

OutSystems

Note: Please explore more and create your own answers for each question and answers give below:

1. Client Interview Questions

1. Difference between Aggregates and Advanced SQL?
2. What are joins in Out systems?
3. Difference between Delete Rule, Protect, Ignore.
4. What is Architecture Canvas?
5. What is the validation, Rules?
6. What are Integration patterns?
7. What is view state/ Cache in Outsystems?
8. Timers?
9. BPT Process?
10. How to fix the performance issues?
11. What is Discovery tool and Architecture Dashboard?
12. BPT vs Light BPT differences?
13. What is Request time out in Outsystems?
14. What is Screen Lifecycle?
15. CSS related question. (Base theme, Screen level, Block level)
16. Timer and Loop Timer usage, Architecture validation
17. Server Action and Service Action difference.
18. Web block communication.
19. TWA and Reactive Difference.
20. Team handling experience.
21. How the hot fixes are being handled?

Scenario base questions

1. Site properties usage and how to use from one module to another
2. Integration requirement and how to get it done.
3. Components of Outsystems.
4. Local Storage configuration in mobile application, Steps of doing this

1. Database, Entities, Joins, Relations

Questions:

- Different types of joins available in Outsystems for tables.
- What is the difference between aggregates and SQL queries? What are the advantages and limitations for both?

- What are the three different delete rules? Explain with the help of a use case.
- Difference between Aggregation functions and Calculated attributes and how does each affect the output of the aggregate?
- Difference between an Entity and Structure and where would you use each? Explain with a use case.
- What are static entities? What are their properties? Can Static and non-static entities be related?

Databases and Data modelling Questions & Answers

Q1. Explain Inner join, Left join and Full Join.

Ans: Inner joins combine records from two tables whenever there are matching values in a field common to both tables.

Left Join is a type of outer join that retrieves all the records from the first table and matches them to the records in second table.

The Full Join basically returns all records from the left table and also from the right table. For example, let's say, we have two tables, Table A and Table B. When Full Join is applied on these two tables, it returns us all records from both Table A and Table B.

Q2. How can we achieve these joins in aggregates?

Ans: We have to specify these joins in aggregates for 2 entities, only with (Inner join), With or without (Left Join), With (Outer join)

Q3. How can we achieve many to many relationships between two entities?

Ans: By creating a Junction entity, which have references for both tables.

If Candidate is unable to explain the Many to Many relationships, we can ask Q5, otherwise ignore.

Q4. What is Junction entity and reference attribute? Explain.

Ans: It is used to establish many to many relationships between 2 entities. It has reference attributes of both entities.

Q5. Explain the scenario where we need to use Advance SQL in place of Aggregates, what are Outsystems recommendation on this?

Ans: The Aggregate is the preferred option, because it is being optimized by the platform. When an attribute is not needed in the code, it will not be added to the output of the Aggregate. When using the Aggregate inside a server action, the output of the aggregate is assigned to the output of the action. The optimization only compares the attributes of the aggregate with the output structure, and the system won't check which attributes of the output structure we are finally going to use inside your screen.

In SQL Queries we must do these kinds of validations ourselves, which requires expertise. Here we can use SQL operators 'IN', 'EXISTS', 'ANY' or 'ALL'.

Q6. What is Aggregate function and Calculated attributes? Explain.

Ans: Calculated attributes are used to add more information in the record. It is created by clicking add attributes in aggregates.

Aggregate function is used to display more meaningful data in aggregate. Below are some aggregate functions.

- Sum: sums all the values in the column
- Average: calculates the average of the values in the column
- Max: finds the maximum value in the column
- Min: finds the minimum value in the column
- Count: counts how many rows there are in the column

Q7. What is the difference between Calculated Attribute and Aggregated Attribute(From an Output perspective)?

Ans: Aggregated attribute will only be the part of output of an aggregate. Calculated attributed can be part of Output of aggregate with all other DB attribute, if it is applied on aggregated attribute then it will be the part of output of an aggregate.

Q8. What is expand in line and where we use that?

Ans: By using expand inline parameter, we can insert SQL content inside OutSystems SQL query. This parameter isn't a SQL parameter, in the sense that it isn't created in the database. It's calculated during runtime and textually expanded inside the SQL call. We should be mindful of the performance impact inline parameters can have when used with dynamic values.

Q9. What is the use case of data action?

Ans: Data action is used to fetch data from external sources, Server action, REST and SOAP API and SQL query.

Q10. Why we need static entity and what are the use cases for static entity?

Ans: A static entity is an entity that has static data associated to it. This static data is managed at design time and can be used directly in the business logic design of the application, benefiting from strong typing.

It can be used where data will be immutable over time, such as the example of statuses: "In Progress", "Finished", "Cancelled". They don't need to be changed; they are fixed.

Q11. Can we use static entity inside expression?

Ans: Yes, we can use.

Q12. If we need to create Static entity with 1000 rows, how we can achieve this without creating each row one by one?

Ans: We will first create an entity and bootstrap data from excel then we convert that entity to static entity.

Q13. Why are we using indexing in entities? What will happen if it is not used correctly.

Ans: We can define a database index in the data model to enforce uniqueness of table attributes across multiple rows or to make searches quicker using those attributes as filters. In OutSystems, we can model a database index in the entity element. Note that adding several indexes to Entities can negatively impact the performance of the database operations, namely write operations.

Q14. If you need to delete 1 million records from table how you can delete that one?

Ans: To delete large number of records we can use timer to do this action. If we use delete action of an entity for deleting large data sets then it can lock the database and it will create an issue in our application. So here we will use Advance SQL for this purpose, and we can also specify small data set that will be deleted using timer iteration.

