# Personal Information

SSN: 609700061

Name: Andy Wang

Address: US, 60 St. Francis Dr., Chico, CA

Zip Code:95926

Phone: 540-751-8691

DOB:24/11/1993

Email: [andynwang.us@gmail.com](mailto:andynwang.us@gmail.com)

# Resume

Andy Wang

Chico, CA | +1 (540) 751-8691 | andynwang.us@gmail.com

## Summary (React, Node Engineer)

Demonstrated history of excellence throughout multiple positions with several companies with a focus on Full Stack Development and Engineering.

- Front end developer with 9 years of experience designing, building, and implementing web and mobile applications

- Innovative problem-solver, designer, and coder with excellent time and task management skills.

- Well-versed in Scrum team participation, remote work communication and project management tools.

### Technical scope

- Frontend Development: React/Next.js, Vue.js/Nuxt.js, JavaScript, TypeScript

- Backend Development: Node.js/Express.js

- Mobile Development: React-Native

- Desktop App Development: React-Electron

## Education

Renmin University of China

Bachelor’s / Jan 2013

## Experience

### Centreville Tech LLC / Jan 2020 ~ Dec 2021

Centreville, Alabama, USA

**Senior Software Engineer**

 Develop rich front-end interfaces for online applications and websites for large clients in industries including medical, insurance, and technology.

• Intuitive, feedback driven feature implementation in React, Vue/Redux, VueX/Typescript

• Built out React/Vue.js SPA to leverage its declarative style, facilitate a modular, component-based design pattern, manage state locally, minimize re-renders, and accelerate front end development process through reusability of components

• Implemented Redux to manage a large state, leverage predictable state updates through use of dispatchers and reducers, and avoid unnecessary prop drilling to improve maintainability and scalability of front-end codebase

• Implemented React Native by leveraging its “out-of-the-box” compiler to architect a responsive, scalable, and cross-platform mobile application

### Level Ex, Inc / Mar 2016 ~ Jan 2020

Chicago, Illinois, USA

**Senior Software Engineer**

Create high quality user experiences by building interfaces that are easy to use, performant, and reliable

• Made an architecture design by using Redux and Redux-Saga for state management

• Delivered user-friendly interfaces (responsive and pixel-perfect pages) to the end-users

• Developed statically typed routing API for React frontend

• Directed conversion of React application to TypeScript

• Implemented React Hooks to accelerate front end rebuild by reducing code while maintaining backward compatibility

• Wrote readable and bug free code by using Typescript which improved the project performance.

• Used various testing frameworks to build more stable project

### One Brick Tech / Dec 2014 ~ Mar 2016

Huntersville, North Carolina, USA

**Software Engineer**

• Implemented stable React components and stand-alone functions to be added to any future pages.

• Designed CSS templates for use in all pages on the website working with CSS background, positioning, text, border, margin, padding, and table.

• Developed user interfaces by using ReactJS, Flux for SPA development.

• Used Web services (SOAP and RESTful) for transmission of large blocks of XML.JSON.

• Used OOP concepts to develop UI components that could be reused across the Web Application.

• Used JIRA as the bug tracking system to track and maintain the history of bugs/issues on an everyday basis.

• Integrated third-party APIs, Twillio, Facebook, Spotify, Twitter, etc.

### Beijing Craig Electronics Co., Ltd. / Oct 2013 ~ Oct 2014

Beijing, CN

**Junior Software Engineer**

Develop front end coding for client websites using HTML5, CSS3 and jQuery. Responsive design/ development using media queries. Cross-browser and legacy Internet Explorer support.

• Produced content pages with CSS3 layout and style markup presentations and also used JS methods and properties.

• Extensive experience using Bootstrap for layout.

• Built pixel-perfect and responsive pages from the designs.

• Updated the website from time to time for special Requirements.

• Wrote the effective JavaScript codes to improve performance.

## Skills

JavaScript, TypeScript, Node, React, React Native, Redux, Web3.js, HTML5/CSS3, AWS, Firebase, GraphQL, MongoDB, Unit Testing, docker, PostgreSQL

## Past Experiences

Apy.vision (React + Node)

Vissla.com (React + Django)

Mailerrize.com (React + Django)

Excoincial.co (React + Rails)

Tabrabbit.com (React + CI)

Arvioliidi.com (Vue + Node)

Thingsboard.io (Angular + Node + Java Spring Boot)

# Introduction

Let me introduce myself.

You know, my name is **Andy** and I am based in **California, US**.

As a **Senior Front-end developer**, I have a demonstrated history of excellence throughout multiple positions with several companies with a focus on **Full-stack development and Engineering**.

I have worked as **a Front-end developer** with 9 years of experience designing, building, and implementing web and mobile applications.

I am proud of my hard problem-solving and design skills and excellent time and task management skills. Also, I am well-versed in Agile/Scrum team participation, remote work communication, and project management tools including **Jira, Jenkins, Slack, GitHub, Bitbucket, and so on.**

Mostly I focused **on JavaScript and JavaScript-based libraries** and frameworks **including React.js/React Native, Next.js, Vue.js, Typescript, Electron.js, etc.**

For the last 2 years, I worked as a **Senior Front-end developer** in **Centreville Tech LLC** which is located in **Alabama, US**.

Here I developed rich Front-end interfaces for online applications and websites for large clients in industries including medical, insurance, and technology.

Working here, I created high-quality user experiences by building interfaces that are easy to use, performant, and reliable. I wrote readable and bug-free code by using Typescript which improved the project performance and used various testing frameworks to build projects more stable including Jest and Mocha.

I believe my experience has trained me to take on a role at your company and I'd love to work in your company in this position.

Is there anything else you need to know about me?

As a senior software engineer with more than 10 years of experience in web/mobile development, I have demonstrated a history of excellence throughout multiple contracts with several companies and freelance positions with a focus on Full Stack Development and Engineering.

As you can see on my profile, I worked as a software engineer at various Software Development Agencies. During that time, I got rich experience in various programming languages, frameworks, and libraries.

I am a big fan of JavaScript, so for the last 8 years, I am focused on full-stack development using several JavaScript frameworks and libraries like React/Next.js, Vue/Nuxt.js, AngularJS/Angular for frontend development, NodeJS based frameworks like Express, Nest, Hapi, and so on.

Moreover, I am a pro-active team player, fast learner, and problem-solver.

I love what I do and I do what I love.

No matter what I am working on I am very invested.

In my previous teams, mostly, we used to follow the agile methodology.

Communication is the cornerstone of the success of a project, as you know. This is why I always emphasize the importance of listening carefully to the requirements and details. We used to communicate with the teammates using Slack and for project management, we used Jira and Trello.

Outside of work, I enjoy playing football and listening to music. I find that music is a wonderful creative outlet and stress-reliever, as well as a perfect balance for my demanding career. That’s me. I am ready to get the ball rolling.

Looking forward to hearing from you very soon.

# Speaking about Experience

## 1. AWS Lambda experience

As an AWS cloud engineer with 4+ years practical experience, I have used Lambda service on the various purposes of my previous projects.

Mostly, I built serverless API endpoint for my web & mobile apps using Lambda (Node.js, Python or .NET Core) with API Gateway.

I am very familiar with its core concepts, advanced usages including concurrency management, throttling, lambda layer, efficient use of memory/cpu resource, diagnosing metrics, etc.

Also, extensively used it in connection with other services like SQS/SNS and Kinesis for data streaming.

## 2. DynamoDB/Cognito experience

DynamoDB: With its fast, scalable and NoSQL based features, I often used DynamoDB in my projects to support the data storage of WIP routines (temporary data store). I am quite familiar with creating/managing tables and indexes, running queries on various scenarios and also using it as a trigger of lambda functions.

Cognito: Used cognito user pool for user authentication / management and identity pool for authorization which means restricting access to various resources (like S3, DynamoDB, etc).

## 3. Serverless architecture

Combining the above mentioned services and others like RDS, SQS/SNS, API Gateway, Step functions, etc, I built several serverless architecture of large scale web / mobile platforms on AWS cloud.

Last year, I worked as a back-end developer/cloud engineer on a construction bid retriever service which runs in serverless architecture with extensive usage of lambda functions, dynamodb and simple queue service.

This platform serves as a SaaS product and in order to process unpredictable and highly scalable amount of requests in an efficient and reliable way, introduced the serverless architecture and orchestrated on the AWS cloud.

## 4. GoLang experience

In the past, I have been committed to some projects as a backend developer. At that time, I have developed microservice APIs using GoLang.

Briefly, I developed microservices to simplify the process of job advertisement on google ads and also bing ads.

And also I created internal GoLang modules to help eliminate boilerplate code and enhance developer productivity.

Wrote unit tests & developed integration tests using docker-compose.

That's my experience in GoLang.

## 5. React Experience

1. Designed CSS templates for use in all pages on the website working with CSS background, positioning, text, border, margin, padding, and table

2. Applied optimization techniques to reduce page size and load times to enhance user experience using sprites

3. Developed user interfaces by using ReactJS, Flux for SPA development

4. Used React-Router to turn the application into Single Page Application

5. Worked in using ReactJS components, Forms, Events, Keys, Router, Animations, and Flux concept.

6. Used Web services (SOAP and RESTful) for transmission of large blocks of XML.JSON

7. Implemented the Drag & Drop functionality using React-Draggable

8. Used React-Autocomplete for creating google maps location search on the webpages.

9. Used OOP concepts to develop UI components that could be reused across the Web Application

10. Used JIRA as the bug tracking system to track and maintain the history of bugs/issues on an everyday basis

11. Involved designing in web pages using HTML5, CSS3, JS, Bootstrap, SASS, LESS, React, Redux, Flex, MongoDB

12. Worked on ReactJS Virtual Dom and React views, rendering using components that contain additional components called custom HTML tags.

13. Implemented stable React components and stand-alone functions to be added to any future pages

14. Used ReactJS for templating for faster compilation and developing reusable components

15. Involved in development of User Interface using HTML5, CSS3, JS, jQuery, Angular

16. Extensively used Angular UI(Angular Bootstrap) for ng-grid, and UI components

17. Extensive experience using Bootstrap for layout

18. Followed SCRUM methodology

19. Produced content pages with CSS3 layout and style markup presentations and also used JS methods and properties

20. Updated the website from time to time for special Requirements

# Important Points for Interview

## While Interview

- Send resume before interview?

- Follow up email within 24 hours

- Never say I don’t know

- Never say It is on my resume

- Never ask what the company does

- Attitude

- Company knowledge

- Rehearse

- Meeting flow

- Follow-up About your self - 1 min, personal characteristics and skills that translate into career strengths

## Asking to Client/Recruiter

1. Ask open-ended questions.

2. Keep it short.

3. Don’t Interrupt

4. Getting to Yes

5. Use Inclusive Language

6. Ask Questions the Interviewer Can Answer

7. Avoid Questions that are Obvious or Easy to Determine

8. Avoid “Why” Questions

9. Avoid Asking Questions that Call for a Superlative

10. Avoid Leading or Loaded Questions

11. Avoid Veiled Threats

12. Avoid Questions that Hint of Desperation

13. Asking Questions that Focus on What the Company Can Do for You

14. Don't Ask Questions that are irrelevant to the job or organization.

15. Relax and smile.

# Intro Call Questions

## Why are you leaving your current job?

My last job has been a great learning experience and I have developed many skills during my 2 years with Centreville Tech LLC. I am now seeking a new opportunity with your company because I have finished all projects and the contract is ended due to date in last company and your company appears to fit well with the direction that I would like to take my career. I have researched and well-known great things about your company.

I am really excited to develop my expertise further with this opportunity.

## What is most important to you in your next position?

Three things that I’m looking for in my next job are a collaborative, team-focused culture, opportunities to learn and grow my skills from a technical standpoint, and a chance to learn more leadership skills over time. After reading the job description for this role, it sounds like this could be a good match for what I’m looking for, so I was excited to come interview for the position and learn more.

## What are you looking for in the next jobs?

Three things that I’m looking for in my next job are a collaborative, team-focused culture, opportunities to learn and grow my skills from a technical standpoint, and a chance to learn more leadership skills over time. After reading the job description for this role, it sounds like this could be a good match for what I’m looking for, so I was excited to come interview for the position and learn more.

## Why Our Company

For many years, I have worked in a Front-end developer role with big and medium clients and companies. That’s why I have proven experience in frontend development especially React and Typescript. You are now looking for a professional react/typescript developer, that’s why I applied for your job.

And the most important reason that I applied for your job is I am a Foodie just like you. :)

## What is your availability?

I'm flexible and available just about any time you need me to work. I am simply looking forward to joining the team and helping whenever I am most needed.

## Why do you think that you will be successful in this job?

As my resume reflects, I have been successful at each of my previous places of employment. Given my research about your company, the job description outlined, and the information we've exchanged today, l believe I have the skills and experience to fulfill what you're looking for, and I'm eager to contribute as an employee.

## What are your career objectives?

My career goal right now is to take the leadership and excellent track record I’ve obtained over the past decade and make the client's ideas real and grow them into a Fortune-500 company.

## Do you consider yourself successful?

Yes, I do consider myself successful. I think my willingness to take on new challenges and work hard sets me up for success. For example, I volunteered to manage a project for my company, which involved managing 20 staff members. I had never managed such a large staff before. However, due to my hard work, effective communication, and clear goals, I effectively managed the team, and we completed our project ahead of schedule. I don’t shy away from a challenge, and I know this would set me up for success at your company.

## What is your greatest strength?

I have a solid work ethic. When I'm working on a project, I don't just want to meet deadlines. Rather, I prefer to complete the project well ahead of schedule. Last year, I even earned a bonus for completing planned products successfully one month ahead of time.

## What is the most challenging problem recently you solved?

## Famous Spots

### Chico

Upper Bidwell Park, The Museum of Northern California, **Chico History Museum**, Gateway Science Museum

*Established in 1904 as a Carnegie Library belonging to the City of Chico, the original Romanesque building was remodeled in 1931 in Mediterranean Revival style and then leased to the Chico Museum Association in 1981, when a new library was built in conjunction with the County.*

### California

**Yosemite National Park**

*Yosemite National Park in Northern California is one of the United States' most scenic and most visited national parks. The mountains, valleys, rivers, and spectacular waterfalls have drawn tourists, artists, and athletes here for decades.*

*Most of the* [*key sights and things to do in Yosemite National Park*](https://www.planetware.com/california/things-to-do-in-yosemite-national-park-us-ca-278.htm) *are in Yosemite Valley.*

**Death Valley National Park**

*Death Valley National Park contains some of California's most inhospitable terrain, with extreme heat that has left this desert area strangely beautiful. Salt fields, dry parched land, sand dunes, mountains, unique rock formations, and a lake that lies below sea level create a unique landscape in this remote valley.*

*Some of the easiest to reach highlights in Death Valley are the sand dunes near Stove Pipe Wells, Badwater Basin, Zabriskie Point, and Dantes View. These and others are all easy to reach with a regular vehicle. If you have a 4WD vehicle, you can head out to more remote places like The Race Track.*

## Asking about Company

Where are you located in?

Can you please explain me about Hiring Process? (How many steps in your Hiring Process?)

What does work/life balance look like at your company?

What kind of Industry do you wanna build? (What kind of Industry do you provide?)

How many developers are there in your team?

Can you provide me clear specifications and designs? Using Figma or Wireframe?

What is your provided compensation? (What is the range of Compensation?)

Don’t you mind if you explain about compensation?

What is the Job Type? Full-time or Contract? (Are you seeking full-time developer or contractors only?)

What benefits do you provide for this job position?

## What gets you excited about coming to work?

Yeah. Hmm.

The first exciting thing is that I will work on what I love and what I'm passionate about.

Top trending technologies like AI, TypeScript, React, Next.js. and so on.

And the next thing is that I will work with people I love working with, I mean you guys. I took some time to familiarize myself with you, I mean, your team and your company. I have noticed that all of you are very kind and have rich experience and skills in your field.

And finally, as same as anyone, it is that I will work for the amount of money I love. Haha. You're right. As you mentioned in your job description, a competitive salary is another exciting thing that I can get. Yup. That's all.

## What was the last really great book you read?

Let me think, maybe, I think it's 'Harry Potter', a really exciting book. I like fantasy movies and books. My favorite fantasy movie is 'Naruto'.

Do you know Naruto Uzumaki? I remember some of his phrases like... "My friends were the first to accept me for who I am" "Failing doesn’t give you a reason to give up, as long as you believe." something like that.

## What surprises people about you?

-As I said before, I really like to work. I spend almost all my time working with challenging projects and learning new technologies.

-What else?

-Yeah, I am a night owl. lol. At SAP, I also worked as a professional remote worker in various global remote agile teams for foreign companies based in the US, UK, Germany, and so on. And the other reason is, you know, at night, there's nobody who bothers me, so I can all-in my time to think and work.

-Something else?

-Let me think. Yeah. The most important thing is writing 100% elegant and optimized code, provide 100% product. That's me.

## If you were going to start your own business, what would it be?

Yeah, it's one of my dreams. I'd love to build my own IT company. I'd like to start my career again step by step, software developer, project manager, CTO, and finally CEO of my own company. Yeah. It is just my dream. haha. Yup. That's one of the reasons why I moved to Hong Kong from Japan. But unfortunately, because of a family emergency, I went back to here, Tokyo again. If things go better, I will move to Hong Kong again asap.

## What’s the biggest problem in most offices today?

As you know, pandemic. So everyone prefers remote work at home. I hope that everyone is safe.

## What did you like most/least about your last company?

Hmm. maybe, working in a global agile team with different developers from all over the world is the most thing that I like at SAP. I worked as a senior software engineer and professional remote worker as I said before, so I have been given the opportunity to take part in various teams with no racism, no levels, no misunderstandings.

I never thought about the least thing at SAP. Everything was okay. Btw, I moved to Hong Kong to achieve my dream.

## Where/when/how do you do your best work?

Haha. Anywhere, anytime, anyway you want.

As I said before, I always prefer 100% complete work. So I put great importance on listening carefully to the requirements and details. And also with a professional vision, I always do my best to deliver exceptional results on time and keep the team members updated with the project progress and any blockers to resolve through discussion.

## When was the last time you made a big mistake at work?

Mistake... I didn't make any mistake till now. But if I must say about this, maybe, it is when I did not meet the timeline that the client wants because there are some misunderstandings between us. And anything else... I think there's no other mistake.

## How could a manager best support you?

-Hmm. When I work as a software engineer at SAP, the team leader in a team gave me recognition and praise. And also he provided feedback, mentorship, and training.

-What else?

-Maybe, providing strong leadership and a clear vision is another main point. Yeah. That's all.

## How do you handle stress or tight deadlines?

Stress... as you know, meeting the deadline is very important for us, I mean, software engineers. But, another important thing is the quality of production. I prefer both of us, but put more importance on quality.

So if the deadline is too tight, I explain the reason why I can not meet the deadline and why the quality is important, what else, how we can find an optimized way to build a perfect production. And also discuss it with teammates.

## What are your plans for the next five years?

I'd like to be a project manager, team leader, or CTO for the next five years.

## What three things do you need to succeed in this position?

Maybe, rich experience, communication skills, and the last thing is challenging to new things.

## What kind of events do you attend outside of work hours?

Before the pandemic, I used to take part in some meetings to exchange experiences with each other and also I like a weekend party. But as you know, nowadays, we should avoid going outside of the home.

so in my free time, I enjoy fitness training and listening to music. I find that music is a wonderful creative outlet and stress-reliever, as well as a perfect balance for my demanding career. It allows me to set personal goals and achieve them, which is also true of physical fitness.

I’ve found that keeping my mind and body sharp improves every single facet of my life. I like cartoon songs like 'Flying to your heart' in TinkerBell, 'Let it go' in Frozen, 'Try Everything' in Zootopia, and so on.

## What blogs or websites do you visit regularly?

Every day, I visit Twitter and I love to twit the phrases about programming.

## How would you describe your group of friends?

Yeah. In one word, 'AWESOME'. We love to exchange the experience with each other and help. You know, if someone has any problems with work or life and he asks us to help, we do all our best to help him. That's our friendship and our property.

## What do you do for fun?

Playing games. Watching cartoons.

## What motivates you to do your best work?

Maybe, work ethic.

## How do you prefer to communicate with coworkers?

I believe communication is the cornerstone of the success of a client-provided project. This is why I always emphasize the importance of listening carefully to the requirements.

In this way, I make sure my deliverance is more than satisfactory. I used to communicate with my teammates using Slack and for the project management, we used Jira and Trello.

If there's something that I need the design from the designers, I am very familiar with Figma.

## Who inspires you and why?

Myself, as a software engineer who loves new challenges, I always prefer to be the 1st in my favorite field, not the 2nd or 3rd.

So, I can say that love for new challenges myself inspires me to do my best.

## How would you describe our company culture?

Yeah, challenging and collaborating. You, I mean your company seems that developers are encouraged to explore the full potential of their skill sets and that they're likely to grow through job experience.

Regarding the collaborating, I have noticed that software engineers and your teams are working well together cross-functionally to accomplish your goals.

## What superpower will you bring to our company?

100% clean and optimized code and 100% perfect product.

## What has been the greatest disappointment of your life to date?

My biggest disappointment is that I wasn't able to follow my dream of being a professional photographer when I was a kid. Even though I was disappointed at the time, I realize now that if I had taken that direction, I would not have achieved my current degrees and developed a career that I love.

## What tools or apps allow you to work more efficiently?

I used to communicate with my teammates using Slack and for the project management, we used Jira and Trello. If there's something that I need the design from the designers, I am very familiar with Figma.

## How do you manage conflict with coworkers?

Good point. As you know, managing conflicts is very important in teamwork. So in that case, I find the conflicts part and take some time to review it.

And after that, I discuss with my teammates who wrote that part to manage these conflicts. After deep discussion, we can handle the conflicts together.

## Would you rather work alone or with a team?

Of course, in a team. As you know, anyone can not do anything alone because everyone has limited knowledge and experience.

So, without any help, without any collaboration, you are nothing. That's my phrase. haha.

## What would be your ideal work schedule?

I would probably want to start at 8 AM and finish up around noon before lunch. And start again at 3 PM and finish the work around 7 PM. From 12 PM to 3 PM, I should handle the problem with my family, as I said before, I came here, Tokyo for handling a family emergency.

## What are the four types of burn-down charts?

The four types of burn-down charts are the product burn-down chart, the sprint burn-down chart, the release burn-down chart, and the defect burn-down chart.

