

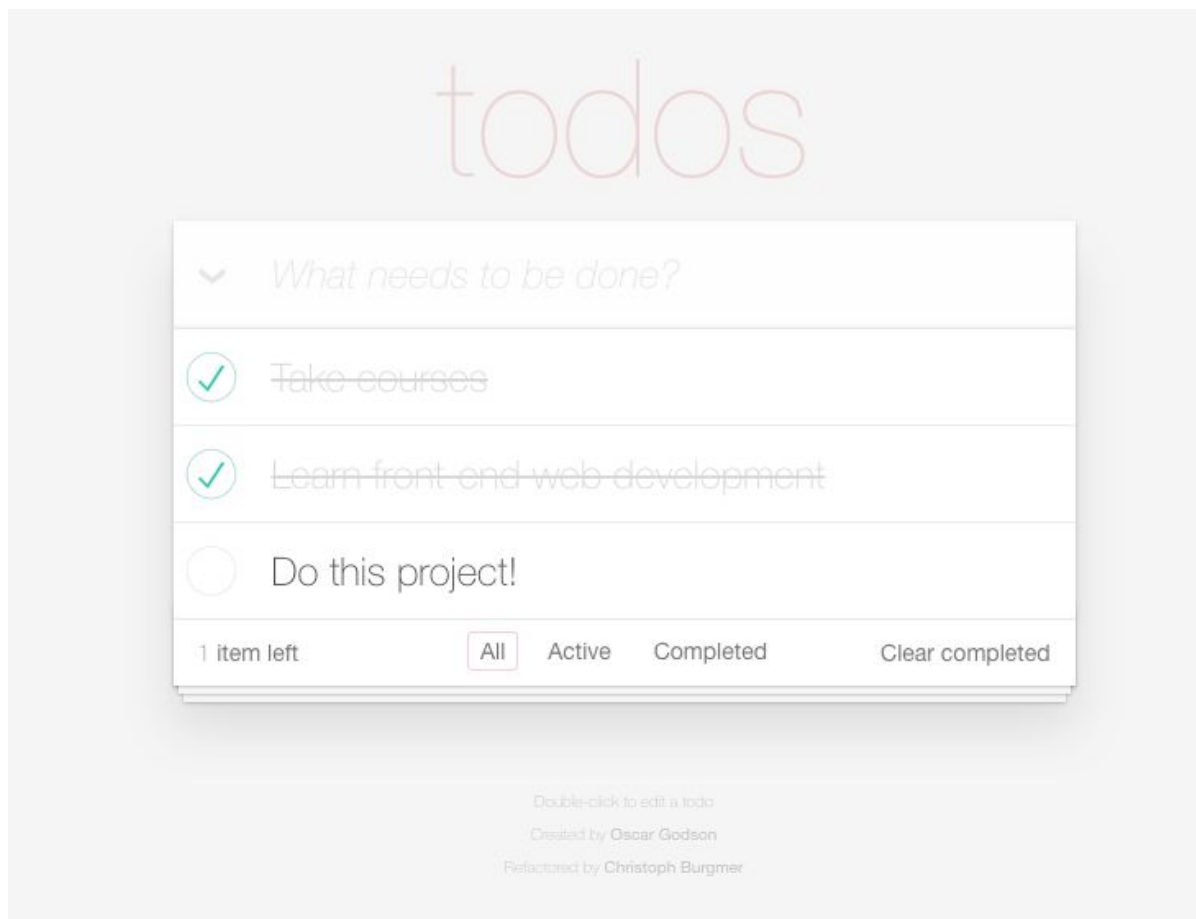
Enhance an existing project

Christine Matta

Overview

The project is a Todo list application that allow the user to do the following:

- Add new Todos
- Modify existing Todos
- Delete Todos
- Mark every separate todo as active or completed
- Mark all todos as active or completed
- Display only all, active or completed Todos
- Clear all completed Todos



Requirements

1. Fix the bugs
2. Add tests
3. Analyze performance
4. Write technical documentation

How to access the Todo App.



1. Write down your Todo task and then press Enter to add a new todo to the todo list
2. You can rename the todo item by double clicking on it and press enter after editing.



