in Microsoft 365	Copilot in Dynamics 365
Copilot in the Power Platform	Microsoft Security Copilot	GitHub Copilot

Copilot products similar to Bing Chat or ChatGPT, but they also combine it with your organization's data to allow Copilot to provide answers and content based on organizational knowledge.

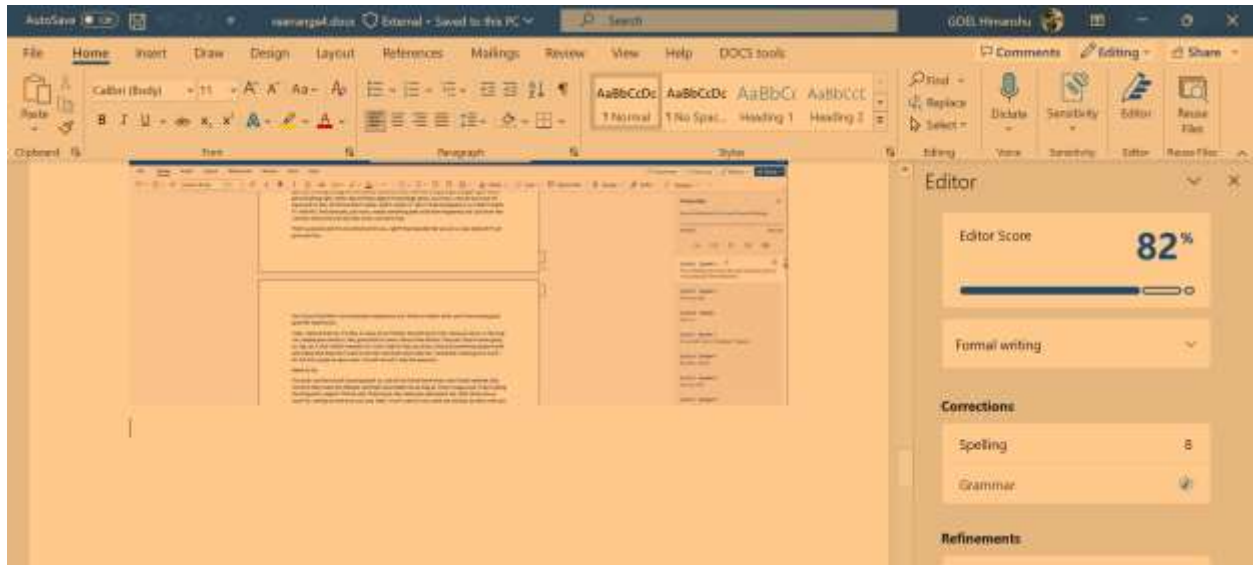
AI and Office

We can extract transcript in word from any video and then can create synopsis from that from ChatGPT

–



We can use powerful editor feature in MS Word to improve the content –



ChatGPT and Generative AI: The Big Picture

Generative artificial intelligence – any type of artificial intelligence that can be used to create new text, images, video, audio, code or synthetic data.

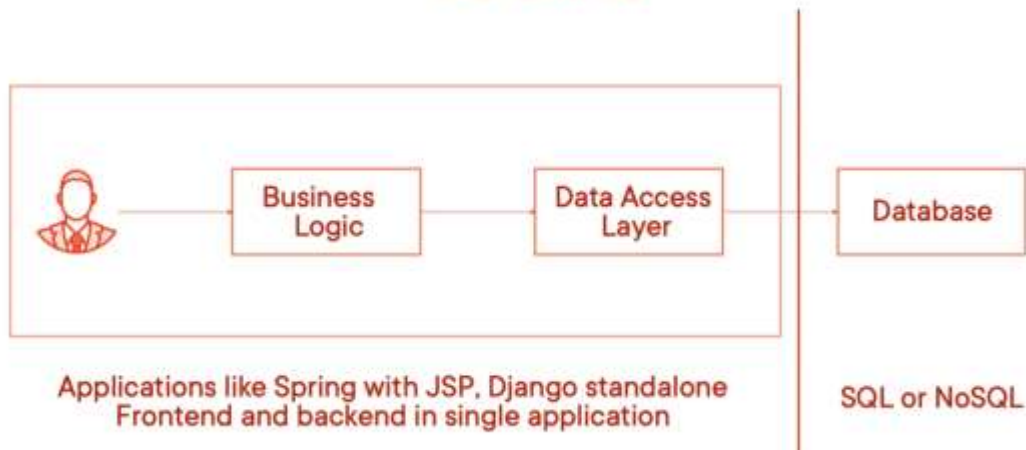
Discriminative AI – categories an existing data like is this transaction fraud or image recognition.