## How do you feel about working in a team environment?

My ideal work environment is one that focuses on teamwork.

I like companies like yours that put an effort in gelling the group together and making sure everyone is supported. I like that sort of structure and formality of knowing what you’re supposed to do and when you are aware of the situation. I work best when I have a group of positive people working around me and where I can focus on putting my talent to work.

## What interested you about the position?

As same as all programmers, I can list the three things that are very important for me. They are production, team, and technology. Since you are building wonderful products for customers, you have a great team of high-skilled developers, and the technology that you are using is one of my main technical skills, I am very interested in this opportunity and I think I can be a good fit for this position.

## Why are you leaving your current position?

I am currently working as a part-time developer at Sidero Startup company. Since all tasks are almost done, I only work 5 hours per week. So a few days ago, I started looking for a new opportunity as a full-time position.

## What are you looking for long term?

My long-term objectives are to become a member of the elite team and work on enterprise-level projects.

## Where do you hope your career will take you in the next 5 years?

For the next 5 or 7 years, I would like to be a project manager or team leader of the team. And after 10 years, I would like to build my own IT company.

## What do you want to do in the next few years?

While working on top-trending tech-stacks with high-skilled developers from all over the world, I would like to learn more about how to make my dream come true.

## What gets you really excited in a job?

As I described before, three things - products, team, and tech-stack.

## What do you do in your current position? What do you like about it? What do you not?

Currently, I am working as a senior full-stack developer(part-time) at Sidero Startup company. You know, this company is my friend's company. I helped him to grow his company and his business. And also I mentored some Junior developers while working with this company. All tasks are almost done.

## Are you comfortable leading a team and have you before? Give us examples of if you have and/or if you were to.

I have some valuable experience in working as a team lead or project manager but for now, I would like to work as an individual developer since I noticed that managing other people is not my main point.

## What's important to you? If you had to pick what your top 6 "values" are in life and at work, what are they and why?

I always follow the core values below.

- Growth: Eagerly learn new things. Have broad interests and be curious about the world. Strive for perfection. Share my knowledge with others.

- Responsibility: Take on uncomfortable tasks that you can handle. Don't be afraid of the risk associated with decision-making. Don't run away from mistakes. Quickly communicate problems.

- Commitment: Take the initiative. Care about the success of the team. Go beyond what is expected. Find passion in what you do.

- Openness: Be brave and question decisions that you don't agree with. Openly talk about problems. Dedicate time to helping others. Approach change positively.

- Delivering value: Recognize what has to be done well from the very beginning. Make complex things simple and know how to choose 20% of work that provides 80% of the results. Deliver high-quality solutions while keeping the picture in mind.

- Love: Do what you love and love what you do. No matting you are working on, you should be very invested.

## Do you prefer to be told exactly what to do, or do you prefer to work with little direction?

Following the core values, I can say that I prefer to be told exactly what to do.

## How do you stay up to date on all the latest technologies?

I always try to keep my tech-stack up-to-date to be ready for any kind of development while reading articles on LinkedIn, Medium, Dev.to and so on.

## Can you rate yourself for the top 5 technologies you are most familiar with - from 1-10? 1 being not at all, and 10 being you could write an O'Reilly book on the tech.

React - 10, React Native - 10, Vue - 8, Node - 9, GoLang - 8

## Are you comfortable in all technologies and would you be willing to work across all technologies? Or, do you prefer to stay in one technology?

As I mentioned before, I always try to learn new technologies so right now I am learning Python and Tensorflow.

## What technologies are you most comfortable with?

Since I am a JS guy so I am most comfortable with JS frameworks and libraries, especially React.

## What would you say your strengths are, and your weaknesses?

I can say my strengths in one word - 100%. I keep the core values in my mind so I always try to provide 100% perfect production. Because of this, sometimes it takes more time to complete the task or ticket.

## Give us an example of when someone told you to do something that was technically wrong and/or you disagreed with the architect. How do you deal with the situation?

As you know, communication is the cornerstone of the success of a project so I always put great importance on listening carefully to the requirements and details from the customers and teammates. After listening carefully, I will explain why it is impossible or why it is wrong with good examples in articles.

Here, the most important thing is respecting for each other.

## What do you think the hardest part about working remotely is?

From my perspective, the hardest part and also the most important part is communicating actively.

## How do you plan on working remotely?

Overlapping working hours with client's business hours - 6 or 8 hours.

## What are some of your pet peeves about working with people?

I think a clear specification is one of the most important aspects to produce a well-organized and clean solution.

## What if you become really frustrated with a teammate? What are you going to do?

While working in a team, I know that I can meet such kinds of situations. In that case, I first try my best to keep professionalism. I will listen to his/her explanation carefully first and will give him my thought in a professional and kind manner. I am sure that losing temper is not the right option in any case.

## What are you looking for financially?

120K - 140K per year

Looking forward to hearing from you very soon.

# React Questions

## Why React?

React is officially defined as a “Javascript library for building user interfaces.” Basically, the idea is to build web applications with rich UI elements. It facilitates smoother communication between developers and web page DOMs (Document Object Models), therefore eliminating the need for the DOM API.

Developers achieve this via composable UIs which are more efficient than traditional ideas of operating the backend of web applications. Using the Model-View-Controller (MVC) architectural model, React represents the ‘View’ section.

## What are components?

Components are the main building blocks.

When building an app, the entire app is divided into components and writing code instructs the components on interaction and so on. Each component is a functional UI element that forms a building block of an application. Further, the two main features of components; they are:

* **Composable** – can be reused and new components can be iterated from former ones.
* **Independent** – each component, though linked with others, still stands apart. So, when you fix the code in one section, you still preserve the rest…

## What are the main features of React?

* **Declarative UI**: as against the traditional imperative mode of operating DOM APIs, this library uses a declarative model. Its components are declarative in that the developer does not have to define step-by-step instructions on how the application must operate. Instead, by declaring each visual element of the UI and its contextual data, the app responds by building a dynamic, interactive UI.
* **State**: the state is the contextual data of a component. Developers can manipulate the state by changing a component’s data properties. Components can also create and manage their own data using this State feature. The setState method is used to modify a component’s data properties.
* **JSX**: this stands for ‘Javascript XML’ and it is what makes it possible to write HTML. It is a Javascript syntax extension that’s used to produce elements. Some React front end developer interview questions may include live JSX coding online test.
* **Redux**: Redux is a JavaScript state management tool most commonly used with React. It’s described as “a predictable state container for JavaScript apps.” Redux often proves useful whenever a developer hits some limitation with the component state.
* **Events**: these are Reactions triggered by a user’s action in interacting with the UI such as clicking a button, pressing a key, etc. Alternatively, events may be system-generated.

## What are the advantages of React?

* **Flexibility**: it allows developers to build upon any desired technology stack. Therefore, unlike other frameworks, developers are free of restrictions to rigid architectures and workflows.
* **Performance optimization**: declarative programming enhances performance by creating a virtual DOM by allowing the developer to write HTML directly into JavaScript files. The virtual DOM is much easier to manipulate than the real DOM and so that improves efficiency. Answers to ReactJS coding interview questions such as this should explain how it enables building rich user interfaces for mobile and web applications.
* **SEO Efficiency**: it is used for building web applications; therefore, SEO is a strong factor in considering it. Thankfully, it is SEO-friendly. This is despite the fact that search engines commonly have problems reading apps that are heavy on JavaScript. The virtual DOM feature makes this less of a challenge. This may also feature in the React JS advanced interview questions and answers section.
* **Productivity Improvement**: developers commonly describe as the best feature the ability to reuse system components. This helps them to deal with issues concerning the library’s complex logic. For companies, the ability to reuse code assets helps in setting internal coding standards and app consistency across the board.
* **Learn once – write anywhere**: while React is popular for web development, there is the React Native framework, which enables building web applications. Again, this improves cross-platform consistency and efficiency. It also improves the work of developers, since, for all platforms, they are working with similar tools. This is an important part of React frontend developer interview questions.

## What is the difference between ReactJS and React Native?

In simple terms, ReactJS is a library, while React Native is a framework. Specifically, the latter is a mobile framework that enables cross-platform mobile development. It is commonly used to build apps meant for Android and iOS. Apps that use it include Skype, Pinterest, Uber Eats, etc.

On the other hand, ReactJS is a library for building the user interface (UI) of web applications. Keep in mind that when people simply use ‘React’, they are most often referring to ReactJS. The ‘native’ factor stems from the fact that it uses native APIs to render components on the app.

## What are the differences between React and AngularJS?

First, while AngularJS is a JavaScript framework, React is a JavaScript library. Therefore, because of the concept of inversion of control, the latter enables developers to have a greater charge of the application flow.

AngularJS uses an actual DOM while React uses the more recent Virtual DOM technology. Therefore, the latter is more suitable for applications requiring dynamic content and regular updates. That’s why some popular apps that use React include Facebook, Netflix, WhatsApp, Discord, etc. More so, a smaller library size makes its apps more lightweight than AngularJS.

AngularJS uses a two-way binding approach, which seems easier at first glance, but it is inefficient in the face of code-heavy applications. Its one-way binding allows developers to efficiently manage large code.

## What are React Component life cycles?

The three main phases of a lifecycle: mounting, updating, and unmounting. Each features different methods, although they can be overridden by the programmer.

* **Mounting**: this happens when elements are entered into the DOM. That is when the component is initially created. The most common methods here are constructor(), render(), and componentDidMount().
* **Updating**: when one triggers a change to state, the component is re-rendered. Common methods are render() and componentDidUpdate().
* **Unmounting**: this is removing a component. The componentWillUnmount() method applies here.

## What differentiates the class component from the functional component?

The preeminent difference between both types of components is syntax. A functional component accepts properties in plain JavaScript functions and returns HTML in the form of JSX elements. As such, they are also known as stateless components. More so, they do not use any render method.

However, class components, also called stateful components, create render functions that extend React’s component classes. While the former is straightforward and easy to read, class components use complex UI logic. Moreover, functional components do not support lifecycle hooks though class components do.

## Explain Controlled vs Uncontrolled Components in React?

One can have a form handled by a React component or alternatively by the DOM directly. Forms implemented in the former manner are called controlled components, while for the latter, they are uncontrolled components.

Uncontrolled components are similar to traditional HTML in that the state is stored internally, although you can still retrieve the current value from the DOM by using a ref. Moreover, controlled components maintain their internal state and allow for validation control, unlike uncontrolled components, which do neither. However, React mostly recommends controlled components for implementing forms.

## How do you render elements in React?

In React, elements are the ‘smallest building blocks’ of apps. Elements are the plain objects returned from components and rendered into the DOM. Elements determine what you see on the browser screen but they themselves are not seen; rather they are stored in the app’s memory.

Therefore, every component has an element that determines the properties, type, etc. of that component. Basically, one renders elements into the root DOM node by using ReactDOM.render(). In addition, since React elements cannot be changed neither can their attributes be updated, one has to use the same code function to create a new element in order to implement an update to the UI.

## What is the difference between props and state?

Properties (props) are read-only. They determine the configuration of a component as received from the parent. If React components are functions, you may think of props are arguments for a function. Props remain immutable throughout the lifecycle of a component.

On the other hand, the state is dynamic. Components don’t have a default state passed down from parents. Hence, the state of a component is mutable, representing a data value that can be updateable when certain conditions are fulfilled. Thus, the state is essentially what makes dynamic rendering possible.

## What are some lifecycle methods of React components?

* **render()** – the most commonly used lifecycle method and the only one required in a class component.
* **constructor()** – a method called before mounting a component. This method is not necessary if you didn’t initialize state yet or bind methods.
* **componentDidMount()** – the first method called after a component is mounted.
* **componentDidUpdate()** – this method is implemented once a component is updated, though not at initial rendering.
* **componentWillUnmount()** – to be implemented before you umount a component is unmounted and also used to perform due cleanups.

Those are the common lifecycle methods. Some are rarer, including shouldComponentUpdate(), static getDerivedStateFromProps(), getSnapshotBeforeUpdate(), static getDerivedStateFromError(), and so on. This question may also easily fall under ReactJS experienced interview questions for senior developers if you need to establish their understanding of components.

## How does Virtual DOM Work?

React allows developers to write HTML into JavaScript files. This empowers developers to maintain control of their HTML code. The Document Object Model (DOM) is an API that works with HTML and XML files. Therefore, it is very useful in building web applications. However, React deviates from the traditional approach to using DOM.

With a virtual DOM, one can keep a representation of the DOM in memory. Thus, virtual DOM is lightweight than the real DOM and does not demand as many resources to maintain. However, through the process of reconciliation, the virtual DOM may be synced with the real DOM via a library.

Unlike a real DOM, one can use a virtual DOM to directly interact with the interface, even though it shares properties with a real DOM.

Many JavaScript framework operations require rapid updates to the DOM. This can slow down the application a lot. However, with virtual DOM, the time cost is greatly reduced.

## What is Higher-Order Component (HOC)?

Higher-Order Components are used to add sophisticated functionalities to existing components. The purpose of the HOC technique, which is common in third-party React libraries, is to reuse component logic.

Therefore, the use of every Higher-Order component function returns a new component. HOCs in React are related to higher-order functions in JavaScript. Higher-order components find implementation in Redux connect (which is a higher-order function), state abstraction, and so on.

## How does conditional rendering work in React?

Conditional rendering is the dynamic output of UI markups depending on the state of a condition. It works similarly to JavaScript conditions. With conditional rendering, one can toggle certain application functions, determine permission levels, show or hide elements, handle authentication, render API data, and so on.

There are various ways to implement conditional rendering in React. Some of the expected answers for this particular one of the React questions for senior developer candidates are:

* Using the **if-else conditional logic** (suitable for smaller and medium-sized apps)
* Using **element variables**, which actually enables writing cleaner code
* Using **ternary operators**, which take some complication away from if-else statements, etc.

## What are the drawbacks of React?

Interview questions on ReactJS for experienced developers such as this help you to determine that a person knows what they are doing. For someone to claim expertise over a platform, they need to be well acquainted with its limitations and challenges, so as to know how to get around them.

JSX: although it is well-established that JSX has its benefits such as enhancing app performance, developers find it difficult to learn and use. Many developers don’t have JSX foundation as a syntax extension. Hence, they find it complex.

The rapid pace of development in React can also be a disadvantage because it means developers have to continually relearn new concepts as updates are released. It also makes it difficult to keep up with documentation if updates always result in significant changes. However, there has been some improvement on this end recently, React’s core API is now much stable than before so it’s indeed getting better.

Also included is how the platform is highly flexible and the lack of conventional standards can make things complex particularly when onboarding new developers. Every React development team seems to have its own separate standards that developers have to learn to work with. In any case, this challenge can be mitigated through proper onboarding, as we do at ProCoders.

React is rich enough as a library, but developers coming from platforms such as Angular might miss some of the features associated with frameworks. For instance, on the Model-View-Controller structure algorithms, React represents only one part: the ‘View’ part. This gap can make React poorly structured and poorly managed.

## What are React Hooks?

React Hooks are still a relatively new feature. That’s why this question is in the mid-level section. Many people haven’t gotten the hang of hooks completely. React Hooks were introduced in version 16.8, which was released in February 2019. For context, the latest version of React is version 17. So, it obviously has not been much longer than hooks were introduced. In any case, hooks are backward-compatible, so that’s one potential problem avoided.

In simple terms, hooks allow a programmer to manipulate the state of stateless functional components without having to convert them into stateful class components. Essentially, with hooks, one no longer needs to write a class to use state and other React features. Some common hooks are:

* **setState** – return a function to update a stateful value
* **useEffect** – implementing side effects in function components
* **useContext** – return the current context value.

Users can also create their own custom hooks, for instance, by using higher-order components. The two rules of hooks are that one must only call hooks at the top level and that one can only call hooks from function components.

## Business Benefits of React?

Because of reusable components, React developers don’t have to write new code every time for different features that have similar functionalities. This improves the speed of the development process and reduces the time-to-market, essentially making the business more competitive in the industry.

React enables building SEO-friendly apps, and the benefits of SEO to marketing and business growth in this day cannot be overstated. By boosting page performance and lowering loading time, a business can improve the performance and user experience of its apps. This is a plus for SEO since website performance is a critical ranking factor of search engines.

React apps are easy to maintain. This lowers incidents of downtime, which may affect customer experience. In addition, React is a cost-optimized solution for building web apps.

Finally, to add to this, when answering these interview questions on ReactJS for experienced developers, they should be able to paint a picture of the future of React and its continued viability in the development stage.

## What is JSX?

JSX is a type of XML/HTML syntax that helps convert ECMAScript to JavaScript code. It is important because it can help create Dom-like structures that allow it to be placed into the same file as JavaScript, where it can be further converted into JavaScript code.

## What is Redux and what are its components?

Redux is a popular library used for front-end development. It has no dependencies and is often used to help create JavaScript applications. Redux is made up of three components: actions, reducers and the store.

* **Store**: Holds the state of the application.
* **Action**: The source information for the store.
* **Reducer**: Specifies how the application's state changes in response to actions sent to the store.

## How do you identify the difference between DOM and virtual DOM?

DOM is considered an abstraction of an HTML code that provides a visual representation of the written text displayed by the HTML code. Virtual DOM is related to HTML DOM in that it is considered to be derived from it, and is therefore considered an abstraction of an abstraction.

## What are stateless components?

Stateless components are components that instead of possessing their own state, they take on whatever form that is applied to them. For this reason, they can also be called reusable components.

## What are synthetic events in React?

* Synthetic events combine the response of different browser's native events into one API, ensuring that the events are consistent across different browsers.
* The application is consistent regardless of the browser it is running in. Here, preventDefault is a synthetic event.

## Why is there a need for using keys in Lists?

Keys are very important in lists for the following reasons:

* A key is a unique identifier and it is used to identify which items have changed, been updated or deleted from the lists
* It also helps to determine which components need to be re-rendered instead of re-rendering all the components every time. Therefore, it increases performance, as only the updated components are re-rendered

## How is React different from Angular?

|  |  |  |
| --- | --- | --- |
|  | Angular | React |
| Author | Google | Facebook |
| Architecture | Complete MVC | View layer of MVC |
| DOM | Real DOM | Virtual DOM |
| Data-Binding | Bi-directional | Uni-directional |
| Rendering | Client-Side | Server-Side |
| Performance | Comparatively slow | Faster due to Virtual DOM |

## What is a state in React?

* The state is a built-in React object that is used to contain data or information about the component. The state in a component can change over time, and whenever it changes, the component re-renders.
* The change in state can happen as a response to user action or system-generated events. It determines the behavior of the component and how it will render.

## What is React Router?

React Router is a routing library built on top of React, which is used to create routes in a React application.

## Why do we need to React Router?

* It maintains consistent structure and behavior and is used to develop single-page web applications.
* Enables multiple views in a single application by defining multiple routes in the React application.

## Differentiate between Real DOM and Virtual DOM.

|  |  |
| --- | --- |
| **Real DOM vs Virtual DOM** | |
| **Real DOM** | **Virtual  DOM** |
| 1. It updates slow. | 1. It updates faster. |
| 2. Can directly update HTML. | 2. Can’t directly update HTML. |
| 3. Creates a new DOM if element updates. | 3. Updates the JSX if element updates. |
| 4. DOM manipulation is very expensive. | 4. DOM manipulation is very easy. |
| 5. Too much of memory wastage. | 5. No memory wastage. |

## What is React?

* React is a front-end JavaScript library developed by Facebook in 2011.
* It follows the component based approach which helps in building reusable UI components.
* It is used for developing complex and interactive web and mobile UI.
* Even though it was open-sourced only in 2015, it has one of the largest communities supporting it.

## What are the features of React?

Major features of React are listed below:

1. It uses the **virtual DOM** instead of the real DOM.
2. It uses **server-side rendering**.
3. It follows **uni-directional data flow** or data binding.

## List some of the major advantages of React.

Some of the major advantages of React are:

1. It increases the application’s performance
2. It can be conveniently used on the client as well as server side
3. Because of JSX, code’s readability increases
4. React is easy to integrate with other frameworks like Meteor, Angular, etc
5. Using React, writing UI test cases become extremely easy

## What are the limitations of React?

Limitations of React are listed below:

1. React is just a library, not a full-blown framework
2. Its library is very large and takes time to understand
3. It can be little difficult for the novice programmers to understand
4. Coding gets complex as it uses inline templating and JSX

## What is JSX?

JSX is a shorthand for JavaScript XML. This is a type of file used by React which utilizes the expressiveness of JavaScript along with HTML like template syntax. This makes the HTML file really easy to understand. This file makes applications robust and boosts its performance. Below is an example of JSX:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | render(){  **return**(    <div>    <h1> Hello World from Edureka!!</h1>             </div>        );  } |

## What do you understand by Virtual DOM? Explain its works.

A virtual DOM is a lightweight JavaScript object which originally is just a copy of the real DOM. It is a node tree that lists the elements, their attributes and content as Objects and their properties. React’s render function creates a node tree out of the React components. It then updates this tree in response to the mutations in the data model which is caused by various actions done by the user or by the system.  
This Virtual DOM works in three simple steps.

1. Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation.



1. Then the difference between the previous DOM representation and the new one is calculated.



1. Once the calculations are done, the real DOM will be updated with only the things that have actually changed.



## Why can’t browsers read JSX?

Browsers can only read JavaScript objects but JSX in not a regular JavaScript object. Thus to enable a browser to read JSX, first, we need to transform JSX file into a JavaScript object using JSX transformers like Babel and then pass it to the browser.

