1. Guitar Tuner

|  |  |  |  |
| --- | --- | --- | --- |
| ■ Logging | ■ Persisted State | ■ Helper functions | ■ Alexa boilerplate |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 | 'use strict'**;**  **const** Alexa **=** require**(**'ask-sdk-v1adapter'**);**  **const** handlers **=** **{**  'LaunchRequest'**:** **function** **()** **{**  logEvent**(this.**event**);**    **this.**emit**(**':ask'**,** 'Welcome to Guitar Tuner, what note would you like me to play?'**);**  **},**  'PlayNoteIntent'**:** **function** **()** **{**  logEvent**(this.**event**);**    **let** note **=** getSlotValue**(this.**event**,** 'note'**);**  setAttrValue**(this.**event**,** 'note'**,** note**);**    **this.**emit**(**':ask'**,** 'What pitch would you like for ' **+** note **+** '?'**);**  **},**  'SpecifyPitchIntent'**:** **function** **()** **{**  logEvent**(this.**event**);**    **let** note **=** getAttrValue**(this.**event**,** 'note'**);**  **let** pitch **=** getSlotValue**(this.**event**,** 'pitch'**);**    **this.**emit**(**':ask'**,** getNotePitchOutput**(**note**,** pitch**));**  **},**  'PlayNoteWithPitchIntent'**:** **function** **()** **{**  logEvent**(this.**event**);**    **let** note **=** getSlotValue**(this.**event**,** 'note'**);**  **let** pitch **=** getSlotValue**(this.**event**,** 'pitch'**);**    **this.**emit**(**':ask'**,** getNotePitchOutput**(**note**,** pitch**));**  **},**  'AMAZON.HelpIntent'**:** **function** **()** **{**  *// This is triggered when users say "Help"*  **},**  'AMAZON.CancelIntent'**:** **function** **()** **{**  *// This is triggered when users say "Cancel"*  **},**  'AMAZON.StopIntent'**:** **function** **()** **{**  *// This is triggered when users say "Stop"*  **},**  'AMAZON.NavigateHomeIntent'**:** **function** **()** **{**  *// This is triggered when users say "Navigate Home"*  **},**  'AMAZON.FallbackIntent'**:** **function** **()** **{**  *// This is triggered when users say something that doesn't map to an intent*  **this.**emit**(**':tell'**,** 'Sorry, I couldn\'t understand you'**);**  **},**  'SessionEndedRequest'**:** **function** **()** **{**  *// This is triggered when users say "Exit"*  **}**  **};**  **function** getNotePitchOutput**(**note**,** pitch**)** **{**  **let** output **=** 'You requested a note ' **+** note **+** ' with pitch ' **+** pitch**;**  console**.**log**(**'Output: ' **+** output**);**  **return** output**;**  **}**  **function** getAttrValue**(**event**,** attrName**)** **{**  **var** attrValue **=** event**.**session**.**attributes**[**attrName**];**  console**.**log**(**'Session has attribute ' **+** attrName **+** ': ' **+** attrValue**);**  **return** attrValue**;**  **}**  **function** setAttrValue**(**event**,** attrName**,** attrValue**)** **{**  event**.**session**.**attributes**[**attrName**]** **=** attrValue**;**  console**.**log**(**'Saved session attribute ' **+** attrName **+** ': ' **+** attrValue**);**  **}**  **function** getSlotValue**(**event**,** slotName**)** **{**  **let** slot **=** event**.**request**.**intent**.**slots**[**slotName**];**  **let** slotValue**;**  **try** **{**  *// Resolve to the canonical value if available*  slotValue **=** slot**.**resolutions**.**resolutionsPerAuthority**[0].**values**[0].**value**.**name**;**  **}** **catch(**err**)** **{**  *// Otherwise fallback to the raw text*  console**.**log**(**err**.**message**);**  slotValue **=** slot**.**value**;**  **}**    console**.**log**(**'User provided ' **+** slotName **+** ': ' **+** slotValue**);**  **return** slotValue**;**  **}**  **function** logEvent**(**event**)** **{**  console**.**log**(**'Request:'**);**  console**.**log**(**JSON**.**stringify**(**event**.**request**));**  console**.**log**(**'Session:'**);**  console**.**log**(**JSON**.**stringify**(**event**.**session**));**  **}**  exports**.**handler **=** **function(**event**,** context**,** callback**)** **{**  **const** alexa **=** Alexa**.**handler**(**event**,** context**,** callback**);**  alexa**.**registerHandlers**(**handlers**);**  alexa**.**execute**();**  **};** |

