

# Web framework to play with dynamic body pose detection

## Description

The framework uses tensorflow.js/posenet to detect one or more bodies from the browser's camera and emits positional data for each body's body parts of which there are 17. The framework can use either of two model architectures from posenet: 'MobileNetV1' (default) or 'ResNet50'. ResNet50 is more precise, but also slower and consumes more resources. The framework can be configured to detect either a single body or multiple bodies in the camera stream. When one or more bodies are detected an array of body data is emitted to all listeners.

## Files and folders

The framework is found 'lib/bodydetection.js'  
Sketches that exemplify use of the framework are found in 'sketches'

## Important classes and objects

The important classes and objects in the framework are:

### *Class BodyStream*

BodyStream loads and sets up the model by hooking it up to the webcam via the video element. When first setup and started it will emit data on bodies found in snapshots of the videostream continuously via the event 'bodiesDetected'.

### *Class Bodies*

Bodies contain all data and methods with regard to bodies detected in one video snapshot.

### *Class Body*

Body contain all data and methods with regard to a single body detected in a video snapshot, i.e. body parts and confidence score. The Body class also has methods for relating body parts to each other (e.g. distance between them).

### *Class BodyPart*

BodyPart contain all data and methods with regard to a single bodypart (e.g. left knee), which is position, speed and confidence score.

### *Object bodyParts*

The object bodyParts enumerates all body parts. When body parts referenced we should use the names in bodyParts, e.g. 'bodyParts.leftFoot'

## Usage

To setup body detection instantiate a new object of the class 'BodyStream':

```
const bodies = new BodyStream ({
  posenet: posenet,
  architecture: modelArchitecture.MobileNetV1,
  detectionType: detectionType.singleBody,
  videoElement: document.getElementById('video'),
  samplingRate: 250})
```

The actual detection is started by calling BodyStream.start and it stops when timeout runs out:

```
BodyStream.start(timeout)
```

If 'start' is called with no timeout it will run indefinitely. The body detection can also be stopped by calling:

```
BodyStream.stop ()
```

To get a list of bodies when bodies are detected listen to the event 'bodiesDetected':

```
bodies.addEventListener('bodiesDetected', (e) => {
  bodies = e.detail.bodies
})
```

The list of bodies is accessible from e.detail.bodies and is of class 'Bodies'

Bodies.getNumOfBodies() returns the number of bodies detected.

Bodies.getBodyAt(index) retrieves a particular body of the class 'Body'

A particular body part can be retrieved by calling Body.bodyPart(bodyPartName). If we want to retrieve data for the right knee, we can write:

```
body.getBodyPart(bodyParts.rightKnee)
```

The distance between two bodyparts can be retrieved by calling:

```
Body.getDistanceBetweenBodyParts(bodyParts.leftWrist,
bodyParts.rightWrist)
```