```
const myTalk = new Talk( {

  title: 'Teach programming like a boss',
  speaker: new Speaker( 'Israel Gutiérrez', '@gootyfer' ),
  date: 'Oct. 20th 2016, 19:00h',
  location: 'Redradix offices',
  audience: programmers.filter(

    programmer => programmer.passions.includes( 'Teaching' )

  )

} );
```

```
myTalk.speaker.describe();
// Very much trainer, very little programmer
```

```
myTalk.speaker.getPassions();
// ['Education innovation', 'Programming']
```

```
myTalk.speaker.getLatestProjects();
// Technology-Enhanced Learning researcher
// Ironhack
// h4ckademy
// Talentum Empleo
// Adalab
```

```
myTalk.audience.ask([
  'Do you teach?',
  'Do you think you teach as you were taught?'
]);
```

```
myTalk.title =
  '6 ways to improve your (programming) teaching practice';
```

```
myTalk.waysToImproveYourTeachingPractice[1] =
  'Focus on the learner, not the content';
```

```
it('should focus on the learner, not the content', () => {

  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.increase(theProgrammingTeacher.focus, 'learner');
  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.decrease(theProgrammingTeacher.focus, 'content');

  expect(theProgrammingTeacher.exploreGroupNeedsBeforeTheCourse)
    .to.be.true;
  expect(theProgrammingTeacher.adaptContentToTheGroup)
    .to.be.true;
  expect(theProgrammingTeacher.defineLearningObjectives)
    .to.be.true;

  expect(theProgrammingTeacher.collectFeedback)
    .to.exist;

});
```

```
it('should focus on the learner, not the content', () => {

  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.increase(theProgrammingTeacher.focus, 'learner');
  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.decrease(theProgrammingTeacher.focus, 'content');

  expect(theProgrammingTeacher.exploreGroupNeedsBeforeTheCourse)
    .to.be.true;
  expect(theProgrammingTeacher.adaptContentToTheGroup)
    .to.be.true;
  expect(theProgrammingTeacher.defineLearningObjectives)
    .to.be.true;

  expect(theProgrammingTeacher.collectFeedback)
    .to.exist;

});
```

```
it('should focus on the learner, not the content', () => {

  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.increase(theProgrammingTeacher.focus, 'learner');
  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.decrease(theProgrammingTeacher.focus, 'content');

  expect(theProgrammingTeacher.exploreGroupNeedsBeforeTheCourse)
    .to.be.true;
  expect(theProgrammingTeacher.adaptContentToTheGroup)
    .to.be.true;
  expect(theProgrammingTeacher.defineLearningObjectives)
    .to.be.true;

  expect(theProgrammingTeacher.collectFeedback)
    .to.exist;

});
```

```
it('should focus on the learner, not the content', () => {

  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.increase(theProgrammingTeacher.focus, 'learner');
  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.decrease(theProgrammingTeacher.focus, 'content');

  expect(theProgrammingTeacher.exploreGroupNeedsBeforeTheCourse)
    .to.be.true;
  expect(theProgrammingTeacher.adaptContentToTheGroup)
    .to.be.true;
  expect(theProgrammingTeacher.defineLearningObjectives)
    .to.be.true;

  expect(theProgrammingTeacher.collectFeedback)
    .to.exist;

});
```

```
myTalk.waysToImproveYourTeachingPractice[2] =
   'Force a methodology shift';
```

```
it('should force a methodology shift', () => {

  expect(theProgrammingTeacher.talkingTime)
    .to.be.at.most(TWENTY_PERCENT);
  expect(theProgrammingTeacher.methodology)
    .to.include('practical exercises');
  expect(theProgrammingTeacher.practiceTime)
    .to.be.at.least(FIFTY_PERCENT);

  expect(theProgrammingTeacher.attitude)
    .to.be.equal('humble');
  expect(theProgrammingTeacher.attitude)
    .not.to.be.equal('fear to make mistakes');

  expect(theProgrammingTeacher.methodology)
    .to.include('flipped classroom');