Notes and Answers:

Entity Relationships:

One to One One-to-Many Many-to-One (Learn how they are created)

Primary key -----is called Identifier/Id in Outsystems

Foreign Key ----- is called Reference Attribute in Outsystems.

Not mandatory for entities to have an Id (Primary key) but to create relations between multiple entities we need it.

Id can be of type: Text, Long Integer and Other entity identifier.

Also learn about Structure and Static entities and their properties.

Like static entities can't be changed during runtime and acts like enumeration, they have attributes and records.

Default Records: Id, Label, Order, Is Active. And each have identifier: Low, Medium, High.

Aggregates and Data Actions:

CRUD Operations (Actions): Create R(Get) Update Delete (In outsystems it's "get" not "retrieve")

Both data actions and aggregates are used to fetch data, they start executing when the screen is initializing and run asynchronously in parallel (Hence don't have a particular order in which the run).

For a more straightforward and optimized data manipulation use Aggregates.

If your use case requires more advanced data fetching, such as subqueries or IN clauses, use a SQL element (Data Action) instead of an Aggregate.

In aggregate u can add multiple filters and sorting methods.

In sorting if there are multiple attributes u must define the ascending and descending criteria for each.

The aggregate is first sorted based on the first attribute and its parameter, if there are multiple same rows in the first attribute then those are again sorted based on the second attributes and sorting parameter.

e.g., Table sorted on Name Ascending and Age Descending. (The people with same name will be later sorted on the age parameter in descending way)

Calculated attributes:

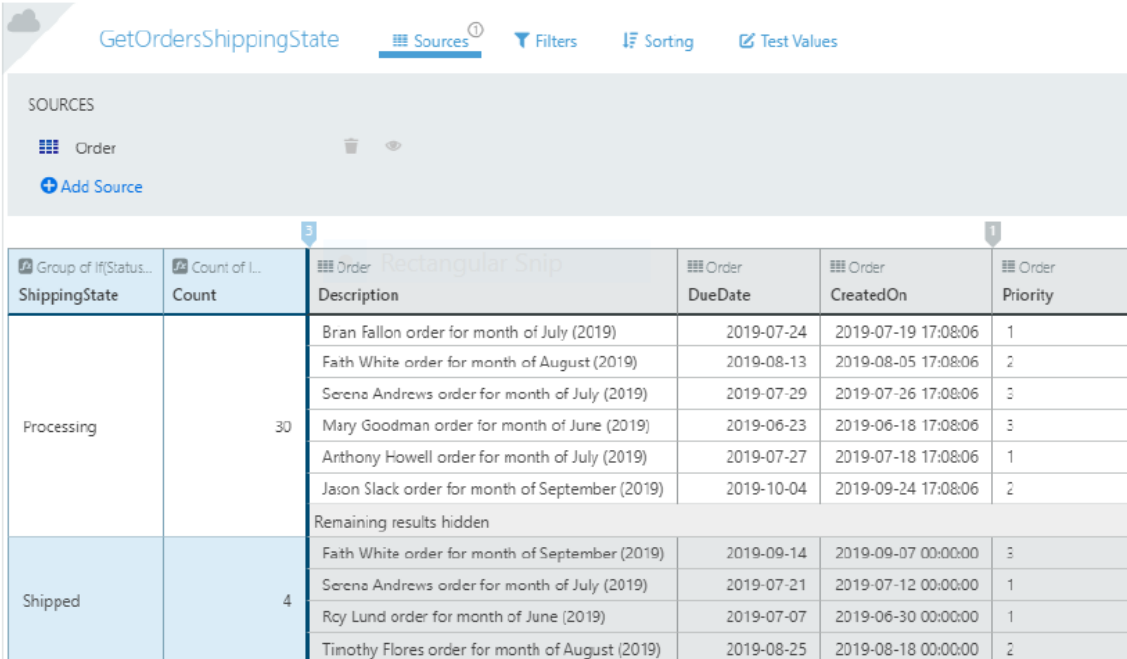
Let's say there is an entity with two attributes "first name" and "last name" and you created a third attribute "full name" = "first name" + "last name" in that case the output of the aggregate would contain all three columns

FirstName Last Name Full Name (First Name + Last Name)

Aggregation Function:

Unlike in calculated attribute if you use aggregation function such as average, sum the output would only contain the new columns that you created and have single value as the output.

e.g., There is an entity with Player name and their runs. And you want to find total runs all the players have scored together and their average. This time you would use sum and average over the run attribute. which will add all the runs of the players and find their average, and that will be the only output of the aggregate the total runs and average runs.



Group of If(Statu...	Count of L...	Description	DueDate	CreatedOn	Priority
Processing	30	Bran Fallon order for month of July (2019)	2019-07-24	2019-07-19 17:08:06	1
		Faith White order for month of August (2019)	2019-08-13	2019-08-05 17:08:06	2
		Serene Andrews order for month of July (2019)	2019-07-29	2019-07-26 17:08:06	3
		Mary Goodman order for month of June (2019)	2019-06-23	2019-06-18 17:08:06	3
		Anthony Howell order for month of July (2019)	2019-07-27	2019-07-18 17:08:06	1
		Jason Slack order for month of September (2019)	2019-10-04	2019-09-24 17:08:06	2
		Remaining results hidden			
Shipped	4	Faith White order for month of September (2019)	2019-09-14	2019-09-07 00:00:00	3
		Serene Andrews order for month of July (2019)	2019-07-21	2019-07-12 00:00:00	1
		Rcy Lund order for month of June (2019)	2019-07-07	2019-06-30 00:00:00	1
		Timothy Flores order for month of August (2019)	2019-08-25	2019-08-18 00:00:00	2

In above image we have used group by and count hence the output would only contain first two columns.