## How different is React’s ES6 syntax when compared to ES5?

Syntax has changed from ES5 to ES6 in the following aspects:

1. require vs import

|  |  |
| --- | --- |
| 1  2  3  4  5 | // ES5  var React = require('react');    // ES6  **import** React from 'react'; |

1. export vs exports

|  |  |
| --- | --- |
| 1  2  3  4  5 | // ES5  module.exports = Component;    // ES6  export **default** Component; |

1. component and function

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | // ES5  var MyComponent = React.createClass({      render: function() {  **return**    <h3>Hello Edureka!</h3>  ;      }  });    // ES6  **class** MyComponent **extends** React.Component {      render() {  **return**    <h3>Hello Edureka!</h3>  ;      }  } |

1. props

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | // ES5  var App = React.createClass({      propTypes: { name: React.PropTypes.string },      render: function() {  **return**    <h3>Hello, {**this**.props.name}!</h3>  ;      }  });    // ES6  **class** App **extends** React.Component {      render() {  **return**    <h3>Hello, {**this**.props.name}!</h3>  ;      }  } |

1. state

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | // ES5  var App = React.createClass({      getInitialState: function() {  **return** { name: 'world' };      },      render: function() {  **return**    <h3>Hello, {**this**.state.name}!</h3>  ;      }  });    // ES6  **class** App **extends** React.Component {      constructor() {  **super**();  **this**.state = { name: 'world' };      }      render() {  **return**    <h3>Hello, {**this**.state.name}!</h3>  ;      }  } |

## How is React different from Angular?

|  |  |  |
| --- | --- | --- |
| **React vs Angular** | | |
| **TOPIC** | **REACT** | **ANGULAR** |
| 1. ARCHITECTURE | Only the View of MVC | Complete MVC |
| 2. RENDERING | Server-side rendering | Client-side rendering |
| 3. DOM | Uses virtual DOM | Uses real DOM |
| 4. DATA BINDING | One-way data binding | Two-way data binding |
| 5. DEBUGGING | Compile time debugging | Runtime debugging |
| 6. AUTHOR | Facebook | Google |

## “In React, everything is a component.” Explain.

Components are the building blocks of a React application’s UI. These components split up the entire UI into small independent and reusable pieces. Then it renders each of these components independent of each other without affecting the rest of the UI.

## What is the purpose of render() in React.

Each React component must have a **render()**mandatorily. It returns a single React element which is the representation of the native DOM component. If more than one HTML element needs to be rendered, then they must be grouped together inside one enclosing tag such as **<form>, <group>,<div>** etc. This function must be kept pure i.e., it must return the same result each time it is invoked.

## How can you embed two or more components into one?

We can embed components into one in the following way:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | **class** MyComponent **extends** React.Component{      render(){  **return**(    <div>    <h1>Hello</h1>                    <Header/>              </div>            );      }  }  **class** Header **extends** React.Component{      render(){  **return**    <h1>Header Component</h1>       };  }  ReactDOM.render(      <MyComponent/>, document.getElementById('content')  ); |

## What is Props?

Props is the shorthand for Properties in React. They are read-only components which must be kept pure i.e. immutable. They are always passed down from the parent to the child components throughout the application. A child component can never send a prop back to the parent component. This help in maintaining the unidirectional data flow and are generally used to render the dynamically generated data.

## What is a state in React and how is it used?

States are the heart of React components. States are the source of data and must be kept as simple as possible. Basically, states are the objects which determine components rendering and behavior. They are mutable unlike the props and create dynamic and interactive components. They are accessed via **this.state().**

## Differentiate between states and props.

|  |  |  |
| --- | --- | --- |
| **States vs Props** | | |
| **Conditions** | **State** | **Props** |
| 1. Receive initial value from parent component | Yes | Yes |
| 2. Parent component can change value | No | Yes |
| 3. Set default values inside component | Yes | Yes |
| 4. Changes inside component | Yes | No |
| 5. Set initial value for child components | Yes | Yes |
| 6. Changes inside child components | No | Yes |

## How can you update the state of a component?

State of a component can be updated using this.setState().

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | **class** MyComponent **extends** React.Component {      constructor() {  **super**();  **this**.state = {              name: 'Maxx',              id: '101'          }      }      render()          {              setTimeout(()=>{**this**.setState({name:'Jaeha', id:'222'})},2000)  **return** (    <div>    <h1>Hello {**this**.state.name}</h1>    <h2>Your Id is {**this**.state.id}</h2>                       </div>                );          }      }  ReactDOM.render(      <MyComponent/>, document.getElementById('content')  ); |

## What is arrow function in React? How is it used?

Arrow functions are more of brief syntax for writing the function expression. They are also called ‘fat arrow‘ (**=>**) the functions. These functions allow to bind the context of the components properly since in ES6 auto binding is not available by default. Arrow functions are mostly useful while working with the higher order functions.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | //General way  render() {  **return**(          <MyInput onChange={**this**.handleChange.bind(**this**) } />      );  }  //With Arrow Function  render() {  **return**(          <MyInput onChange={ (e) => **this**.handleOnChange(e) } />      );  } |

## Differentiate between stateful and stateless components.

|  |  |
| --- | --- |
| **Stateful vs Stateless** | |
| **Stateful Component** | **Stateless Component** |
| 1. Stores info about component’s state change in memory | 1. Calculates the internal state of the components |
| 2. Have authority to change state | 2. Do not have the authority to change state |
| 3. Contains the knowledge of past, current and possible future changes in state | 3. Contains no knowledge of past, current and possible future state changes |
| 4. Stateless components notify them about the requirement of the state change, then they send down the props to them. | 4. They receive the props from the Stateful components and treat them as callback functions. |

## What are the different phases of React component’s lifecycle?

There are three different phases of React component’s lifecycle:

1. Initial Rendering Phase: This is the phase when the component is about to start its life journey and make its way to the DOM.
2. Updating Phase: Once the component gets added to the DOM, it can potentially update and re-render only when a prop or state change occurs. That happens only in this phase.
3. Unmounting Phase: This is the final phase of a component’s life cycle in which the component is destroyed and removed from the DOM.

## Explain the lifecycle methods of React components in detail.

Some of the most important lifecycle methods are:

1. **componentWillMount()**–Executed just before rendering takes place both on the client as well as server-side.
2. **componentDidMount()**–Executed on the client side only after the first render.
3. **componentWillReceiveProps()**– Invoked as soon as the props are received from the parent class and before another render is called.
4. **shouldComponentUpdate()**–Returns true or false value based on certain conditions. If you want your component to update, return **true** else return **false**. By default, it returns false.
5. **componentWillUpdate()**– Called just before rendering takes place in the DOM.
6. **componentDidUpdate()**–Called immediately after rendering takes place.
7. **componentWillUnmount()**– Called after the component is unmounted from the DOM. It is used to clear up the memory spaces.

## What is an event in React?

In React, events are the triggered reactions to specific actions like mouse hover, mouse click, key press, etc. Handling these events are similar to handling events in DOM elements. But there are some syntactical differences like:

1. Events are named using camel case instead of just using the lowercase.
2. Events are passed as functions instead of strings.

The event argument contains a set of properties, which are specific to an event. Each event type contains its own properties and behavior which can be accessed via its event handler only.

## How do you create an event in React?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | **class** Display **extends** React.Component({      show(evt) {          // code      },      render() {          // Render the div with an onClick prop (value is a function)  **return** (    <div onClick={**this**.show}>Click Me!</div>            );      }  }); |

## What are synthetic events in React?

Synthetic events are the objects which act as a cross-browser wrapper around the browser’s native event. They combine the behavior of different browsers into one API. This is done to make sure that the events show consistent properties across different browsers.

## What do you understand by refs in React?

Refs is the short hand for References in React. It is an attribute which helps to store a reference to a particular React element or component, which will be returned by the components render configuration function. It is used to return references to a particular element or component returned by render(). They come in handy when we need DOM measurements or to add methods to the components.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | **class** ReferenceDemo **extends** React.Component{       display() {  **const** name = **this**.inputDemo.value;           document.getElementById('disp').innerHTML = name;       }  render() {  **return**(    <div>              Name: <input type="text" ref={input => **this**.inputDemo = input} />              <button name="Click" onClick={**this**.display}>Click</button>    <h2>Hello <span id="disp"></span> !!!</h2>          </div>      );     }   } |

## List some of the cases when you should use Refs.

Following are the cases when refs should be used:

* When you need to manage focus, select text or media playback
* To trigger imperative animations
* Integrate with third-party DOM libraries

## How do you modularize code in React?

We can modularize code by using the export and import properties. They help in writing the components separately in different files.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | //ChildComponent.jsx  export **default** **class** ChildComponent **extends** React.Component {      render() {  **return**(    <div>    <h1>This is a child component</h1>               </div>            );      }  }    //ParentComponent.jsx  **import** ChildComponent from './childcomponent.js';  **class** ParentComponent **extends** React.Component {      render() {  **return**(    <div>                  <App />              </div>            );      }  } |

## How are forms created in React?

React forms are similar to HTML forms. But in React, the state is contained in the state property of the component and is only updated via setState(). Thus the elements can’t directly update their state and their submission is handled by a JavaScript function. This function has full access to the data that is entered by the user into a form.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | handleSubmit(event) {      alert('A name was submitted: ' + **this**.state.value);      event.preventDefault();  }    render() {  **return** (    <form onSubmit={**this**.handleSubmit}>              <label>                  Name:                  <input type="text" value={**this**.state.value} onChange={**this**.handleSubmit} />              </label>              <input type="submit" value="Submit" />          </form>        );  } |

## What do you know about controlled and uncontrolled components?

|  |  |
| --- | --- |
| **Controlled vs Uncontrolled Components** | |
| **Controlled Components** | **Uncontrolled Components** |
| 1. They do not maintain their own state | 1. They maintain their own state |
| 2. Data is controlled by the parent component | 2. Data is controlled by the DOM |
| 3. They take in the current values through props and then notify the changes via callbacks | 3. Refs are used to get their current values |

## What are Higher Order Components(HOC)?

Higher Order Component is an advanced way of reusing the component logic. Basically, it’s a pattern that is derived from React’s compositional nature. HOC are custom components which wrap another component within it. They can accept any dynamically provided child component but they won’t modify or copy any behavior from their input components. You can say that HOC are ‘pure’ components.

## What can you do with HOC?

HOC can be used for many tasks like:

* Code reuse, logic and bootstrap abstraction
* Render High jacking
* State abstraction and manipulation
* Props manipulation

## What are Pure Components?

*Pure*components are the simplest and fastest components which can be written. They can replace any component which only has a **render().**These components enhance the simplicity of the code and performance of the application.

## What is the significance of keys in React?

Keys are used for identifying unique Virtual DOM Elements with their corresponding data driving the UI. They help React to optimize the rendering by recycling all the existing elements in the DOM. These keys must be a unique number or string, using which React just reorders the elements instead of re-rendering them. This leads to increase in application’s performance.

## What were the major problems with MVC framework?

Following are some of the major problems with MVC framework:

* DOM manipulation was very expensive
* Applications were slow and inefficient
* There was huge memory wastage
* Because of circular dependencies, a complicated model was created around models and views

## Explain Flux.

Flux is an architectural pattern which enforces the uni-directional data flow. It controls derived data and enables communication between multiple components using a central Store which has authority for all data. Any update in data throughout the application must occur here only. Flux provides stability to the application and reduces run-time errors.

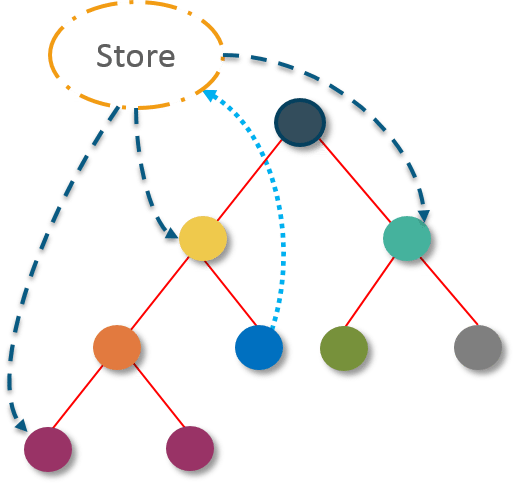


## What is Redux?

Redux is one of the most trending libraries for front-end development in today’s marketplace. It is a predictable state container for JavaScript applications and is used for the entire applications state management. Applications developed with Redux are easy to test and can run in different environments showing consistent behavior.

## What are the three principles that Redux follows?

1. ***Single source of truth:***The state of the entire application is stored in an object/ state tree within a single store. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.
2. **State is read-only:**The only way to change the state is to trigger an action. An action is a plain JS object describing the change. Just like state is the minimal representation of data, the action is the minimal representation of the change to that data.
3. ***Changes are made with pure functions:*** In order to specify how the state tree is transformed by actions, you need pure functions. Pure functions are those whose return value depends solely on the values of their arguments.



## What do you understand by “Single source of truth”?

Redux uses ‘Store’ for storing the application’s entire state at one place. So all the component’s state are stored in the Store and they receive updates from the Store itself. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.

## List down the components of Redux.

Redux is composed of the following components:

1. **Action** – It’s an object that describes what happened.
2. **Reducer**–  It is a place to determine how the state will change.
3. **Store** – State/ Object tree of the entire application is saved in the Store.
4. **View** – Simply displays the data provided by the Store.

In case you are facing any challenges with these React interview questions, please comment on your problems in the section below.

## Show how the data flows through Redux?



## How are Actions defined in Redux?

Actions in React must have a type property that indicates the type of ACTION being performed. They must be defined as a String constant and you can add more properties to it as well. In Redux, actions are created using the functions called Action Creators. Below is an example of Action and Action Creator:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | function addTodo(text) {  **return** {                  type: ADD\_TODO,                   text      }  } |

## Explain the role of Reducer.

Reducers are pure functions which specify how the application’s state changes in response to an ACTION. Reducers work by taking in the previous state and action, and then it returns a new state. It determines what sort of update needs to be done based on the type of the action, and then returns new values. It returns the previous state as it is, if no work needs to be done.

## What is the significance of Store in Redux?

A store is a JavaScript object which can hold the application’s state and provide a few helper methods to access the state, dispatch actions and register listeners. The entire state/ object tree of an application is saved in a single store. As a result of this, Redux is very simple and predictable. We can pass middleware to the store to handle the processing of data as well as to keep a log of various actions that change the state of stores. All the actions return a new state via reducers.

## How is Redux different from Flux?

|  |  |
| --- | --- |
| **Flux vs Redux** | |
| **Flux** | **Redux** |
| 1. The Store contains state and change logic | 1. Store and change logic are separate |
| 2. There are multiple stores | 2. There is only one store |
| 3. All the stores are disconnected and flat | 3. Single store with hierarchical reducers |
| 4. Has singleton dispatcher | 4. No concept of dispatcher |
| 5. React components subscribe to the store | 5. Container components utilize connect |
| 6. State is mutable | 6. State is immutable |

In case you are facing any challenges with these React interview questions, please comment on your problems in the section below.

## What are the advantages of Redux?

Advantages of Redux are listed below:

* **Predictability of outcome –**Since there is always one source of truth, i.e. the store, there is no confusion about how to sync the current state with actions and other parts of the application.
* **Maintainability –**The code becomes easier to maintain with a predictable outcome and strict structure.
* **Server-side rendering –** You just need to pass the store created on the server, to the client side. This is very useful for initial render and provides a better user experience as it optimizes the application performance.
* **Developer tools –**From actions to state changes, developers can track everything going on in the application in real time.
* **Community and ecosystem –**Redux has a huge community behind it which makes it even more captivating to use. A large community of talented individuals contribute to the betterment of the library and develop various applications with it.
* **Ease of testing –**Redux’s code is mostly functions which are small, pure and isolated. This makes the code testable and independent.
* **Organization –**Redux is precise about how code should be organized, this makes the code more consistent and easier when a team works with it.

## What is React Router?

React Router is a powerful routing library built on top of React, which helps in adding new screens and flows to the application. This keeps the URL in sync with data that’s being displayed on the web page. It maintains a standardized structure and behavior and is used for developing single page web applications. React Router has a simple API.

## Why is switch keyword used in React Router v4?

Although a **<div>** is used to encapsulate multiple routes inside the Router. The ‘switch’ keyword is used when you want to display only a single route to be rendered amongst the several defined routes. The **<switch>**tag when in use matches the typed URL with the defined routes in sequential order. When the first match is found, it renders the specified route. Thereby bypassing the remaining routes.

## Why do we need a Router in React?

A Router is used to define multiple routes and when a user types a specific URL, if this URL matches the path of any ‘route’ defined inside the router, then the user is redirected to that particular route. So basically, we need to add a Router library to our app that allows creating multiple routes with each leading to us a unique view.

|  |  |
| --- | --- |
| 1  2  3  4  5 | <**switch**>      <route exact path=&rsquo;/&rsquo;&nbsp;component={Home}/>      <route path=&rsquo;/posts/:id&rsquo; component={Newpost}/>      <route path=&rsquo;/posts&rsquo;&nbsp;&nbsp; component={Post}/>  </**switch**> |

## List down the advantages of React Router.

Few advantages are:

1. Just like how React is based on components, in React Router v4, the API is ‘All About Components’. A Router can be visualized as a single root component (**<BrowserRouter>**) in which we enclose the specific child routes (**<route>**).
2. No need to manually set History value: In React Router v4, all we need to do is wrap our routes within the **<BrowserRouter>** component.
3. The packages are split: Three packages one each for Web, Native and Core. This supports the compact size of our application. It is easy to switch over based on a similar coding style.

## How is React Router different from conventional routing?

|  |  |  |
| --- | --- | --- |
| **Conventional Routing vs React Routing** | | |
| **Topic** | **Conventional Routing** | **React Routing** |
| **PAGES INVOLVED** | Each view corresponds to a new file | Only single HTML page is involved |
| **URL CHANGES** | A HTTP request is sent to a server and corresponding HTML page is received | Only the History attribute is changed |
| **FEEL** | User actually navigates across different pages for each view | User is duped thinking he is navigating across different pages |

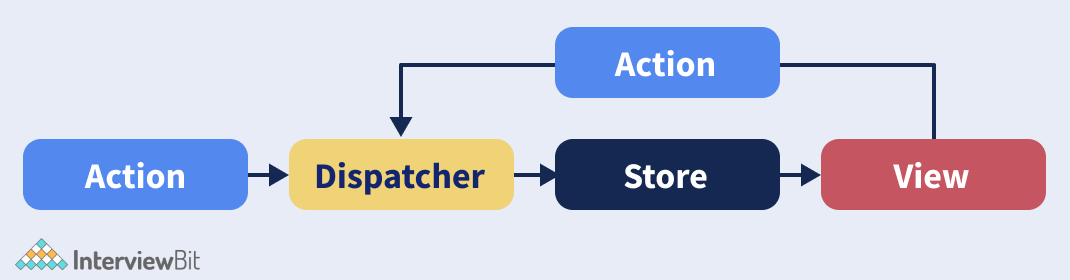
# Redux Questions

## 1. What is Redux?

Redux is an open-source library made using the scripting language JavaScript. Redux's primary use lies in managing and centralizing application state and it is usually used along with JavaScript libraries, for instance, React or Angular in order to build UIs (User Interfaces). It is a predictable state container for applications built using JavaScript. It is based on the Flux design pattern. Redux is very small in size (around 2 kilobytes) and has no dependencies.

## 2. What is Flux?

Flux is an application design paradigm just like the Model View Controller design pattern. Flux is nothing but a new kind of architecture that complements React and the concept of Unidirectional Data Flow. The image given below shows how the workflow between dispatcher, stores and views components with distinct inputs and outputs are in Flux:



## 3. Difference between Redux and Flux.

The main differences in the comparison: Redux vs Flux are as follows:

|  |  |  |
| --- | --- | --- |
| COMPARISON PARAMETER | REDUX | FLUX |
| Number of stores per application | Redux includes a single Store per application. Rather than placing state information in multiple Stores across the app, Redux keeps everything in one region of the application | Flux includes multiple Stores per application. |
| Architecture | Redux is an open-source JavaScript library used for creating User Interfaces. | Flux's architecture is designed to build client-side web apps. |
| Place where Business Logic of the Application Resides | In Redux, the business logic of the application resides in the Reducer. | In Flux, the business logic of the application resides in the Store. |

## 4. What is Redux in React js?

Redux in React is the official React binding for Redux which allows the components in React to read data from a Redux Store, and dispatch Actions to the Store for updating the data. The purpose of Redux is to help applications scale well by providing means to manage the state via a unidirectional data flow model.

## 5. State the core principles of Redux.

The core principles of Redux are as follows:

* **Single source of truth:** The global state of our application is always put away in an object tree inside one store.
* **The state is read-only:** The only way to change the state of our application is by emitting an action, an object explaining what has happened.
* **Changes are made with pure functions:** This principle means that in order to define how the state tree is being transformed by the actions, we have to write pure reducers.

## 6. What are the advantages of using Redux?

Some of the advantages of using Redux are as follows:

* Redux provides extremely easy state transfer between the components.
* The states are always predictable in Redux and its maintenance is relatively easy.
* Debugging and testing code in Redux is simple through logging behaviour and status.
* Redux provides great performance. It might occur to us that keeping the application's state global would result in bad performance. However, usually, that is not the case as React Redux implements a lot of performance optimizations internally so that our own connected component only re-renders when it actually needs to.
* Redux also offers state persistence by storing the application's state to local storage and restoring it after a refresh.

## 7. Is it true that Redux can only be used with React?

No, it is not true that Redux can only be used with React. Redux is being used as a data store for lots of UI layers. There are bindings available in Redux for React, Angular, Angular 2, Vue, etc.

## 8. What do you understand about Redux Toolkit?

Redux Toolkit is Redux's official, opinionated, batteries included toolset for efficient Redux development. It also consists of the most widely used Redux add-ons, for instance, Redux Thunk for asynchronous logic,  Reselect for writing selector functions and many more for making development easy for developers and saving them time.

## 9. What are some of the major features of Redux DevTools?

Some of the major features of Redux DevTools are as follows:

* Redux DevTools is nothing but a time travel environment that makes it possible for us to live edit in Redux with a variety of functionalities like action replay, hot reloading, and customizable UI.
* Redux DevTools makes it possible for us to inspect all the states and action payload. We can go back into the time simply by cancelling the actions.
* Each stage action is re-evaluated in case we change the code of the reducer.
* We can also continue our debug sessions across page reloads with the help of persistState() store enhancer.

## 10. Is it necessary to keep all the component states in the Redux store?

No, it is not necessary to keep all the component states in the Redux store. We have to keep your application state as small as possible and therefore, we should do it only if it makes a difference for us to keep something there or maybe if it makes the use of Dev Tools easier.

## 11. Highlight the key differences between mapStateToProps() and mapDispatchToProps()?

The key differences between mapStateToProps() and mapDispatchToProps() are given below:

|  |  |
| --- | --- |
| **mapStateToProps()** | **mapDispatchToProps()** |
| The mapStateToProps() method is used to render the stored data to the component. | The mapDispatchToProps() method is used to render the action creators with props to the component. |
| The entirety of the results of the mapStateToProps() method is a plain object which is later merged into the component’s prop. | In the mapDispatchToProps() method, each action creator is wrapped in the dispatcher call so that they can be called upon directly and later merged into the component’s prop. |
| This method's use is to connect the redux state to the props of the react component. | This method's use is to connect redux actions to the react props. |

## 12. What do you understand by an action in Redux's architecture?

In the Redux architecture, actions are nothing but the plain JavaScript objects which contain a type field. They can be thought of as an event that is used to describe something which has happened in the application. Actions contain only a tiny bit of information that is required to mention what has happened.