  expect(theProgrammingTeacher.methodology)
    .to.include('project based learning');

});
```

```
it('should force a methodology shift', () => {

  expect(theProgrammingTeacher.talkingTime)
    .to.be.at.most(TWENTY_PERCENT);
  expect(theProgrammingTeacher.methodology)
    .to.include('practical exercises');
  expect(theProgrammingTeacher.practiceTime)
    .to.be.at.least(FIFTY_PERCENT);

  expect(theProgrammingTeacher.attitude)
    .to.be.equal('humble');
  expect(theProgrammingTeacher.attitude)
    .not.to.be.equal('fear to make mistakes');

  expect(theProgrammingTeacher.methodology)
    .to.include('flipped classroom');
  expect(theProgrammingTeacher.methodology)
    .to.include('project based learning');

});
```

```
it('should force a methodology shift', () => {

  expect(theProgrammingTeacher.talkingTime)
    .to.be.at.most(TWENTY_PERCENT);
  expect(theProgrammingTeacher.methodology)
    .to.include('practical exercises');
  expect(theProgrammingTeacher.practiceTime)
    .to.be.at.least(FIFTY_PERCENT);

  expect(theProgrammingTeacher.attitude)
    .to.be.equal('humble');
  expect(theProgrammingTeacher.attitude)
    .not.to.be.equal('fear to make mistakes');

  expect(theProgrammingTeacher.methodology)
    .to.include('flipped classroom');
  expect(theProgrammingTeacher.methodology)
    .to.include('project based learning');

});
```

```
it('should force a methodology shift', () => {

  expect(theProgrammingTeacher.talkingTime)
    .to.be.at.most(TWENTY_PERCENT);
  expect(theProgrammingTeacher.methodology)
    .to.include('practical exercises');
  expect(theProgrammingTeacher.practiceTime)
    .to.be.at.least(FIFTY_PERCENT);

  expect(theProgrammingTeacher.attitude)
    .to.be.equal('humble');
  expect(theProgrammingTeacher.attitude)
    .not.to.be.equal('fear to make mistakes');

  expect(theProgrammingTeacher.methodology)
    .to.include('flipped classroom');
  expect(theProgrammingTeacher.methodology)
    .to.include('project based learning');

});
```

```
myTalk.waysToImproveYourTeachingPractice[3] =
   'Change the role of the teacher';
```

```
it('should change his/her role in the learning process', () => {

  expect(theProgrammingTeacher.role)
    .to.not.include('sage on the stage');
  expect(theProgrammingTeacher.role)
    .to.include('guide on the side');

  expect(theCourse.materials)
    .to.be.equal('commodity');

  expect(theProgrammingTeacher.answer)
    .to.be.equal('questions');

  expect(theProgrammingTeacher.actions)
    .to.include('mentoring');

});
```

```
it('should change his/her role in the learning process', () => {

  expect(theProgrammingTeacher.role)
    .to.not.include('sage on the stage');
  expect(theProgrammingTeacher.role)
    .to.include('guide on the side');

  expect(theCourse.materials)
    .to.be.equal('commodity');

  expect(theProgrammingTeacher.answer)
    .to.be.equal('questions');

  expect(theProgrammingTeacher.actions)
    .to.include('mentoring');

});
```

```
it('should change his/her role in the learning process', () => {

  expect(theProgrammingTeacher.role)
    .to.not.include('sage on the stage');
  expect(theProgrammingTeacher.role)
    .to.include('guide on the side');

  expect(theCourse.materials)
    .to.be.equal('commodity');

  expect(theProgrammingTeacher.answer)
    .to.be.equal('questions');

  expect(theProgrammingTeacher.actions)
    .to.include('mentoring');

});
```

```
it('should change his/her role in the learning process', () => {

  expect(theProgrammingTeacher.role)
    .to.not.include('sage on the stage');
  expect(theProgrammingTeacher.role)
    .to.include('guide on the side');

  expect(theCourse.materials)
    .to.be.equal('commodity');

  expect(theProgrammingTeacher.answer)
    .to.be.equal('questions');

  expect(theProgrammingTeacher.actions)
