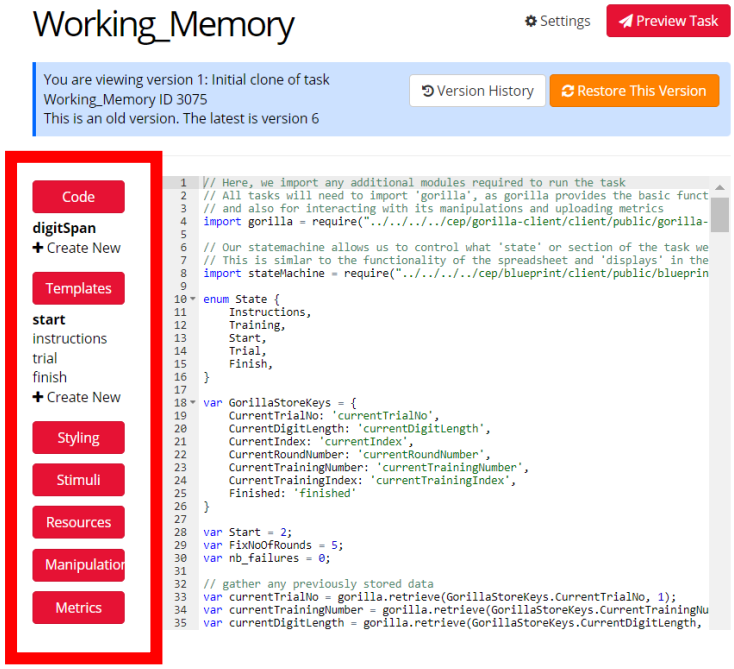
実験のソースを見てみると下図の赤枠で囲ったものの内の Code, Templates, Styling と Metrics にコードの記述がありました。以下にそれぞれのコードを貼り付けました。



Code

digitSpan

```
// Here, we import any additional modules required to run the task
// All tasks will need to import 'gorilla', as gorilla provides the basic functionality for running the task
// and also for interacting with its manipulations and uploading metrics
import gorilla = require("../../../../cep/gorilla-client/client/public/gorilla-client/js/gorilla");

// Our statemachine allows us to control what 'state' or section of the task we are in
// This is simlar to the functionality of the spreadsheet and 'displays' in the task builder
import stateMachine =
require("../../../../cep/blueprint/client/public/blueprint/js/state_machine");

enum State {
   Instructions,
   Training,
   Start,
   Trial,
   Finish,
}

var GorillaStoreKeys = {
   CurrentTrialNo: 'currentTrialNo',
   CurrentDigitLength: 'currentDigitLength',
   CurrentIndex: 'currentIndex',
```

```
   CurrentRoundNumber: 'currentRoundNumber',
   CurrentTrainingNumber: 'currentTrainingNumber',
   CurrentTrainingIndex: 'currentTrainingIndex',
   Finished: 'finished'
}

var Start = 2;
var FixNoOfRounds = 5;
var nb_failures = 0;

// gather any previously stored data
var currentTrialNo = gorilla.retrieve(GorillaStoreKeys.CurrentTrialNo, 1);
var currentTrainingNumber = gorilla.retrieve(GorillaStoreKeys.CurrentTrainingNumber, 1);
var currentDigitLength = gorilla.retrieve(GorillaStoreKeys.CurrentDigitLength, Start);
var currentRoundNumber = gorilla.retrieve(GorillaStoreKeys.CurrentRoundNumber, 1);
var finishedFlag = gorilla.retrieve(GorillaStoreKeys.Finished, false);
var score = 0;
var Trainingscore = 0;

// In this function we create the logic for our task: it's states and functionality
gorilla.ready(()=> {

   // attach the responsive resizing logic to the container
   gorilla.responsiveFrame('#gorilla');
   var SM = new stateMachine.StateMachine();

   // In this state, we will display our instructions for the task
   // these will need to change depending on whether we want the digits entered in the forward
or reversed order
   SM.addState(State.Instructions, {
      // The onEnter functions is executed when a state is entered.  It is the first thing a state will
do.