## 13. Give an example of the usage of Actions in Redux's architecture.

An example of the usage of Actions in Redux's architecture is given below:

const addingTodoAction = {

type: 'ADD',

payload: 'Do-homework'

}

## 14. Where can we use Redux?

Redux is primarily being used along with React. However, we can also use it along with Angular, Vue, Meteor, etc. using the bindings it has to offer to us.

## 15. What are constants in Redux?

Constants in Redux help us in easily finding all the usages of a particular functionality across our entire project when we are using an Integrated Development Environment (IDE). Using constants, we can avoid silly bugs caused because of typing errors or typos as it shows a "ReferenceError" whenever a typo is made.

## 16. Show using code how constants can be used in Redux.

First of all, we can store all the constants in a single file in our project named constants.js or something else as follows:

export const ADDING\_TODO = 'ADDING\_TODO';

export const DELETING\_TODO = 'DELETING\_TODO';

export const EDITING\_TODO = 'EDITING\_TODO';

export const COMPLETING\_TODO = 'COMPLETING\_TODO';

export const COMPLETING\_ALL = 'COMPLETING\_ALL';

export const CLEARING\_COMPLETED = 'CLEARING\_COMPLETED';

After storing the constants in one place, we can use them in two ways in our project:

* During actions creation (in actions.js file of our project):

 import { DELETING\_TODO } from './constants';

export function deletingTodo(text) {

return { type: DELETING\_TODO, text };

}

* In Reducers (in reducer.js file of our project):

 import { EDITING\_TODO } from './constants';

export default (state = [], action) => {

switch (action.type) {

case EDITING\_TODO:

return [

...state,

{

text: action.text,

completed: false

}

];

default:

return state

}

};

## 17. What are reducers in Redux's architecture?

Reducers in Redux's architecture are pure functions that are used to take the previous state and an action and return the next state. Its syntax is given below:

 (previous\_state, action) => new\_state

## 18. Show with an example how reducers are used in Redux.

An example of how reducers are used in Redux is given below:

const initial\_state = { value: 0 }

function countReducer(state = initial\_state, action) {

// Checking to see if the reducer cares about this action

if (action.type === 'counter/incremented') {

// If the action is of type "counter" or "incremented", we make a copy of `state`

return {

...state,

// We also update the copy with the new value

value: state.value + 1

}

}

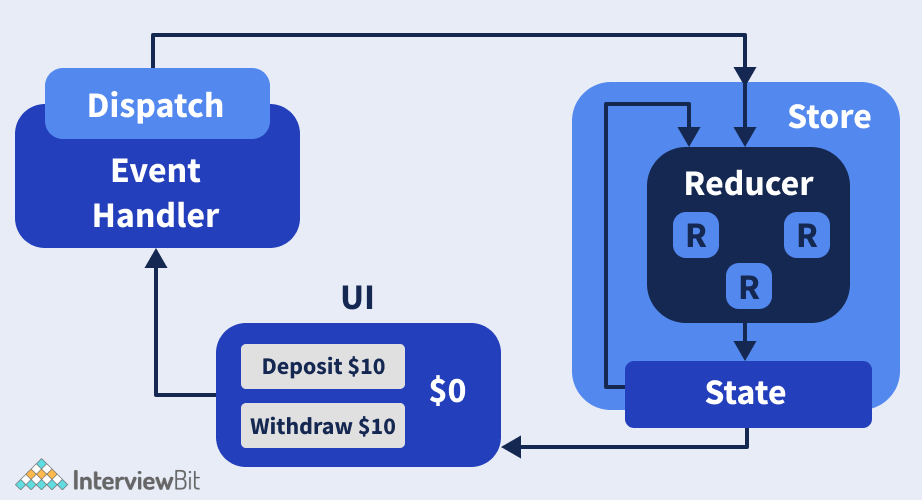
// If not, we return the original state unchanged

return state

}

## 19. Explain the typical data flow in an application made using React and Redux (Redux Lifecycle for an application).

The typical data flow in Redux starts with a call back from the User Interface component which dispatches an action with a payload. After that, the reducers intercept and receive the dispatched actions, generating a new application state. After that, the actions are propagated down through a hierarchy of components from the Redux store.



## 20. What is the mental model of redux saga?

Redux Saga functions as a separate thread in our programme which is solely responsible for side effects. Redux Saga is a redux middleware. In other words, it means that it can be started, paused, and aborted from the main application using standard Redux actions, has access to the entire Redux application state, and can also dispatch Redux actions.

## 21. Describe what is meant by a "store" in Redux.

“Store” in Redux is used to carry together all the states, reducers, and actions which create the app. Some of the responsibilities of the store are as follows:

* The state of the current application from inside is held by the Redux Store.
* We can access the current state using store.getState().
* We can update the state using store.dispatch(action).
* We can also register listener callbacks using the store.subscriber(listener).

## 22. Name all the Redux Store methods.

All the Redux Store Methods are as follows:

* getState()
* subscribe(listener)
* dispatch(action)
* replaceReducer(nextReducer)

## 23. Give an example depicting the usage of a Redux store.

An example depicting the usage of a Redux store is given below:

import { createStore } from 'redux'

const store = createStore(todos, ['Use Redux'])

function deletingTodo(text) {

return {

type: 'DELETING\_TODO',

text

}

}

store.dispatch(deletingTodo('Do the homework'))

store.dispatch(deletingTodo('Buy coffee'))

## 24. Explain with an example how to set the initial state in Redux.

In order to set the initial state in Redux, we have to pass the initial state as the second argument to createStore as shown below:

 const rootReducer = combineReducers({

todos: todos,

visibilityFilter: visibilityFilter

});

const initialState = {

todos: [{id:100, name:'ritik', completed: true}]

};

const store = createStore(

rootReducer,

initialState

);

## 25. What do you understand about Redux Thunk?

Using Redux Thunk middleware, we can write action creators returning a function instead of an action. This thunk can postpone the dispatch of an action, or do conditional dispatchment. The arguments passed to the inner function are the store methods dispatch and getState(). In the event of an action creator returning a function, the function gets executed by the Redux Thunk middleware and it does not have to be pure. In other words, the function is allowed to have side effects, including executing asynchronous API calls. It can even dispatch actions. Redux thunk is used to delay the dispatch of an action, or to dispatch in the event of a certain condition being met. At the time of dispatch of a function instead of an action object, if Redux Thunk middleware is enabled, the middleware will call that function with the dispatch method itself as the first argument.

## 26. What are the workflow features in Redux?

The workflow features in Redux are as follows:

* **Reset**: The state of the store is allowed to be reset.
* **Revert**: Revert or Rollback to the last committed state is allowed.
* **Sweep**: Every disabled action which we have fired unintentionally will be removed.
* **Commit**: The current state is made the initial state.

## 27. Differentiate between Relay and Redux.

The key differences between Relay and Redux are given below:

|  |  |
| --- | --- |
| **Relay** | **Redux** |
| The state originating from the server is taken care of by Relay. | All the states of the application are taken care of by Redux. |
| Relay can be used for caching and optimizing the data. | Redux is not responsible for handling data fetching (it can be done manually though). |

## 28. How can we use connect from react redux?

In order to use connect from React Redux, we will have to follow a couple of steps to use our store in our container:

* Firstly, we use the mapStateToProps(): This would map the state variables from our store to the props which we specify.
* Secondly, we connect the above props to our container: The object returned by the mapStateToProps component is connected to the container.

A sample code for connect from react-redux is given below:

import React from 'react';

import { connect } from 'react-redux';

class App extends React.Component {

render() {

return <div>{this.props.containerData}</div>;

}

}

function mapStateToProps(state) {

return { containerData: state.appData };

}

export default connect(mapStateToProps)(App);

function mapStateToProps(state) {

return { containerData: state.data };

}

export default connect(mapStateToProps)(App);

## 29. What are Redux forms? What are its major features?

Redux Forms present in React and Redux help in enabling a form in React to use Redux for storing all of its states. They can be used with raw inputs in HTML5. Redux forms work extremely well with User Interface (UI) frameworks, for instance, Material UI, React Widgets, React Bootstrap and many more.

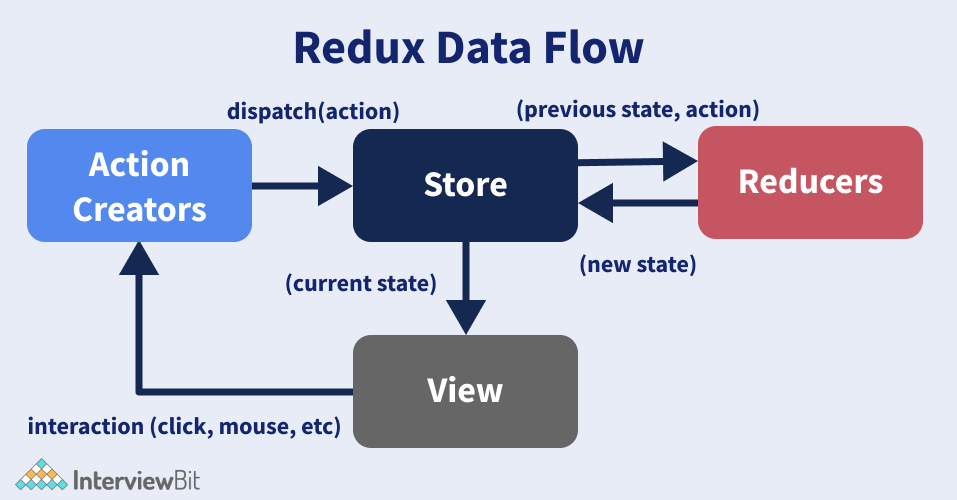
The **major features of Redux forms** are as follows:

* Field values persistence through the Redux store.
* Validation (synchronous/asynchronous) and submission.
* Formatting, parsing and normalization of field values.

## 30. How can we structure the top level directories in Redux?

Every Redux application has multiple top-level directories as given below:

* **Components**: Components are used for “dumb” React components unfamiliar with Redux.
* **Containers**: Containers are used for “smart” React components that are connected to Redux.
* **Actions**: Actions are used for all the action creators, where the file name should be corresponding to the part of the app.
* **Reducers**: Reducers are used for all the reducers where the file name is corresponding to the state key.
* **Store**: Stores are used for store initialization. This directory works best in small and mid-level size apps.



## 31. How can we access a redux store outside a react component?

For accessing a redux store outside a react component, we can export the store from the module where it has been created with createStore as done in the following example:

store = createStore(reducer);

export default store;

## 32. Differentiate between React Redux and React's Context API.

The key differences in the comparison: Context Api vs Redux are as follows:

|  |  |
| --- | --- |
| **React Redux** | **React Context API** |
| In order to use React-Redux in our application, we need to code it separately and then merge it into the main project. | React Context can be used in the application directly. |
| The number of features provided by React-Redux is comparatively more than React Context. | The number of features provided by React Context is comparatively less than React-Redux. |

## 33. What do you understand about the Redux Saga?

Redux Saga is a middleware library that can be useful for allowing a Redux store to interact with the resources outside of itself in an asynchronous manner, for example, making HTTP requests to external services, accessing browser storage, executing Input/Output operations and many more. These operations are also called side effects.

## 34. What are the things which we should never do inside a reducer?

The things which we should never do inside a reducer are as follows:

* Modify the argument of the reducer
* We should assure that we do not perform any side operations such as routing transitions, API calls, etc.
* We should not call non-pure functions, for instance Date.now(), Math.random(), etc.

## 35. How can the addition of multiple middlewares to Redux be done?

In order to add multiple middlewares to Redux, the usage of applyMiddleware can do the work. In applyMiddleware, we can pass every piece of middleware as a new argument. Therefore, our job is to just pass each piece of middleware that we want. In the example given below, we have added Redux Thunk and logger middlewares as an argument:

import { createStore, applyMiddleware } from 'redux'

const createStoreWithMiddleware = applyMiddleware(ReduxThunk, logger)(createStore);

# Next.js Questions

## 1) What is Next.js?

Next.js is an open-source, lightweight React.js framework that facilitates developers to build static and server-side rendering web applications. It was created by Zeit. Next.js framework is based on React, Webpack, and Babel and allows us to write server-rendered React apps easily. It doesn't require any webpack configuration and only needs npm run dev start building your next feature-filled web application.

## 2) Why is Next.js used for? / Why do world's leading companies prefer Next.js?

If you want to build a complete web application with React from scratch, you have to fulfill the following points:

* Your code has to be bundled using a bundler like webpack and transformed using a compiler like Babel.
* You have to do production optimizations such as code splitting.
* You have to pre-render some pages for performance and SEO statically. You might also want to use server-side rendering or client-side rendering.
* You might have to write some server-side code to connect your React app to your data store.

Next.js fulfills the above all requirements.

**Reasons why the world's leading companies prefer Next.js:**

**Zero Setup:** Next.js provides automatic code-splitting, filesystem-based routing, hot code reloading, and universal rendering; that's why the world's leading companies prefer it.

**Fully Extensible:** Next.js is fully extensible and has complete control over Babel and Webpack. It provides a customizable server, routing, and next plugins.

**Ready for Production:** Next.js is optimized for smaller build sizes, faster dev compilation, and many other improvements, making it a popular choice.

**Next.js can Deploy Anywhere:** Next.js is an open-source, lightweight React.js framework that facilitates developers to build static and server-side rendering web applications.

## 3) What is the process to install Next.js? / How to install Next.js?

Before installing Next.js, you must have installed Node.js on your system. Learn how to install Node.js on your system: <https://www.javatpoint.com/install-nodejs>

It requires NPM to start installing Next.js with all its dependencies.

**Follow the steps given below to install Next.js:**

* First, create a directory to keep the Next.js project and go into it:

1. mkdir my-portfolio-site
2. cd my-portfolio-site

* Now, initialize this with a package.json file.
* Use the y flag by npm init -y
* Use the following syntax to install Next.js

1. npm install react react-dom next

* Update package.json with run script languages to start the initialization of the Next.js application.
* Now, find the package.json file on the root folder and add the below mentioned script

1. {
2. "scripts": {
3. "dev": "next",
4. "build": "next build",
5. "start": "next start"
6. }
7. }

* After that, the filesystem is the main API. Every ".js" file becomes a route that gets automatically processed and rendered.
* Now, the process is completed, and Next.js is installed on your system.

## 4) What are the most prominent features of Next.js?

Following is a list of the most prominent features of Next.js that excite the developers most:

* js provides the by default and easy server rendering.
* js supports static exporting.
* It provides a Webpack-based dev environment which supports Hot Module Replacement (HMR)
* It seaports automatic code-splitting for faster page loads.
* It supports simple client-side routing (page-based) or file system-based routing.
* It provides complete Webpack and Babel control.
* It provides a faster and optimized development compilation.
* It can be implemented with Express or any other Node.js HTTP server.
* You can easily customize it with your own Babel and Webpack configurations.
* It supports hot code reloading.

**Besides this, Next.js also has some awesome features such as:**

* Dynamic styles and themes support
* Built-in CSS vendor prefixing
* CSS Preprocessing via Plugins
* Full CSS support, no tradeoffs in power
* Its runtime size is very small. It is just 3kb (zipped from 12kb)
* It provides source maps support

## 5) Which types of websites most popularly use Next.js?

Next.js is a popular framework of React.js that is most popularly used for building the following types of apps and websites:

* Static Websites
* Desktop Websites
* SEO Friendly Websites
* Server Rendered Apps
* Progressive Web Apps (PWA) etc.

## 6) Is it possible to use Next.js with Redux?

Yes. You can easily use Next.js with Redux.

## 7) What is the recommended method to fetch data in Next.js?

There are multiple ways to fetch data in Next.js, but Next.js itself recommends **getInitialProps,** an async function to retrieve data from anywhere. When we use **getInitialProps** to retrieve data, it receives a context object which has the following properties:

* **pathname-** It specifies the path section of the URL.
* **query-** It is used to specify the query string section of URL parsed as an object.
* **asPath-** It specifies the string of the actual path (including the query) shows in the browser.
* **req-** It is used to specify the HTTP request object (server only).
* **res-** It is used to specify the HTTP response object (server only).
* **err-** It is used to specify the error object if any error is encountered during the rendering.

## 8) Give an example to demonstrate how do you set up CDN in Next.js?

To setup CDN in Next.js, the developers have to follow the steps given below:

To start, we have to first set up the "assetPrefix" setting and configure our CDN origin to support and resolve the domain that our Next.js is hosted on.

1. const isProd = process.env.NODE\_ENV === 'production';
2. module.exports = {
3. // You may only need to add assetPrefix in the production.
4. assetPrefix: isProd ? 'https://cdn.mydomain.com' : ''
5. };

If the CDN is present on a separate domain, we have to set a configuration option as following:

1. // next.config.js
2. odule.exports = {
3. crossOrigin: 'anonymous'
4. ;

## 9) Are Create-React-App and Next.js used for the same thing?

The Create-React-App is basically React with an integrated build system. It acts like a good boilerplate, so we don't need to set up Webpack, Babel, and other dependent packages to run React. Other than that, if you require extra functionalities such as routing, server-side rendering, and so on, you just need to add packages on top of Create-React-App. On the other hand, The Next.js is an open-source, lightweight full-stack React framework that comes bundled with an efficient build system, server-side rendering, routing, API routing, and many other awesome features that make the production environment easy.

## 10) How can you install and use Next.js?

There are mainly two ways to install and run Next.js on your system. If you're new to Next.js, we recommend that you make sure that your development environment is ready. Next.js is a React framework, and it requires Node.js to be installed on your system. If you don't have Node.js installed, you can install it from here: <https://www.javatpoint.com/install-nodejs>. Next.js requires Node.js version 10.13 or later. You should also have your text editor and terminal.

**System Requirement for Next.js**

* You must have installed Node.js version 12.0 or later.
* js supports Windows, MacOS, and Linux also.

**The simplest way to install Next.js**

* The simplest way to install and run Next.js on your system is by creating a new Next.js app using create-next-app. It sets up everything automatically for you. To create a project, run the following command on the Node.js console.

1. npx create-next-app
2. # or
3. yarn create next-app

* If you want to start with a TypeScript project, you can use the following command:

1. npx create-next-app --typescript
2. # or
3. yarn create next-app -typescript

* After completing the installation, follow the instructions to start the development server. You can also try to edit the pages/index.js and see the result on your browser, which by default runs on http://localhost:3000

**Manual Installation and Setup of Next.js**

* First, install the next react and react-dom in your project by running the following command.

1. npm install next react react-dom
2. or
3. yarn add next react react-dom

* Now, open the package.json file and add the following scripts:

1. "scripts": {
2. "dev": "next dev",
3. "build": "next build",
4. "start": "next start",
5. "lint": "next lint"
6. }

**The above script specifies the different stages of developing an application:**

* **dev -** It runs the next dev, which starts Next.js in development mode.
* **build -** It runs "next build", which builds the application for production usage.
* **start -** It runs "next start", which starts a Next.js production server.
* **lint -** It runs "next lint", which sets up Next.js' built-in ESLint configuration.

Next.js is built around the concept of pages. A page is a React Component exported from a .js, .jsx, .ts, or .tsx file in the pages directory. Pages are associated with a route based on their filename. For example, pages/about.js is mapped to /about. You can even add dynamic route parameters with the filename.

## 11) How can you disable the etag generation in Next.js?

Generally, we use the app.disable('etag') syntax to disable the etag generation in Next.js. But, this may not work for all static contents. So, we should use the following syntax to disable the etag for all static contents.

**Syntax:**

1. app.use(express.static(path.join(\_\_dirname, 'public'), {
2. etag: false
3. }));

## 12) How can you configure the build-id in Next.js?

To configure the build-id in Next.js, we must configure a static ID between our builds. So, we have to provide the "generateBuildId" function with the following configuration.

**Syntax:**

1. // next.config.js
2. module.exports = {
3. generateBuildId: async () => {
4. // For example get the latest git commit hash here
5. return 'my-build-id';
6. }
7. };

## 13) How can you create a page directory inside your project?

To create a page directory inside our project we have to populate the ./pages/index.js with the following contents:

1. function HomePage() {
2. return <div>Welcome to Next.js!</div>
3. }

To start developing our application, we have to run the npm run dev or yarn dev command. This will start the development server on http://localhost:3000. Now we can visit the localhost: http://localhost:3000 to view our application.

## 14) Give an example to demonstrate how to create a custom error page in Next.js?

We can create our custom error page by defining a \_error.js in the pages folder. See the following example:

1. **import** React from "react";
2. **class** Error **extends** React.Component {
3. **static** getInitialProps({ res, err }) {
4. **const** statusCode = res ? res.statusCode : err ? err.statusCode : **null**;
5. **return** { statusCode };
6. }
7. render() {
8. **return** (
10. {**this**.props.statusCode
11. ? `An error ${**this**.props.statusCode} has occurred on the server`
12. : "An error occurred on client-side"}
14. );
15. }
16. }
17. export **default** Error;

## 15) What do you understand by code splitting in Next.js?

Generally, code splitting is one of the most compelling features of webpack. This feature facilitates us to split our code into various bundles, which can be loaded only on-demand or in parallel. This is mainly used to achieve the smaller bundles and facilitates us to control resource load prioritization which finally has a great impact on the load time.

**There are mainly three approaches to code splitting:**

* **Entry Points:** It is used to split code using entry configuration manually.
* **Prevent Duplication:** It uses Entry dependencies or SplitChunksPlugin to dedupe and split chunks.
* **Dynamic Imports:** It splits the code via inline function calls within modules

It is mainly used to enable pages that can never load unnecessary code.

## 16) How can you enable AMP in Next.js?

This is an important question and is asked in many Next.js interview questions. There are two ways to enable AMP in Next.js.

* AMP-First Pages
* Hybrid AMP Pages

**AMP-First Pages:** The AMP-First Pages are served to the primary traffic of the website as well as traffic generated from the search engine. Use the following syntax to implement AMP-first pages.