1. High Low Game

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ■ Logging | ■ Input validation | ■ Persisted state | ■ Helper functions | ■ Alexa boilerplate |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14    15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36    37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130 | 'use strict'**;**  **const** Alexa **=** require**(**'ask-sdk-v1adapter'**);**  **const** ALLOWED\_HINTS **=** **[** 'low'**,** 'high' **];**  **function** avg**(**n1**,** n2**)** **{**  **return** Math**.**floor**((**n1 **+** n2**)** **/** **2);**  **}**  **const** handlers **=** **{**  'LaunchRequest'**:** **function** **()** **{**  logEvent**(this.**event**);**    **this.**emit**(**':ask'**,** 'Welcome to High Low Game. Think of a number between 1 and 100, and say "start" when ready.'**);**  **},**  'StartIntent'**:** **function** **()** **{**  logEvent**(this.**event**);**    **let** minGuess **=** **1;**  **let** maxGuess **=** **100;**  **let** guess **=** avg**(**minGuess**,** maxGuess**);**    setAttrValue**(this.**event**,** 'guessesLeft'**,** **4);**  setAttrValue**(this.**event**,** 'minGuess'**,** minGuess**);**  setAttrValue**(this.**event**,** 'maxGuess'**,** maxGuess**);**  setAttrValue**(this.**event**,** 'lastGuess'**,** guess**);**    **this.**emit**(**':ask'**,** produceGuessOutput**(**guess**));**  **},**  'HintIntent'**:** **function** **()** **{**  logEvent**(this.**event**);**    **let** lastGuess **=** getAttrValue**(this.**event**,** 'lastGuess'**);**  **let** hint **=** getSlotValue**(this.**event**,** 'hint'**);**  **if** **(!**ALLOWED\_HINTS**.**includes**(**hint**))** **{**  **this.**emit**(**':ask'**,** 'Sorry, I didn\'t get your hint. Is ' **+** lastGuess **+** ' high, low or correct?'**);**  **return;**  **}**    **let** guessesLeft **=** getAttrValue**(this.**event**,** 'guessesLeft'**);**  **if** **(**guessesLeft **===** **0)** **{**  **this.**emit**(**':tell'**,** 'I lost. Good game, thank you for playing!'**);**  **return;**  **}**  setAttrValue**(this.**event**,** 'guessesLeft'**,** guessesLeft **-** **1);**    **let** minGuess**,** maxGuess**;**    **if** **(**hint **===** 'low'**)** **{**  minGuess **=** lastGuess**;**  setAttrValue**(this.**event**,** 'minGuess'**,** minGuess**);**  maxGuess **=** getAttrValue**(this.**event**,** 'maxGuess'**);**  **}** **else** **{**  minGuess **=** getAttrValue**(this.**event**,** 'minGuess'**);**  maxGuess **=** lastGuess**;**  setAttrValue**(this.**event**,** 'maxGuess'**,** maxGuess**);**  **}**    **let** guess **=** avg**(**minGuess**,** maxGuess**);**  setAttrValue**(this.**event**,** 'lastGuess'**,** guess**);**  **this.**emit**(**':ask'**,** produceGuessOutput**(**guess**));**  **},**  'CorrectIntent'**:** **function** **()** **{**  logEvent**(this.**event**);**    **this.**emit**(**':tell'**,** 'Hooray! Thank you for playing!'**);**  **},**  'AMAZON.HelpIntent'**:** **function** **()** **{**  *// This is triggered when users say "Help"*  **},**  'AMAZON.CancelIntent'**:** **function** **()** **{**  *// This is triggered when users say "Cancel"*  **},**  'AMAZON.StopIntent'**:** **function** **()** **{**  *// This is triggered when users say "Stop"*  **},**  'AMAZON.NavigateHomeIntent'**:** **function** **()** **{**  *// This is triggered when users say "Navigate Home"*  **},**  'AMAZON.FallbackIntent'**:** **function** **()** **{**  *// This is triggered when users say something that doesn't map to an intent*  **},**  'SessionEndedRequest'**:** **function** **()** **{**  *// This is triggered when users say "Exit"*  **}**  **};**  **function** produceGuessOutput**(**guess**)** **{**  **return** guess **+** '. Is it high, low or correct?'**;**  **}**  **function** getAttrValue**(**event**,** attrName**)** **{**  **var** attrValue **=** event**.**session**.**attributes**[**attrName**];**  console**.**log**(**'Session has attribute ' **+** attrName **+** ': ' **+** attrValue**);**  **return** attrValue**;**  **}**  **function** setAttrValue**(**event**,** attrName**,** attrValue**)** **{**  event**.