    .to.include('mentoring');

});
```

```
it('should change his/her role in the learning process', () => {

  expect(theProgrammingTeacher.role)
    .to.not.include('sage on the stage');
  expect(theProgrammingTeacher.role)
    .to.include('guide on the side');

  expect(theCourse.materials)
    .to.be.equal('commodity');

  expect(theProgrammingTeacher.answer)
    .to.be.equal('questions');

  expect(theProgrammingTeacher.actions)
    .to.include('mentoring');

});
```

```
myTalk.waysToImproveYourTeachingPractice[4] =
   'Empower the role of the peers';
```

```
it('should empower the role of the peers in the learning process', () => {

  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.change(theGroupOfLearners, 'role');

  expect(theProgrammingTeacher.activities)
    .to.include('group');

  const groupActivities = theProgrammingTeacher.groupActivities;
  expect(groupActivities)
    .to.include('group changes');
  expect(groupActivities)
    .to.include('groups created by the teacher');

  expect(theProgrammingTeacher.doGroupActivities.bind(theProgrammingTeacher))
    .to.increase(theGroupOfLearners, 'motivation');

});
```

```
it('should empower the role of the peers in the learning process', () => {

  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.change(theGroupOfLearners, 'role');

  expect(theProgrammingTeacher.activities)
    .to.include('group');

  const groupActivities = theProgrammingTeacher.groupActivities;
  expect(groupActivities)
    .to.include('group changes');
  expect(groupActivities)
    .to.include('groups created by the teacher');

  expect(theProgrammingTeacher.doGroupActivities.bind(theProgrammingTeacher))
    .to.increase(theGroupOfLearners, 'motivation');

});
```

```
it('should empower the role of the peers in the learning process', () => {

  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.change(theGroupOfLearners, 'role');

  expect(theProgrammingTeacher.activities)
    .to.include('group');

  const groupActivities = theProgrammingTeacher.groupActivities;
  expect(groupActivities)
    .to.include('group changes');
  expect(groupActivities)
    .to.include('groups created by the teacher');

  expect(theProgrammingTeacher.doGroupActivities.bind(theProgrammingTeacher))
    .to.increase(theGroupOfLearners, 'motivation');

});
```

```
it('should empower the role of the peers in the learning process', () => {

  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.change(theGroupOfLearners, 'role');

  expect(theProgrammingTeacher.activities)
    .to.include('group');

  const groupActivities = theProgrammingTeacher.groupActivities;
  expect(groupActivities)
    .to.include('group changes');
  expect(groupActivities)
    .to.include('groups created by the teacher');

  expect(theProgrammingTeacher.doGroupActivities.bind(theProgrammingTeacher))
    .to.increase(theGroupOfLearners, 'motivation');

});
```

```
it('should empower the role of the peers in the learning process', () => {

  expect(theProgrammingTeacher.action.bind(theProgrammingTeacher))
    .to.change(theGroupOfLearners, 'role');

  expect(theProgrammingTeacher.activities)
    .to.include('group');

  const groupActivities = theProgrammingTeacher.groupActivities;
  expect(groupActivities)
    .to.include('group changes');
  expect(groupActivities)
    .to.include('groups created by the teacher');

  expect(theProgrammingTeacher.doGroupActivities.bind(theProgrammingTeacher))
    .to.increase(theGroupOfLearners, 'motivation');

});
```

```
myTalk.waysToImproveYourTeachingPractice[5] =
  'Assessment is feedback to improve';
```

```
it('should assess learners in order to improve learning', () => {

  expect(theProgrammingTeacher.doAssessment.bind(theProgrammingTeacher))
    .to.change(theGroupOfLearners, 'awareness');