Joins:

Only with (Inner join)

With or without (Left Join)

With (Outer join)

You are creating a one-to-many relation b/w two entities, if the reference attribute is mandatory an inner join will be created and if it's not mandatory a left join will be created.

In advance SQL queries YOU USE BRACKETS AND OTHER SIGNS like:

{Entity} [Attribute] and @Inputparameter (for dynamic query)

SELECT{Person}. [Name] from {Person} where {Person}. [Department] = @Department

Delete Rules: Delete, Ignore, protect (Maintain the integrity of the database).

When you create relationships between the Entities in your module, you must define which kind of referential integrity you want to use when deleting records. The referential integrity specifies what happens to a record of an Entity B that references a record of Entity A, when the Entity A record is deleted.

Protect

Setting the Delete Rule to Protect prevents deleting records of the main Entity while there are associated records in the related Entity.

This behaviour is ensured by a database constraint created on the reference attribute. If you try to delete a record in the main Entity that still has associated records in the related Entity, the Platform Server returns a database exception, and the operation is not executed.

Delete

Setting the Delete Rule to Delete guarantees that when a record of the main Entity is deleted, all the associated records in the related Entities are also deleted. This mechanism is commonly known as Cascade Delete.

This behaviour is ensured by a database constraint created on the reference attribute.

Ignore

Setting the Delete Rule to Ignore allows deleting records of the main Entity keeping the associated records in the related Entity.

The Ignore value does not guarantee referential integrity and, therefore, no database constraint is created. Accordingly, when you change the Delete Rule property from a previous value to Ignore, the corresponding automatic index is deleted (unless you have manually changed any of its properties).

If the foreign key attribute references an external Entity exposed by an Extension, the only possible value for the Delete Rule property is Ignore, as the referential integrity can't be guaranteed.

Refer:

https://success.outsystems.com/documentation/11/developing_an_application/use_data/data_modeling/entity_relationships/delete_rules/

For examples.

Structure is a compound data type that you create for your need.

While developing your application, you may find it interesting to have a variable holding a collection of variables with different data types grouped together and use it in the logic. For example, to assign the values returned by an action where you don't have to create one output for each value.

For advance SQL query you must create the structure of the output that u expect the query to return. Even if it's a non-select query.

.....

2. Reactive programming model, Actions and Web Blocks

Questions:

- Define the architecture of an OutSystems application/ Define OutSystems architecture Canvas. What is the need/ advantages of dividing an application in different levels.
- What is the reactive programming model? What are the different events that occur in the lifecycle of screen in chronological order? Explain them.
- What are web blocks? What are the advantages of web block?
- How does a parent and web block communicate? How does a web block communicate with another web block?
- Different type of actions available in OutSystems. In which order can they be called?
- What is form validation? Describe how to do proper form validation in client and server actions? Who/How to set the "form.valid" property to false?

Screen lifecycle and events questions & Answers:

Q1. What are the different lifecycle events in OutSystems screen and what is its sequence?

Ans:

https://success.outsystems.com/documentation/11/developing_an_application/implement_application_logic/screen_and_block_lifecycle_events/

Q2. Suppose a block have an input parameter named as search (It is taking the input from parent block/ screen's input field) and an aggregate in that block is using that input parameter to show the values when user enter the values in input field of the screen displayed data should change, but it is not changing what you will do?

Ans: In web block Onparameterchange event we need to refresh the required aggregate.

Q3. What is the difference between screen and web block?

Ans: A web block can be used inside a screen or web block. In web block we have reusable codes which can be used in any screen by just using the web block. It can trigger event communicate with the parent block or screen. It have on Parameter changed event which get call when any input parameter gets changed for the block. Web block need screen to initialize it. No role-based authentication will be applied on the web block it can only applied in screen.

Screen cannot be reused like web block but yes it can be used in iFrame. It doesn't have on Parameters changed event.

Q4. One web block is made for forms having multiple input fields, how it will send data to parent block?

Ans: Web block will use event to send the data to the parent block. In parent block we need to handle the data.

Action, Logic and Variable

Q1. What are the different actions in reactive web application?

Ans: Server actions, service actions, data action, screen action and client action.

Q2. What is the difference between server action and service action?

Ans:

	Server Actions	Service Actions
Release cycles	Changes in implementation requires consumers to also be deployed.	Changes in the implementation can be deployed independently from the consumers.
Service communication	Consumer and producer modules run in a single process.	Consumer and producer modules run in different processes.
DB transactions	All transactions in a single process.	Multiple processes require multiple transactions.
Development effort	Simpler logic and faster development.	Requires additional logic to handle transactionality and networking.

Q3. What is the use case of service action?

Ans:

https://success.outsystems.com/documentation/11/developing_an_application/reuse_and_refactor/use_services_to_expose_functionality/

Q4. Can we use client action as functions if it is using list operation?

Ans: No, we can't use client action as a function if it is doing the list operation and calling any server or service action.

Q5. What is the difference between screen action and client action?

Ans: Screen actions can access screen's local variable, have input parameter only and it can be used within the scope of the screen. Whereas client action can't access local variable of screen, have both input and output parameter and can be used with module. Client action can be used as function, and we can put our reusable code under client action.

Q6. When we use service action, will it create a REST log?

Ans: Yes, it will create a REST log. We can check logs of service action service centre.