**Example:**

1. // pages/index.js
2. import { withAmp } from 'next/amp'
3. function HomePage() {
4. return <p> Welcome to AMP + Next.js.</p>
5. }
6. export default withAmp(HomePage)

**Hybrid AMP Pages:** The Hybrid AMP pages allow the users to have coexisted AMP version of a traditional page so that the search engines can display the AMP version or the page in different mobile search results.

See the following example to understand how to implement the Hybrid AMP to pages:

**Example:**

1. // pages/index.js
2. import { withAmp } from 'next/amp'
3. function HomePage() {
4. return <p> Welcome to AMP + Next.js.</p>
5. }
6. export default withAmp(HomePage)

## 17) Is it possible to host Next.js in a web server like Nginx?

Next.js is not as simple as static html files. It requires an application server that runs Node.js to deploy and run a Next.js application. Here, we get requests that have to be processed on the server.

# Typescript Questions

## What are the primitive types in TypeScript?

TypeScript has three primitive types that are frequently used: string, number, and boolean. These correspond to the similarly named types in JavaScript.

* **string**: represents text values such as “javascript”, “typescript”, etc.
* **number**: represents numeric values like 1, 2, 32, 43, etc.
* **boolean:** represents a variable that can have either a ‘true’ or ‘false’ value.

## Explain how the arrays work in TypeScript.

We use arrays to store values of the same type. Arrays are ordered and indexed collections of values. The indexing starts at 0, i.e., the first element has index 0, the second has index 1, and so on.

Here is the syntax to declare and initialize an array in TypeScript.

let values: number[] = [];

values[0] = 10;

values[1] = 20;

values[2] = 30;

You can also create an array using the short-hand syntax as follows:

let values: number[] = [15, 20, 25, 30];

TypeScript provides an alternate syntax to specify the Array type.

let values: Array<number> = [15, 20, 25, 30];

## What is any type, and when to use it?

There are times when you want to store a value in a variable but don’t know the type of that variable in advance. For example, the value is coming from an API call or the user input. The ‘any’ type allows you to assign a value of any type to the variable of type any.

TypeScript assumes a variable is of type **any** when you don’t explicitly provide the type, and the compiler cannot infer the type from the surrounding context.

## What is void, and when to use the void type?

The void indicates the absence of type on a variable. It acts as the opposite type to any. It is especially useful in functions that don’t return a value.

If a variable is of type void, you can only assign the null or undefined values to that variable.

## What is an unknown type, and when to use it in TypeScript?

The unknown type is the type-safe counterpart of any type. You can assign anything to the unknown, but the unknown isn’t assignable to anything but itself and any, without performing a type assertion of a control-flow-based narrowing. You cannot perform any operations on a variable of an unknown type without first asserting or narrowing it to a more specific type.

Consider the following example. We create the foo variable of unknown type and assign a string value to it. If we try to assign that unknown variable to a string variable bar, the compiler gives an error.

You can narrow down a variable of an unknown type to something specific by doing typeof checks or comparison checks or using type guards.

## What are the different keywords to declare variables in TypeScript?

**var:** Declares a function-scoped or global variable. You can optionally set its value during the declaration. Its behavior and scoping rules are similar to the var keyword in JavaScript.

**let:** Declares a block-scoped local variable. Similar to var, you can optionally set the value of a variable during the declaration.

**const:** Declares a block-scoped constant value that cannot be changed after it’s initialized.

## Provide the syntax of a function with the type annotations.

Functions are blocks of code to perform a specific code. Functions can optionally take one or more arguments, process them, and optionally return a value.

## How to create objects in TypeScript?

Objects are dictionary-like collections of keys and values. The keys have to be unique. They are similar to arrays and are also sometimes called associative arrays. However, an array uses numbers to index the values, whereas an object allows you to use any other type as the key.

In TypeScript, an Object type refers to any value with properties. It can be defined by simply listing the properties and their types.

For example,

let pt: { x: number; y: number } = {

x: 10,

y: 20

};

## How to specify optional properties in TypeScript?

An object type can have zero or more optional properties by adding a ‘?’ after the property name.

let pt: { x: number; y: number; z?: number } = {

x: 10,

y: 20

};

console.log(pt);

In the example above, because the property ‘z’ is marked as optional, the compiler won’t complain if we don’t provide it during the initialization.

## Explain the concept of null and its use in TypeScript.

In programming, a null value indicates an absence of value. A null variable doesn’t point to any object. Hence you cannot access any properties on the variable or call a method on it.

In TypeScript, the null value is indicated by the ‘null’ keyword.

## What is undefined in TypeScript?

When a variable is declared without initialization, it’s assigned the undefined value. It’s not very useful on its own. A variable is undefined if it’s declared, but no value has been assigned to it. In contrast, null is assigned to a variable, and it represents no value.

## Explain the purpose of the never type in TypeScript.

As the name suggests, the never type represents the type of values that never occur. For example, a function that never returns a value or that always throws an exception can mark its return type as never.

function error(message: string): never {

throw new Error(message);

}

You might wonder why we need a ‘never’ type when we already have ‘void’. Though both types look similar, they represent two very different concepts.

A function that doesn't return a value implicitly returns the value undefined in JavaScript. Hence, even though we are saying it’s not returning anything, it’s returning ‘undefined’. We usually ignore the return value in these cases. Such a function is inferred to have a void return type in TypeScript.

// This function returns undefined

function greet(name: string) {

console.log(`Hello, ${name}`);

}

let greeting = greet("David");

console.log(greeting); // undefined

In contrast, a function that has a never return type never returns. It doesn't return undefined, either. There are 2 cases where functions should return never type:

1. In an unending loop e.g a while(true){} type loop.
2. A function that throws an error e.g function foo(){throw new Exception('Error message')}

## What is the typeof operator? How is it used in TypeScript?

Similar to JavaScript, the typeof operator in TypeScript returns the type of the operand as a string.

console.log(typeof 10); // "number"

console.log(typeof 'foo'); // "string"

console.log(typeof false); // "boolean"

console.log(typeof bar); // "undefined"

In TypeScript, you can use the typeof operator in a type context to refer to the type of a property or a variable.

let greeting = "hello";

let typeOfGreeting: typeof greeting; // similar to let typeOfGreeting: string

## What are the rest parameters and arguments in TypeScript?

A rest parameter allows a function to accept an indefinite number of arguments as an array. It is denoted by the ‘…’ syntax and indicates that the function can accept one or more arguments.

function add(...values: number[]) {

let sum = 0;

values.forEach(val => sum += val);

return sum;

}

const sum = add(5, 10, 15, 20);

console.log(sum); // 50

In contrast, the rest arguments allow a function caller to provide a variable number of arguments from an array. Consider the following example.

const first = [1, 2, 3];

const second = [4, 5, 6];

first.push(...second);

console.log(first); // [1, 2, 3, 4, 5, 6]

## What is parameter destructuring?

Parameter destructing allows a function to unpack the object provided as an argument into one or more local variables.

function multiply({ a, b, c }: { a: number; b: number; c: number }) {

console.log(a \* b \* c);

}

multiply({ a: 1, b: 2, c: 3 });

You can simplify the above code by using an interface or a named type, as follows:

type ABC = { a: number; b: number; c: number };

function multiply({ a, b, c }: ABC) {

console.log(a \* b \* c);

}

multiply({ a: 1, b: 2, c: 3 });

## Explain the TypeScript class syntax.

TypeScript fully supports classes. The TypeScript syntax for class declaration is similar to that of JavaScript, with the added type support for the member declarations.

Here is a simple class that defines an Employee type.

class Employee {

name: string;

salary: number;

constructor(name: string, salary: number) {

this.name = name;

this.salary = salary;

}

promote() : void {

this.salary += 10000;

}

}

You can create an instance (or object) of a class by using the new keyword.

// Create a new employee

let john = new Employee("John", 60000);

console.log(john.salary); // 60000

john.promote();

console.log(john.salary); // 70000

## Explain the arrow function syntax in TypeScript.

Arrow functions provide a short and convenient syntax to declare functions. They are also called lambdas in other programming languages.

Consider a regular function that adds two numbers and returns a number.

function add(x: number, y: number): number {

let sum = x + y;

return sum;

}

Using arrow functions syntax, the same function can be defined as:

let add = (x: number, y: number): number => {

let sum = x + y;

return sum;

}

You can further simplify the syntax by getting rid of the brackets and the return statement. This is allowed when the function body consists of only one statement. For example, if we remove the temporary sum variable, we can rewrite the above function as:

let add = (x: number, y: number): number => x + y;

Arrow functions are often used to create anonymous callback functions in TypeScript. Consider the example below that loops over and filters an array of numbers and returns an array containing multiples of five. The filter function takes an arrow function.

let numbers = [3, 5, 9, 15, 34, 35];

let fiveMultiples = numbers.filter(num => (num % 5) == 0);

console.log(fiveMultiples); // [5, 15, 35]

## Provide the syntax for optional parameters in TypeScript.

A function can mark one or more of its parameters as optional by suffixing its name with ‘?’. In the example below, the parameter greeting is marked optional.

function greet(name: string, greeting?: string) {

if (!greeting)

greeting = "Hello";

console.log(`${greeting}, ${name}`);

}

greet("John", "Hi"); // Hi, John

greet("Mary", "Hola"); // Hola, Mary

greet("Jane"); // Hello, Jane

## What is the purpose of the tsconfig.json file?

A tsconfig.json file in a directory marks that directory as the root of a TypeScript project. It provides the compiler options to compile the project.

Here is a sample tsconfig.json file:

{

"compilerOptions": {

"module": "system",

"noImplicitAny": true,

"removeComments": true,

"outFile": "../../built/local/tsc.js",

"sourceMap": true

},

"include": ["src/\*\*/\*"],

"exclude": ["node\_modules", "\*\*/\*.spec.ts"]

}

## Explain the different variants of the for loop in TypeScript.

TypeScript provides the following three ways to loop over collections.

* ‘for’ loop

let values = [10, "foo", true];

for(let i=0; i<values.length; i++) {

console.log(values[i]); // 10, "foo", true

}

* ‘forEach’ function

let values = [10, "foo", true];

values.forEach(val => {

console.log(val); // 10, "foo", true

})

* ‘for..of’ statement

let values = [10, "foo", true];

for (let val of values) {

console.log(val); // 10, "foo", true

}

## Explain the symbol type in TypeScript.

Symbols were introduced in ES6 and are supported by TypeScript. Similar to numbers and strings, symbols are primitive types. You can use Symbols to create unique properties for objects.

You can create symbol values by calling the Symbol() constructor, optionally providing a string key.

let foo = Symbol();

let bar = Symbol("bar"); // optional string key

A key characteristic of symbols is that they are unique and immutable.

let foo = Symbol("foo");

let newFoo = Symbol("foo");

let areEqual = foo === newFoo;

console.log(areEqual); // false, symbols are unique

## Explain how optional chaining works in TypeScript.

Optional chaining allows you to access properties and call methods on them in a chain-like fashion. You can do this using the ‘?.’ operator.

TypeScript immediately stops running some expression if it runs into a ‘null’ or ‘undefined’ value and returns ‘undefined’ for the entire expression chain.

Using optional chaining, the following expression

let x = foo === null || foo === undefined ? undefined : foo.bar.baz();

can be expressed as:

let x = foo?.bar.baz();

## Provide the TypeScript syntax to create function overloads.

Function overloading allows us to define multiple functions with the same name, as long as their number of parameters or the types of parameters are different.

The following example defines two overloads for the function buildDate. The first overload takes a number as a parameter, whereas the second takes three numbers as parameters. These are called overload signatures.

The body of the function also called an implementation signature, follows the overload signatures. You can’t call this signature directly, as it’s not visible from the outside. It should be compatible with the overload signatures.

function buildDate(timestamp: number): Date;

function buildDate(m: number, d: number, y: number): Date;

function buildDate(mOrTimestamp: number, d?: number, y?: number): Date {

if (d !== undefined && y !== undefined) {

return new Date(y, mOrTimestamp, d);

} else {

return new Date(mOrTimestamp);

}

}

const d1 = buildDate(87654321);

const d2 = buildDate(2, 2, 2);

## What is meant by type inference?

TypeScript can infer the type of a variable when you don’t provide an explicit type. This is known as type inference. This is usually done when the variables or parameters are initialized during the declaration.

For example, TypeScript knows that the variable foo is a string, even though we don’t mention string as a type.

let foo = "this is a string";

console.log(typeof foo); // "string"

## What is meant by contextual typing?

When the TypeScript compiler uses the location (or context) of a variable to infer its type, it’s called contextual typing.

In the following example, TypeScript uses the Window.onmousedown function type information to infer the type of the function expression on the right-hand side of the assignment. This allows it to infer the type of the e parameter, which does have a button property but not a property named foo.

window.onmousedown = function (e) {

console.log(e.button); //<- OK

console.log(e.foo); //<- Error!

};

## What is the purpose of noImplicitAny?

Usually, when we don’t provide any type on a variable, TypeScript assumes ‘any’ type. For example, TypeScript compiles the following code, assuming the parameter ‘**s’** is of any type. It works as long as the caller passes a string.

function parse(s) {

console.log(s.split(' '));

}

parse("Hello world"); // ["Hello", "world"]

However, the code breaks down as soon as we pass a number or other type than a string that doesn’t have a split() method on it. For example,

function parse(s) {

console.log(s.split(' ')); // [ERR]: s.split is not a function

}

parse(10);

**noImplicitAny** is a compiler option that you set in the tsconfig.json file. It forces the TypeScript compiler to raise an error whenever it infers a variable is of any type. This prevents us from accidentally causing similar errors.

Parameter 's' implicitly has an 'any' type.(7006)

function parse(s) {

console.log(s.split(' ')); // [ERR]: s.split is not a function

}

## What is an interface?

An interface defines a contract by specifying the type of data an object can have and its operations. In TypeScript, you can specify an object’s shape by creating an interface and using it as its type. It’s also called “duck typing”.

In TypeScript, you can create and use an interface as follows:

interface Employee {

name: string;

salary: number;

}

function process(employee: Employee) {

console.log(`${employee.name}'s salary = ${employee.salary}`);

}

let john: Employee = {

name: "John Doe",

salary: 150000

}

process(john); // "John Doe's salary = 150000"

Interfaces are an effective way to specify contracts within your code as well as outside your code.

## Explain the various ways to control member visibility in TypeScript.

TypeScript provides three keywords to control the visibility of class members, such as properties or methods.

* **public:** You can access a public member anywhere outside the class. All class members are public by default.
* **protected:** A protected member is visible only to the subclasses of the class containing that member. Outside code that doesn’t extend the container class can’t access a protected member.
* **private:** A private member is only visible inside the class. No outside code can access the private members of a class.

## Does TypeScript support static classes? If not, why?

TypeScript doesn’t support static classes, unlike the popular object-oriented programming languages like C# and Java.

These languages need static classes because all code, i.e., data and functions, need to be inside a class and cannot exist independently. Static classes provide a way to allow these functions without associating them with any objects.

In TypeScript, you can create any data and functions as simple objects without creating a containing class. Hence TypeScript doesn’t need static classes. A singleton class is just a simple object in TypeScript.

## What are abstract classes? When should you use one?

Abstract classes are similar to interfaces in that they specify a contract for the objects, and you cannot instantiate them directly. However, unlike interfaces, an abstract class may provide implementation details for one or more of its members.

An abstract class marks one or more of its members as abstract. Any classes that extend an abstract class have to provide an implementation for the abstract members of the superclass.

Here is an example of an abstract class Writer with two member functions. The write() method is marked as abstract, whereas the greet() method has an implementation. Both the FictionWriter and RomanceWriter classes that extend from Writer have to provide their specific implementation for the write method.

abstract class Writer {

abstract write(): void;

greet(): void {

console.log("Hello, there. I am a writer.");

}

}

class FictionWriter extends Writer {

write(): void {

console.log("Writing a fiction.");

}

}

class RomanceWriter extends Writer {

write(): void {

console.log("Writing a romance novel.");

}

}

const john = new FictionWriter();

john.greet(); // "Hello, there. I am a writer."

john.write(); // "Writing a fiction."

const mary = new RomanceWriter();

mary.greet(); // "Hello, there. I am a writer."

mary.write(); // "Writing a romance novel."

## What are anonymous functions? Provide their syntax in TypeScript.

An anonymous function is a function without a name. Anonymous functions are typically used as callback functions, i.e., they are passed around to other functions, only to be invoked by the other function at a later point in time. For example,

setTimeout(function () {

console.log('Run after 2 seconds')

}, 2000);

You can invoke an anonymous function as soon as it’s created. It’s called ‘immediately invoked function execution (IIFE)’, For example:

(function() {

console.log('Invoked immediately after creation');

})();

## What are union types in TypeScript?

A union type is a special construct in TypeScript that indicates that a value can be one of several types. A vertical bar (|) separates these types.

Consider the following example where the variable value belongs to a union type consisting of strings and numbers. The value is initialized to string “Foo”. Because it can only be a string or a number, we can change it to a number later, and the TypeScript compiler doesn’t complain.

let value: string | number = "Foo";

value = 10; // Okay

However, if we try to set the value to a type not included in the union types, we get the following error.

value = true; // Type 'boolean' is not assignable to type 'string | number'.(2322)

Union types allow you to create new types out of existing types. This removes a lot of boilerplate code as you don’t have to create new classes and type hierarchies.

## What are intersection types?

Intersection types let you combine the members of two or more types by using the ‘&’ operator. This allows you to combine existing types to get a single type with all the features you need.

The following example creates a new type Supervisor that has the members of types Employee and Manager.

interface Employee {

work: () => string;

}

interface Manager {

manage: () => string;

}

type Supervisor = Employee & Manager;

// john can both work and manage

let john: Supervisor;

## What are type aliases? How do you create one?

Type aliases give a new, meaningful name for a type. They don’t create new types but create new names that refer to that type.

For example, you can alias a union type to avoid typing all the types everywhere that value is being used.

type alphanumeric = string | number;

let value: alphanumeric = "";

value = 10;

## Explain the tuple types in TypeScript.

Tuples are a special type in TypeScript. They are similar to arrays with a fixed number of elements with a known type. However, the types need not be the same.

// Declare a tuple type and initialize it

let values: [string, number] = ["Foo", 15];

// Type 'boolean' is not assignable to type 'string'.(2322)

// Type 'string' is not assignable to type 'number'.(2322)

let wrongValues: [string, number] = [true, "hello"]; // Error

Since TypeScript 3.0, a tuple can specify one or more optional types using the ? as shown below.

let values: [string, number, boolean?] = ["Foo", 15];

## Explain how tuple destructuring works in TypeScript.

You can destructure tuple elements by using the assignment operator (=). The destructuring variables get the types of the corresponding tuple elements.

let employeeRecord: [string, number] = ["John Doe", 50000];

let [emp\_name, emp\_salary] = employeeRecord;

console.log(`Name: ${emp\_name}`); // "Name: John Doe"

console.log(`Salary: ${emp\_salary}`); // "Salary: 50000"

After destructuring, you can’t assign a value of a different type to the destructured variable. For example,

emp\_name = true; // Type 'boolean' is not assignable to type 'string'.(2322)

## What are type assertions in TypeScript?

Sometimes, you as a programmer might know more about the type of a variable than TypeScript can infer. Usually, this happens when you know the type of an object is more specific than its current type. In such cases, you can tell the TypeScript compiler not to infer the type of the variable by using type assertions.

TypeScript provides two forms to assert the types.

* as syntax:

let value: unknown = "Foo";

let len: number = (value as string).length;

* <> syntax:

let value: unknown = "Foo";

let len: number = (<string>value).length;

Type assertions are similar to typecasting in other programming languages such as C# or Java. However, unlike those languages, there’s no runtime penalty of boxing and unboxing variables to fit the types. Type assertions simply let the TypeScript compiler know the type of the variable.

## How to enforce strict null checks in TypeScript?

Null pointers are one of the most common sources of unexpected runtime errors in programming. TypeScript helps you avoid them to a large degree by enforcing strict null checks.

You can enforce strict null checks in two ways:

* providing the --strictNullChecks flag to the TypeScript (tsc) compiler
* setting the strictNullChecks property to true in the tsconfig.json configuration file.

When the flag is false, TypeScript ignores null and undefined values in the code. When it is true, null and undefined have their distinct types. The compiler throws a type error if you try to use them where a concrete value is expected.

## How to make object properties immutable in TypeScript? (hint: readonly)

You can mark object properties as immutable by using the readonly keyword before the property name. For example:

interface Coordinate {

readonly x: number;

readonly y: number;

}

When you mark a property as readonly, it can only be set when you initialize the object. Once the object is created, you cannot change it.

let c: Coordinate = { x: 5, y: 15 };

c.x = 20; // Cannot assign to 'x' because it is a read-only property.(2540)

## What is a type declaration file?

A typical TypeScript project references other third-party TypeScript libraries such as JQuery to perform routine tasks. Having type information for the library code helps you in coding by providing detailed information about the types, method signatures, etc., and provides IntelliSense.

A type declaration file is a text file ending with a .d.ts extension providing a way to declare the existence of some types or values without actually providing implementations for those values. It contains the type declarations but doesn’t have any source code. It doesn’t produce a .js file after compilation.

## What are triple-slash directives?

Triple-slash directives are single-line comments that contain a single XML tag. TypeScript uses this XML tag as a compiler directive.

You can only place a triple-slash directive at the top of the containing file. Only single or multi-line comments can come before a triple-slash directive. TypeScript treats them as regular comments if it occurs in the middle of a code block, after a statement.

The primary use of triple-slash directives is to include other files in the compilation process. For example, the following directive instructs the compiler to include a file specified by the path in the containing TypeScript file.

/// <reference path="..." />

Triple-slash directives also order the output when using --out or --outFile. The output files are produced to the output file location in the same order as the input files.

## Explain the purpose of the ‘in’ operator.

The in operator is used to find if a property is in the specified object. It returns true if the property belongs to the object. Otherwise, it returns false.

const car = { make: 'Hyundai', model: 'Elantra', year: 2017 };

console.log('model' in car); // true

console.log('test' in car); // false

## What are the ‘implements’ clauses in TypeScript?

An implements clause is used to check that a class satisfies the contract specified by an interface. If a class implements an interface and doesn’t implement that interface, the TypeScript compiler issues an error.

interface Runnable {

run(): void;

}

class Job implements Runnable {

run() {

console.log("running the scheduled job!");

}

}

// Class 'Task' incorrectly implements interface 'Runnable'.

// Property 'run' is missing in type 'Task' but required in type 'Runnable'.(2420)

class Task implements Runnable {

perform() {

console.log("pong!");

}

}

A class can implement more than one interface. In this case, the class has to specify all the contracts of those interfaces.