  expect(theProgrammingTeacher.assessment.objective)
    .to.equal('learn');
  expect(theProgrammingTeacher.assessment.objective)
    .not.to.equal('a mark');

  expect(theProgrammingTeacher.assessment.feedback)
    .to.equal('quick');

  expect(theProgrammingTeacher.assessment.result)
    .to.change(theCourse, 'content');

  expect(theProgrammingTeacher.assessment.preparation)
    .to.include('learning objectives');

});
```

```
it('should assess learners in order to improve learning', () => {

  expect(theProgrammingTeacher.doAssessment.bind(theProgrammingTeacher))
    .to.change(theGroupOfLearners, 'awareness');

  expect(theProgrammingTeacher.assessment.objective)
    .to.equal('learn');
  expect(theProgrammingTeacher.assessment.objective)
    .not.to.equal('a mark');

  expect(theProgrammingTeacher.assessment.feedback)
    .to.equal('quick');

  expect(theProgrammingTeacher.assessment.result)
    .to.change(theCourse, 'content');

  expect(theProgrammingTeacher.assessment.preparation)
    .to.include('learning objectives');

});
```

```
it('should assess learners in order to improve learning', () => {

  expect(theProgrammingTeacher.doAssessment.bind(theProgrammingTeacher))
    .to.change(theGroupOfLearners, 'awareness');

  expect(theProgrammingTeacher.assessment.objective)
    .to.equal('learn');
  expect(theProgrammingTeacher.assessment.objective)
    .not.to.equal('a mark');

  expect(theProgrammingTeacher.assessment.feedback)
    .to.equal('quick');

  expect(theProgrammingTeacher.assessment.result)
    .to.change(theCourse, 'content');

  expect(theProgrammingTeacher.assessment.preparation)
    .to.include('learning objectives');

});
```

```
it('should assess learners in order to improve learning', () => {

  expect(theProgrammingTeacher.doAssessment.bind(theProgrammingTeacher))
    .to.change(theGroupOfLearners, 'awareness');

  expect(theProgrammingTeacher.assessment.objective)
    .to.equal('learn');
  expect(theProgrammingTeacher.assessment.objective)
    .not.to.equal('a mark');

  expect(theProgrammingTeacher.assessment.feedback)
    .to.equal('quick');

  expect(theProgrammingTeacher.assessment.result)
    .to.change(theCourse, 'content');

  expect(theProgrammingTeacher.assessment.preparation)
    .to.include('learning objectives');

});
```

```
it('should assess learners in order to improve learning', () => {

  expect(theProgrammingTeacher.doAssessment.bind(theProgrammingTeacher))
    .to.change(theGroupOfLearners, 'awareness');

  expect(theProgrammingTeacher.assessment.objective)
    .to.equal('learn');
  expect(theProgrammingTeacher.assessment.objective)
    .not.to.equal('a mark');

  expect(theProgrammingTeacher.assessment.feedback)
    .to.equal('quick');

  expect(theProgrammingTeacher.assessment.result)
    .to.change(theCourse, 'content');

  expect(theProgrammingTeacher.assessment.preparation)
    .to.include('learning objectives');

});
```

```
it('should assess learners in order to improve learning', () => {

  expect(theProgrammingTeacher.doAssessment.bind(theProgrammingTeacher))
    .to.change(theGroupOfLearners, 'awareness');

  expect(theProgrammingTeacher.assessment.objective)
    .to.equal('learn');
  expect(theProgrammingTeacher.assessment.objective)
    .not.to.equal('a mark');

  expect(theProgrammingTeacher.assessment.feedback)
    .to.equal('quick');

  expect(theProgrammingTeacher.assessment.result)
    .to.change(theCourse, 'content');

  expect(theProgrammingTeacher.assessment.preparation)
    .to.include('learning objectives');

});
```

```
myTalk.waysToImproveYourTeachingPractice[6] =
   'To motivate > To teach';
```

```
it('should prioritize motivating learners over teaching them', () => {

  const motivation = theProgrammingTeacher.priorities['motivate learners'];
  const teaching = theProgrammingTeacher.priorities['teaching'];
