Valid JS-Dev-101 Exam Materials | JS-Dev-101 Valid Braindumps Ebook

BONUS!!! Download part of NewPassLeader JS-Dev-101 dumps for free: https://drive.google.com/open?id=1biCkEryv_XQtMmvNyFlW28ZxOFoSFF1n

I know that all your considerations are in order to finally pass the JS-Dev-101 exam. Our JS-Dev-101 study materials have helped many people pass the exam and is about to help you. The 99% pass rate of our JS-Dev-101 training prep is enough to make you feel at ease. Of course, we do everything we could do to ensure that you could think through it and that you also needed to pay a bit of your effort. And with our JS-Dev-101 Exam Questions, you will pass the exam for sure.

Salesforce JS-Dev-101 Exam Overview:

Certification Vendor:	Salesforce
Exam Name:	Salesforce Certified JavaScript Developer - Multiple Choice
Exam Number:	JS-Dev-101
Exam Price:	USD 200
Related Certifications:	Salesforce Certified Platform Developer I Salesforce Certified Developer
Certificate Validity Period:	Does not expire; requires maintenance updates
Exam Format:	Multiple choice, Multiple select
Passing Score:	65%
Real Exam Qty:	60 scored + up to 5 unscored
Available Languages:	English, Japanese
Exam Duration:	105 minutes
Recommended Training:	Trailhead JavaScript Developer I Certification Prep
Exam Registration:	Salesforce Certification Registration
Sample Questions:	Salesforce JS-Dev-101 Sample Questions
Exam Way:	Online proctored or onsite at authorized testing centers
Pre Condition:	No formal prerequisites; recommended: strong JavaScript fundamentals, ES6+ experience, web development familiarity
Official Syllabus URL:	https://trailheadacademy.salesforce.com/certificate/exam-javascript-dev---JS-Dev-101

>> Valid JS-Dev-101 Exam Materials <<

New Valid JS-Dev-101 Exam Materials | Pass-Sure Salesforce JS-Dev-101 Valid Braindumps Ebook: Salesforce Certified JavaScript Developer - Multiple Choice

Due to the shortage of useful practice materials or being scanty for them, many candidates may choose the bad quality exam

materials, but more and more candidates can choose our JS-Dev-101 study materials. Actually, some practice materials are shooting the breeze about their effectiveness, but our JS-Dev-101 training quiz are real high quality practice materials with passing rate up to 98 to 100 percent. And you will be amazed to find that our JS-Dev-101 exam questions are exactly the same ones in the real exam.

Salesforce JS-Dev-101 Exam Syllabus Topics:

Topic	Details
Topic 1	• Testing: Covers evaluating unit test effectiveness against a block of code and modifying tests to improve their coverage and reliability.
Topic 2	• Variables, Types, and Collections: Covers declaring and initializing variables, working with strings, numbers, dates, arrays, and JSON, along with understanding type coercion and truthy • falsy evaluations.
Topic 3	• Debugging and Error Handling: Covers proper error handling techniques and the use of the console and breakpoints to debug code.
Topic 4	• Objects, Functions, and Classes: Covers function, object, and class implementations to meet business requirements, along with the use of modules, decorators, variable scope, and execution flow.
Topic 5	• Asynchronous Programming: Covers asynchronous programming concepts and understanding how the event loop controls execution flow and determines outcomes.
Topic 6	• Server Side JavaScript: Covers Node.js implementations, CLI commands, core modules, and package management solutions for given scenarios.

Salesforce Certified JavaScript Developer - Multiple Choice Sample Questions (Q77-Q82):

NEW QUESTION # 77
A developer writes the code below to return a message to a user attempting to register a new username. If the username is available, a variable named nag is declared and assigned a value on line 03.

```
01  function getAvailabilityMessage(item) {
02    if (getAvailability(item)) {
03      var msg = "Username available";
04    }
05    return msg;
06  }
```

What is the value of msg when getAvailableabilityMessage ("newUserName") is executed and get Availability ("newUserName") returns true?

- A. "User-name available"
- B. "newUserName"
- C. "msg is not defined"
- D. undefined

Answer: A

NEW QUESTION # 78
Refer to the code below:
01 new Promise((resolve, reject) => {
02 const fraction = Math.random();
03 if (fraction > 0.5) reject('fraction > 0.5, ' + fraction);
04 resolve(fraction);

```
05 })
06 .then(() => console.log('resolved'))
07 .catch((error) => console.error(error))
08 .finally(() => console.log('when am I called?'));
```
When does Promise.finally on line 08 get called?

- A. When resolved
- B. When rejected
- C. When resolved or rejected
- D. When resolved and settled

Answer: C

Explanation:
Behavior of Promise.prototype.finally:
.finally(handler) registers a callback that runs when the promise is settled, meaning:
Either fulfilled (resolved), or
rejected.
Important points:
The finally callback does not receive the promise's value or error (unlike then and catch).
It is executed after the promise is settled, but before the resolution value or rejection reason is passed further down the chain.
It runs in both success and failure paths.
In the given code:
The promise may either:
Call reject('fraction > 0.5, ' + fraction) if fraction > 0.5, or
Call resolve(fraction) otherwise.
In both cases:
If it resolves, .then(() => console.log('resolved')) runs, and then .finally(...) is executed.
If it rejects, .catch((error) => console.error(error)) runs, and then .finally(...) is executed.
So .finally runs:
Not just "when rejected".
Not just "when resolved".
But whenever the promise is resolved or rejected.
Therefore, the correct choice is:
D . When resolved or rejected.