      onEnter: (machine: stateMachine.Machine) => {
         //var text = (DigitDirection == 'forward') ? "repeat the digits in the SAME order" : "repeat
the digits in the REVERSED order";
         gorilla.populate('#gorilla', 'instructions');
         gorilla.refreshLayout();
         $('#start-button').one('click', (event: JQueryEventObject) => {
            machine.transition(State.Training);
         });
      }
   })

   // in this state, the digits will be displayed and the response panel will be accessible
   SM.addState(State.Training, {
      onEnter: (machine: stateMachine.Machine) => {
         var numberSet = [ '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'];
         // Choose the digits that will be displayed to the participant
         // This needs a no. of digits equal to our current digit length
         var trialArray = [];
         for(var i = 0; i< 2; i++) {
```

```
            var numberIndex = randomIntFromInterval(0, 9 - trialArray.length);
            trialArray.push(numberSet[numberIndex]);
            numberSet.splice(numberIndex, 1);
        }

        // Choose the correct answers - either the trial array, or the trial array reversed
        var TraininganswerArray = [];
            for(j = trialArray.length - 1; j > -1; j--){
                TraininganswerArray.push(trialArray[j]);
            }

        // Populate our trial screen
        gorilla.populate('#gorilla', 'trial', {});
        $('.numberDisplay').hide();
        $('.gorilla-fixation-cross').hide();
        gorilla.refreshLayout();

        // if the screen refreshes we always want to start from 0
        var currentTrainingIndex = 0;

        // Display the fixation cross
        $('#gorilla')
              .queue(function(){
                $('.gorilla-fixation-cross').show();
                gorilla.refreshLayout();
                $(this).dequeue();
              })
              .delay(450)
              .queue(function(){
                $('.gorilla-fixation-cross').hide();
                gorilla.refreshLayout();
                $(this).dequeue();
              })
              .delay(500);

        // Display the numbers to the participant

        // this function displays a number on the screen, cycling through each element in our trial
array
        // the function is recursive, calling it self again and again until we have displayed all of the
trial numbers
        function numberDisplay(callback){
            $('#gorilla')
              .queue(function(){
                $('.numberDisplay').show();
                $('.numberDisplay').html(trialArray[currentTrainingIndex]);
                gorilla.refreshLayout();
                $(this).dequeue();
              })
              .delay(1500)
              .queue(function() {
```

```
            currentTrainingIndex++;
            $('.numberDisplay').hide();
            gorilla.refreshLayout();
            $(this).dequeue();
          })
          .delay(500)
          .queue(function() {
            if(currentTrainingIndex < trialArray.length){
              numberDisplay(callback);
            }
            else {
              callback();
            }
            $(this).dequeue();
          });
      }

      // once the display of numbers has been completed, we run the function below
      numberDisplay( function() {
        // Indicate the number of digits we are expecting
        var answerTrainingIndex = 0;
        var responseTrainingTextArray = [];
        var responseTrainingText = '';
        for(var i=0; i<2; i++){
          responseTrainingTextArray.push('<span style="width: 70px; display:inline-
block;">_</span>');
          responseTrainingText += responseTrainingTextArray[i];
        }

        $('.numberDisplay').show();
        $('.numberDisplay').html(responseTrainingText);
        gorilla.refreshLayout();

        var responseTrainingSet = [];
        var setTrainingCorrect = true;
        var totalTrainingCorrect = 0;

        $('.c-answer-button').on('click', (event: JQueryEventObject) => {
          // begin with logic to display digit on screen
          var Traininganswer = $(event.currentTarget).data('answer');
          responseTrainingSet.push(Traininganswer);
          responseTrainingTextArray[answerTrainingIndex] = '<span style="width: 70px;
display:inline-block;">' + Traininganswer + '</span>';
          responseTrainingText = '';
          for(var i=0; i<responseTrainingTextArray.length; i++){
            responseTrainingText+= responseTrainingTextArray[i];