ChatGPT is a generative AI language model that interacts in a conversational way.

Micro Frontend Fundamentals

Monolith to micro frontend journey –

Monolithic



Pros	Cons
Development	Code base size IDE loading
Deployment	Commitment to a single tech stack
Scalability	Deployment of whole application for a small change

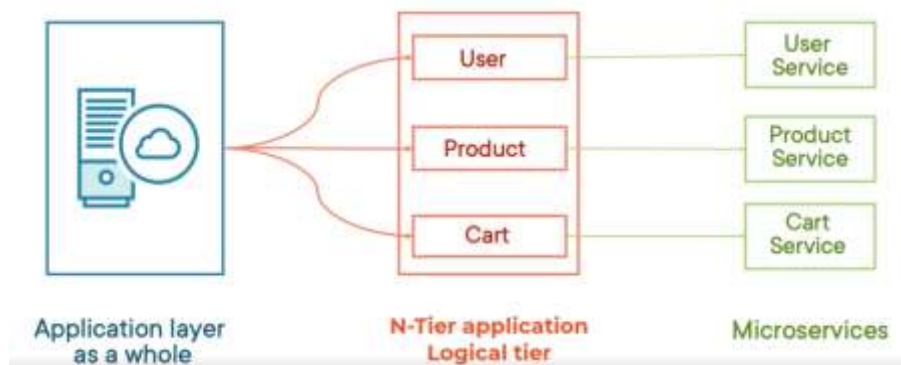
N-tier application – unlike monolith, it has separate servers for frontend and backend.



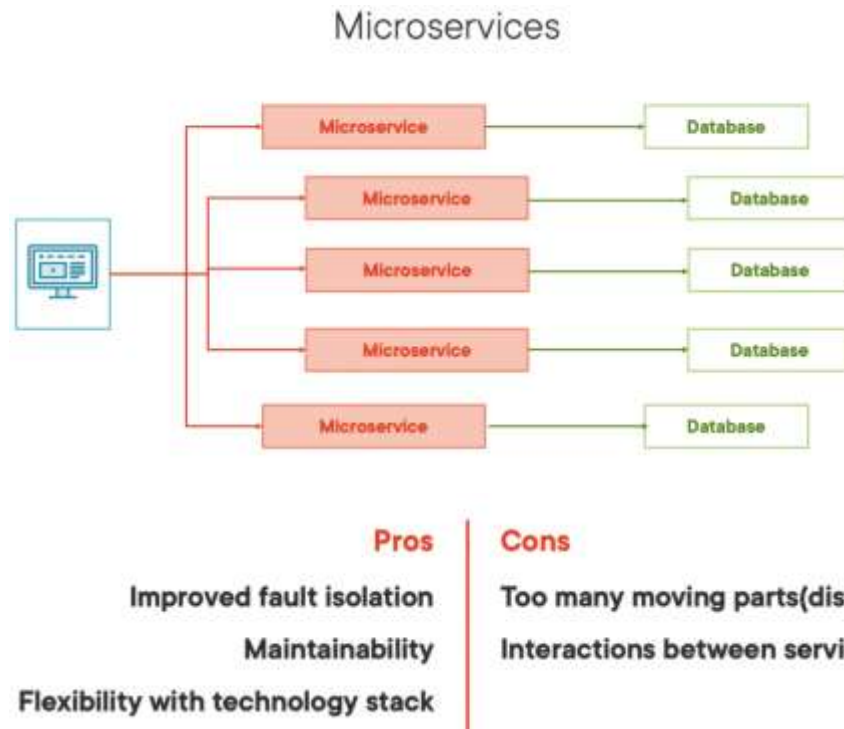
In this also, the logic tier is still huge and for a small change whole tier need to be re-deployed. Logic tier has been involved into micro services.

In microservices, we would be having 3 teams with 3 separate repository to handle user, product and cart functionalities.