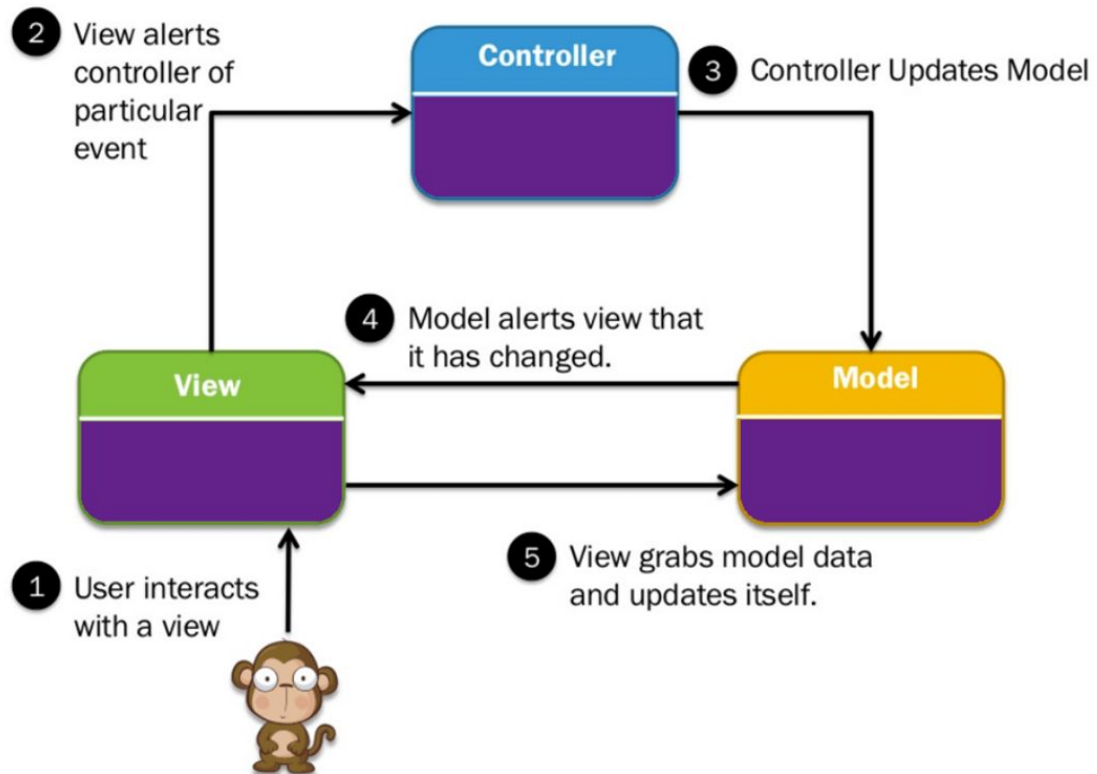
3. When change the status to completed the css style change to dimed label
4. To change the status of active todo to completed or vice versa click this box.
5. To change the status of all active todos to completed or vice versa click this field.
6. You can view how many active Todos are still remaining.
7. This filter buttons have 3 taps All, active and completed, you can filter depending on your desire
8. To remove a toto item from the list , hover on this item and then click the X button
9. To remove all completed todos click this element.

MVC architecture

I. Features of MVC

- Easy and frictionless testability. Highly testable framework
- Offers full control over your HTML as well as your URLs
- Clear separation of logic: Model, View, Controller.
 - Separation of application tasks
 - Business logic
 - UI logic
 - input logic
- Supports for Test Driven Development (TDD)

II. MVC Architecture



Three important MVC the components are:

- Model: It includes all the data and its related logic
- View: Present data to the user or handles user interaction
- Controller: An interface between Model and View components

MODE

Model is responsible for managing the data of the application. It receives user input from the controller. He is responsible for CRUD (create, read, update and delete) operations.

VIEW

View is a presentation of the model in a particular format - in our case displaying All, Active or Completed todos (route). It also lets the user interact with displayed data.

CONTROLLER

Controller responds to the user input from View and performs interactions with Model. It also reads data from Model and passes it to the view.

By using MVC architecture our application works like Single Page Application (SPA) - that means user can interact with todos without reloading webpage.

