Unit 1: Study Up on Browsers and Events

Learning Objectives

After completing this unit, you'll be able to:

- Utilize events, event handlers, and propagation.
- Evaluate and manipulate the Document Object Model (DOM).
- Implement the browser dev tools to investigate code behavior.
- Describe browser-specific APIs.

Key Topics

This unit prepares you for the Browsers and Events section of the Salesforce JavaScript Developer I multiple-choice exam, which makes up 17% of the overall exam. This section of the exam tests these topics.

- Event handlers
- DOM manipulation
- Event objects
- Default actions
- Pointer events
- Timers
- Debouncing

This unit provides a number of interactive, real-world, scenario-based questions that are a lot like the ones you'll encounter as a JavaScript developer. Looking at these scenarios helps prepare you to take the Browser and Events section of the Salesforce JavaScript Developer I multiple-choice exam. As you tackle the practice questions, you get immediate feedback on your answers, along with detailed information on why your answers are correct (or incorrect).

- The unit also contains interactive flashcards centered around study topics that help you prepare for the Browser and Events section of the exam.

Download the Guide

Would you like a hard copy of this module's content as a study aid? Download the JavaScript Developer I Certification Prep: Browsers and Events, Asynchronous Programming, and Server-Side JavaScript guide. (Each module in this trail includes a link to a printable version of the content that you can download.)

Exam Practice Questions

Ready to jump in? The sample tool below is not scored—it's just an easy way to quiz yourself. To use it, read the scenario, then click on the answer you think is correct. Some questions may have more than one correct answer. Click **Submit** to learn whether the answer you chose is correct or incorrect, and why; if there's a longer explanation, click and then click anywhere in the window to close it. When you reach the end, you can review the answers or retake the questions.

Question 1

Given two nested divs and the code below:

```
window.onload = (event) => {
  document.querySelector('.outerDiv')
    .addEventListener('click', displayOuterMessage, true);
  document.querySelector('.innerDiv')
    .addEventListener('click', displayInnerMessage, true);
};
```

What order will the event listeners be called when the innerDiv is clicked?

Option		Feedback
A	displayInnerMessage, displayOuterMessage	Incorrect. Events in the target phase will trigger all listeners on an element in the order they were registered.
B	displayInnerMessage only	Incorrect. Both event listeners will get called.
C	displayOuterMessage only	Incorrect. Both event listeners will get called.
D	displayOuterMessage, displayInnerMessage	**Correct. Events in the target phase will trigger all listeners on an element in the order they were registered.**

Question 2

If an application manipulates the browser history using the History API, which event should a developer use to detect when the browser's native back or forward button is clicked?

Option		Feedback
A	popstate	**Correct. A popstate event is dispatched to the window each time the active history entry changes between two history entries for the same document.**
B	navigate	Incorrect. The navigate() method loads a specified URL.
C	pushstate	Incorrect. The pushstate() method pushes the given data onto the session history stack with the specified title.
D	change	Incorrect. The change event is fired for changes to <input>, <select>, and <textarea> elements.

Did you get a scenario wrong? Check out the table below for related study material.

Scenario 1	Learn about event targets by reading MDN web docs: EventTarget.addEventListener.
Scenario 2	Review browser history to learn how to manipulate and monitor it.

Exam Topic Flashcards

The following flashcards cover Browsers and Events. Use these interactive flashcards to brush up on some of the key topics you'll find on this part of the exam.

Read the question or term on each card, then click or tap the card to reveal the correct answer. Click the right-facing arrow to move to the next card, and the left-facing arrow to return to the previous card.

Card 1	
Front of card (term or question)	Events are fired inside the browser window and tend to attach to specific items residing in the browser. What are some of these items?
Back of card (definition or answer)	A single element, set of elements, the HTML document loaded in the current tab, or the entire browser window.

Card 2	
Front of card (term or question)	What are some of the basic DOM data types?
Back of card (definition or answer)	Document, node, element, nodeList, attribute, and namedNodemap.

Did you get a flashcard wrong? Check out the table below for related study material.

Flashcard 1	Study up on events by reviewing MDN web docs: Introduction to events.
Flashcard 2	Review the various parts of the Document Object Model (DOM).

Related Badges

Looking for more information? Explore these related badges.

Badge	Content Type
 JavaScript Skills for Salesforce Developers	Module
 Modern JavaScript Development	Module

You've reviewed the browser and events. Next, let's take a look at asynchronous programming techniques.

Resources

- *External Site:* MDN web docs: Event reference
- *External Site:* MDN web docs: Introduction to events
- *External Site:* JavaScript.info: Developer console
- *External Site:* MDN web docs: Introduction to the DOM

Unit 2: Explore Asynchronous Programming

Learning Objectives

After completing this unit, you'll be able to:

- Apply asynchronous programming concepts, given a scenario.
- Use event loop and event monitor or determine loop outcomes, given a scenario.