Application Layer Segregation to Microservices

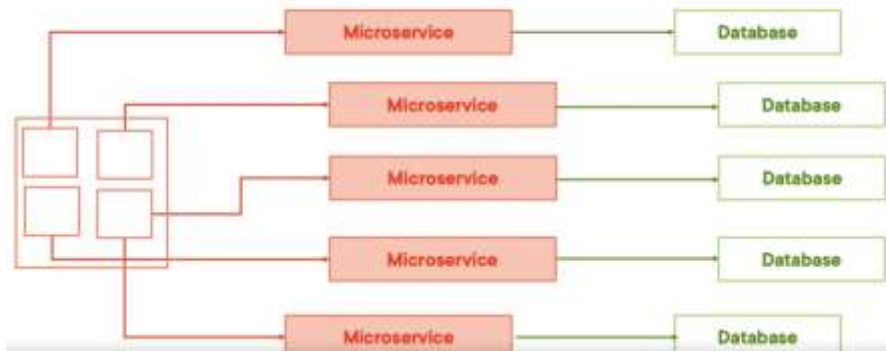


Each microservices might or might not have its own database –



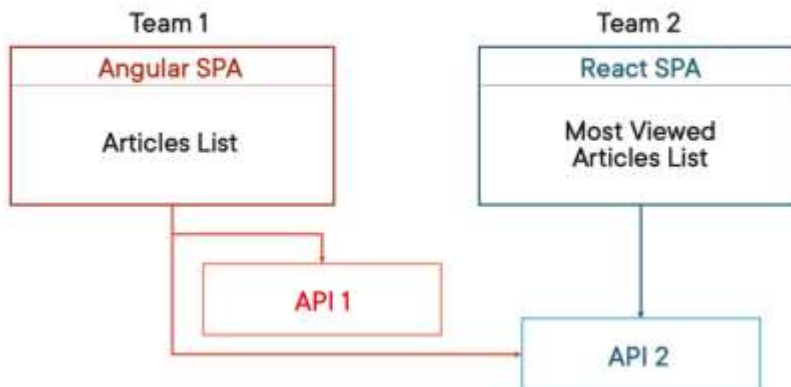
As the logic layer has been segregated into micro services, it is time for presentation layer which is called micro frontend. In this we create separate application for each micro service like for user service, creating page for user login, logout or details. For product service, creating pages for updating product details, etc.

Micro Frontend

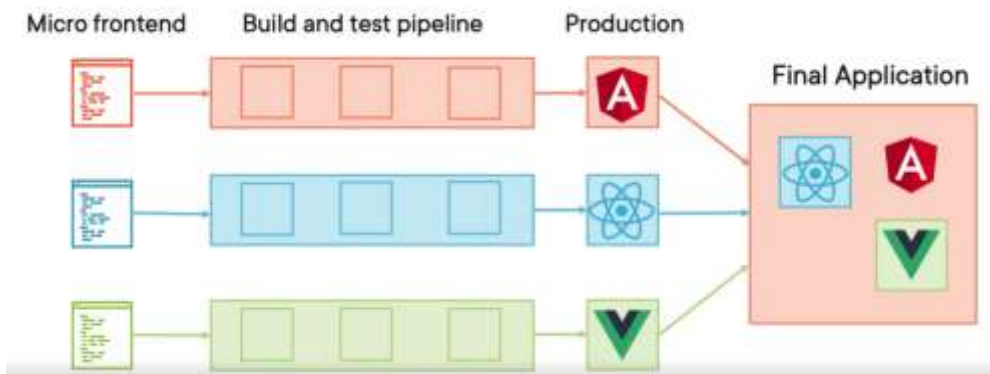


In this every interaction happens through the APIs

Micro Frontend



It provides benefits of individual deployments –

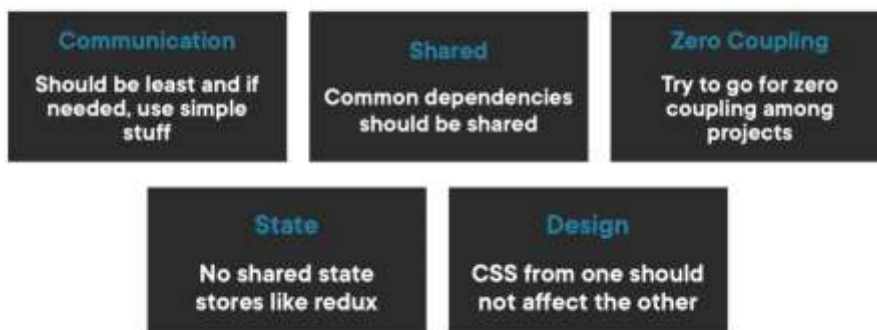


Before moving to micro frontend, we need to be sure that frontend is so huge that it can't be manage properly.