NEW QUESTION # 79
Refer to the code below (assuming Promise.race is intended):
```
let cat3 = new Promise(resolve =>
setTimeout(resolve, 3000, "Cat 3 completes")
);
Promise.race([cat1, cat2, cat3])
.then(value => {
let result = `${value} the race.`;
})
.catch(err => {
console.log("Race is cancelled:", err);
});
```
(Assuming cat1 and cat2 are similar to earlier examples: cat2 resolves fastest.) What is the value of result when Promise.race executes?

- A. Car 3 completed the race.
- B. Race is cancelled.
- C. Car 2 completed the race.
- D. Car 1 crashed on the race.

Answer: C

Explanation:
From earlier pattern (cars/cats race):

One promise resolves the fastest with "Car 2 completed" (or analogous "Cat 2 completes").
Promise.race resolves with the first settled promise's value.
In .then, value is "Car 2 completed" and:
let result = `${value} the race.`;
// "Car 2 completed the race."
Thus result is "Car 2 completed the race." → option A.

NEW QUESTION # 80
Original constructor function:
01 function Vehicle(name, price) {
02 this.name = name;
03 this.price = price;
04 }
05 Vehicle.prototype.priceInfo = function () {
06 return `Cost of the $(this.name) is $(this.price)$`;
07 }
08 var ford = new Vehicle('Ford Fiesta', '20,000');
Which class definition is correct?

- A. class Vehicle {
  constructor(name, price) {
  this.name = name;
  this.price = price;
  }
  priceInfo() {
  return `Cost of the ${this.name} is ${this.price}$`;
  }
  }
- B. class Vehicle {
  constructor(name, price) {
  this.name = name;
  this.price = price;
  }
  priceInfo() {
  return 'Cost of the ${this.name} is ${this.price}$';
  }
  }
- C. class Vehicle {
  constructor() {
  this.name = name;
  this.price = price;
  }
  priceInfo() {
  return `Cost of the ${this.name} is ${this.price}$`;
  }
  }
- D. class Vehicle {
  vehicle(name, price) {
  this.name = name;
  this.price = price;
  }
  priceInfo() {
  return 'Cost of the ${this.name} is ${this.price}$';
  }
  }

Answer: A

Explanation:
A correct conversion from constructor-function syntax to ES6 class syntax must satisfy:

Use a constructor(name, price) method.
Assign instance properties inside the constructor.
Define methods without function keyword directly in the class body.
Use proper JavaScript template literals (backticks).
Evaluate each option:
Option A
Incorrect template literal syntax: uses single quotes with ${} which will not interpolate.
Option B
Incorrect:
The method vehicle() is not recognized as a constructor.
The constructor method is missing.
Option C
Incorrect:
The constructor has no parameters, but uses name and price which are undefined.
Option D
Correct:
Uses proper constructor with parameters.
Correct assignment of instance properties.
Correct template literal using backticks.
This is the valid ES6 class translation.
JavaScript Knowledge Reference (text-only)
ES6 classes require a constructor method for initialization.
Template literals must use backticks.
Methods inside classes are defined without the function keyword.
Omitted constructor parameters lead to undefined instance fields.

NEW QUESTION # 81
A developer uses a parsed JSON string to work with user information as in the block below:
01 const userInformation = {
02 "id": "user-01",
03 "email": "user01@universalcontainers.demo",
04 "age": 25
05 };
Which two options access the email attribute in the object?

- A. userInformation.get("email")
- B. userInformation["email"]
- C. userInformation.email
- D. userInformation[email]

Answer: B,C

Explanation:
Comprehensive and Detailed Explanation From JavaScript Knowledge:
userInformation is a plain JavaScript object:
{
id: "user-01",
email: "user01@universalcontainers.demo",
age: 25
}
You can access object properties in two standard ways:
Dot notation: object.propertyName
Bracket notation: object['propertyName']
Apply to each option:
A . userInformation.email
Correct dot notation; accesses "user01@universalcontainers.demo".
B . userInformation.get("email")
.get() is a method on Map instances, not on plain objects.
A regular object does not have a .get method; this would be TypeError.
C . userInformation["email"]

Correct bracket notation; string "email" is used as the property key.
Returns "user01@universalcontainers.demo".
D . userInformation[email]
Here email is treated as a variable, not a string literal.
Unless there is a variable named email defined with a value matching a property key, this will either:
Throw a ReferenceError (if email is not defined), or
Use the value of email as a computed key (not what is intended here).
It is not the correct way to access the "email" property literal.
Therefore, the correct ways are:
Answe r: A, C
Relevant concepts: object property access (. vs []), plain objects vs Map, literal property names vs computed property names.

NEW QUESTION # 82
.....