Notes and Answers:

Actions

Three types of actions: Screen Action (Have Input parameter and local variable), Client Action (Have Input parameter and local variable and Output Parameters) and Server Action (Have Input parameter and local variable and Output parameter)

Screen Action can call other Screen Actions, Client Action and Server Action.

Client Action can call other Client Actions and Server Action.

Server Action can call other Server Actions.

But the reverse call can't happen.

e.g., server action can't call a screen action or client action, or client action can't call a screen action.

Reactive programming Model:

Total 5 event calls.

4 are for screens (On Initialize>On Ready>On Render>On Destroy)

1 is for Aggregates/Data action (On After Fetch).

On render is called multiple times on screen whenever underlying data on screen or block changes.

On destroy event is called on the current screen to destroy the data of the previous screen. This happens only after the data on the current screen is fully rendered.

Blocks and Events

Blocks and screens have different scopes hence they don't know of the changes occurring inside each other. To tell each other of the changes occurring inside their scope they pass messages to each other using input parameter that we called events. It happens to maintain the proper functionality.

A block triggers an event (Using trigger event) and parent handle it (Event handler).

Scope of trigger event is within block and event handler is within parent screen.

The parent must mandatorily handle the event using event handler if the trigger event of the block is mandatory.

Two types of events:

Trigger event: Tells parent of change inside the block scope

On Parameter Change event: Tells the Block of change inside the parent scope.

Other Important points:

Site Property: Stored on server, not frequently changed, Modified in service centre.

Application property: Stored on browser, can be changed frequently.

Both have them are of basic datatypes like long integer or text which is same as that of identifier, hence we can say that they can be of type identifier.

.....

3. Best Practices, Timers and Processes, Debugging, Performance, CSS, Others.

Questions:

Process and Timers

Q1. Why do we use timers in our application? What are the purposes of using timer?

Timers are asynchronous logic in Outsystems. It is useful in running the batches, loading bulk data in tables, doing automated checks for data, deleting data, sending emails at a predetermined time. It

can also help in fetching the environment report through email. It could be used for configuring application after deployment.

Q2. What will happen when timer gets an exception?

Once timer gets an exception the dB operations done by the timer will be roll back. It will retry 2 more times then it got aborted. We can change timer's execution attempts from Service Center.

Q3. Suppose a timer is scheduled at an interval of 5 mins and it is taking more than 5 mins due to large number of data coming through an API. When will next scheduled timer run?

It will wait for the current timer to finish the execution then it will run.

Q4. What is BPT and why we use this?

OutSystems allows you to design and manage your business processes and integrate them into your applications. A business process is simply called a Process in OutSystems and is understood as the way that a particular task is carried out in your organization, such as handling invoices, processing orders, or handling complaints. Processes are also known as BPT (Business Process Technology).

It can be used inside action; it will run in background.

Q5. What is the difference between BPT and Light BPT?

A Process in OutSystems can be a Light BPT process or a BPT Process.

Light BPT will (can) activate a process immediately after Data manipulation on an Entity. The big difference with triggers is that (normally) triggers are executed within the same transaction, where a Light BPT Process will start separately.

BPT is more for digitalizing workflows

<https://william-antunes.medium.com/timers-vs-bpt-vs-light-bpt-a-comparison-of-which-to-use-when-in-outsystems-dbbac72738cb>

Q6. Can we run BPT on click?

Yes, we can run BPT on click through the action. It can be run inside the timer but it needs to be handled properly for Timeout, idle process etc.

Q7. Suppose a scenario a user is uploading a large excel document from screen, data in excel will get validated and then inserted in tables but using server/service action user is getting the connection time out error. How can you solve this issue?

Server / service action with validation for large amount of data will get timed out. We can use BPT in this scenario which will run in background, it will do validation for the data and insert the data.

For very large amount of data BPT can get timed out so we might need to break the data in smaller chunks inside BPT, so that it will not get timed out.

Q8. How will you debug a Timer?

We can add break points inside the action and start debugging from the same module and run timer from service Center.

Q9. What happen if Timer's schedule is set on When published?

It is not a best practice to keep scheduled of timer set to when published. Whenever module get published, it will get triggered automatically. It could get triggered after deployment in higher environment. If the timer can have cleared the data from table and inserting bulk record, then it will create serious impact on higher environment.

Q10. How to handle heavy timer?

If any timer is taking more time than default timeout (20mins) then we need to set timeout for that timer.

We can make data set of 1000-2000 and run them and commit them.

<https://www.outsystems.com/training/lesson/1749/odc-2018-heavy-timers>

Q11. Can we schedule a timer in 1 min interval?

Yes, we can schedule timer to 1 min interval, but we need to provide the schedule by ourselves and it can max 300 – 310 interval or approx. 5hrs. In this scenario to cover 24hrs we need to schedule 5 timers.

<https://william-antunes.medium.com/timers-vs-bpt-vs-light-bpt-a-comparison-of-which-to-use-when-in-outsystems-dbbac72738cb>

CSS

Q1. What is the order of precedence of CSS style?

Precedence/loading order:

- Blocks (lowest)
- Base theme(s)
- Theme and Theme Editor
- Screen
- Styles Editor
- Inline styles

Q2. Suppose a scenario where a class in theme named **XYZ** have background color **RED** and same class have background color as **PINK** in screen CSS and in web block CSS same class have background color have **YELLOW** what would be the background color of container in Web block which is using **XYZ** class.

Ans: It will take web block CSS and the background color would be Yellow.

Q3. What are selectors in CSS?

Ans: A selector is used in CSS to find and select HTML elements which needs to be styled. We can divide CSS selectors in 5 categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

Q4. What are Pseudo-classes?

Ans: A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

Q5. How to highlight an element in different color on mouse over?