Pros	Cons
Incremental upgrades	Payload size
Simple, decoupled code bases	Operational complexity
Independent deployment	Increased cost
Autonomous teams	

Micro frontend expectations –

Micro Frontend Expectations



Challenges with Micro Frontends –

Communication among frontends

Sharing CSS or design issues

Sharing dependencies

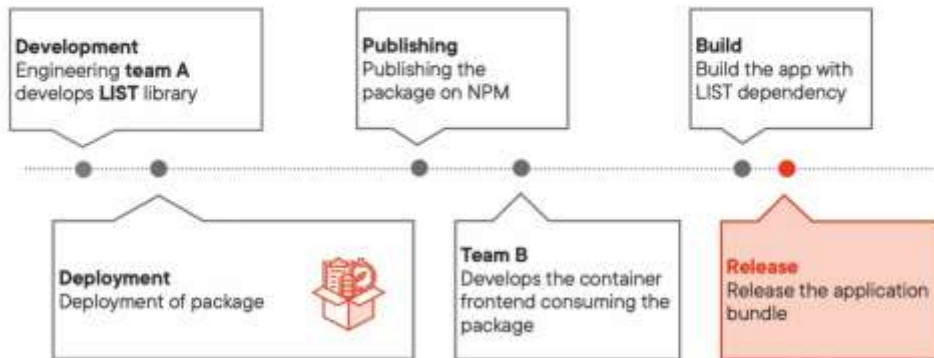
The communication should happen via callbacks or events, if communication has to happen or must. Ideally we should avoid the communication between frontends.

Integration approaches for these different frontend applications – server-side template composition (not recommended), build time integration, run time integration (via iFrames, javascript or web components).

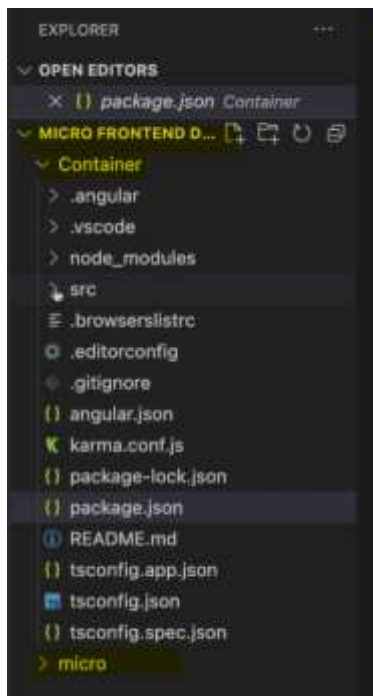
Integration refers to how different parts of a micro frontend get assembled to work as a single application.

In below approach we have a overhead that if team a release some new changes then team b need to take those changes in new package, build it and use it in their app then make a new release. Most of the companies follows this approach -

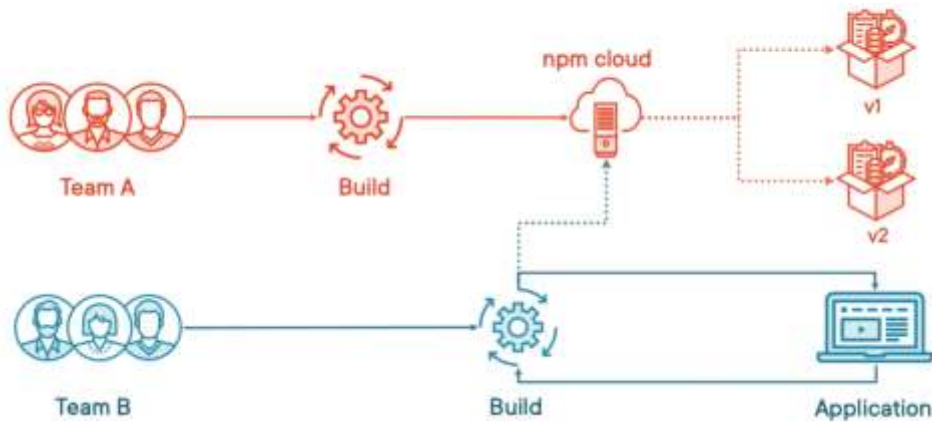
Build Time Integration Timeline



In container folder we will be having a standard angular application, in micro folder we will having library projects –