Bugs fixing

1. Bug which does not allow adding new todos to the list

LOCATION: controller.js *line 95*

```
Controller.prototype.addItem = function (title) {
```

has been changed into

```
Controller.prototype.addItem = function (title) {
```

2. Bug which may leads to potential conflict between duplicate IDs

LOCATION: store.js starting from line 84

```
var newId = "";  
  
var charset = "0123456789";  
  
for (var i = 0; i < 6; i++) {  
  
    newId += charset.charAt(Math.floor(Math.random() * charset.length));  
  
}
```

has been changed into

```
var newId = new Date().getTime();
```

Adding tests

Tests have been created by using Jasmine.

Jasmine is an open source testing framework for JavaScript.

To start the test simply start SpecRunner.html file located in application main directory in test folder.

To create or modify tests open and modify ControllerSpec.js file located in the same directory.

- should show entries on start-up

```
it('should show entries on start-up', function () {  
    // TODO: write test  
    var todo = {id: 42, title: 'my todo', completed: true};  
    setUpModel([ todo ]);  
    subject.setView('/');  
    expect(model.read).toHaveBeenCalled(); // test if the read function is called or not  
    expect(view.render).toHaveBeenCalledWith('showEntries', [ todo ]); //  
});
```

- should show active entries

```
it('should show active entries', function () {  
    // TODO: write test  
    var todo = [{id:42, title: 'my todo', completed: false}, //active  
                {id:43, title: 'my todo 2', completed: true}]; //not active  
    setUpModel(todo);  
    subject.setView('/#/active'); // set active view  
    expect(view.render).toHaveBeenCalledWith('setFilter','active');  
    expect(view.render).toHaveBeenCalledWith('showEntries', todo);  
});
```


- should show completed entries

```
it('should show completed entries', function () {
  // TODO: write test
  var todo = [{id:42, title: 'my todo', completed: false}, //active
               {id:43, title: 'my todo 2', completed: true}]; //not active

  setUpModel(todo);
  subject.setView('#/completed');

  expect(view.render).toHaveBeenCalledWith('setFilter', 'completed');
  expect(view.render).toHaveBeenCalledWith('showEntries', todo);
  console.log(todo)
});
});
```

- should highlight "All" filter by default

```
it('should highlight "All" filter by default', function () {
  // TODO: write test
  setUpModel([{title: 'my todo', completed: true}]);
  subject.setView(''); //home-page
  expect(view.render).toHaveBeenCalledWith('setFilter', '');
});
```

- should highlight "Active" filter when switching to active view

```
it('should highlight "Active" filter when switching to active view', function () {
  // TODO: write test
  setUpModel([{title: 'my todo', completed: true}]);
  subject.setView('#/active');
  expect(view.render).toHaveBeenCalledWith('setFilter', 'active');
});
```

- should toggle all todos to completed

```
describe('toggle all', function () {
  it('should toggle all todos to completed', function () {
    // TODO: write test
    // if all items are completed then will toggle to true i.e. set them to uncompleted
    const todos = [{id:1, title: 'my todo', completed: false},
    {id : 2, title: 'my todo', completed: true}];
    setUpModel(todos);
    subject.setView('');
    view.render.calls.reset();
    model.read.calls.reset();
    view.trigger('toggleAll', {completed: true}); // toggle to true, but all not completed
    expect(model.read).toHaveBeenCalledOnceWith({completed: false}, jasmine.any(Function));
    expect(model.update).toHaveBeenCalledOnceWith(1, { completed: true}, jasmine.any(Function));
    expect(model.update).toHaveBeenCalledOnceWith(2, { completed: true}, jasmine.any(Function));
    // make sure its called as many times as todos length
    expect(model.update).toHaveBeenCalledTimes(todos.length);
  });
});
```

- should add a new todo to the model

```
describe('new todo', function () {
  it('should add a new todo to the model', function () {
    // TODO: write test
    setUpModel([]);

    subject.setView('');
    view.render.calls.reset();
    model.read.calls.reset();

    view.trigger('newTodo', 'a new todo');

    expect(model.create).toHaveBeenCalledOnceWith('a new todo', jasmine.any(Function));
  });
});
```