Ans: Using pseudo class we can highlight element on Mouse over.

Example: `div: hover {background-color: Red;}`

Q6. Change first child's color of an element.

Ans: The: first-**child** pseudo-class matches a specified element that is the first child of another element.

```
p i: first-child {  
    color: blue ;  
}
```

Q7. What are CSS combinators?

Ans: A combinator is something that explains the relationship between the selectors. A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

Q8. What are Pseudo-Elements?

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

Q9. Change first line's color of an element?

Ans: The **::first-line** pseudo-element is used to add a special style to the first line of a text.

```
Example: p::first-line {  
    color: #ff0000;  
    font-variant: small-caps;  
}
```

Q10. Select nth child element.

```
Ans: div:nth-child(2) {  
    background: red;  
}
```

Q11. What is Universal selector?

Ans: The universal selector (*) selects all HTML elements on the page.

```
Example: * {  
    text-align: center;  
    color: blue;  
}
```

Q12: What is the difference between Flex and Grid?

Ans: <https://www.srijan.net/resources/blog/css-grid-vs-flexbox#:~:text=Grid%20is%20made%20for%20two,here%2C%20gives%20you%20more%20flexibility.>

Q13. What is position static, relative, and absolute?

Ans: **position: static**

By default, position an element based on its current position in the flow. The top, right, bottom, left and z-index properties do not apply. — source MDN

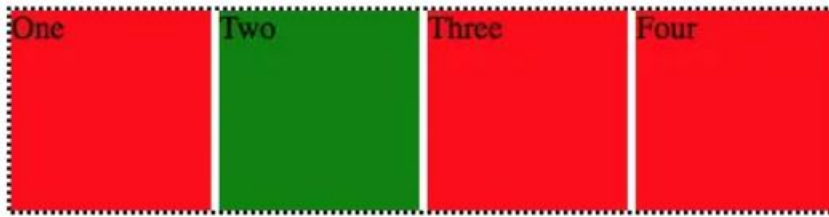
position: relative

Position an element based on its current position without changing layout.

position: absolute

Position an element based on its closest positioned ancestor position.

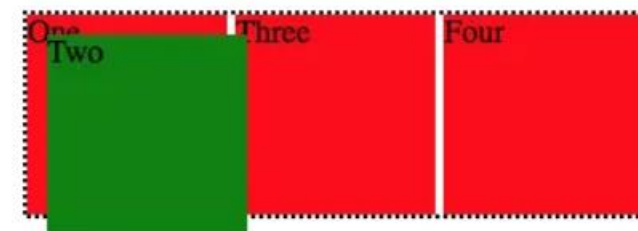
position: static



position: relative



position: absolute



JavaScript

Q1. In which scenario we are using JS in Outsystems?

Q2. How does OutSystems **integrate** JavaScript into its development platform?

OutSystems integrates JavaScript into its development platform by allowing developers to write custom JavaScript code (In **JavaScript Node**) within their applications. This code can be used to enhance the functionality, interactivity, and customization of the application's front-end.

Q3. What is the **purpose/** benefits of using JavaScript in OutSystems applications?

JavaScript is used in OutSystems applications to add client-side interactivity, perform **validations**, **manipulate the DOM** and extend the capabilities of OutSystems components.

Q4. How can you include custom JavaScript code in an OutSystems application?

Custom JavaScript code can be included in an OutSystems application by creating a JavaScript file or by adding JavaScript directly in the "Scripts" section of a web block or screen. The JavaScript code can then be referenced and executed within the application's logic.

Q5. Can you explain the process of invoking JavaScript functions from OutSystems actions or expressions? (In Traditional)

In OutSystems, JavaScript functions can be invoked from actions or expressions using the "RunJavaScript" (HTTPRequestHandlerAPI) action or the "JavaScript" expression. This allows developers to execute custom JavaScript code and retrieve the result back into the OutSystems logic.

Q6. How can you access and manipulate HTML elements using JavaScript in OutSystems?

JavaScript in OutSystems can access and manipulate HTML elements by using standard DOM manipulation techniques. Developers can use JavaScript methods like `getElementById`, `querySelector`, or event listeners to interact with HTML elements and modify their properties, content, or styles.

Q7. Are there any **limitations** or considerations when using JavaScript in OutSystems applications?

While using JavaScript in OutSystems applications, it's important to consider **security risks**, **cross-browser compatibility**, performance implications, and code maintainability. Developers should follow best practices, validate inputs, sanitize data, and ensure proper error handling.

Q8. Can you provide an example of using JavaScript to perform client-side validation in an OutSystems form?

Yes, an example of using JavaScript for client-side validation in an OutSystems form could be validating a phone number input field to ensure it contains a valid format before submitting the form. JavaScript can be used to check the input value against a regular expression or perform other validation logic.

Q9. How can JavaScript be utilized to **enhance the user interface (UI)** and user experience (UX) in OutSystems applications?

JavaScript can be used in OutSystems applications to create **dynamic UI components**, add animations and transitions, implement drag-and-drop functionality, provide real-time updates, handle user interactions, and create responsive interfaces that enhance the overall user experience.

Q10. Can you explain the concept of JavaScript callbacks and how they can be used in OutSystems integrations?

JavaScript callbacks are functions that are passed as arguments to other functions and executed later when a certain event or operation is completed. In OutSystems integrations, JavaScript callbacks can be used to handle the asynchronous response from APIs or external systems, allowing you to process the returned data and trigger subsequent actions.