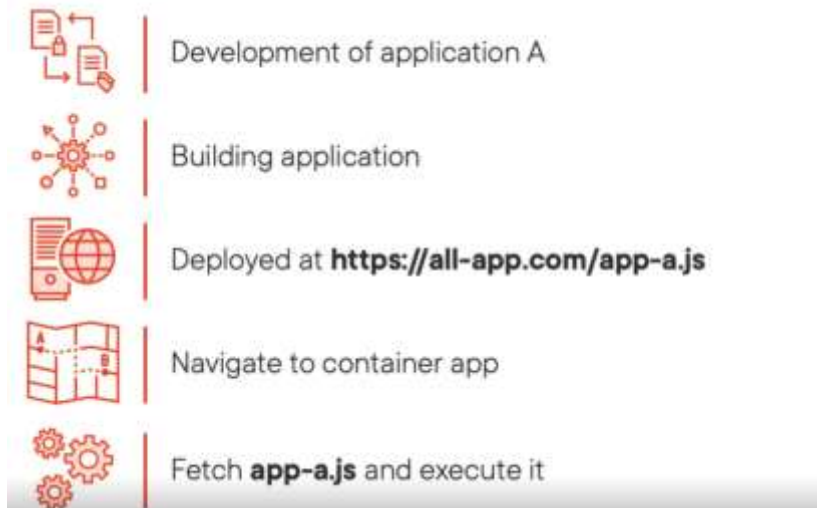


Build Time Integration



Runtime integration using iFrames – in the container application we will use a iFrame and will open a iFrame application server running URL to view the live changes.

Runtime integration via JavaScript – application A will be deployed on same URL which will be referred by the container application.



Difference

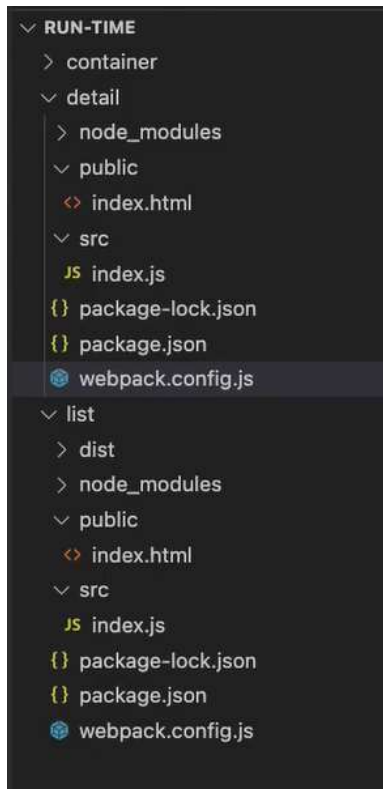
Build time	Runtime
Gives the container access to child app's source code before it's loaded in the browser	Gives the container access to child app's source code after it's loaded in the browser
Approach: using npm packages	Approach: webpack module federation, iframe

Experience is the name everyone gives to their mistakes. Oscan Wilde.

Implementing module federation using Webpack –



```
1 const HtmlWebpackPlugin = require('html-webpack-plugin');
2 const FederationPlugin = require('webpack/lib/container/ModuleFederationF
3 module.exports = {
4   mode: "development",
5   devServer: {
6     port: 4000
7   },
8   plugins: [
9     new FederationPlugin({
10      name: 'container',
11      remotes: {
12        list: 'list@http://localhost:4200/'
13      },
14    }),
15     new HtmlWebpackPlugin({
16       template: './public/index.html'
17     })
18   ]
19 }
```



If two applications use the sample NPM package, then we can mention it in the webpack configuration so that it doesn't load two copies of the library –

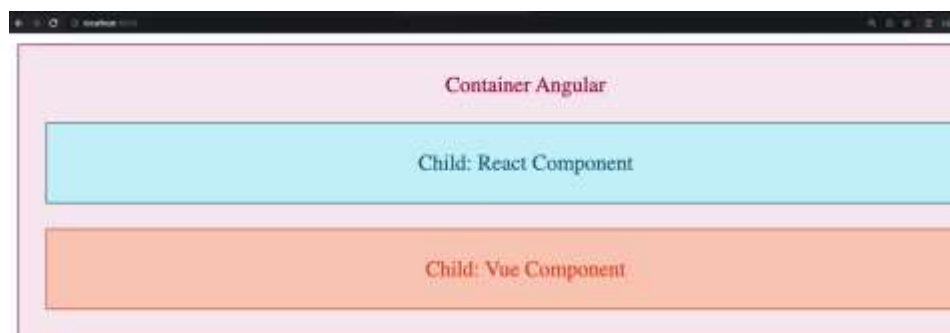
```
plugins: [  
  new FederationPlugin(  
    {  
      name: 'list',  
      filename: 'list',  
      exposes: {  
        './listIndex':  
          FederationPlugin,  
        './listIndexOptions':  
          options,  
      },  
      shared: ['lodash'],  
    },  
  ),  
  new HtmlWebpackPlugin(  
    {  
      template: './public/index.html',  
    },  
  ),  
]
```