          }

          $('.numberDisplay').html(responseTrainingText);
          gorilla.refreshLayout();
```

```
            // Check whether the digit is correct
            var instanceTrainingCorrect = (Traininganswer ==
TrainingAnswerArray[answerTrainingIndex]) ? true : false;
            if(instanceTrainingCorrect){
              totalTrainingCorrect++;
            }

            answerTrainingIndex++;
            // if we have reached the final number of the answer array
            if (answerTrainingIndex == TrainingAnswerArray.length) {
              $("input[type=button]").attr("disabled", "disabled");

              // if we have gotten any answers wrong, we need to acknowledge that the set was
incorrect
              if(totalTrainingCorrect == TrainingAnswerArray.length) {
                Trainingscore++;
              }
              else {
                setTrainingCorrect = false;
              }

              if(setTrainingCorrect){
                $('.feedback-correct').show();
              } else {
                $('.feedback-incorrect').show();
              }

            gorilla.refreshLayout();

            gorilla.metric({
              trainingtarget: TrainingAnswerArray,
              trainingresponse: responseTrainingSet,
              totaltrainingcorrect: totalTrainingCorrect,
              length: 2,
              trainingcorrect: setTrainingCorrect,
              trainingScore: Trainingscore,
            });

            currentRoundNumber++;

            //   currentTrainingNo++;
            //   gorilla.store(GorillaStoreKeys.CurrentTrainingNo, currentTrainingNo);

            // add a small delay so that the final entered number is briefly displayed before moving
on
              $('#gorilla')
                .delay(500)
                .queue(function(){
                  machine.transition(State.Training);
                  $(this).dequeue();
                });
```

```
            };
          });
        });
    },
    onExit: (machine: stateMachine.Machine) => {
      // we want to check if we've reached the maximum allowed number of trials.  If we have,
we finish the task
      if(currentRoundNumber > 2) {
        finishedFlag = true;
        gorilla.store(GorillaStoreKeys.Finished, finishedFlag);
        machine.transition(State.Start);
        currentRoundNumber = 1;
      //}
    }
  })

  SM.addState(State.Start, {
    // The onEnter functions is executed when a state is entered.  It is the first thing a state will
do.
    onEnter: (machine: stateMachine.Machine) => {
      gorilla.populate('#gorilla', 'start');
      gorilla.refreshLayout();
      $('#start-button').one('click', (event: JQueryEventObject) => {
        machine.transition(State.Trial);
      });
    }
  })

  // in this state, the digits will be displayed and the response panel will be accessible
  SM.addState(State.Trial, {
    onEnter: (machine: stateMachine.Machine) => {
      var numberSet = [ '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'];
      if currentRoundNumber == 1 {
        nb_failures = 0;
      };

      // Choose the digits that will be displayed to the participant
      // This needs a no. of digits equal to our current digit length
      var trialArray = [];
      for(var i = 0; i< currentDigitLength; i++) {
        var numberIndex = randomIntFromInterval(0, 9 - trialArray.length);
        trialArray.push(numberSet[numberIndex]);
        numberSet.splice(numberIndex, 1);
      }

      // Choose the correct answers - either the trial array, or the trial array reversed
      var answerArray = [];
        for(j = trialArray.length - 1; j > -1; j--){
          answerArray.push(trialArray[j]);
        }
```

```
        // Populate our trial screen
        gorilla.populate('#gorilla', 'trial', {});
        $('.numberDisplay').hide();
        $('.gorilla-fixation-cross').hide();
        gorilla.refreshLayout();

        // if the screen refreshes we always want to start from 0
        var currentIndex = 0;

        // Display the fixation cross
        $('#gorilla')
            .queue(function(){
              $('.gorilla-fixation-cross').show();
              gorilla.refreshLayout();
              $(this).dequeue();
            })
            .delay(450)
            .queue(function(){
              $('.gorilla-fixation-cross').hide();
              gorilla.refreshLayout();
              $(this).dequeue();
            })
            .delay(500);

        // Display the numbers to the participant

