React.js interview questions and answers that are commonly asked in interviews. I'll keep them simple, practical, and easy to remember.

□ 1. What is React?

Answer:
React is a **JavaScript library** used to build **user interfaces (UI)**, especially for single-page applications.

It was created by **Meta (Facebook)**.

It helps developers build reusable UI components.

□ Example: Facebook, Instagram, Netflix use React.

□ 2. What are components in React?

Answer:
Components are **independent, reusable building blocks** of a React app.

There are 2 types:

- Functional Components (modern approach)
- Class Components (older approach)

Example:

```
function Hello() {
  return <h1>Hello World</h1>;
}
```

□ 3. What is JSX?

Answer:
JSX stands for **JavaScript XML**.

It allows writing HTML inside JavaScript.

Example:

```
const element = <h1>Hello React</h1>;
```

 JSX is not required but makes React code easier to read.

---

4. What is the Virtual DOM?

Answer:
The Virtual DOM is a **lightweight copy of the real DOM**.

React updates the Virtual DOM first, compares changes, and then updates only the necessary parts of the real DOM.

 This makes React very fast.

---

5. Difference between Real DOM and Virtual DOM

Real DOM	Virtual DOM
Slow updates	Fast updates
Updates entire page	Updates only changed part
Direct browser manipulation	Memory-based copy

Real DOM vs Virtual DOM (Basic Idea)

Real DOM	Virtual DOM
Actual structure of webpage in browser	Light copy of Real DOM in memory
Slow when updating	Fast updates
Re-renders whole page or large part	Updates only changed part

---

□ Simple Analogy

- **Real DOM** → Printed book □ (you rewrite whole page if something changes)
- **Virtual DOM** → Digital document □ (only edits changed words)

□ Real Example (Counter App)

□ Real DOM Approach (Traditional JavaScript)

```
<button onclick="increase()">Click</button>
<h1 id="count">0</h1>

<script>
let count = 0;

function increase() {
  count++;
  document.getElementById("count").innerHTML = count;
}
</script>
```

□ What happens here?

- Browser directly updates DOM
- Each change touches real page structure
- If page is large → performance slows

□ 3. Virtual DOM Example (React)

```
import { useState } from "react";

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <button onClick={() => setCount(count + 1)}>
        Click
      </button>
      <h1>{count}</h1>
```

```
    </div>
  );
}
```

□ What React does internally:

Step-by-step:

1. Creates **Virtual DOM copy**
2. User clicks button
3. Virtual DOM updates first
4. React compares old vs new Virtual DOM (**Diffing algorithm**)
5. Only `<h1>` updates in real DOM

□ Key Difference with Real Behavior

□ Real DOM

If page has 100 elements:

- Any update → browser may re-render large part or whole DOM
- Slower performance in big apps

□ Virtual DOM

If page has 100 elements:

- Only **changed element updates**
- Rest stays untouched

6. What are Props in React?

Answer:
Props (properties) are used to **pass data from parent to child component**.

Example:

```
function Hello(props) {
  return <h1>Hello {props.name}</h1>;
}
```

Usage:

```
<Hello name="John" />
```

---

7. What isState in React?

Answer:
State is **internal data of a component** that can change over time.

Example:

```
import { useState } from "react";

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <button onClick={() => setCount(count + 1)}>
      Count: {count}
    </button>
  );
}
```

---

8. Difference between Props and State

Props	State
Passed from parent	Managed inside component
Immutable	Mutable
Read-only	Can change

□ 9. What are Hooks in React?

Answer:
Hooks are special functions that let you use **state and lifecycle features in functional components**.

Common hooks:

- `useState`
- `useEffect`

□ 10. What is useEffect?

Answer:
`useEffect` is used to perform **side effects** like:

- API calls
- DOM updates
- timers

Example:

```
useEffect(() => {
  console.log("Component loaded");
}, []);
```

□ 11. Whatis an event in React?

Answer:
Events handle user actions like clicks, typing, etc.

Example:

```
function clickMe() {
  alert("Clicked!");
}

<button onClick={clickMe}>Click</button>
```

□ 12. What is conditional rendering?

Answer:
It means showing different UI based on conditions.

Example:

```
function App() {
  const loggedIn = true;

  return (
    <h1>{loggedIn ? "Welcome Back" : "Please Login"}</h1>
  );
}
```

□ 13. What is React Router?

Answer:
React Router is used for **navigation between pages** in React apps.

Example pages:

- Home
- About
- Contact

□ 14. What is lifting state up?

Answer:
It means moving state to a **common parent component** so multiple child components can share data.

□ 15. Why is React popular?