Q11. How can you leverage JavaScript libraries or frameworks like jQuery or React

You can leverage JavaScript libraries or frameworks like jQuery or React within an OutSystems application by importing and using them as external libraries. OutSystems provides integration capabilities to include these libraries and interact with their APIs, enabling you to leverage the features and components they offer.

Q12. Are there any specific considerations when using JavaScript in mobile applications developed with OutSystems?

When using JavaScript in mobile applications developed with OutSystems, it's important to consider device compatibility, performance optimization, and the specific capabilities and limitations of the target mobile platforms (e.g., iOS, Android). OutSystems provides mobile-specific features and APIs to facilitate mobile development using JavaScript.

Q13. Can you explain how to handle cross-browser **compatibility issues** when using JavaScript in OutSystems?

Cross-browser compatibility issues when using JavaScript in OutSystems can be handled by following best practices, **using feature detection** instead of browser detection, using standardized JavaScript APIs, and testing the application on different browsers and devices. Additionally, using JavaScript libraries or frameworks that handle cross-browser compatibility can simplify this process.

[Implementing feature detection - Learn web development | MDN \(mozilla.org\)](#)

Others

Q1. What is AI Mentor Studio? What is Discovery tool? Give a brief introduction about them and their use? What is the difference and similarities between them?

Q2. Explain how you would debug and do different kind testing in an application on Outsystems.

Best Practices

Q1. Suppose a scenario where screen (In reactive web app) is getting frequent connection timeout for a particular data action. When you check that action you see that 20 thousand records of tables

are getting iterated for some specific purpose to filter out some data which can't be applied using Aggregate or SQL. Someone tried to use list filter in data action, but they again got the connection time out. How can you resolve the issue?

Q2. A screen is rendering very slow, how you will investigate the issue?

Ans: First we will check any aggregate refresh or server action is called in on initialize or on ready events of the screen we will check the logs for the screen and module and check whether we are getting a slow SQL warning or not. We can also check if there is any data action what time it is taking to get the data. If data action using some APIs to fetch the data, we need to check the integration log for that API, whether API is slow, or we are getting a timeout for that API.

Q3. While creating Static entity, if we set ID attribute as Auto number "Yes" then, is there any chance that it might create issue in Upper environment?

Ans: If we set static entity's ID attribute to Auto number Yes then it's ID will be generated automatically after deployment and if we did any delete any static entity's attribute in development then ID attributes of Static entity is different from development environment and in any place of our code we explicitly define the ID value for inserting data for any entity then that data might be creating issue in higher environment and it will be hard to find out that one.

Q4. Can we use inline CSS for the widgets? Is it a good practice?

Ans: Yes, we use inline CSS but is quite hard to maintain.

Q5. What is sanitization API? Where are we using it?

Ans: API that provides methods to help you avoid code injection in HTML, JavaScript, and SQL snippets that need to include untrusted content, for example, content gathered from end users

Q6. Why should not we change site properties value programmatically and frequently?

Ans: When we are changing a site properties value it will restart the IIS to reflect the value of site properties. Due to this it will affect the application's performance.

DevOps

Q1. How to do deployment in Outsystems?

Ans:

https://success.outsystems.com/documentation/11/managing_the_applications_lifecycle/deploy_applications/deploy_an_application/

Q2. You need to create a team of developers and need to provide access to them in different application. How will you do this?

Ans: We can create Team in Lifetime under user management tab.

https://success.outsystems.com/documentation/11/managing_the_applications_lifecycle/manage_users/create_an_it_team/

Q3. Before deployment what best practices you need to follow?

Ans: Before deployment we need to republish the application. First core then business and after that UI. We also need to inform dev team regarding deployment so that they will not publish any changes during deployment from dev to QA environment.

Q4. You are doing deployment dev to QA but at the same time some other team member is publishing the changes for the same application which you are deploying. Will it impact your deployment?

Ans: While deployment it would throw error if any module is publishing during that time. So in best practice no changes to be published till the deployment gets over.

Integrations

Q1. What is onBeforeRequest and onRequest in REST API?

Ans: onRequest: Outsystems allows us to run logic before executing each request of an exposed REST API. We need to make sure to set the "CustomizedRequest" output to the request after preprocessing the request.

onBeforeRequest: Use to modify the information of the original request, such as the URL, the request text or the headers. An example of this kind of customization is when we are consuming a REST API that uses token-based authentication and we need to add the token to the URL before sending the request. In this situation we should use the OnBeforeRequest.

Q2. How to do tokenized authentication for consumed REST API?

Ans:

https://success.outsystems.com/documentation/10/extensibility_and_integration/rest/consume_rest_apis/simple_customizations/

Q3. What is on Authentication method in Outsystems?

Ans: onAuthentication method is used to authenticate the consumer in Outsystems.

Q4. How can we check the performance of exposed and consumed APIs.

Ans: We can check the integration logs for the APIs, how much time it is taking to process the request.

Q5. If any APIs are getting exception, where you will check that one?

Ans: In service center we can check the integration logs for that API.

Notes and Answers:

Extras

Data archiving is the process of identifying and moving data from the primary storage into a secondary storage, for long term storage. The major benefit of having a data archiving strategy is runtime performance and cost savings.

Data purging is a mechanism that permanently deletes inactive or obsolete records from the database. The major benefit of having a data archiving strategy is runtime performance and cost savings.

Mobile Data Layer Best Practices

While designing the database for a mobile app you can improve the User Experience by following a Local Storage First strategy. This approach allows the user to have smaller loading times while navigating through screens and minimizing server-side calls by requesting the updated data only.

Use local storage whenever possible.