        // this function displays a number on the screen, cycling through each element in our trial
array
        // the function is recursive, calling it self again and again until we have displayed all of the
trial numbers
        function numberDisplay(callback){
          $('#gorilla')
            .queue(function(){
              $('.numberDisplay').show();
              $('.numberDisplay').html(trialArray[currentIndex]);
              gorilla.refreshLayout();
              $(this).dequeue();
            })
            .delay(1500)
            .queue(function() {
              currentIndex++;
              $('.numberDisplay').hide();
              gorilla.refreshLayout();
              $(this).dequeue();
            })
            .delay(500)
            .queue(function() {
              if(currentIndex < trialArray.length){
                numberDisplay(callback);
              }
              else {
```

```
                callback();
            }
            $(this).dequeue();
        });
    }

    // once the display of numbers has been completed, we run the function below
    numberDisplay( function() {
        // Indicate the number of digits we are expecting
        var answerIndex = 0;
        var responseTextArray = [];
        var responseText = '';
        for(var i=0; i<currentDigitLength; i++){
            responseTextArray.push('<span style="width: 70px; display:inline-block;">_</span>');
            responseText += responseTextArray[i];
        }

        $('.numberDisplay').show();
        $('.numberDisplay').html(responseText);
        gorilla.refreshLayout();

        var responseSet = [];
        var setCorrect = true;
        var totalCorrect = 0;

        $('.c-answer-button').on('click', (event: JQueryEventObject) => {
            // begin with logic to display digit on screen
            var answer = $(event.currentTarget).data('answer');
            responseSet.push(answer);
            responseTextArray[answerIndex] = '<span style="width: 70px; display:inline-block;">' +
answer + '</span>';
            responseText = '';
            for(var i=0; i<responseTextArray.length; i++){
                responseText+= responseTextArray[i];
            }

            $('.numberDisplay').html(responseText);
            gorilla.refreshLayout();

            // Check whether the digit is correct
            var instanceCorrect = (answer == answerArray[answerIndex]) ? true : false;
            if(instanceCorrect){
                totalCorrect++;
            }

            answerIndex ++;
            // if we have reached the final number of the answer array
            if (answerIndex == answerArray.length) {
                $("input[type=button]").attr("disabled", "disabled");
```

```
                // if we have gotten any answers wrong, we need to acknowledge that the set was
incorrect
                if(totalCorrect == answerArray.length) {
                  score++;
                }
                else {
                  setCorrect = false;
                  nb_failures++;
                }

                gorilla.refreshLayout();

                gorilla.metric({
                  target: answerArray,
                  response: responseSet,
                  length: answerArray.length,
                  correct: setCorrect,
                  totalCorrect: totalCorrect,
                  Score: score,
                });

                currentTrialNo++;
                gorilla.store(GorillaStoreKeys.CurrentTrialNo, currentTrialNo);

                // Logic to determine what happens in the next round
                  // need to check if we've reached the end of a round, if so, increase digit length
                  if(currentRoundNumber == FixNoOfRounds & nb_failures < 3) {
                    currentRoundNumber = 1;
                    currentDigitLength++;
                    gorilla.store(GorillaStoreKeys.CurrentDigitLength, currentDigitLength);
                  } else {
                    currentRoundNumber++;
                  }
                  gorilla.store(GorillaStoreKeys.CurrentRoundNumber, currentRoundNumber);


                // add a small delay so that the final entered number is briefly displayed before
moving on
                $('#gorilla')
                  .delay(500)
                  .queue(function(){
                    machine.transition(State.Trial);
                    $(this).dequeue();
                  });
              }
            });
          });
      },
      // The onExit function is executed whenever a state is left.  It is the last thing a state will do