With an Angular application instead of setting up the webpack files manually, we can use the `@angular-architects/module-federation` package to make it compatible with module federation.

```
bhubaneshpinani@shubhams-macbook container-angular % ng add @angular-architects/module-federation --port 4
200 --project container-angular
Skipping installation: Package already installed
CREATE webpack.config.js (1604 bytes)
CREATE webpack.prod.config.js (46 bytes)
CREATE src/bootstrap.ts (214 bytes)
UPDATE tsconfig.json (823 bytes)
UPDATE tsconfig.app.json (209 bytes)
UPDATE angular.json (2548 bytes)
UPDATE package.json (1237 bytes)
UPDATE src/main.ts (58 bytes)
✓ Packages installed successfully.
bhubaneshpinani@shubhams-macbook container-angular %
```

```
28 new ModuleFederationPlugin({
29   // library: { type: "module" },
30
31   // For remotes (please adjust)
32   // name: "containerAngular",
33   // filename: "remoteEntry.js",
34   // exposes: {
35   //   './Component': './src/app/app.component.ts',
36   // },
37
38   // For hosts (please adjust)
39   remotes: {
40     childVue: "childVue@http://localhost:8081/remoteEntry.js",
41     childReact: "childReact@http://localhost:8082/remoteEntry.js"
42   },
43 }
```

```
1 <router-outlet></router-outlet>
2 <app-react-app></app-react-app>
3 <app-vue-app></app-vue-app>
```

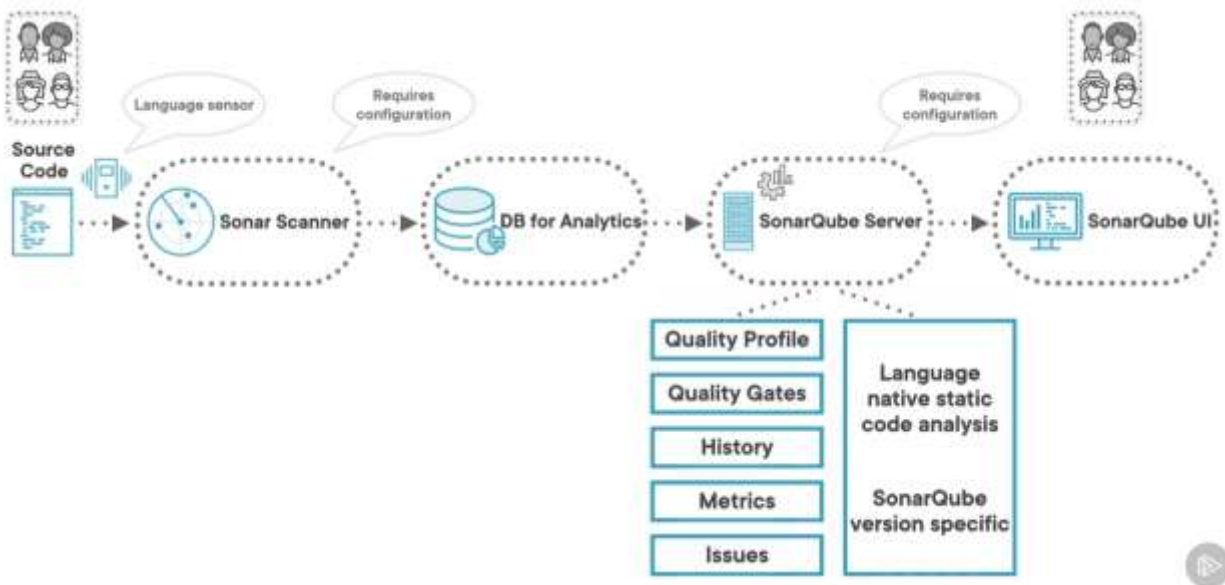


We can refer this repository to look more examples regarding the module federation - <https://github.com/module-federation/module-federation-examples>

Application Analysis with SonarQube

SonarQube allows us to do integrating the tool into our existing CI/CD pipeline and running it against every build to guarantee high quality of our codebase from a security standpoint. It is created by SonarSource. It detects bugs, code smells, vulnerabilities or hot spots.

Workflow –



SonarQube is based upon SAST (Static application security analysis). SAST solution analyse an application from the “inside out” and do not operate on a running system to perform a scan.