**session**.**attributes**[**attrName**]** **=** attrValue**;**  console**.**log**(**'Saved session attribute ' **+** attrName **+** ': ' **+** attrValue**);**  **}**  **function** getSlotValue**(**event**,** slotName**)** **{**  **let** slot **=** event**.**request**.**intent**.**slots**[**slotName**];**  **let** slotValue**;**  **try** **{**  *// Resolve to the canonical value if available*  slotValue **=** slot**.**resolutions**.**resolutionsPerAuthority**[0].**values**[0].**value**.**name**;**  **}** **catch(**err**)** **{**  *// Otherwise fallback to the raw text*  console**.**log**(**err**.**message**);**  slotValue **=** slot**.**value**;**  **}**    console**.**log**(**'User provided ' **+** slotName **+** ': ' **+** slotValue**);**  **return** slotValue**;**  **}**  **function** logEvent**(**event**)** **{**  console**.**log**(**'Request:'**);**  console**.**log**(**JSON**.**stringify**(**event**.**request**));**  console**.**log**(**'Session:'**);**  console**.**log**(**JSON**.**stringify**(**event**.**session**));**  **}**  exports**.**handler **=** **function(**event**,** context**,** callback**)** **{**  **const** alexa **=** Alexa**.**handler**(**event**,** context**,** callback**);**  alexa**.**registerHandlers**(**handlers**);**  alexa**.**execute**();**  **};** |

1. Artistic Joke

|  |  |  |  |
| --- | --- | --- | --- |
| ■ Logging | ■ Input validation | ■ Random choice | ■ Dictionary lookup |
| ■ Modified speech (SSML) | ■ Helper functions | ■ Alexa boilerplate |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16    17  18  19    20  21  22    23    24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53    54    55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131 | 'use strict'**;**  **const** Alexa **=** require**(**'ask-sdk-v1adapter'**);**  **const** MODIFIER\_TO\_PROSODY **=** **{**  'quiet'**:** 'volume="x-soft"'**,**  'loud'**:** 'volume="x-loud"'**,**  'quick'**:** 'rate="x-fast"'**,**  'slow'**:** 'rate="x-slow"'**,**  'high'**:** 'pitch="x-high"'**,**  'low'**:** 'pitch="x-low"'  **};**  **const** ALLOWED\_MODIFIERS **=** Object**.**keys**(**MODIFIER\_TO\_PROSODY**);**  **const** JOKES **=** **[**  'Today at the bank, an old lady asked me to help check her balance. So I pushed her over.'**,**  'I couldn\'t figure out why the baseball kept getting larger. Then it hit me.'**,**  'I told my girlfriend she drew her eyebrows too high. She seemed surprised.'**,**  'My dog used to chase people on a bike a lot. It got so bad, finally I had to take his bike away.'**,**  'I\'m so good at sleeping. I can do it with my eyes closed.'**,**  'My boss told me to have a good day... so I went home.'**,**  'A woman walks into a library and asked if they had any books about paranoia. The librarian says "They\'re right behind you!"'**,**  'The other day, my wife asked me to pass her lipstick but I accidentally passed her a glue stick. She still isn\'t talking to me.'**,**  'Why do blind people hate skydiving? It scares the hell out of their dogs.'**,**  'When you look really closely, all mirrors look like eyeballs.'  **];**  **function** getRandomInt**(**max**)** **{**  **return** Math**.**floor**(**Math**.**random**()** **\*** Math**.**floor**(**max**));**  **}**  **function** pickAJoke**()** **{**  **return** JOKES**[**getRandomInt**(**JOKES**.**length**)];**  **}**  **const** handlers **=** **{**  'LaunchRequest'**:** **function** **()** **{**  logEvent**(this.**event**);**  **this.**emit**(**':ask'**,** 'What kind of joke do you want to hear?'**);**  **},**  'TellAJokeIntent'**:** **function** **()** **{**  logEvent**(this.**event**);**    **let** joke **=** pickAJoke**();**  **this.**emit**(**':ask'**,** joke**);**  **},**  'WhisperAJokeIntent'**:** **function** **()** **{**  logEvent**(this.**event**);**    **let** joke **=** pickAJoke**();**    // Docs: https://developer.amazon.com/docs/custom-skills/speech-synthesis-markup-language-ssml-reference.html'  **let** ssmlResponse **=** '<amazon:effect name="whispered">' **+** joke **+** '</amazon:effect>**;**  **this.