- It is best for performance
- Reduces the number of server calls
- It is offline-ready

Following the good practices of storing data in the device's local storage, provide you with the following benefits:

- Smooth navigation between screens
- Increased performance
- Server-side call avoidance
- Offline-ready

Development Best Practices



OutSystems provides out-of-the-box capabilities to integrate with external databases such as SQL Server, Oracle, MySQL, PostgreSQL and iDB2, as well as a myriad of enterprise systems of record (SoR), including SAP, Salesforce, Microsoft Dynamics 365, Microsoft Dataverse, and SharePoint Online

The available integrations in OutSystems include:

- Enterprise SoR (System of Record) for managing sales and collaboration, including SAP, Salesforce, Microsoft Dynamics, Microsoft Dataverse, and Sharepoint Online
- Web services to both consume and expose REST and SOAP
- External databases
- End-user authentication
- DevOps and CI/CD processes and tools for building, deploying, automating, testing, monitoring, and logging
- AI and chatbots

There are many other integrations that don't exactly fit into any particular category, including

- Augmented reality frameworks
- Messaging formats and protocols
- IoT gateways

OutSystems must know best practices:

https://success.outsystems.com/documentation/best_practices/development/outsystems_platform_best_practices/

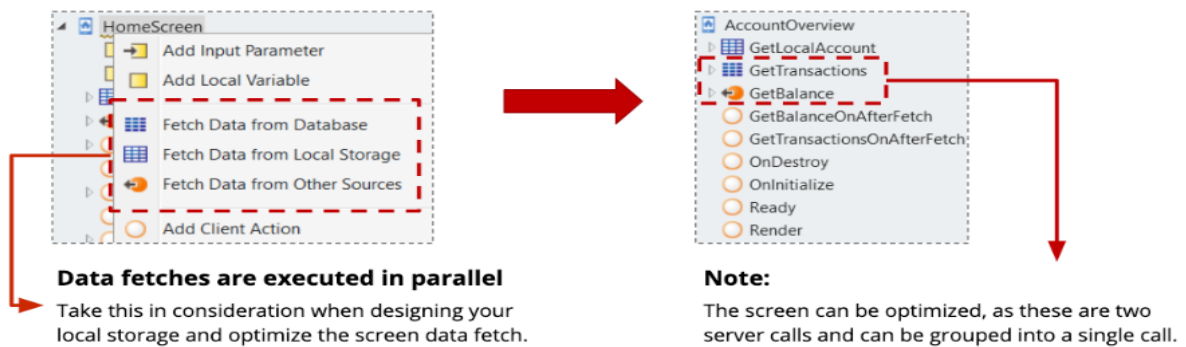
Mobile app best practices:

- Focus on mobile use cases to guide your development
- Keep heavy logic on the server side
- Create Local Storage
- Choose OutSystems low code over JavaScript
- Test your app in real world scenarios

For mobile app database use the following best practices:

- Use Fetch Data actions to take advantage of parallel and asynchronous execution
- Use the On After Fetch event for post-processing
- Group server calls in a single server call to decrease request latency and the number of server calls.

The following diagram shows an example of a timeline for events and actions:



OutSystems monitoring tools

The built-in monitoring and analysis included in every OutSystems installation:

- Service Center provides monitoring logs and analytics reports about performance issues of the platform and applications.
- Lifetime gives an aggregated view into Web application performance and end-user experience.
- Performance Monitoring API provides access to detailed performance data of request events and deep analysis of performance issues.
- App Feedback can be enabled in Lifetime so key users, during the acceptance stage, deployment, and after release, can send suggestions or report problems.
- Asynchronous Logging API can be used in your OutSystems applications for more advanced monitoring by referencing the module which provides actions.
- Business Activity Monitoring enables developers to start monitoring and optimizing processes immediately after deployment without the need for any development effort.

You can examine different types of logs either via Service Center or the Lifetime console. These logs provide information about many aspects of your applications, including:

- Errors logs
- General logs
- Service Actions logs
- Integrations logs

Other logs are also generated, but typically these are the most used.

Performance Best Practices - Data model

- Create indexes for the most commonly used attributes to improve the overall query speed. However, be aware that indexes cause inserts and updates to take longer.
- Avoid text attributes with more than 2000 characters. Store binary and large text attributes in their respective separate entities.

- When handling large Excel files consider recording the data in an entity and using a timer to process the data in background.
 - Whenever possible set Reference Attributes Delete Rule to Ignore instead of Delete. This is an advanced scenario that must be used with care, since it may cause data inconsistency which can result in serious issues in your application.
 - You should consider using a separate entity to archive old data.
-

Performance Best Practices – Infrastructure

- Setup database maintenance plans: Database queries over indexed attributes are still slow and there's evidence that the indexes are not being used by the database engine because they are too fragmented and full table scans are being performed instead. The indexes must be defragmented.
 - Configure web server memory settings: Apply the recommended tuning setting to decrease the number of worker process recycles, unless memory limits have been reached.
 - Backup database transaction logs often: Setup regular transaction log backups for databases with high volume of data changes
 - Exclude some folders from antivirus scanning
 - If available and for high load environments, use 64-bit architecture servers and operating systems, instead of 32-bit architectures.
 - For heavy background processing applications that rely on timers, try to isolate those timers in eSpaces. Configure dedicated servers to run those timers.
-

Performance Best Practices – Logic

- Use Service Center reports to find performance problems and guide the optimization process.
 - Avoid long-running timers and batch jobs: Background processing shouldn't be done all in a row. Use control tables and process data in chunks, rescheduling itself at the end of a run.
 - Simplify the screen preparation actions to improve the screen loading speed.
 - Avoid using site properties as logic variables that frequently change
 - Place as little information in session as possible and especially avoid large records and record lists.
 - Never create an action to execute an Aggregate. When the Aggregate is isolated in an action by itself the platform can't optimize the number of fields to be retrieved.
 - Avoid using queries inside 'If' branches in preparation: Avoid using the conditional queries in preparation to prevent the extra query data from going to the viewstate.
-