## What are string literal types?

In TypeScript, you can refer to specific strings and numbers as types.

let foo: "bar" = "bar";

// OK

foo = "bar";

// Error: Type '"baz"' is not assignable to type '"bar"'.(2322)

foo = "baz";

String literal types on their own are not that useful. However, you can combine them into unions. This allows you to specify all the string values that a variable can take, in turn acting like enums. This can be useful for function parameters.

function greet(name: string, greeting: "hi" | "hello" | "hola") {

// ...

}

greet("John", "hello");

// Error: Argument of type '"Howdy?"' is not assignable to parameter of type '"hi" | "hello" | "hola"'.(2345)

greet("Mary", "Howdy?");

String literal types can help us spell-check the string values.

## What are template literal types?

Template literal types are similar to the string literal types. You can combine them with concrete, literal types to produce a new string literal type. Template literal types allow us to use the string literal types as building blocks to create new string literal types.

type Point = "GraphPoint";

// type Shape = "Grid GraphPoint"

type Shape = `Grid ${Point}`;

Template literal types can also expand into multiple strings via unions. It helps us create the set of every possible string literal that each union member can represent.

type Color = "green" | "yellow";

type Quantity = "five" | "six";

// type ItemTwo = "five item" | "six item" | "green item" | "yellow item"

type ItemOne = `${Quantity | Color} item`;

## Explain the concept of inheritance in TypeScript.

Inheritance allows a class to extend another class and reuse and modify the behavior defined in the other class. The class which inherits another class is called the derived class, and the class getting inherited is called the base class.

In TypeScript, a class can only extend one class. TypeScript uses the keyword ‘extends’ to specify the relationship between the base class and the derived classes.

class Rectangle {

length: number;

breadth: number

constructor(length: number, breadth: number) {

this.length = length;

this.breadth = breadth

}

area(): number {

return this.length \* this.breadth;

}

}

class Square extends Rectangle {

constructor(side: number) {

super(side, side);

}

volume() {

return "Square doesn't have a volume!"

}

}

const sq = new Square(10);

console.log(sq.area()); // 100

console.log(sq.volume()); // "Square doesn't have a volume!"

In the above example, because the class Square extends functionality from Rectangle, we can create an instance of square and call both the area() and volume() methods.

## What are conditional types? How do you create them?

A conditional type allows you to dynamically select one of two possible types based on a condition. The condition is expressed as a type relationship test.

C extends B ? TypeX : TypeY

Here, if type C extends B, the value of the above type is TypeX. Otherwise, it is TypeY.

## What is the Function type in TypeScript?

Function is a global type in TypeScript. It has properties like bind, call, and apply, along with the other properties present on all function values.

function perform(fn: Function) {

fn(10);

}

You can always call a value of the Function type, and it returns a value of ‘any’ type.

## List some of the utility types provided by TypeScript and explain their usage.

TypeScript provides various utility types that make common type transformations easy. These utility types are available globally. Here are some of the essential utility types included in TypeScript.

|  |  |
| --- | --- |
| **Utility Type** | **Description** |
| Partial<Type> | Constructs a type with all properties of Type set to optional. |
| Required<Type> | Constructs a type consisting of all properties of Type set to required. |
| Readonly<Type> | Constructs a type with all properties of Type set to readonly. |
| Record<Keys, Type> | Constructs an object type with property keys are of type Keys, and values are Type. |

# Node.js Questions

## 1. What is a first class function in Javascript?

When functions can be treated like any other variable then those functions are first-class functions. There are many other programming languages, for example, scala, Haskell, etc which follow this including JS. Now because of this function can be passed as a param to another function(callback) or a function can return another function(higher-order function). map() and filter() are higher-order functions that are popularly used.

## 2. What is Node.js and how it works?

Node.js is a virtual machine that uses JavaScript as its scripting language and runs Chrome’s V8 JavaScript engine. Basically, Node.js is based on an event-driven architecture where I/O runs asynchronously making it lightweight and efficient. It is being used in developing desktop applications as well with a popular framework called electron as it provides API to access OS-level features such as file system, network, etc.

## 3. How do you manage packages in your node.js project?

It can be managed by a number of package installers and their configuration file accordingly. Out of them mostly use npm or yarn. Both provide almost all libraries of javascript with extended features of controlling environment-specific configurations. To maintain versions of libs being installed in a project we use package.json and package-lock.json so that there is no issue in porting that app to a different environment.

## 4. How is Node.js better than other frameworks most popularly used?

* Node.js provides simplicity in development because of its non-blocking I/O and even-based model results in short response time and concurrent processing, unlike other frameworks where developers have to use thread management.
* It runs on a chrome v8 engine which is written in c++ and is highly performant with constant improvement.
* Also since we will use Javascript in both the frontend and backend the development will be much faster.
* And at last, there are ample libraries so that we don’t need to reinvent the wheel.

## 5. Explain the steps how “Control Flow” controls the functions calls?

* Control the order of execution
* Collect data
* Limit concurrency
* Call the following step in the program.

## 6. What are some commonly used timing features of Node.js?

* **setTimeout/clearTimeout**– This is used to implement delays in code execution.
* **setInterval/clearInterval** – This is used to run a code block multiple times.
* **setImmediate/clearImmediate** – Any function passed as the setImmediate() argument is a callback that's executed in the next iterationof the event loop.
* **process.nextTick** – Any function passed as the setImmediate() argument is a callback that's executed in the next iteration of the event loop.

## 7. What are the advantages of using promises instead of callbacks?

The main advantage of using promise is you get an object to decide the action that needs to be taken after the async task completes. This gives more manageable code and avoids callback hell.

## 8. What is fork in node JS?

A fork in general is used to spawn child processes. In node it is used to create a new instance of v8 engine to run multiple workers to execute the code.

## 9. Why is Node.js single-threaded?

Node.js was created explicitly as an experiment in async processing. This was to try a new theory of doing async processing on a single thread over the existing thread-based implementation of scaling via different frameworks.

## 10. How do you create a simple server in Node.js that returns Hello World?

var http = require("http");

http.createServer(function (request, response) {

response.writeHead(200, {'Content-Type': 'text/plain'});

response.end('Hello World\n');

}).listen(3000);

## 11. How many types of API functions are there in Node.js?

There are two types of API functions:

* **Asynchronous, non-blocking functions** - mostly I/O operations which can be fork out of the main loop.
* **Synchronous, blocking functions** - mostly operations that influence the process running in the main loop.

## 12. What is REPL?

REPL in Node.js stands for **R**ead, **E**val, **P**rint, and **L**oop, which further means evaluating code on the go.

## 13. List down the two arguments that async.queue takes as input?

* Task Function
* Concurrency Value

## 14. What is the purpose of module.exports?

This is used to expose functions of a particular module or file to be used elsewhere in the project. This can be used to encapsulate all similar functions in a file which further improves the project structure.  
  
For example, you have a file for all utils functions with util to get solutions in a different programming language of a problem statement.

const getSolutionInJavaScript = async ({

problem\_id

}) => {

...

};

const getSolutionInPython = async ({

problem\_id

}) => {

...

};

module.exports = { getSolutionInJavaScript, getSolutionInPython }

Thus using module.exports we can use these functions in some other file:

const { getSolutionInJavaScript, getSolutionInPython} = require("./utils")

## 15. What tools can be used to assure consistent code style?

ESLint can be used with any IDE to ensure a consistent coding style which further helps in maintaining the codebase.

## 16. What do you understand by callback hell?

async\_A(function(){

async\_B(function(){

async\_C(function(){

async\_D(function(){

....

});

});

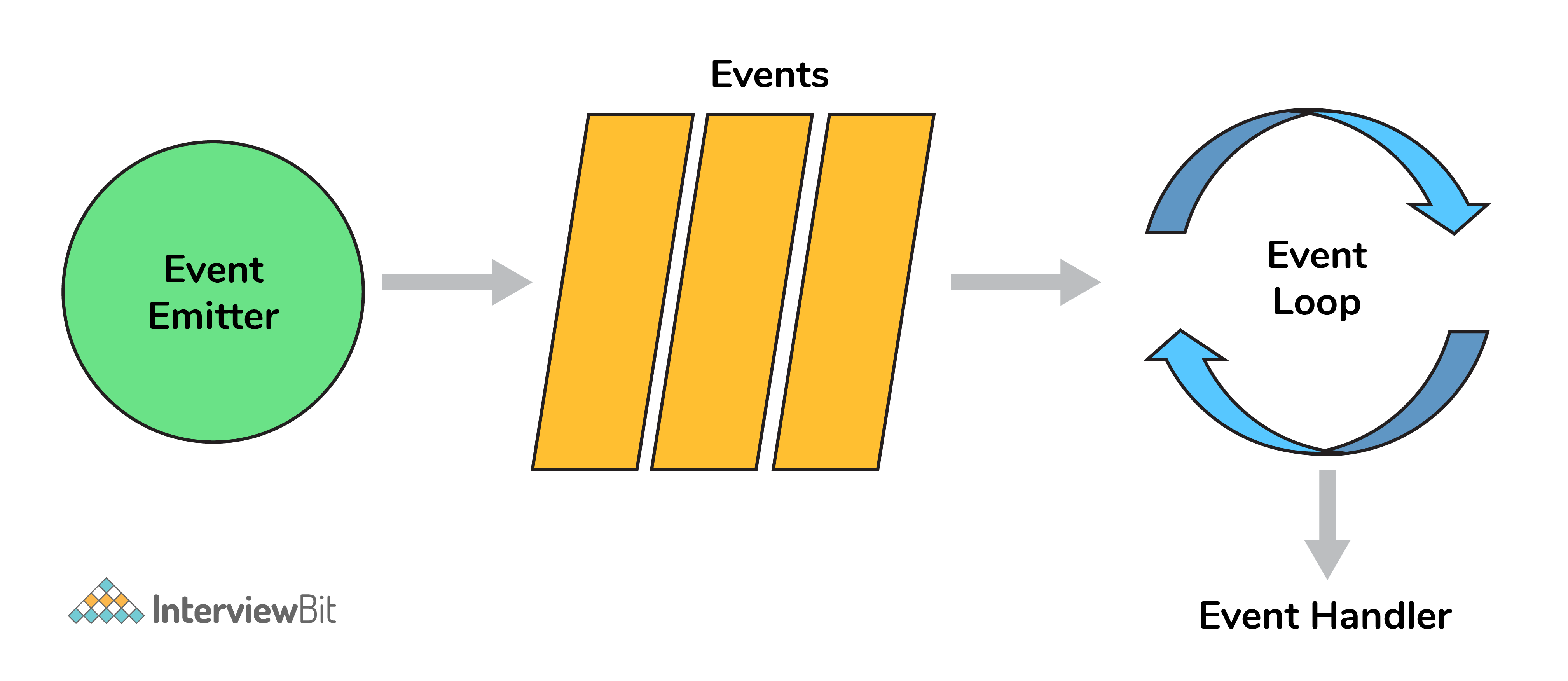
});

});

For the above example, we are passing callback functions and it makes the code unreadable and not maintainable, thus we should change the async logic to avoid this.

## 17. What is an event-loop in Node JS?

Whatever that is async is managed by event-loop using a queue and listener.  We can get the idea using the following diagram:

**Node.js Event Loop**

So when an async function needs to be executed(or I/O) the main thread sends it to a different thread allowing v8 to keep executing the main code. Event loop involves different phases with specific tasks such as timers, pending callbacks, idle or prepare, poll, check, close callbacks with different FIFO queues. Also in between iterations it checks for async I/O or timers and shuts down cleanly if there aren't any.

## 18. If Node.js is single threaded then how does it handle concurrency?

The main loop is single-threaded and all async calls are managed by libuv library.

For example:

const crypto = require("crypto");

const start = Date.now();

function logHashTime() {

crypto.pbkdf2("a", "b", 100000, 512, "sha512", () => {

console.log("Hash: ", Date.now() - start);

});

}

logHashTime();

logHashTime();

logHashTime();

logHashTime();

This gives the output:

Hash: 1213

Hash: 1225

Hash: 1212

Hash: 1222

This is because libuv sets up a thread pool to handle such concurrency. How many threads will be there in the thread pool depends upon the number of cores but you can override this.

## 19. Differentiate between process.nextTick() and setImmediate()?

Both can be used to switch to an asynchronous mode of operation by listener functions.   
process.nextTick() sets the callback to execute but setImmediate pushes the callback in the queue to be executed. So the event loop runs in the following manner  
**timers–>pending callbacks–>idle,prepare–>connections(poll,data,etc)–>check–>close callbacks**

In this process.nextTick() method adds the callback function to the start of the next event queue and setImmediate() method to place the function in the check phase of the next event queue.

## 20. How does Node.js overcome the problem of blocking of I/O operations?

Since the node has an event loop that can be used to handle all the I/O operations in an asynchronous manner without blocking the main function.   
  
So for example, if some network call needs to happen it will be scheduled in the event loop instead of the main thread(single thread). And if there are multiple such I/O calls each one will be queued accordingly to be executed separately(other than the main thread).

Thus even though we have single-threaded JS, I/O ops are handled in a nonblocking way.

## 21. How can we use async await in node.js?

Here is an example of using async-await pattern:

// this code is to retry with exponential backoff

function wait (timeout) {

return new Promise((resolve) => {

setTimeout(() => {

resolve()

}, timeout);

});

}

async function requestWithRetry (url) {

const MAX\_RETRIES = 10;

for (let i = 0; i <= MAX\_RETRIES; i++) {

try {

return await request(url);

} catch (err) {

const timeout = Math.pow(2, i);

console.log('Waiting', timeout, 'ms');

await wait(timeout);

console.log('Retrying', err.message, i);

}

}

}

## 22. What is node.js streams?

Streams are instances of EventEmitter which can be used to work with streaming data in Node.js. They can be used for handling and manipulating streaming large files(videos, mp3, etc) over the network. They use buffers as their temporary storage.  
  
There are mainly four types of the stream:

* **Writable:** streams to which data can be written (for example, fs.createWriteStream()).
* **Readable:** streams from which data can be read (for example, fs.createReadStream()).
* **Duplex:**streams that are both Readable and Writable (for example, net.Socket).
* **Transform:** Duplex streams that can modify or transform the data as it is written and read (for example, zlib.createDeflate()).

## 23. What are node.js buffers?

In general, buffers is a temporary memory that is mainly used by stream to hold on to some data until consumed. Buffers are introduced with additional use cases than JavaScript’s Unit8Array and are mainly used to represent a fixed-length sequence of bytes. This also supports legacy encodings like ASCII, utf-8, etc. It is a fixed(non-resizable) allocated memory outside the v8.

## 24. What is middleware?

Middleware comes in between your request and business logic. It is mainly used to capture logs and enable rate limit, routing, authentication, basically whatever that is not a part of business logic. There are third-party middleware also such as body-parser and you can write your own middleware for a specific use case.

## 25. Explain what a Reactor Pattern in Node.js?

Reactor pattern again a pattern for nonblocking I/O operations. But in general, this is used in any event-driven architecture.   
  
There are two components in this: 1. Reactor 2. Handler.  
  
**Reactor**: Its job is to dispatch the I/O event to appropriate handlers  
**Handler**: Its job is to actually work on those events

## 26. Why should you separate Express app and server?

The server is responsible for initializing the routes, middleware, and other application logic whereas the app has all the business logic which will be served by the routes initiated by the server. This ensures that the business logic is encapsulated and decoupled from the application logic which makes the project more readable and maintainable.

## 27. For Node.js, why Google uses V8 engine?

Well, are there any other options available? Yes, of course, we have [Spidermonkey](https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey) from Firefox, Chakra from Edge but Google’s v8 is the most evolved(since it’s open-source so there’s a huge community helping in developing features and fixing bugs) and fastest(since it’s written in c++) we got till now as a JavaScript and WebAssembly engine. And it is portable to almost every machine known.

## 28. Describe the exit codes of Node.js?

Exit codes give us an idea of how a process got terminated/the reason behind termination.   
  
A few of them are:

* Uncaught fatal exception - (code - 1) - There has been an exception that is not handled
* Unused - (code - 2) - This is reserved by bash
* Fatal Error - (code - 5) - There has been an error in V8 with stderr output of the description
* Internal Exception handler Run-time failure - (code - 7) - There has been an exception when bootstrapping function was called
* Internal JavaScript Evaluation Failure - (code - 4) - There has been an exception when the bootstrapping process failed to return function value when evaluated.

## 29. Explain the concept of stub in Node.js?

Stubs are used in writing tests which are an important part of development. It replaces the whole function which is getting tested.    
  
This helps in scenarios where we need to test:

* External calls which make tests slow and difficult to write (e.g HTTP calls/ DB calls)
* Triggering different outcomes for a piece of code (e.g. what happens if an error is thrown/ if it passes)

For example, this is the function:

const request = require('request');

const getPhotosByAlbumId = (id) => {

const requestUrl = `https://jsonplaceholder.typicode.com/albums/${id}/photos?\_limit=3`;

return new Promise((resolve, reject) => {

request.get(requestUrl, (err, res, body) => {

if (err) {

return reject(err);

}

resolve(JSON.parse(body));

});

});

};

module.exports = getPhotosByAlbumId;

To test this function this is the stub

const expect = require('chai').expect;

const request = require('request');

const sinon = require('sinon');

const getPhotosByAlbumId = require('./index');

describe('with Stub: getPhotosByAlbumId', () => {

before(() => {

sinon.stub(request, 'get')

.yields(null, null, JSON.stringify([

{

"albumId": 1,

"id": 1,

"title": "A real photo 1",

"url": "https://via.placeholder.com/600/92c952",

"thumbnailUrl": "https://via.placeholder.com/150/92c952"

},

{

"albumId": 1,

"id": 2,

"title": "A real photo 2",

"url": "https://via.placeholder.com/600/771796",

"thumbnailUrl": "https://via.placeholder.com/150/771796"

},

{

"albumId": 1,

"id": 3,

"title": "A real photo 3",

"url": "https://via.placeholder.com/600/24f355",

"thumbnailUrl": "https://via.placeholder.com/150/24f355"

}

]));

});

after(() => {

request.get.restore();

});

it('should getPhotosByAlbumId', (done) => {

getPhotosByAlbumId(1).then((photos) => {

expect(photos.length).to.equal(3);

photos.forEach(photo => {

expect(photo).to.have.property('id');

expect(photo).to.have.property('title');

expect(photo).to.have.property('url');

});

done();

});

});

});

## 30. What is an Event Emitter in Node.js?

EventEmitter is a Node.js class that includes all the objects that are basically capable of emitting events. This can be done by attaching named events that are emitted by the object using an eventEmitter.on() function. Thus whenever this object throws an even the attached functions are invoked synchronously.

const EventEmitter = require('events');

class MyEmitter extends EventEmitter {}

const myEmitter = new MyEmitter();

myEmitter.on('event', () => {

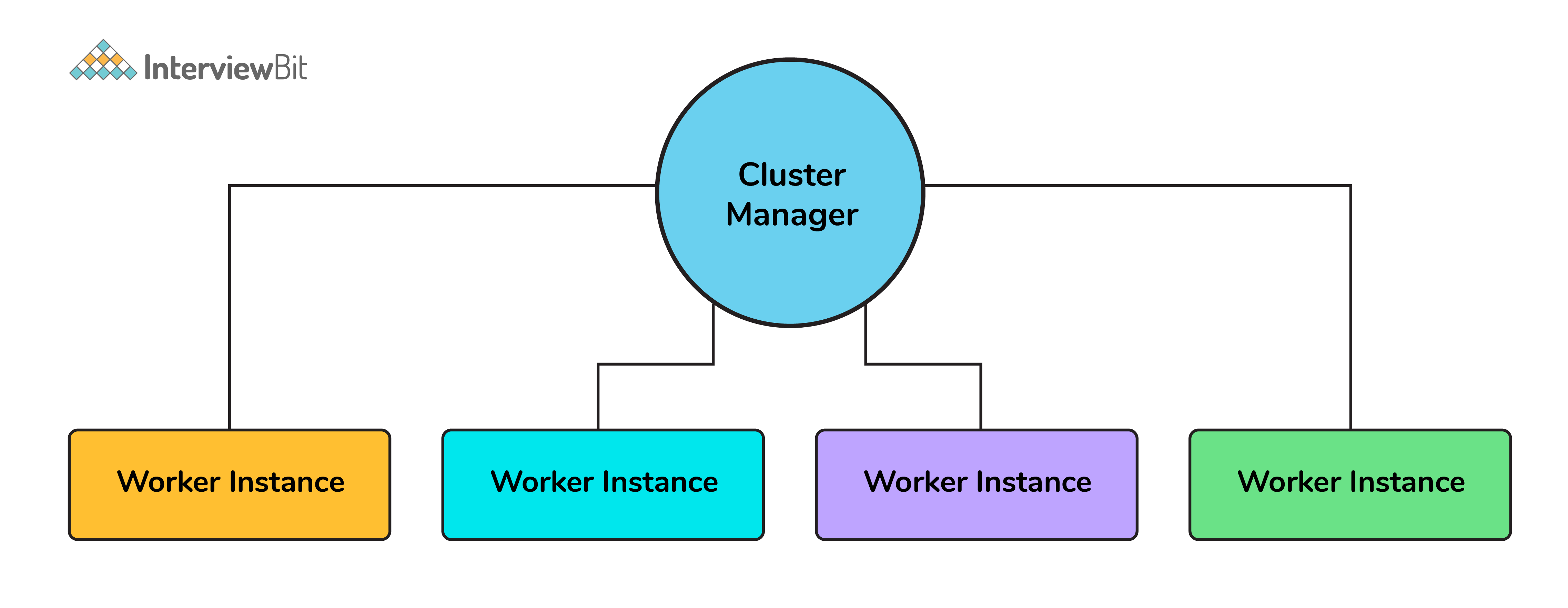
console.log('an event occurred!');

});

myEmitter.emit('event');

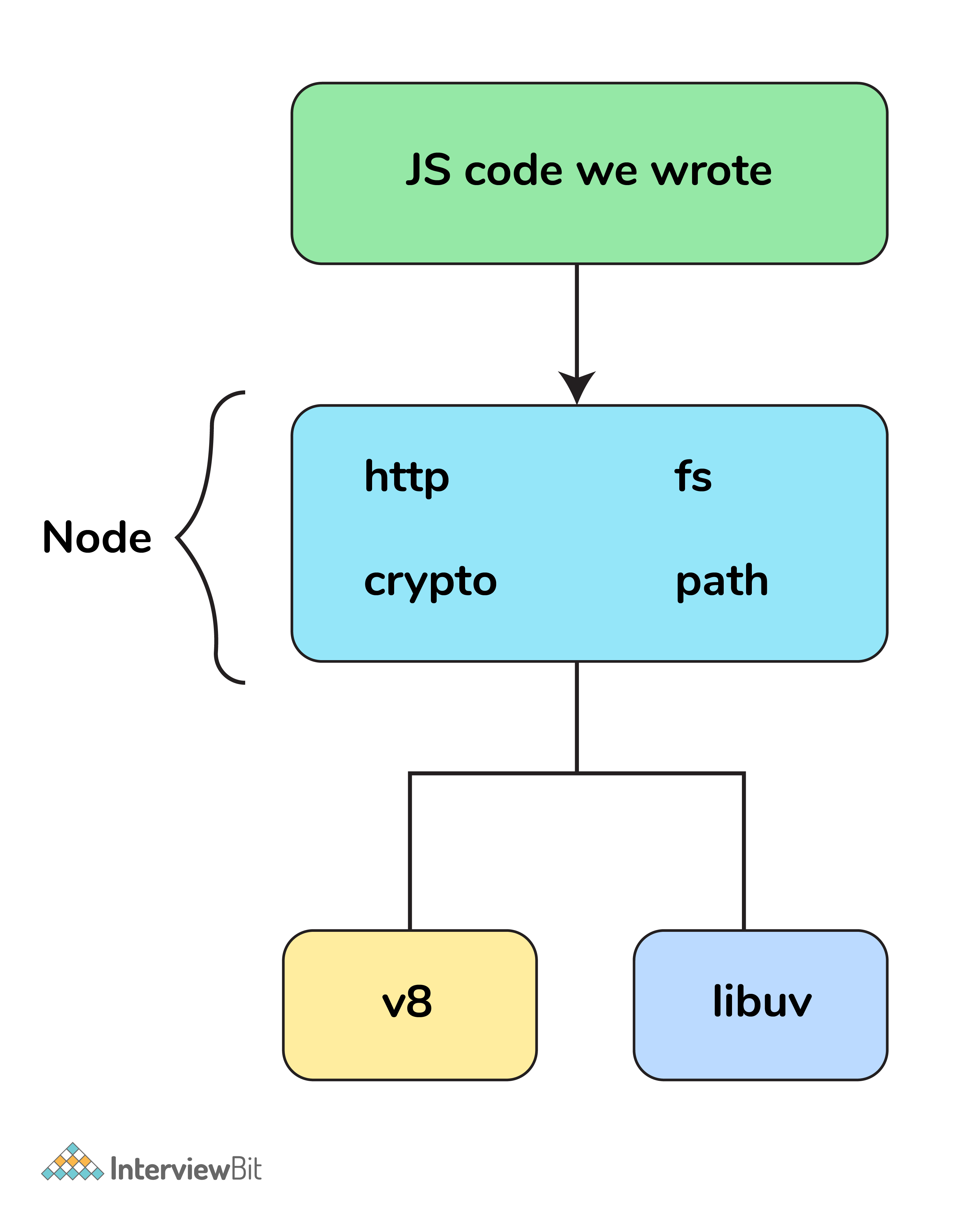
## 31. Enhancing Node.js performance through clustering.