  expect(motivation)
    .to.be.above(teaching);

  expect(theProgrammingTeacher.role)
    .to.include('community manager');

  expect(theGroupOfLearners.feelings)
    .to.include('main character');
  expect(theProgrammingTeacher.activityTopics)
    .to.include('learners interests');
  expect(theProgrammingTeacher.preparation)
    .to.include('little details');

  expect(theProgrammingTeacher.actions)
    .to.include('suprise learners');

});
```

```
it('should prioritize motivating learners over teaching them', () => {

  const motivation = theProgrammingTeacher.priorities['motivate learners'];
  const teaching = theProgrammingTeacher.priorities['teaching'];
  expect(motivation)
    .to.be.above(teaching);

  expect(theProgrammingTeacher.role)
    .to.include('community manager');

  expect(theGroupOfLearners.feelings)
    .to.include('main character');
  expect(theProgrammingTeacher.activityTopics)
    .to.include('learners interests');
  expect(theProgrammingTeacher.preparation)
    .to.include('little details');

  expect(theProgrammingTeacher.actions)
    .to.include('suprise learners');

});
```

```
it('should prioritize motivating learners over teaching them', () => {

  const motivation = theProgrammingTeacher.priorities['motivate learners'];
  const teaching = theProgrammingTeacher.priorities['teaching'];
  expect(motivation)
    .to.be.above(teaching);

  expect(theProgrammingTeacher.role)
    .to.include('community manager');

  expect(theGroupOfLearners.feelings)
    .to.include('main character');
  expect(theProgrammingTeacher.activityTopics)
    .to.include('learners interests');
  expect(theProgrammingTeacher.preparation)
    .to.include('little details');

  expect(theProgrammingTeacher.actions)
    .to.include('suprise learners');

});
```

```
it('should prioritize motivating learners over teaching them', () => {

  const motivation = theProgrammingTeacher.priorities['motivate learners'];
  const teaching = theProgrammingTeacher.priorities['teaching'];
  expect(motivation)
    .to.be.above(teaching);

  expect(theProgrammingTeacher.role)
    .to.include('community manager');

  expect(theGroupOfLearners.feelings)
    .to.include('main character');
  expect(theProgrammingTeacher.activityTopics)
    .to.include('learners interests');
  expect(theProgrammingTeacher.preparation)
    .to.include('little details');

  expect(theProgrammingTeacher.actions)
    .to.include('suprise learners');

});
```

```
it('should prioritize motivating learners over teaching them', () => {

  const motivation = theProgrammingTeacher.priorities['motivate learners'];
  const teaching = theProgrammingTeacher.priorities['teaching'];
  expect(motivation)
    .to.be.above(teaching);

  expect(theProgrammingTeacher.role)
    .to.include('community manager');

  expect(theGroupOfLearners.feelings)
    .to.include('main character');
  expect(theProgrammingTeacher.activityTopics)
    .to.include('learners interests');
  expect(theProgrammingTeacher.preparation)
    .to.include('little details');

  expect(theProgrammingTeacher.actions)
    .to.include('suprise learners');

});
```

```
➜  teaching-programming-talk git:(master) ✗ npm test
```

```
➜  teaching-programming-talk git:(master) ✗ npm test

> teaching-programming-talk@1.0.0 test /Users/israel/code/teaching-programming-t
alk
> mocha --require babel-register src/**/*.test.js



  the (programming) teacher
    ✓ should focus on the learner, not the content
    ✓ should force a methodology shift
    ✓ should change his/her role in the learning process
    ✓ should empower the role of the peers in the learning process
    ✓ should assess learners in order to improve learning
    ✓ should prioritize motivating learners over teaching them


  6 passing (25ms)
```

```
myTalk.close();
//Thank you!
```