**emit**(**':ask'**,** ssmlResponse**);**  **},**  'ModifiedJokeIntent'**:** **function** **()** **{**  logEvent**(this.**event**);**    **let** modifier **=** getSlotValue**(this.**event**,** 'modifier'**);**  **if** **(!**ALLOWED\_MODIFIERS**.**includes**(**modifier**))** **{**  **this.**emit**(**':ask'**,** 'I don\'t know that kind of jokes'**);**  **return;**  **}**    **let** joke **=** pickAJoke**();**  **let** prosody **=** MODIFIER\_TO\_PROSODY**[**modifier**];**    *// Example: <prosody pitch="high">Pinnochio drowned</prosody>*  **let** ssmlResponse **=** '<prosody ' **+** prosody **+** '>' **+** joke **+** '</prosody>'**;**  **this.**emit**(**':ask'**,** ssmlResponse**);**  **},**  'AMAZON.HelpIntent'**:** **function** **()** **{**  *// This is triggered when users say "Help"*  **},**  'AMAZON.CancelIntent'**:** **function** **()** **{**  *// This is triggered when users say "Cancel"*  **},**  'AMAZON.StopIntent'**:** **function** **()** **{**  *// This is triggered when users say "Stop"*  **},**  'AMAZON.NavigateHomeIntent'**:** **function** **()** **{**  *// This is triggered when users say "Navigate Home"*  **},**  'AMAZON.FallbackIntent'**:** **function** **()** **{**  *// This is triggered when users say something that doesn't map to an intent*  **},**  'SessionEndedRequest'**:** **function** **()** **{**  *// This is triggered when users say "Exit"*  **}**  **};**  **function** getSlotValue**(**event**,** slotName**)** **{**  **let** slot **=** event**.**request**.**intent**.**slots**[**slotName**];**  **let** slotValue**;**  **try** **{**  *// Resolve to the canonical value if available*  slotValue **=** slot**.**resolutions**.**resolutionsPerAuthority**[0].**values**[0].**value**.**name**;**  **}** **catch(**err**)** **{**  *// Otherwise fallback to the raw text*  console**.**log**(**err**.**message**);**  slotValue **=** slot**.**value**;**  **}**    console**.**log**(**'User provided ' **+** slotName **+** ': ' **+** slotValue**);**  **return** slotValue**;**  **}**  **function** getAttrValue**(**event**,** attrName**)** **{**  **var** attrValue **=** event**.**session**.**attributes**[**attrName**];**  console**.**log**(**'Session has attribute ' **+** attrName **+** ': ' **+** attrValue**);**  **return** attrValue**;**  **}**  **function** setAttrValue**(**event**,** attrName**,** attrValue**)** **{**  event**.**session**.**attributes**[**attrName**]** **=** attrValue**;**  console**.**log**(**'Saved session attribute ' **+** attrName **+** ': ' **+** attrValue**);**  **}**  **function** logEvent**(**event**)** **{**  console**.**log**(**'Request:'**);**  console**.**log**(**JSON**.**stringify**(**event**.**request**));**  console**.**log**(**'Session:'**);**  console**.**log**(**JSON**.**stringify**(**event**.**session**));**  **}**  exports**.**handler **=** **function(**event**,** context**,** callback**)** **{**  **const** alexa **=** Alexa**.**handler**(**event**,** context**,** callback**);**  alexa**.**registerHandlers**(**handlers**);**  alexa**.**execute**();**  **};** | |  | |  |

1. Seattle Getaway

|  |  |  |  |
| --- | --- | --- | --- |
| ■ Logging | ■ Web request | ■ Helper functions | ■ Alexa boilerplate |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27    28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123 | 'use strict'**;**  **const** Alexa **=** require**(**'ask-sdk-v1adapter'**);**  **const** https **=** require**(**'https'**);**  **function** distanceCallback**(**responseString**,** callbackParam**)** **{**  **let** city **=** callbackParam**[0];**  **let** emit **=** callbackParam**[1];**  **let** responseJson **=** JSON**.**parse**(**responseString**);**  **let** durationText **=** responseJson**.**rows**[0].**elements**[0].**duration**.**text**;**  emit**(**':ask'**,** 'Drive to ' **+** city **+** ' is going to be ' **+** durationText**);**  **}**  **const** handlers **=** **{**  'LaunchRequest'**:** **function** **()** **{**  logEvent**(this.