Answer:

- Fast performance (Virtual DOM)
- Reusable components
- Strong community
- Easy to learn basics
- Used in real-world apps

□ BONUS (Very common interview question)

What is the difference between React and JavaScript?

Answer:

- JavaScript is a programming language
- React is a library built using JavaScript to create UI

Django interview questions and answers for beginners, written in a simple, practical way so you can revise quickly and use them in interviews.

---

1. What is Django?

Answer:
Django is a **high-level Python web framework** used to build **secure and scalable web applications quickly**.

It follows the **MVT (Model-View-Template)** architecture.

Example: Instagram, Pinterest, Mozilla use Django.

---

2. What is MVT in Django?

Answer:
MVT stands for:

- **Model** → Database structure
- **View** → Business logic
- **Template** → Frontend (HTML)

Flow:

```
User → URL → View → Model → Template → Response
```

---

3. Django vs Flask?

Django	Flask
Full-stack framework	Micro framework
Built-in admin, ORM	Minimal features
Faster development	More flexibility
Best for large apps	Best for small apps

---

4. What is a Django project vs app?

Answer:

- **Project** → Whole website (container)
- **App** → Feature inside project

Example:

- Project: ecommerce_site
- Apps:
  - users
  - products
  - payments

---

5. What is a model in Django?

Answer:
A model is a **Python class that represents database tables**.

Example:

```
from django.db import models

class Student(models.Model):
    name = models.CharField(max_length=100)
    age = models.IntegerField()
```

 This creates a database table automatically.

□ 6. What is Django ORM?

Answer:
ORM (Object Relational Mapper) allows you to interact with database using Python instead of SQL.

Example:

```
Student.objects.all()
```

□ No need to write SQL queries.

□ 7. What are Django views?

Answer:
Views handle **logic and request processing**.

Example:

```
from django.http import HttpResponse

def home(request):
    return HttpResponse("Hello Django")
```

□ 8. What are Django templates?

Answer:
Templates are **HTML files with dynamic data support**.

Example:

```
<h1>Hello {{ name }}</h1>
```

□ 9. What are URLs in Django?

Answer:
URLs map requests to views.

Example:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home),
]
```

□ 10. What is Django Admin Panel?

Answer:
Django provides a **built-in admin interface** to manage database easily.

Features:

- Add data
- Edit data
- Delete data

□ No frontend coding needed.

□ 11. What is migrations in Django?

Answer:
Migrations are used to **apply model changes to database**.

Commands:

```
python manage.py makemigrations
python manage.py migrate
```

□ 12. What is static and media files?

Answer:

- **Static files** → CSS, JS, images (design files)
- **Media files** → User uploaded files

---

13. What is Django middleware?

Answer:
Middleware is a layer that processes **requests and responses globally**.

Example use:

- Authentication
- Security
- Session handling

---

14. What is a Django form?

Answer:
Forms are used to **take input from users**.

Example:

```
from django import forms

class ContactForm(forms.Form):
    name = forms.CharField()
```

---

15. What is CSRF token?

Answer:
CSRF (Cross-Site Request Forgery) token protects forms from **unauthorized attacks**.

Example in template:

```
<form method="POST">
  {% csrf_token %}
</form>
```

---

16. What is request and response in Django?

Answer:

- **Request** → Data sent by user
- **Response** → Data sent back by server

---

17. What is QuerySet?

Answer:
QuerySet is a collection of database queries.

Example:

```
Student.objects.filter(age=20)
```

---

18. Why is Django popular?

Answer:

- Fast development
- Secure framework
- Built-in admin panel
- Scalable
- Python-based (easy to learn)

BONUS (Very Important Interview Question)

What is difference between Django and JavaScript backend?

Answer:

- Django → Python-based backend framework
- JavaScript (Node.js) → JS-based backend runtime

Django interview coding questions (beginner → slightly intermediate) with **clear solutions and explanations**. These are the types commonly asked in interviews and coding rounds.

---

1. Create a simple Django view that returns “Hello World”

Question:

Write a Django view that returns "Hello World".

Answer:

```
from django.http import HttpResponse

def home(request):
    return HttpResponse("Hello World")
```

Explanation:

- `HttpResponse` sends text response to browser
- `request` is required in every Django view

---

2. Create a modelfor a Student

Question:

Create a Django model for Student with name, age, and email.

Answer:

```
from django.db import models

class Student(models.Model):
    name = models.CharField(max_length=100)
    age = models.IntegerField()
    email = models.EmailField()

    def __str__(self):
        return self.name
```

□ Explanation:

- `CharField` → text
- `IntegerField` → numbers
- `EmailField` → email validation

□ 3. Fetch all records from a model

□ Question:

How do you get all students from database?

□ Answer:

```
students = Student.objects.all()
```

□ Output (conceptually):

```
<QuerySet [Student1, Student2, Student3]>
```

□ 4. Filter students by age

□ Question:

Get all students whose age is 20.

□ Answer:

```
students = Student.objects.filter(age=20)
```

□ Explanation:

- `filter()` is used for conditions

□ 5. Get a single student record

□ Question:

Get student with id = 1.

□ Answer:

```
student = Student.objects.get(id=1)
```

□ Note:

- Raises error if object not found

□ 6. Create a new student record

□ Question:

How to insert data in Django ORM?

□ Answer:

```
Student.objects.create(
    name="John",
    age=21,
    email="john@gmail.com"
)
```

□ 7. Update a student record

□ Question:

Update student name.

□ Answer:

```
student = Student.objects.get(id=1)
student.name = "Alex"
student.save()
```

□ 8. Delete astudent record

□ Question:

How to delete a record?

Answer:

```
student = Student.objects.get(id=1)
student.delete()
```

---

9. Simple Django URL mapping

Question:

Map a URL to a view.

Answer:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home),
]
```

---

10. Simple form handling in Django

Question:

Create a simple form view.

Answer:

```
from django.http import HttpResponse

def submit_form(request):
    if request.method == "POST":
        name = request.POST.get("name")
        return HttpResponse("Hello " + name)

    return HttpResponse("Send POST request")
```

---

11. Render HTML template

Question:

Render a template with context data.

□ Answer:

```
from django.shortcuts import render

def home(request):
    return render(request, "home.html", {"name": "John"})
```

□ Template:

```
<h1>Hello {{ name }}</h1>
```

□ 12. Login check logic (basic)

□ Question:

Check if user is authenticated.

□ Answer:

```
def dashboard(request):
    if request.user.is_authenticated:
        return HttpResponse("Welcome User")
    else:
        return HttpResponse("Please Login")
```

□ 13. Count total objects in table

□ Question:

Get total number of students.

□ Answer:

```
count = Student.objects.count()
```

□ 14. Order data in Django

□ Question:

Get students sorted by age.

✅ Answer:

```
students = Student.objects.order_by('age')
```

Descending order:

```
students = Student.objects.order_by('-age')
```

✅ 15. Limit results (Top 3 students)

✅ Question:

Get first 3 students.

✅ Answer:

```
students = Student.objects.all()[:3]
```

✅ BONUS (Very common interview coding task)

✅ Create API-like JSON response

✅ Answer:

```
from django.http import JsonResponse

def students_api(request):
    data = {
        "name": "John",
        "age": 21
    }
    return JsonResponse(data)
```

ORM queries (VERY IMPORTANT) CRUD operations Views + URLs

1. Django ORM Queries (VERY IMPORTANT)

Django ORM = database operations using Python instead of SQL.

Assume this model:

```
from django.db import models

class Student(models.Model):
    name = models.CharField(max_length=100)
    age = models.IntegerField()
    marks = models.IntegerField()
```

---

1. Get all records

```
Student.objects.all()
```

Output:

```
<QuerySet [<Student: John>, <Student: Alice>]>
```

---

2. Filter data (WHERE condition)

```
Student.objects.filter(age=20)
```

SQL equivalent:

```
SELECT * FROM student WHERE age = 20;
```

---

3. Get single record

```
Student.objects.get(id=1)
```

Returns error if not found

---

4. Order data

```
Student.objects.order_by('age')    # ascending
Student.objects.order_by('-age')   # descending
```

□ 5. Limit results

```
Student.objects.all()[:3]
```

□ 6. Aggregate (IMPORTANT)

Count students:

```
Student.objects.count()
```

Max marks:

```
from django.db.models import Max

Student.objects.aggregate(Max('marks'))
```

□ 7. Update records

```
Student.objects.filter(id=1).update(name="Alex")
```

□ 8. Delete records

```
Student.objects.filter(id=1).delete()
```

□ 2. CRUD Operations in Django

CRUD = Create, Read, Update, Delete

□ CREATE

```
Student.objects.create(
    name="John",
    age=21,
    marks=85
)
```

□ READ

```
students = Student.objects.all()
```

□ UPDATE

```
student = Student.objects.get(id=1)
student.name = "Mike"
student.save()
```

□ DELETE

```
student = Student.objects.get(id=1)
student.delete()
```

□ CRUD Flow Example (Interview Favorite)

```
# Create
Student.objects.create(name="A", age=20, marks=90)

# Read
Student.objects.all()

# Update
Student.objects.filter(name="A").update(marks=95)

# Delete
Student.objects.filter(name="A").delete()
```