Node.js applications run on a single processor, which means that by default they don’t take advantage of a multiple-core system. Cluster mode is used to start up multiple node.js processes thereby having multiple instances of the event loop. When we start using cluster in a nodejs app behind the scene multiple node.js processes are created but there is also a parent process called the **cluster manager** which is responsible for monitoring the health of the individual instances of our application.

**Clustering in Node.js**

## 32. What is a thread pool and which library handles it in Node.js

The Thread pool is handled by the libuv library. libuv is a multi-platform C library that provides support for asynchronous I/O-based operations such as file systems, networking, and concurrency.

**  
Thread Pool**

## 33. What is WASI and why is it being introduced?

Web assembly provides an implementation of [WebAssembly System Interface](https://wasi.dev/) specification through WASI API in node.js implemented using WASI class. The introduction of WASI was done by keeping in mind its possible to use the underlying operating system via a collection of POSIX-like functions thus further enabling the application to use resources more efficiently and features that require system-level access.

## 34. How are worker threads different from clusters?

**Cluster:**

* There is one process on each CPU with an IPC to communicate.
* In case we want to have multiple servers accepting HTTP requests via a single port, clusters can be helpful.
* The processes are spawned in each CPU thus will have separate memory and node instance which further will lead to memory issues.

**Worker threads:**

* There is only one process in total with multiple threads.
* Each thread has one Node instance (one event loop, one JS engine) with most of the APIs accessible.
* Shares memory with other threads (e.g. SharedArrayBuffer)
* This can be used for CPU-intensive tasks like processing data or accessing the file system since NodeJS is single-threaded, synchronous tasks can be made more efficient leveraging the worker's threads.

## 35. How to measure the duration of async operations?

Performance API provides us with tools to figure out the necessary performance metrics. A simple example would be using async\_hooks and perf\_hooks

'use strict';

const async\_hooks = require('async\_hooks');

const {

performance,

PerformanceObserver

} = require('perf\_hooks');

const set = new Set();

const hook = async\_hooks.createHook({

init(id, type) {

if (type === 'Timeout') {

performance.mark(`Timeout-${id}-Init`);

set.add(id);

}

},

destroy(id) {

if (set.has(id)) {

set.delete(id);

performance.mark(`Timeout-${id}-Destroy`);

performance.measure(`Timeout-${id}`,

`Timeout-${id}-Init`,

`Timeout-${id}-Destroy`);

}

}

});

hook.enable();

const obs = new PerformanceObserver((list, observer) => {

console.log(list.getEntries()[0]);

performance.clearMarks();

observer.disconnect();

});

obs.observe({ entryTypes: ['measure'], buffered: true });

setTimeout(() => {}, 1000);

This would give us the exact time it took to execute the callback.

## 36. How to measure the performance of async operations?

Performance API provides us with tools to figure out the necessary performance metrics.   
  
A simple example would be:

const { PerformanceObserver, performance } = require('perf\_hooks');

const obs = new PerformanceObserver((items) => {

console.log(items.getEntries()[0].duration);

performance.clearMarks();

});

obs.observe({ entryTypes: ['measure'] });

performance.measure('Start to Now');

performance.mark('A');

doSomeLongRunningProcess(() => {

performance.measure('A to Now', 'A');

performance.mark('B');

performance.measure('A to B', 'A', 'B');

});

# Behavioral Questions

## Behavioral job interview questions about time management

### Question #1 - How do you accomplish tasks when under a tight deadline? Give me an example.

Sample Answer:

**Situation:** Well, typically, I try to never commit to a deadline I don’t think I can make. But sometimes, unexpected things happen and you’re forced to think on your feet. For example, at my last job, my coworker had to take some time off work because of an emergency, and his project was left without a manager.

**Task:** My supervisor then instructed me to take over his project and complete what work was left. Suddenly, I had a new project on my hands, and I wasn’t really sure how to handle it, as the deadline was in 1 week.

**Action:** First, I requested a reduction on my own daily sales goals - which I was granted. This way, I could pay more attention to the project, and only a few hours per day to my original tasks. Once I had a consistent schedule and hours set for each of my tasks, it was mostly easy from there.

**Results:** Thanks to my teammates and my good time management skills, I managed to finish up 2 days early before the deadline. And once my coworker came back to work, I was able to review the whole thing with him before submitting it. For what it’s worth, he was thoroughly impressed. And a few months later, I even got promoted based on my performance.

### Question #2 - Describe a long-term project you managed. How did you make sure everything was running smoothly?

Sample Answer:

**Situation:** When I was at Company X, I was managing the web development team in charge of setting up a new website for one of our biggest clients at the time. With most projects, we had a process set up and we would get most sites done in up to 2 months. This project, however, was a bit different, as the website was supposed to be more detailed, with a lot of unique pages. So, we had to be a lot more careful with our time-management.

**Task:** We had a strict deadline of 15 weeks, and I had to make sure that we used up our time as efficiently as possible.

**Action:** Before getting to actual work, I decided that we should plan everything out by the week. After some research and consulting with our team of developers, we decided to split the workload between different stages. We would devote around 1 week to the discovery phase, 5 weeks to design, 3 weeks to initial development and the rest to any modifications and updates.

**Results:** In the end, we actually finished the website with all the promised functionalities in just under 3 months. The client was very satisfied with the result and eventually ended up recommending partners to our firm.

### Question #3 - Sometimes, it’s almost impossible to get everything done on your to-do list. What do you do when your list of responsibilities becomes overwhelming?

**Situation:** As a senior at University X, there were times when I just couldn’t physically get everything done on time. For example, towards the end of my final semester, I was the Student Council President and I was also writing my University thesis. I had to submit my thesis the next day, and I was also working with my fellow student council members to organize the end-of-the-year ceremony for the University.

**Task:** If I had tried to multitask both, I would just have done a poor job. Now, for me, the University thesis was clearly higher up in my list of priorities. After all, this was what my studies were building up to for so long. But I couldn’t just abandon my council members either. With 24 hours until my thesis deadline, I had to think fast.

**Action:** I decided that the best approach was to send all of my notes and outlines for the event to the Student Council VP, who was also a close friend of mine. Luckily, he understood my situation and took over my event-management responsibilities. Then, I had just enough time to edit and finalize my paper.

**Results:** Thanks to the VP, I was able to fix and finalize my Thesis. And fortunately, the event went without a hitch too. In the end, I learned a valuable lesson on time-management, and the importance of having the right team around you who you can rely on.

### Question #4 - Tell me about a time you set a personal goal for yourself. How did you ensure you would meet your objectives and what steps did you take?

**Situation:** I think the most recent, and important, personal goal that comes to mind is that I managed to teach myself web development from scratch. You see, I wasn’t very satisfied as a sales rep at Company X. My coworkers were nice, and the pay was decent too, but I just didn’t see myself growing there.

**Task:** So, I decided that I wanted a career change in a field I’ve always been interested in - web development. Now, because I was working full-time, I had to be very efficient with my time-management skills.

**Action:** I did some research, and all that was left to do was just follow my routine and stay committed. I set up a personal calendar and made sure to study HTML, CSS, and JavaScript for at least 2 hours every day. I gathered a list of beginner-friendly books to start with, and once I was done with those, I paid for some advanced online courses to improve my React and Vue.js skills. When I felt comfortable, I started working on some personal projects for my portfolio and did some freelance work part-time while I was still working at Company X.

**Results:** In the end, I’m glad I stuck to my plan and continued with my set curriculum. If I did not have my calendar planned out with specific objectives, I surely would have been overwhelmed. Sure, at times, it felt like I was basically working 2 jobs and that a lot of the material wasn’t making sense. But I just kept moving forward, and then, I got my first real break as a junior web dev at Company Y.

### Question #5 - Can you describe an instance where your supervisor or manager just gave you too much work with not enough time? What did you do?

**Situation:** I had a pretty rocky start with my manager at Agency X, as we had different expectations for my workload. Normally, I don’t have a problem with a fast-paced working environment, and I tend to thrive when I’m thinking on my feet. But at the agency, I had just finished onboarding, and I was already bombarded with tasks and weekly reports. For the most part, I was managing to get everything done on time, but I realized the quality would suffer if my list of tasks kept getting longer.

**Task:** So, I had to take up my work schedule issue with my manager and let him know about my concern. I decided that being direct, and also respectful was the best approach, and booked the meeting.

**Action:**During the meeting, I remained calm, and just went straight to the point. I explained how I liked my work, but the heavy workload was really impacting the quality of the work.

**Results:** Luckily, he was understanding. I was the first in-house designer they’d hired, and they weren’t 100% sure what was a lot of work, and what wasn’t. We ended up working together to better define my responsibilities. From then on, I was, for the most part, only getting the workload I could handle without diminishing the quality of my work.

### Question #6 - How do you deal with unexpected changes to deadlines?

Interviewers use this question to assess how you react to time-sensitive projects and how well you work under pressure. In your response, highlight your ability to react quickly without sacrificing quality, and emphasize strategies that have helped you adapt to changes in your previous work experience.

**Example**: *"Last April, my manager assigned me to increase a software's efficiency by 3% in 30 days. However, the deadline changed a few days later, and I only had two weeks to complete the task. While I was a little nervous about my ability to complete the task in time, I asked fellow team members for their advice and learned a new strategy that allowed me to increase software efficiency by 4% and exceed my goal in two weeks."*

## Behavioral job interview questions about communication skills

### Question #1 - How do you handle a disagreement with your colleagues? Give me an example of when you successfully persuaded someone to see things your way at work.

**Situation:** When I was working as a recruiter at Company X, I noticed that one of the candidates who had sent in their application was perfect for the role. Though he didn’t have a university diploma and his resume wasn’t too polished, reading his cover letter, it was obvious he knew the industry and had delivered clear results.

**Task:** I thought it was worth giving him a shot, but my supervisor didn’t see it that way. She skimmed through the resume and told me not to waste time, and just discard the candidate.

**Action:** I was, however, still pretty confident in the candidate, so I talked to the supervisor over lunch. I took a bit of an indirect approach, though. Instead of trying to directly pitch the candidate, I asked her to clarify the job description a bit more. We went a bit in-depth on what, exactly, we were looking for in the candidate, and once we were done discussing it, I told her that we happened to have a candidate that possessed all the relevant experience, but his resume was a bit weak.

**Results:** Convinced, the supervisor decided to give the candidate’s application a more in-depth look and realized that they were, in fact, very qualified. She thanked me for bringing it up and agreed with me that the candidate was worth calling in for an interview.

### Question #2 - What would you do if you misunderstood an important task on the job? Give me an example.

**Situation:** At my previous internship at Company X, I underestimated the amount of time it would take me to finish a presentation for a team meeting. The deadline my boss gave me was around a week, which was completely fair and I didn’t think it would be a problem. However, apparently, we had some miscommunication with what he’d meant with the deadline. I thought it was the date where we would go through the presentation, edit it together, and submit it like that. Apparently what he’d meant, though, was to have the presentation 100% ready on that date.

**Task:** So, I had to submit a draft presentation first, edit it based on my manager’s comments, and then present the report, all within 2 days.

**Action:** I booked a meeting with the manager for the following day, and spent 4 extra hours at the office to make sure that the first draft of the presentation was spotless. We held the meeting the next day, and went through the presentation together to make sure it’s spotless.

**Results:** The manager loved the work, and it only took us around 30 minutes to finalize the whole thing.

### Question #3 - Have you ever had to work under someone who wasn’t very good at communicating? What happened?

**Situation:** Yes, at my last job as a tech recruiter the hiring manager I was working directly with was somewhat more difficult to communicate with. He had very strict and precise requirements on the type of candidates he wanted to invite for interviews. He wasn’t open to much communication on the matter or trying new things even when the company desperately needed new hires. This one time, I got a candidate that was a pretty good fit for the job, but was lacking in some aspects.

**Task:** I wanted to make sure that we got the person in for an interview, but I was 100% sure that my hiring manager would shut me down.

**Action:** So, before running the candidate through him, I called them and collected his biggest strengths to present to the hiring manager.

**Results:** The hiring manager did, indeed, end up liking the candidate and calling them in for an interview.

### Question #4 - Tell me about a time when you successfully explained a technical problem to a colleague or a customer who didn’t have a tech background?

**Situation:** I’ve worked as a tech support specialist before, so I really excel at this. I’ve had to explain complex concepts to customers on a regular basis, but to give you one single example, I’ve had to explain to clients with next to no understanding of computers how to delete a virus on their computer.

**Task:** After trying to give basic instructions to the client, they still didn’t really understand much, so I had to come up with a smarter solution.

**Action:** So what I did was, I walked them through the entire thing step by step while explaining it simply but in no condescending terms. Instead of making them do most of the work, I walked them through the process of getting me to connect with their computer, and then I explained to them what, exactly, I did.

**Results:** The customer was very happy with my work, and we managed to fix the issue with their computer.

### Question #5 - Can you tell me about a time you gave a presentation that was particularly successful? Why do you think it went well?

**Situation:** Sure thing. As the business development manager at Firm X, there were quite a few opportunities when I had to speak in front of a crowd. The most recent, and successful, one was for the new project we were launching.

**Task:** I was called on to speak for a 2 department-wide meeting, of up to 50 people. Now, I had never delivered a presentation to this many people, but luckily, I knew most of them quite well after years of working with them.

**Action:** Working with 2 other members of my team, I decided to take a more creative approach, and create a short video (a skit) to hook the audience. That was the intro, and then we used PowerPoint and hands-on examples to show what to expect from the new project launch. And finally, we dedicated the last 5 minutes to a Q&A session.

**Results:** It felt longer, but the whole speech took about 15 minutes in total. We got great feedback from the audience, and I was later asked to present at the all-hands meeting the next month. I knew my colleagues well enough and I tried to make the speech as if I was having a one-on-one conversation with a friend - with a few jokes in-between.

## Behavioral job interview questions about teamwork

### Question #1 - Tell me about a time when you had to work with someone completely different from you. How did you adapt to collaborate better?

**Situation:** Sure, I always enjoy working with new and different people. Usually, because they bring something new to the table. At Company X, there was a particularly young developer who was assigned to work with me on a new software development project, and I was to run him through what our typical coding process was like.

**Task:** It was also my job to get to know him, and find common ground so that we could effectively work together. The fact that he was younger wasn’t an issue for me, but because he was completely self-taught, he didn’t know a lot about the industry methodologies we used.

**Action:** Teaching him everything from scratch would take too much time. So, instead, I briefly explained the development process (waterfall model) we were using for that specific project, and taught him how to write tests for our code-base. Writing tests is the number 1 way to learn what code does. After all, that’s how I got started with development.

**Results:** I also sat down and helped him go through the material at times, but in the end, he surprised me by how much of a fast-learner he was. He just needed a bit of encouragement and guidance. Through this approach, he learned our whole routine in less than a week, while most of our new hires needed at least up to 2 weeks. In return, I learned a lot about multitasking and time-management from him. The whole thing was a win-win situation, and it was all smooth sailing the next time we worked together (which was quite often).

### Question #2 - What do you do when your team member refuses to, or just can’t complete their part of the work? Give me an example.

**Situation:** There was one co-worker at Company X who was notorious for being bad at deadlines. But she would always end up delivering exceptional work, just a few hours (or worse - days) late. For some reason, the company was ok with this as her work was just too good. So, this one time, the management put us together to work on a time-sensitive project.

**Task:** Our task was to turn in a sales presentation together and have our manager go over it before sending the client the final version. Because of how important the project was, I didn’t want to risk going over the deadline - as this would also directly impact other people. Either way, for everyone’s sake, I had to somehow get her to hurry up with the project. So, I decided to try and push her a little and see what would happen.

**Action:** I started regularly checking in on her to see where she was with work. I would bring it up at times over lunch, send a quick Slack message, and so on. She wasn’t taking this quite well, but it DID get her to work faster and more efficiently.

**Results:** At the end, the constant check-ins and pushing did have a positive effect, even though the co-worker didn’t particularly like me too much once we were finished with the work. We even managed to submit the final version of the presentation 2 days before the deadline.

## Behavioral job interview questions about working with clients

### Question #1 - Clients can be difficult to work with sometimes. Can you describe a situation when a client was wrong and you had to correct them?

**Situation:** Absolutely. One of our past clients at Agency X came to us because his Facebook advertising strategy wasn’t working. He was driving traffic but wasn’t getting any conversions, so they thought that it was because they weren’t reaching the right audience. We realized, though, that it was actually because their product homepage wasn’t really that convincing. The client, however, was adamant about “not fixing what wasn’t broken.”

**Task:** I had to somehow communicate with the client that the service he wanted wasn’t what he wanted - there was no way for us to fix his Facebook ads if his homepage wasn’t selling the product.

**Action:** We had to give the client an ultimatum - they either go with our approach, or we wouldn’t be able to get the results (and hence, work with them).

**Results:** After some back and forth, the client grudgingly agreed to do an A/B test between the existing landing page, and one that we’d propose. So, we tested the two landing pages with the same ads he’d been running, and ended up getting 2.5x better results. From then on, the client was a lot more willing to allow us to experiment with whatever we proposed.

### Question #2 - How do you handle irate customers? Give me an example.

**Situation:** Working in customer support, you really get to talk with many different kinds of people. I remember I had one angry customer that called the helpdesk once to complain. He kept repeating the product he bought was faulty and demanded me to resolve the situation then and there.

**Task:** Customers calling for refunds happen all the time, but this one was different as he just kept shouting over the phone the whole time. I had to get him to calm down if I wanted the call to go anywhere.

**Action:** Fortunately, I had experience dealing with loud customers, and knew the first thing I had to do was listen to his story. Halfway through telling his story, he calmed down once he realized I was trying to help. He explained that the product was supposed to be a gift, and that’s why he was so frustrated. Then, I offered 2 solutions: a refund or a replacement for his product with express delivery.

**Results:** The customer opted for the replacement option. I called him back once they received the order just to check-in if he was happy with the product. He turned out to be happy both with the product and our service, and thanked me for the help.

### Question #3 - We all make mistakes sometimes we wish we could take back. Is there a time that comes to mind where you wish you had handled a situation with a client or colleague differently?

**Situation:** This one client we worked with was particularly difficult. They were extremely unpleasant to work with and treated our staff pretty badly. The management, however, insisted on sticking with them, since they made up for a good chunk of our income. At one point, though, the client just barged into our office and started yelling at their account manager for a small mistake on their end.

**Task:** At this point, I realized that working with the client was really affecting our staff negatively, and we’d be losing some good employees if we kept working with them.

**Action:** So, I set up a meeting with the management team, and gave them concrete facts and figures about the client. Sure, they were paying us good money, but they were really hurting the workplace morale.

**Results:** After hearing me out, the management agreed and fired the client. They decided that overall, the impact such clients had on the company wasn’t worth it, and started doing stricter vetting during discovery calls.

## Behavioral job interview questions about adaptability

### Question #1 - Tell me about your first job in the industry. What did you do to learn the ropes?

**Situation:** Well, my first job in the field was as a junior dev ops engineer. While I did have extensive knowledge of the field, I didn’t have too much experience doing it.