Key Topics

This unit prepares you for the Asynchronous Programming section of the Salesforce JavaScript Developer I multiple-choice exam, which makes up 13% of the overall exam. This section of the exam tests these topics.

- Callbacks
- Promises
- Failures
- Asynchronous functions

Like the previous unit, this unit contains practice scenario-based questions and flashcards.

Exam Practice Questions

Ready to jump in? The sample tool below is not scored—it's just an easy way to quiz yourself. To use it, read the scenario, then click on the answer you think is correct. Some questions may have more than one correct answer. Click **Submit** to learn whether the answer you chose is correct or incorrect, and why; if there's a longer explanation, click  and then click anywhere in the window to close it. When you reach the end, you can review the answers or retake the questions.

Question 1

Refer to this code:

```
const p1 = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve('P1 Resolved');
  }, 1500);
});
const p2 = (data) => new Promise((resolve, reject) => {
  setTimeout(() => resolve('${data}, P2 Resolved'), 1500, data);
});
```

Which two answers correctly execute p1 and p2?

Option		Feedback
A	p1.then((data) => p2(data)).then(result => result);	**Correct. The method promise.then() is used to associate further action with a promise that becomes settled.**

B	async function getResult() {
	  const data = await p1;
	  return await p2(data);
	}
	getResult();
		Correct. Using await inside the function causes both to execute.

C	p1().then(function() {
	  p2().then(function(result) {
	  return result;
	  });
	});
		Incorrect. The syntax is wrong for calling p1 and p2.

D	async function getResult() {
	  const data = p1;
	  const result = p2(data);
	}
	await getResult();
		Incorrect. The await is outside the function calling the data.

Question 2

```
const getId = new Promise((resolve, reject) => {
  setTimeout(() => resolve(15), 1500);
});
const getBook = bookId => new Promise((resolve, reject) => {
  setTimeout(() => resolve('${bookId}:JavaScript Algorithms'), 1500);
});
getId.then(id => getBook(id)).then(data => data);
```

What is the correct code to replace the last line with an async/await function?

Option		Feedback

A	async function getBookInfo() {
	  const Id = await getId;
	  const result = await getBook(id);
	}
	getBookInfo();
		Correct. Correct use of await with an async function. The await expression is always inside an async function.

B	async function getBookInfo() {
	  const Id = await getId;
	  const result = await getBook(id);
	}
	await getBookInfo();
		Incorrect. Extra await calling getBookInfo() not necessary.

C	await function getBookInfo() {
	  const Id = getId;
	  const result = getBook(id);
	}
	async getBookInfo();
		Incorrect. Wrong use of await and async to get info. The await expression is always inside an async function.

D	async function getBookInfo() {
	  const Id = getId;
	  const result = getBook(id);
	}
	await getBookInfo();
		Incorrect. Wrong use of await and async to get info. The await expression is always inside an async function.

JavaScript Dev I Cert Prep: Browsers, Async Programming, Server Side JavaScript

Did you get a scenario wrong? Check out the table below for related study material.

Scenario 1	Study up on promise options by reading through MDN web docs: Promise.
Scenario 2	Review async functions in the MDN web docs: async function docs.

Exam Topic Flashcards

The following flashcards cover Asynchronous Programming. Use these interactive flashcards to brush up on some of the key topics you'll find on this part of the exam.

Read the question or term on each card, then click or tap the card to reveal the correct answer. Click the right-facing arrow to move to the next card, and the left-facing arrow to return to the previous card.

Card 1	
Front of card (term or question)	The promise object returned by the new Promise constructor has internal properties. What are they?
Back of card (definition or answer)	state — initially "pending", then changes to either "fulfilled" when resolve is called or "rejected" when reject is called. result — initially undefined, then changes to value when resolve(value) called or error when reject(error) is called.

Card 2	
Front of card (term or question)	Where can you use the await keyword and what does it do?
Back of card (definition or answer)	Keyword await only works inside async functions and makes JavaScript wait until that promise settles and returns its result.

Did you get a flashcard wrong? Check out the table below for related study material.

Flashcard 1	Understand promise returned values by studying JavaScript.info: Promise.
Flashcard 2	Learn how to use async and await keywords by reviewing the JavaScript.info: Async/await docs.

Related Badges

Looking for more information? Explore these related badges.

Badge	Content Type
 JavaScript Skills for Salesforce Developers	Module
 Modern JavaScript Development	Module

Congratulations! You've studied up on asynchronous programming techniques. Next, let's take a look at server-side JavaScript.