□ 3. Views + URLs in Django

□ Step 1: Create a View

Function-based view:

```
from django.http import HttpResponse

def home(request):
    return HttpResponse("Hello Django")
```

□ Step 2: Map URL to View

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home),
]
```

□ Flow:

```
Browser → URL → View → Response
```

□ Example with parameter in URL

View:

```
def greet(request, name):
    return HttpResponse("Hello " + name)
```

URL:

```
urlpatterns = [
    path('greet/<str:name>/', views.greet),
]
```

Output:

```
/greet/John → Hello John
```

□ Rendering HTML template

View:

```
from django.shortcuts import render

def home(request):
    return render(request, "home.html", {"name": "John"})
```

Template:

```
<h1>Welcome {{ name }}</h1>
```

□ POST request view example

```
def submit(request):
    if request.method == "POST":
        name = request.POST.get("name")
        return HttpResponse("Hello " + name)

    return HttpResponse("Send POST request")
```

□ FINAL INTERVIEW CHEAT SHEET

□ ORM Must Know

- `all()`
- `filter()`
- `get()`
- `create()`
- `update()`
- `delete()`
- `order_by()`
- `aggregate()`

□ CRUD Must Know

- Create → `create()`
- Read → `all()`
- Update → `update()` or `.save()`
- Delete → `delete()`

Django Function-Based View (FBV) interview questions and answers for **Login, Signup, and Logout**—with **simple code + explanations** that are commonly asked in interviews.

□ 1. How do you create a signup (registration) system in Django using FBV?

□ Answer:

In Django, signup is done using the built-in `User` model.

□ Signup View (Function-Based)

```
from django.contrib.auth.models import User
from django.shortcuts import render, redirect
from django.contrib import messages

def signup(request):
    if request.method == "POST":
        username = request.POST.get("username")
        password = request.POST.get("password")

        if User.objects.filter(username=username).exists():
            messages.error(request, "Username already exists")
            return redirect("signup")

        user = User.objects.create_user(username=username, password=password)
        user.save()

        messages.success(request, "Account created successfully")
```

```
        return redirect("login")

    return render(request, "signup.html")
```

□ Interview Explanation:

- Uses Django built-in `User` model
- `create_user()` automatically hashes password
- Checks duplicate username
- Redirects to login after signup

□ 2. How do you implement login in Django FBV?

□ Answer:

Django provides `authenticate()` and `login()` functions.

□ Login View

```
from django.contrib.auth import authenticate, login
from django.shortcuts import render, redirect
from django.contrib import messages

def user_login(request):
    if request.method == "POST":
        username = request.POST.get("username")
        password = request.POST.get("password")

        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect("home")
        else:
            messages.error(request, "Invalid credentials")
            return redirect("login")

    return render(request, "login.html")
```

□ Interview Explanation:

- `authenticate()` → checks credentials
- `login()` → creates session
- If user is valid → redirect to home

□ 3. How do you implement logout in Django FBV?

□ Answer:

Django provides a built-in `logout()` function.

□ Logout View

```
from django.contrib.auth import logout
from django.shortcuts import redirect

def user_logout(request):
    logout(request)
    return redirect("login")
```

□ Interview Explanation:

- Clears session data
- Logs user out completely
- Redirects to login page

---

4. How does Django authentication work internally?

Answer:

Django authentication works using:

- Session framework
- User model
- Middleware

Flow:

```
Login → authenticate() → login() → session created → user stays logged in
```

---

5. What is the difference between authenticate() and login()?

Function	Purpose
authenticate()	Verifies username & password
login()	Creates session for user

Example:

```
user = authenticate(request, username="john", password="1234")
login(request, user)
```

---

6. How do you restrict login-required pages?

Answer:

Use `login_required` decorator.

Example:

```
from django.contrib.auth.decorators import login_required

@login_required
def dashboard(request):
    return render(request, "dashboard.html")
```

Explanation:

- If user not logged in → redirected to login page
- Protects private pages

---

7. How do you check if user is logged in?

```
def home(request):
    if request.user.is_authenticated:
        return render(request, "home.html")
    else:
        return redirect("login")
```

---

8. What is request.user in Django?

Answer:

- `request.user` represents the **current logged-in user**
- If not logged in → AnonymousUser

Example:

```
print(request.user.username)
```

Signup vs Login vs Logout (Interview Summary)

Feature	Signup	Login	Logout
Purpose	Create account	Access account	Exit account
Function	create_user()	authenticate() + login()	logout()
Session	Not created	Created	Destroyed

Full Flow (Most Asked Interview Question)

User Authentication Flow:

```
Signup → Login → Session created → Access pages → Logout → Session destroyed
```