      onExit: (machine: stateMachine.Machine) => {
```

```
        // we want to check if we've reached the maximum allowed number of trials.  If we have,
we finish the task
        if(nb_failures === 3) {
           var span = currentDigitLength - 1;
           var finalscore = score;
           gorilla.metric({
              Span: span,
              FinalScore: score;
           });
           finishedFlag = true;
           gorilla.store(GorillaStoreKeys.Finished, finishedFlag);
           machine.transition(State.Finish);
        }
     }
   })

  // this is the state we enter when we have finished the task
  SM.addState(State.Finish, {
     onEnter: (machine: stateMachine.Machine) => {
        gorilla.populate('#gorilla', 'finish', {});
        gorilla.refreshLayout();
        $('#finish-button').one('click', (event: JQueryEventObject) => {
           gorilla.finish();
        });
     }
  })

  // calling this function starts gorilla and the task as a whole
  gorilla.run(function() {
     // if we refreshed the page, having finished the task, we want to go back to the finish screen,
not the instructions
      if(!finishedFlag) {
         SM.start(State.Instructions);
      } else {
         SM.start(State.Finish);
      }
  });
})

function randomIntFromInterval(min,max)
{
   return Math.floor(Math.random()*(max-min+1)+min);
}
```

Templates

start

```
<div class="fullheight fullsize" style="position:relative;">
   <div class="gorilla-zone" data-left="0%" data-right="0%" data-top="30%" data-bottom="30%">
        <p style = "font-size: 30px">Now you will do the task for real.</p>
```

```
        <p style = "font-size: 30px">You should still repeat the digits in the REVERSED order </p>
        <p style = "font-size: 30px">Just do your best!</p>
    </div>
    <div class="gorilla-zone" data-left="0%" data-right="0%" data-top="70%" data-bottom="0%">
      <div class="gorilla-float centered button">
        <button class="btn btn-primary" id="start-button">Start</button>
      </div>
    </div>
</div>
```

instruction

```
<div class="fullheight fullsize" style="position:relative;">
    <div class="gorilla-zone" data-left="0%" data-right="0%" data-top="30%" data-bottom="30%">
        <p style = "font-size: 30px">In this task, you will see a sequence of numbers.</p>
        <p style = "font-size: 30px">Using the number pad on the screen, you should repeat </p>
        <p style = "font-size: 30px"> the digits in the REVERSED order </p>
        <p style = "font-size: 30px">Let's try!</p>
    </div>
    <div class="gorilla-zone" data-left="0%" data-right="0%" data-top="70%" data-bottom="0%">
      <div class="gorilla-float centered button">
        <button class="btn btn-primary" id="start-button">Start</button>
      </div>
    </div>

    <div class="gorilla-zone" data-left="82%" data-right="0%" data-top="10%" data-
bottom="20%">
      <div class="gorilla-float centered">
          <button class="btn btn-lg btn-primary c-answer-button" data-answer="1" style="margin-
bottom: 4px; margin-right: 4px;">1</button>
          <button class="btn btn-lg btn-primary c-answer-button" data-answer="2" style="margin-
bottom: 4px; margin-right: 4px;">2</button>
          <button class="btn btn-lg btn-primary c-answer-button" data-answer="3" style="margin-
bottom: 4px; margin-right: 4px;">3</button>
        <br/>
          <button class="btn btn-lg btn-primary c-answer-button" data-answer="4" style="margin-
bottom: 4px; margin-right: 4px;">4</button>
          <button class="btn btn-lg btn-primary c-answer-button" data-answer="5" style="margin-
bottom: 4px; margin-right: 4px;">5</button>
          <button class="btn btn-lg btn-primary c-answer-button" data-answer="6" style="margin-
bottom: 4px; margin-right: 4px;">6</button>
        <br/>
          <button class="btn btn-lg btn-primary c-answer-button" data-answer="7" style="margin-
bottom: 4px; margin-right: 4px;">7</button>
          <button class="btn btn-lg btn-primary c-answer-button" data-answer="8" style="margin-
bottom: 4px; margin-right: 4px;">8</button>
          <button class="btn btn-lg btn-primary c-answer-button" data-answer="9" style="margin-
bottom: 4px; margin-right: 4px;">9</button>
        <br/>
          <button class="btn btn-lg btn-primary c-answer-button" data-answer="0" style="margin-
right: 4px">0</button>
```

```
        </div>
    </div>
</div>
```

trial

```
<div class="gorilla-zone" data-left="0%" data-right="30%" data-top="15%" data-bottom="40%">