Performance Best Practices – Queries

- Don't do joins over linked servers. Cross server joins are very inefficient. Avoid them at all costs.
 - Minimize the number of fields fetched from the database.
 - Keep the Max Records property of Aggregates consistent with the amount of data that you're displaying.
 - When performing massive operations use SQL queries instead of using a Foreach loop with Aggregates (keep in mind that Aggregates are always more built to change, though). Also updates and massive deletes are faster in SQL queries than using Entity Actions. Write queries that update as many rows as possible in a single statement rather than using multiple queries to update the same rows.
 - Never create an action to execute an Aggregate. When the query is isolated in an action by itself, the Platform can't optimize the number of fields to be retrieved.
 - Iterating more than once over a query result isn't a good practice. By doing so, the query results are copied into the memory. The same applies when using direct indexers (like `query[2].value` expressions).
 - Minimize the number of executed queries. Often, it's possible to fetch all necessary data in a single query execution instead of multiple ones.
-

Performance Top 10 Rules

Good applications have at least one thing in common – they are really fast. Beyond built-in OutSystems Platform optimizations, keep these best practices in mind.

- **Index your entities**
Creating indexes for the most used entity attributes will significantly improve the performance of select queries at the cost of slight overhead in insert and update operations.
- **Setup database maintenance plans**
Maintenance plans with reorganization of indexes and statistics update allow the database engine to optimize queries and take advantage of the existing entity indexes. Don't forget to involve the resident DBA.
- **Use Service Center reports**
Use Service Center reports as the starting point for all application tuning efforts. These reports provide a full view of the slowest operations; are divided by application screens, web services, timers, queries and extensions.
- **Don't join over linked server**
Cross server joins are very inefficient because the table in the linked server is completely loaded to the local DB server and then the join is performed. Cross server joins may be

acceptable if the tables are small and unavoidable but as a rule cross server joins should be avoided as much as possible.

- **Configure web server memory settings**
Apply the webserver tuning settings to optimize memory usage and application server performance according to the checklist best practices. Using .NET stack, process recycling affects performance and should be minimized, while in the Java stack, the virtual machine memory settings should be fine-tuned to improve availability.
 - **Isolate large text and binary data**
Avoid the use of 2000+ characters in a text field; data fields greater than 2000 bytes are converted into a Text data type which in turn are much slower to process and decrease performance overall. As a rule, isolate binary and/or large text fields in separate entities and only retrieve them when they are strictly necessary.
 - **Avoid long running timers and batch jobs**
Timers and batch jobs should be built having partial processing capability and auto-timeout mechanisms. These features will prevent run-away jobs, endless data reprocessing and data inconsistencies.
 - **Simplify screen preparations**
Screen preparation processing affects the user experience and therefore should be simple and fast to execute. This is especially important for web screens having a high access volume like home pages
 - **Avoid using preparation data in screen actions**
Don't use preparation data in screen actions because it increases the network traffic between the server and the browser. This in turn increases network usage and degrade server request processing performance.
 - **Minimize the number of fields fetched from the database**
Use specific structures with the necessary attributes, instead of returning the whole database entity as I/O parameters of actions. This allows the platform to optimize the queries, improving database performance.
-

Performance Best Practices - User Interface

- **Avoid using preparation data in screen actions;** most of the time it's better just to fetch the data again.

- AJAX is a great tool to improve usability and user interaction but needs to be used with care to avoid performance issues.
 - Split large screens to reduce Viewstate size and the number of queries in the preparation.
 - Use cached WebScreens/WebBlocks when repeated use yields the same result therefore saving the cost of processing the screen/block over and over again. Particularly useful in AJAX queries that retrieve lookup data.
 - Reduce the number of screen parameters and their sizes to a minimum to improve performance and memory.
 - Avoid the use of JavaScript in screen expression; place JavaScript code in files included through extensions or eSpace resources.
 - Use CSS sprites that replace several images to reduce the number of server requests and improve screen load times.
 - Move the JavaScript that is not needed at load time to the bottom of the page to make the page load faster.
-

Reactive web security best practices:

- Don't expose sensitive data on the client-side: Client-side code (UI, client actions, screen and block variables, and input parameters) converts into web resources (HTML, JavaScript and CSS). This means that it can be manipulated. That's why we need to make sure that all security sensitive code and data is on the server side.
 - When you need data, don't fetch all fields. Only fetch the data that is needed for the screen.
 - When you fetch data from the server (queries or API calls), don't use input parameters that have impact on the data that is returned. An attacker can change these values and fetch some other data.
 - Protect screens and aggregates with roles
-

Most Important tools:

{Architecture dashboard is now AI mentor studio}

AI mentor is used to track your technical debt its reviewing and doing coding analysis to show any violation inside your code for different aspects performance, security, architecture, and user experience of applications. It shows you and suggest implementation to follow best practice while you are building your application.

What AI Mentor Studio analyses

AI Mentor Studio analyses the code produced by developers and provides insights regarding code quality that may impact team agility.

Code Analysis

AI Mentor Studio runs a set of predefined rules throughout the produced low-code, with the goal of uncovering code patterns in the following categories:

- Performance
- Architecture
- Maintainability
- Security

Discovery tool is used for reviewing solution architecture and showing any violation related to solution architecture which helps you to find violation very easily and fix it.