**event**);**  **this.**emit**(**':ask'**,** 'Which city would you like to go to?'**);**  **},**  'PickCityIntent'**:** **function** **()** **{**  logEvent**(this.**event**);**    **let** city **=** getSlotValue**(this.**event**,** 'city'**);**  **let** encodedCity **=** encodeURIComponent**(**city**);**    httpGet**(**  'maps.googleapis.com'**,**  '/maps/api/distancematrix/json?origins=Seattle,WA&destinations=' **+** encodedCity **+** '&key=AIzaSyB4nTfp2M\_t2L\_zKj4Z4Wpb61mGs\_wOQAw'**,**  distanceCallback**,**  **[**city**,** **this.**emit**]);**    *// httpGet() returns before the request is finished*  *// We don't know the response here, callback will know it*  *// Alexa will wait till callback calls emit()*  **},**  'AMAZON.HelpIntent'**:** **function** **()** **{**  *// This is triggered when users say "Help"*  **},**  'AMAZON.CancelIntent'**:** **function** **()** **{**  *// This is triggered when users say "Cancel"*  **},**  'AMAZON.StopIntent'**:** **function** **()** **{**  *// This is triggered when users say "Stop"*  **},**  'AMAZON.NavigateHomeIntent'**:** **function** **()** **{**  *// This is triggered when users say "Navigate Home"*  **},**  'AMAZON.FallbackIntent'**:** **function** **()** **{**  *// This is triggered when users say something that doesn't map to an intent*  **},**  'SessionEndedRequest'**:** **function** **()** **{**  *// This is triggered when users say "Exit"*  **}**  **};**  **function** httpGet**(**host**,** path**,** callback**,** callbackParam**)** **{**  console**.**log**(**'Outgoing request host: ' **+** host**);**  console**.**log**(**'Outgoing request path: ' **+** path**);**  console**.**log**(**'Outgoing request callback param: ' **+** callbackParam**);**    **let** options **=** **{**  host**:** host**,**  path**:** path**,**  method**:** 'GET'**,**  **};**  **let** req **=** https**.**request**(**options**,** res **=>** **{**  res**.**setEncoding**(**'utf8'**);**  **let** responseString **=** ''**;**    *//accept incoming data asynchronously*  res**.**on**(**'data'**,** chunk **=>** **{**  responseString **=** responseString **+** chunk**;**  **});**    *//return the data when streaming is complete*  res**.**on**(**'end'**,** **()** **=>** **{**  console**.**log**(**'Received response: ' **+** responseString**);**  callback**(**responseString**,** callbackParam**);**  **});**  **});**  req**.**end**();**  **}**  **function** getSlotValue**(**event**,** slotName**)** **{**  **let** slot **=** event**.**request**.**intent**.**slots**[**slotName**];**  **let** slotValue**;**  **try** **{**  *// Resolve to the canonical value if available*  slotValue **=** slot**.**resolutions**.**resolutionsPerAuthority**[0].**values**[0].**value**.**name**;**  **}** **catch(**err**)** **{**  *// Otherwise fallback to the raw text*  console**.**log**(**err**.**message**);**  slotValue **=** slot**.**value**;**  **}**    console**.**log**(**'User provided ' **+** slotName **+** ': ' **+** slotValue**);**  **return** slotValue**;**  **}**  **function** getAttrValue**(**event**,** attrName**)** **{**  **var** attrValue **=** event**.**session**.**attributes**[**attrName**];**  console**.**log**(**'Session has attribute ' **+** attrName **+** ': ' **+** attrValue**);**  **return** attrValue**;**  **}**  **function** setAttrValue**(**event**,** attrName**,** attrValue**)** **{**  event**.**session**.**attributes**[**attrName**]** **=** attrValue**;**  console**.**log**(**'Saved session attribute ' **+** attrName **+** ': ' **+** attrValue**);**  **}**  **function** logEvent**(**event**)** **{**  console**.**log**(**'Request:'**);**  console**.**log**(**JSON**.**stringify**(**event**.**request**));**  console**.**log**(**'Session:'**);**  console**.**log**(**JSON**.**stringify**(**event**.**session**));**  **}**  exports**.**handler **=** **function(**event**,** context**,** callback**)** **{**  **const** alexa **=** Alexa**.**handler**(**event**,** context**,** callback**);**  alexa**.**registerHandlers**(**handlers**);**  alexa**.**execute**();**  **};** |