        <div class="gorilla-float centered">
                <div class="numberDisplay s-question" style="font-size: 84px; display:inline-
block"></div>
                <span class="gorilla-fixation-cross" style="font-size: 84px;">+</span>
        </div>
</div>
<div class="gorilla-zone" data-left="70%" data-right="0%" data-top="10%" data-bottom="20%">
    <div class="gorilla-float centered">
        <button class="btn btn-lg btn-primary c-answer-button" data-answer="1" style="margin-
bottom: 4px; margin-right: 4px;">1</button>
        <button class="btn btn-lg btn-primary c-answer-button" data-answer="2" style="margin-
bottom: 4px; margin-right: 4px;">2</button>
        <button class="btn btn-lg btn-primary c-answer-button" data-answer="3" style="margin-
bottom: 4px; margin-right: 4px;">3</button>
    <br/>
        <button class="btn btn-lg btn-primary c-answer-button" data-answer="4" style="margin-
bottom: 4px; margin-right: 4px;">4</button>
        <button class="btn btn-lg btn-primary c-answer-button" data-answer="5" style="margin-
bottom: 4px; margin-right: 4px;">5</button>
        <button class="btn btn-lg btn-primary c-answer-button" data-answer="6" style="margin-
bottom: 4px; margin-right: 4px;">6</button>
    <br/>
        <button class="btn btn-lg btn-primary c-answer-button" data-answer="7" style="margin-
bottom: 4px; margin-right: 4px;">7</button>
        <button class="btn btn-lg btn-primary c-answer-button" data-answer="8" style="margin-
bottom: 4px; margin-right: 4px;">8</button>
        <button class="btn btn-lg btn-primary c-answer-button" data-answer="9" style="margin-
bottom: 4px; margin-right: 4px;">9</button>
    <br/>
        <button class="btn btn-lg btn-primary c-answer-button" data-answer="0" style="margin-
right: 4px">0</button>
        </div>
</div>
<div class="gorilla-zone" data-left="0%" data-right="40%" data-top="60%" data-bottom="20%">
    <div class="fullsize">
      <div class="gorilla-content-feedback feedback-correct gorilla-float centered" hidden>
        <span class="fa fa-check"></span>
      </div>
      <div class="gorilla-content-feedback feedback-incorrect gorilla-float centered" hidden>
        <span class="fa fa-times"></span>
      </div>
    </div>
</div>
```

finish

```
<div class="fullheight fullsize" style="position:relative;">
   <div class="gorilla-zone" data-left="0%" data-right="0%" data-top="0%" data-bottom="30%">
      <div class="gorilla-float content">
         <h1>Finished</h1>
         <h4>The task is now complete</h4>
         <h4>Thank you for participating</h4>
      </div>
   </div>
   <div class="gorilla-zone" data-left="0%" data-right="0%" data-top="70%" data-bottom="0%">
      <div class="gorilla-float centered button">
         <button class="btn btn-primary" id="finish-button">Finish</button>
      </div>
   </div>
</div>
```

styling

```
@import "/style/style.less";

.c-answer-button {
   font-size:64px;
   padding-top: 0px;
   padding-bottom: 0px;
   padding-left: 3px;
   padding-right: 3px;
}

.s-fixation {
         display:block;
         position:relative;
         top:30%;
         margin-top:0;
         font-size: 200px;
         text-align:center;
}

.s-question-container {
   display:block;
   position:relative;
   top:30%;
   margin-top:0;
         text-align:left;
}

.s-buttons-container {
         text-align:center;
}

.s-answer-green {
```

```
            .btn-success();
}

.s-answer-red {
            .btn-danger();
}
```

Metrics

```
return {
   "Training Target" : function(m) {
      return m.results_data["trainingtarget"];
   },
   "Training Response" : function(m) {
      return m.results_data["trainingresponse"];
   },
   "Total Training Correct": function(m) {
      return m.results_data["totaltrainingcorrect"];
   },
   "Training Correct?": function(m) {
      return m.results_data["trainingcorrect"];
   },
   "Training Score": function(m) {
      return m.results_data["trainingScore"];
   },
   "Target": function(m){
      return m.results_data["target"];
   },
   "Response": function(m) {
      return m.results_data["response"];
   },
   "Total correct": function(m) {
      return m.results_data["totalCorrect"];
   },
   "Correct?": function(m) {
      return m.results_data["correct"];
   },
   "Score": function(m) {
      return m.results_data["Score"];
   },
   "Final Score": function(m) {
      return m.results_data["FinalScore"];
   },
   "Span": function(m) {
      return m.results_data["Span"];
   },
   "Length": function(m) {
      return m.results_data["length"];
   },
};
```