Resources

- *External Site:* JavaScript.info: Promise
- *External Site*: JavaScript.info: Async/await
- *External Site*: MDN web docs: Using Promises

Unit 3: Review Server Side JavaScript

Learning Objectives

After completing this unit, you'll be able to:

- Given a scenario and requirements, infer which Node.js implementation is a good solution.
- Given a scenario and requirements, infer which Node.js CLI command is a good solution.
- Know the core Node.js modules and given requirements, infer which Node.js library/framework is a good solution.
- Given a scenario and requirements, distinguish which Node.Js Package Management solution is the most fitting.

Key Topics

This unit prepares you for the Server-Side JavaScript section of the Salesforce JavaScript Developer I multiple-choice exam, which makes up 8% of the overall exam. This section of the exam tests these topics:

- Node.js fundamentals
- Package files
- File system and HTTP modules
- Streams

Like the previous units, this unit contains practice scenario-based questions and flashcards.

Exam Practice Questions

Ready to jump in? The sample tool below is not scored—it's just an easy way to quiz yourself. To use it, read the scenario, then click on the answer you think is correct. Some questions may have more than one correct answer. Click **Submit** to learn whether the answer you chose is correct or incorrect, and why; if there's a longer explanation, click  and then click anywhere in the window to close it. When you reach the end, you can review the answers or retake the questions.

Question 1

Here is the package.json for the bar.awesome module:

{"name": "bar.awesome","version": "1.3.5","peerDependencies": { "baz": "5.x" }}
A particular project has the package.json definition below.
{"name": "UC Project Extra","version": "0.0.5","dependencies": { "bar.awesome": "1.3.5", "baz": "6.0.0" }}

What happens when a developer executes npm install?

Option		Feedback
A	The command fails because bar.awesome does not have any dependency.	Incorrect. bar does have a dependency on baz.
B	The command fails because bar versions are not compatible.	Incorrect. The bar versions are compatible, but the baz versions are not.
C	The command succeeds but displays a warning about a version mismatch.	**Correct. There is a mismatch on the baz dependency that causes the warning.**
D	The command succeeds with no errors or warnings.	Incorrect. The command does succeed but there is a warning for the mismatched baz dependency.

Question 2

Refer to the code below:

```
01 const https = require('https');
02 const server = https.createServer((req, res) => {
03   // code goes here
04     let reqData = JSON.parse(chunk);
05     console.log(reqData);
06   });
07   res.end('OK');
08 });
09 server.listen(8000);
```

Which code does the developer need to add to line 03 to receive incoming request data?

Option		Feedback
A	req.on('data', (chunk) => {	**Correct. The chunk argument is passed in for JSON.parse to use.**
B	req.get((chunk) => {	Incorrect. The get method is used with http requests.
C	req.data((chunk) => {	Incorrect. Data is what is getting passed in, not a method.
D	req.on('get', (chunk) => {	Incorrect. The get argument is not passed in so can not be used this way.

Did you get a scenario wrong? Check out the table below for related study material.

Scenario 1	Review the package.json configuration and dependencies issues by reading the npm package.json documentation.
Scenario 2	Describe the server call functions by reviewing the Node.js https documentation.

Exam Topic Flashcards

The following flashcards cover server-side JavaScript. Use these interactive flashcards to brush up on some of the key topics you'll find on this part of the exam.

Read the question or term on each card, then click or tap the card to reveal the correct answer. Click the right-facing arrow to move to the next card, and the left-facing arrow to return to the previous card.

Card 1	
Front of card (term or question)	What are the most important items in your npm package.json file if you plan to publish a package?
Back of card (definition or answer)	The name and version. Together they form a unique identifier for the package.

Card 2	
Front of card (term or question)	When it comes to frameworks, what is one of the main benefits of using popular frameworks?
Back of card (definition or answer)	One of the main benefits is a strong community supporting the framework. This usually results in good documentation and resources.

Did you get a flashcard wrong? Check out the table below for related study material.

Flashcard 1	Study up on the npm package.json settings by reviewing the Specifics of npm's package.json handling.
Flashcard 2	Discover the popular frameworks that are available by reviewing the best front-end frameworks and the best back-end frameworks.

Related Badges

Looking for more information? Explore these related badges.

Badge	Content Type
 JavaScript Skills for Salesforce Developers	Module
 Modern JavaScript Development	Module
 Salesforce & Heroku Integration	Module

Congratulations. You've covered 38% of the JavaScript Developer I multiple-choice exam material in this badge.

You've reviewed these sections.

- Browsers and Events
- Asynchronous Programming
- Server-Side JavaScript

Be sure to review the other two JavaScript Developer I Certification Prep badges. Good luck on your exam!

Resources

- *External Site*: Node: Documentation
- *External Site*: npm: CLI documentation