**Task:** This made it very hard for me to get started with the job. While I was working almost all the time, I wasn’t getting too much done.

**Action:** So, what I did was, taking a lot of my personal time to really work and learn the ins and outs of dev ops. I also made sure to talk to my team members and get their input on daily tasks.

**Results:** A few months into the job, I managed to learn the ropes and ended up being a lot more productive.

### Question #2 - Can you give me an example of when you had to adapt to a new and sudden change in the workplace? What happened?

**Situation:** Sure thing. In my previous position as an account manager at Company X, we had to suddenly change all of our CRM software and move all the data to a new tool. The CRM tool we’d been using till now wasn’t fit for a growing team, and on top of that, they were upping their pricing, so it wasn’t really worthwhile for us.

**Task:** I was put in charge of finding the replacement CRM, as I was the one who knew the previous one inside-out. And this was also an opportunity for me to clean up our outdated info and start fresh. All the while, I still had to handle my daily responsibilities and as usual.

**Action:** So, the first thing I did was ask our sales associates and lead generation teams what they thought of the old CRM, and if there were any new features they were lacking. After doing a bit of research and asking around, I found the perfect tool that had it all - sales analytics, email integration, and more. And because I typically have no problem with learning new tools, I stayed in one evening, transferred our data to the new tool, and wiped the old account. Finally, I sent a new announcement to the entire team about the new software, as well as a video on how to use it.

**Results:** We completed the transfer with 4 days to spare, the team was satisfied with the new CRM, and my daily responsibilities as an account manager didn’t suffer.

### Question #3 - Give me an example of when you had to suddenly perform under pressure. What happened and how did you handle it?

**Situation:** As a seasonal worker, there have been a lot of times where I had to juggle extra responsibilities. My last position as a line cook at Restaurant X comes to mind. During summer, we were pretty much always full, and sometimes, even understaffed to handle all the customers. To make things worse, we didn’t have the best shift system at the time either. So, if someone were to unexpectedly not show up for their shift, we’d have to put out the fires as they came up.

**Task:** Which is exactly what happened when one of our waitresses had to cancel her shift due to an emergency.

**Action:** So, I stepped up and took her shift as soon as I had clocked out of mine as one of the line cooks. Luckily, I had previous experience working as a waiter.

**Results:** I was tired and a bit uncoordinated at the beginning, but at the end of the day, everything worked out just fine.

### Question #4 - Tell me about a time you were new to a situation or environment. How did you adapt?

When I started my last job, I had never worked as a full-time software engineer and knew that I had a lot to learn. However, I made sure to ask many questions and take notes about what I learned, reviewing them at the end of each workday. Eventually, I became familiar with the systems and protocol and exceeded my goals within the first six months of employment.

### Question #5 - As a software engineer, you must be both predictable and innovative. How can these traits coexist in your work?

By asking this question, the interviewer is likely trying to assess your ability to balance consistency with creativity and gain an understanding of your philosophy toward your work. Try to provide an answer that's true to your philosophy but also reflects the values of the company with which you're interviewing.

**Example**: *"I believe that the balance of predictability and innovation is the foundation of my work as a software engineer. It's important that my team delivers high-quality software within a predictable time frame, but our everyday work requires us to be innovative and develop alternative systems and processes. From my experience, it is an engineer's ability to balance predictability and innovation that leads to their success and potential to be an effective team member."*

## Behavioral job interview questions about leadership

### Question #1 - Tell me about a time when you successfully delegated tasks to your team.

**Situation:** Well, at my first job as a team lead, I had to really get to know most of my team in order to delegate tasks appropriately.

**Task:** Most team members were new to the company, so I didn’t have much to go with.

**Action:** So, I sat down with each team member individually, and really got to know them and their strengths and weaknesses, and distributed tasks based on their personality.

**Results:** Team members were pretty happy with the tasks they got, and started off their relationship with our company on a positive note.

### Question #2 - Can you tell me about a time when you had to perform a task or work on a project you had no previous experience before? How did you approach this situation and what did you learn?

**Situation:** In my previous position at Company X, my manager had to leave unexpectedly for about a month due to a medical condition. Fortunately, she was able to give us a week's notice.

**Task:** Because of that, our director asked me to fill in as the interim manager. I was familiar with the basics of management on a theoretical level, and I had worked with my manager closely before, but I certainly wasn’t trained to be a manager yet. Though, I wasn’t going to say no, and I, more or less, felt confident about my ability to take on the new challenge.

**Action:** So, I accepted the position. The first thing I did was gather the team and let them know about the situation. I was very open about my lack of experience, and asked them to be open about giving feedback when possible. I also asked a manager for an hour of their time to pick their brain and make sure I’m doing everything right.

**Results:** In the end, we managed to get through the month without any problems, and delivered all the projects on time. When my manager returned, she was very pleased with the work, and I even got compliments from our director. Because of my success with the role, I was then promoted to team manager at the end of that year.

### Question #3 - What are your organizational strategies?

When answering this question, be honest and reflect on what strategies have worked for you in the past. Discuss which methods were the most successful and why, and show that you understand your own work style and best practices. You can explain your time management strategies and give past examples.

**Example**: *"Over the years, I've found that I perform the best at work when I'm the most organized. For this reason, I have developed a variety of strategies to make sure that I'm managing my time and projects accordingly. For example, I use time management applications on my phone that help me stay focused, and I limit multitasking when possible so that I remain concentrated on one task until I fully execute it."*

# Cultural Questions

## Would you like to work overtime or odd hours?

 I know that in the company being asked to work for an extended number of hours comes with a good reason, so I am ok with it. It an extra effort means I am doing something for the company, I'll be happy to do it.

## What is more important to you: the money or the work?

- "Money is always important, but the work is most important for me."

- "I would say that work is more important. If we work and achieve Company goals then obviously money would follow. I believe work to be prior."

- "Work is more important for me. Working just for money may not be fulfilled if I don't feel satisfied with my job. My work makes me stay productive, and money would naturally come along well."

- "I think money probably matters to me about as much as it does to anyone. It's vital and necessary for us to live and prosper but, at the same time, it's not my single most important driving force. I believe that money is rewarded for work."

## Why should we hire you?

"I believe that everyone starts with a beginning, I need a platform to prove my abilities and skills. I think your company is the right place to explore my abilities. I need to be a part of your growth. I will do my level best."

"I have a good experience in that particular field (field of your specialization), and I think my talents will be a big contribution to the continuing pursuit of excellence of your company."

## Assume you are hired, then how long would you expect to work for us?

"I will do my best for the growth of your company as long as I have the career growth, job satisfaction, respect and a healthy environment, then I don't need to change my company."

"I will work with the company as long as my presence benefits the company and I get ample opportunity to grow and develop both professionally and monetarily."

"Everyone looks for a bright future, healthy work environment, good salary, job satisfaction and I am pretty sure that your company gives such things, so I don't need to change the company."

"I will work with the company as long as my presence benefits both the company and mine in parallel. So your company gains good results, and I can be in a good position to improve my skills."

## How would you rate yourself on a scale of 1 to 10?

“I will rate myself 8 out of 10 because I would never like to think that there should be a room left for putting in more efforts. That thought will create an interest in learning the things. Thank you very much for giving me this wonderful opportunity.”

“I will answer this question based on some parameters. As far as hard work is concerned, I will rate myself as 8 because there should always be a scope to increase our skills which will create an interest in learning the things. When it comes to creativity, I would like to rate myself as 9. In the past, I have designed banners and brochures which were appreciated by the clients. To talk about patience, I will tag myself with 6 because I am an entry-level professional. Same as personal life, even professional life needs more experience for more patience. That is probably why in most companies, senior management looks more patient than entry level or even middle level. Overall, I would rate myself as 8 on a scale of 1 to 10. “

## What is your objective in life?

"My short-term goal is to work in a reputed organization like yours where I can enhance my technical skills and knowledge.

My long-term goal is to see the company at a topmost position where I am one of the people responsible for that."

"My goal is to become a successful person and make my family proud of me."

## What are your greatest strengths

“I am highly organized. I always take notes, and I use a series of tools to help myself stay on top of deadlines. I like to keep a clean workspace and create a logical filing method so I’m always able to find what I need.  
I find this increases efficiency and helps the rest of the team stay on track as well.”

“My main strengths are the ability to use my initiative to take on challenges. I am always proactive at what I do, and that keeps my mind stimulated and focused.”

“My greatest strengths would be my intelligence and thoughtfulness. I believe that in every work environment you need to process every step and be detailed in your work. “

“My time management skills are excellent, and I'm organized, efficient, and take pride in excelling at my work.”

“My greatest strength is my ability to focus on my work. I'm not easily distracted, and this means that my performance is very high.”

“My biggest strength is my Confidence. Apart from that, I am Hardworking, self-motivated with a positive attitude towards my career and my life. If I have a problem, then I think its an opportunity for me to express my ability. “

## What is your greatest weakness

“In the past, I had great opportunities to work with startups. Sometimes, they had very tight deadlines because they had to show MVP or PoC to investors on time. But I usually strived for the perfect result and pay too much attention to detail and sometimes, it caused missing the deadline.”

 "I used to be very disorganized, always forgetting assignments and birthdays. But I managed to work out a computerized system of to-do lists and reminders that keep me on top of everything."

"I am a straightforward person, and I cannot say no when someone asks me for help."

## What are your hobbies?

My hobbies are dancing, Internet surfing, playing Chess, listening to music, watching the news channel. In my spare time, I like to read news on my phone and traveling to my hometown. Thank you for giving this opportunity to introduce myself.

## Explain, how would you be an asset to this organization?

"To become an asset for an organization, we have to punctual, dedicated, quickly adapt of the environment and positive working attitude I have all of these qualities so I will prove an asset for this company."

"My skill in XYZ company is outstanding. I have earned a lot of certificates and awards from my past employers. As an employee, I can handle pressure with ease and can work with minimal supervision."

## Would you lie for the company?

"It depends on the situation if my lie creates a positive impact on the company and It was useful for many people, then I will lie."

"It depends on the situation if my lie creates a positive impact on the company and It was useful for many people, then I will lie."

## How do you get to know about our company?

I get to know about your company from several online websites.

## What does success mean to you?

“If I feel I am making a difference working with a team of people to make a more profitable company. It is a success for me.”

“If I put a smile on someone face and make him happy, it is a success for me.”

## Describe yourself in one word?

Original, genuine, logical, incredible, focused, curious, active, quick, balanced, achiever, etc.

## What is the difference between confidence and overconfidence?

Confidence is based on facts and knowledge, and overconfidence is based on speculation. There is a small difference between confidence and overconfidence.

“Yes, I can do this work is self-confidence. But only I can do this work is overconfidence.”

“Confidence is an internal belief that I am a right person for this job and overconfidence is thought that I am only the right person for this job.”

## Just imagine that you have enough money to retire right now. Would you?

 "No sir, I don't think so. I am a professional, and I love my work, so there is no question to leave my work. Yes, it may be that I would take a break to spend quality time with my family."

## What makes you angry?

"Sir, I am not a short-tempered person, but I feel a bit of annoyance when someone disturbs me in my work without a genuine reason. Although I am an even-tempered person, when I get angry, I try to channel my negative feeling in my work."

## What was the most difficult decision you have made in your past life?

“After completing my graduation, the toughest decision is whether to go for higher studies or do a job. Then I chose the job because getting trained is better than educated and it was also the demand of that time.”

“My toughest decision was to take admission in B.tech. I belong to a middle-class family, and my father was not in favor of taking admission, but I convinced him, and today he is very happy.”

“Before some time when I had to choose between joining a group of employees protesting some issue, and staying away from the issue. I ended up being a mediator between our immediate supervisor and employees, and I am glad I made that decision because it all ended well and without further conflicts in the company.”

## Why did you apply to this job? (Why should we hire you for this position?)

In my next position, I’d like to continue building my project management and leadership skills. I reviewed the job description and saw an emphasis on team leadership and project management, so it seemed like a great fit. And since I’ve already spent 3 years managing 7-figure client projects and leading a team of five people in my most recent company, I’m confident that I could get up to speed very quickly and start contributing results for you in this role.

## Where do you see yourself in 5 years

My career goal is to be a staff engineer in next 5 years by growing architect and management skills.

## How do you handle conflicts with team members?

When I faced with a conflict, I like to ask questions and understand my coworker’s perspective. This helps keep the situation calm, helps them feel like they’re being heard, and after this, I’ve found it’s much easier to come to an agreement or compromise while both staying a lot calmer.

## Why did you leave your last job

It was a contract role and worked on last position for 2 years. The project has been completed and I had to spend up to 5 hours a week.

## What motivates you most

Learning new things, coming up with creative ideas to improve something, or make something new, analyzing complex data in order to draw clear and simple conclusions, working well as part of a team.

## In your opinion, what are the principles of good software engineering? What are some basic principles everyone should follow?

I think one of the main principles of software engineering, and one I try to live by, is to keep things as simple as possible.  
You're often already dealing with complex algorithms and design concerns, so no need to make things even more difficult with overly complicated, resource-heavy code.  
Your code should be simple, lean and easy to read.  
If you start there, the rest will follow.

## What do you do outside of work

A lot of my spare time lately has been dedicated to teaching myself how to keep work-life balance.

Besides, I love watching football matches as well as playing the game with friends. My favorite football team is Real Madrid.

## How do you work with non-technical clients? (Tell me about a time you had to work with difficult client)

First, I would try to understand the big picture of his/her requirements. And going into details, relate his business requirements to technical field and let the client understand the approach that I am going to vet. It is very important to understand the difference between me and the client (technician vs non-technician) and always try to keep consistent and assertive communication. After discussing the requirements, it would be good to draw development roadmap to the client and upon mutual agreement, start the work.

Sir, it's a career move. I have learned a lot from my last job, but now I am looking for new challenges to broaden my horizons and to gain a new skill-set.

## How do you handle urgent deadlines?

I am used to working under tight deadlines, so I set my most urgent tasks at the top of my to-do list every morning when I get to work. Then, I establish a clear deadline for myself that's usually a day in advance of company due date. Recently, I had to shift around my workload to accommodate an urgent product order. The client wanted custom modifications to the product completed in a week, but our normal lead time for implementing modifications is around 10 to 14 days. I communicated with my production department and manufacturing team to implement urgent changes in the production timeline. These adjustments allowed us to ship the product to the client on time, effectively reducing the time it took to apply the customizations by three days.

## What do you value most?

I value happiness. By keeping this value at the center of my life, I am able to easily make decisions in the best interest of my family, my business and myself. A happy family, a happy home and a happy work environment add up to a happy life. I value making a difference in life and living with integrity.

## Do you have any experience in mentoring others? or onboarding juniors to your project?

Yes, in my last position, I organized the onboarding process of junior developers in the company. Introduced the general development flow, projects we work on, version and task management, etc. Also mentored their work for 6 months to ensure they meet the company standard and produce efficiency work results accordingly.

## What are the various types of software maintenance?

Maintenance types are corrective, adaptive, perfective and preventive.  
  
**Corrective**: This type of maintenance is used to remove the errors spotted by business users.  
**Adaptive**: This maintenance activity is performed to check the changes made in the hardware and software environment.  
**Perfective**: This type of maintenance is used to implement changes in existing or new user requirements  
**Preventive**: This maintenance activity is performed to avoid any issues in future implementations.

## What is the worst job you ever had?

If I had to pick the worst job I’ve ever had, it was probably back to my very early job, working as a volunteer for an insurance company. They often scheduled me to work immediately after class, which cut into the time I had to study, and they often had very tight deadline.

## Tell me about a time you made a mistake (failure)

In my first job, my manager asked me to develop several forms of tracking projects. I said “yes” immediately because I wanted to deliver result and demonstrate my skill asap. But as I started working on the project, I realized that I didn't understand the overall goal. And the project ended up creating templates that didn't meet my manager's expectations.  
My manager was disappointed in me. She told me that if I had asked some clarifying questions, I would have delivered better result. I admitted to my manager that I had made a mistake and learned that it doesn't make me look stupid when I ask questions and it's better to speak up quickly. That's how I've handled those situations ever since.

## How would you describe yourself with one word?

I am highly organized.  
I always take notes, and I use a series of tools to help myself stay on top of deadlines.  
I like to keep a clean workspace and create a logical filing method so I’m always able to find what I need.  
I find this increases efficiency and helps the rest of the team stay on track as well.

## What salary (expectation) are you looking for?

I’m more interested in the role itself than the pay.  
That said, I’d expect to be paid the appropriate range for this role, based on my 8 years of experience.  
I also think a fair salary would bear in mind the cost of living here in Hong Kong.

## What are you passionate about?

One area that I’m passionate about is self-development and self-improvement in my personal life. I’m always looking to challenge myself and learn new things. That’s one reason I enjoy working in software engineering; I’ve learned great communication skills, architecting skills, and problem-solving skills that help me in my day-to-day work but also in my personal life in terms of confidence, communication, etc.

## What makes you unique from other candidates

Working at a startup gave me an opportunity to understand the ins-and-outs of the industry, and to take on tasks I might not have at a larger company. I think this experience gives me a slight edge over other applicants.

## How long will you stay with us?

As long as there's a lot of work to do, then I'd be happy to stay for a long time. Since I'm a busybody, I like to being productive most of the time.

## Are you a team player? or do you want to work alone?

It depends on the task.  
When it comes to brainstorming, teams produce great ideas with multiple input, and teams can highlight people’s strengths.  
But I certainly enjoy working on my tasks as an individual as well, since in many ways it takes the ability to work alone for the team to fully succeed.

## What is your workflow process for sticking to a project timeline?

Time-management is one of my strengths. I will divide the project requirements into smaller chunks and schedule the work process according to them. Practically, I am a fan of Agile methodology which handles it pretty nicely

## What would you do if you encountered unexpected difficulties on a project?

Rather than trying to find the solution by myself and spending much time on it, I will share it with the team and find the solution together.

## What do you do when you sense a project is going to take longer than expected?

I let the team-lead/product-owners know about the possible delay in the product delivery with proper reasons. And find the possible solution together.

## Tell about how you work under pressure

This is a story when I worked as a lead developer at a US company. We were an agile team and worked on sprint by sprint, which usually took 3 weeks to complete. One day, Scrum Master said that the client wanted the sprint delivered in 2 weeks. I met with our team and reviewed the calendar. We eliminated team meetings and shifted lower-priority tasks until the end of the 2 weeks period to add extra hours to our workweeks.   
I challenged my team to complete the project within 14 days or left and as a reward promised two days of extra PTO time. Our team got the job done in 12 days.

## Do you have any questions for me?

“Thank you for giving me this opportunity. After my overall performance till now if I got selected what I need to improve and if I'm not selected how can I succeed further. Can you give any advice sir?”

Well prepared questions. Very important moment to move forward or not.  
1. What’s the makeup of the team as far as experience? Am I going to be a mentor or will I be mentored?  
2. What does this company value the most and how do you think my work for you will further these values?  
3. What kinds of processes are in place to help me work collaboratively?  
4. What should be the most challenging part of this position?  
6. What’s the most important thing I can do to help within the first 90 days of my employment?  
7. Do you have any questions or concerns about my ability to perform this job?  
8. When top performers leave the company why do they leave and where do they usually go?  
9. What do you see in me? What are my strongest assets and possible weaknesses? Do you have any concerns that I need to clear up in order to be the top candidate?  
10. Who are the “coolest” people on my team? What makes him or her “cool? Can I meet them?

## What is the most inventive or innovative thing you have done? Describe something that was your idea, e.g., a process change, a product idea, a new metric, or a novel customer interface. It does not need to be something that is patented. Do not write about anything your current or previous employer would deem confidential information. Provide relevant context for us to understand the invention/innovation. What problem were you seeking to solve, and what was the result? Why was it an important problem to solve? How did it make a difference to the business or organization?

I was committed to one project that develops the web platform which provides the document encrypting service when I was at Ceridian.

After launching the production successfully, we should decide how to provide the APIs to the customers. Most team members agreed to provide the API documentation since we have well-structured swagger documentation about the APIs. But I had another idea that can provide a more user-experienced and user-friendly interface.

The idea was to develop the node and python packages that customers can easily integrate into their projects and access the APIs using wrapper functions. Of course, it requires more effort and time to develop new packages. But as you know, we should always care about customers when we release in production, so that we can get more rating in the market. After careful consideration, the company had decided to go with my idea.

Finally, we can deliver an easy-to-use and qualified product to the customers. And we can get a bunch of subscribers and 5-star feedback at the same time.

## What was the biggest mistake of your life?

I think the worst mistake I ever made at work was in my first ever job - five years ago now. A more senior member of the team seemed to take an instant dislike to me from the start, and one day she was particularly unpleasant to me in front of several colleagues.

Later on, I was talking to one of those colleagues who was, I thought, attempting to console me. Angry and hurt, I foolishly vented my feelings and told her what I thought of the lady in question. I was naturally shocked to find out that she went on to tell everyone what I had said and this certainly didn't help my relationship with the team member who was causing me problems.

Rather than let the situation carry on, I chose to have a quiet word with this lady to find out what her problem was with me and to see if we could put it behind us. It turned out it was nothing personal; she just resented the fact that a friend of hers had also been interviewed for my position and had been turned down. Once we had got matters out into the air, her behavior changed, and we got on quite well after that. However, I certainly learned a lot from experience. I learned that careful communication is vital in managing interpersonal relationships and that if I have a problem with someone, it's always best to talk it over with them rather than with someone else.

## Describe the three things that are most important for you in a job?

“According to me, Honesty, Loyalty, and determination to achieve my team's target are the three important things in a job.”

“According to me, Professionalism, growth and a healthy work-life balance are the three important things in a job.”

## What are your expectations from the company?

I have always wanted to work with an organization which provides a very comfortable and home like work environment. I would like to work in the company where I can get the opportunity to learn and enhance my skill to become a better professional in the future.

## What gets you up in the morning?

*It is my promise that gets up me in the morning. My promise is to learn something new and someone in need. It provides me the satisfaction that I am making a difference in someone life.*

## What is your favorite book?

*This question is asked to judge your taste about reading books. The interviewer wants to know what types of book you like. Would you fit for the company culture?*

*Answer this question according to your sense, your knowledge about the book. Only named the books you have really read. You should choose something from a reputable author that your interviewer has probably heard of.*

## As you said, internet surfing is your hobby. Which site do you surf mostly?

*This is your choice that which sites you surf most, but while answering this question always refers to sites which are relevant to your field of job. Don't take the name of social networking sites or other irrelevant sites.*