- should update the view

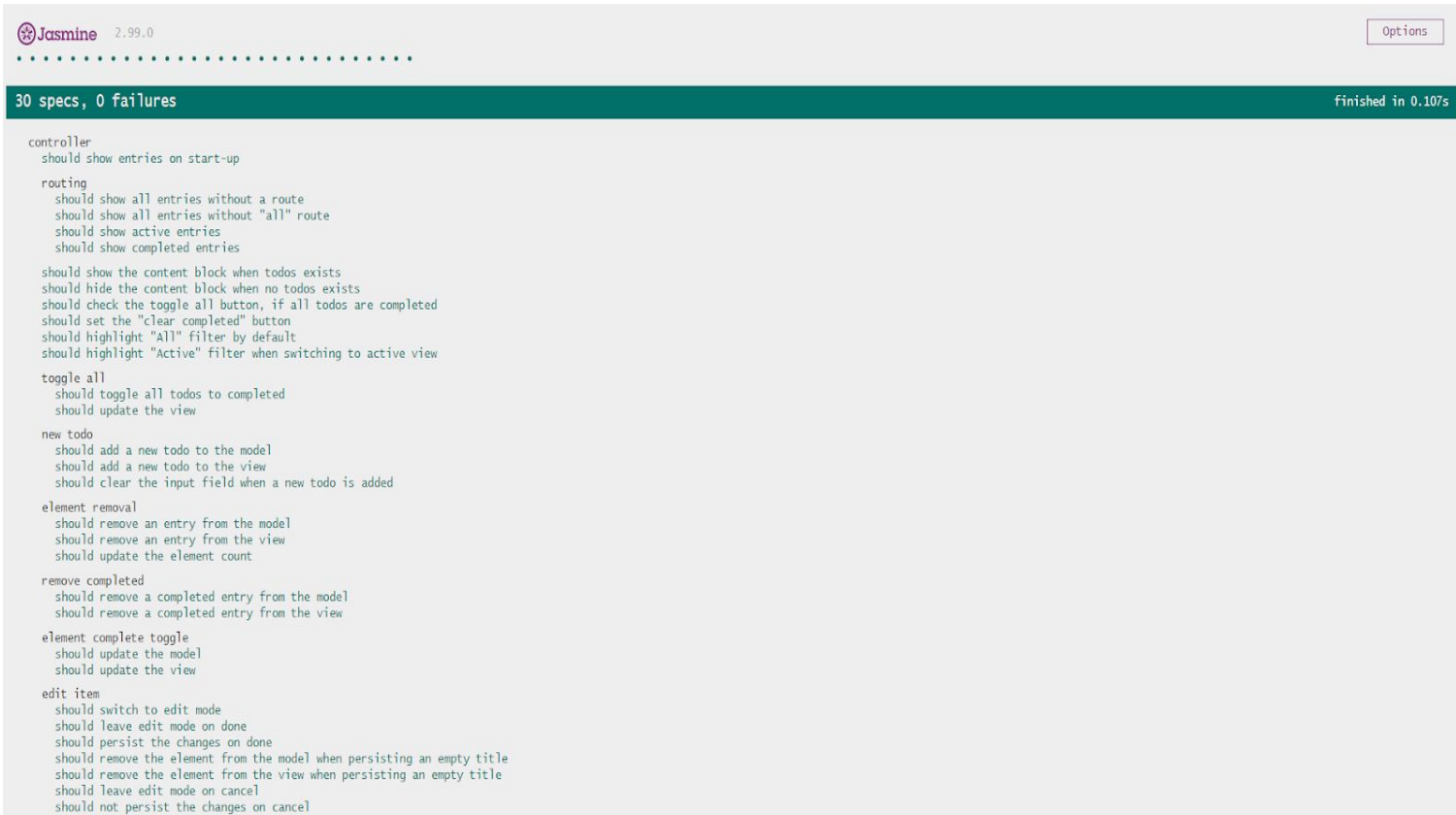
```
it('should update the view', function () {
  // TODO: write test
  setUpModel([{id : 1, title: 'my todo', completed: false},
    {id : 2, title: 'my todo', completed: true}]);
  subject.setView('');
  view.render.calls.reset();
  model.read.calls.reset();
  view.trigger('toggleAll',{completed : false}); // not all are completed
  expect(view.render).toHaveBeenCalled('toggleAll', {checked: false});
});
```

- should remove an entry from the model

```
describe('element removal', function () {
  it('should remove an entry from the model', function () {
    // TODO: write test
    var todo = {id: 42, title: 'my todo', completed: true};
    setUpModel([todo]);

    subject.setView('');
    view.trigger('itemRemove', {id: 42});
    expect(model.remove).toHaveBeenCalled(42, jasmine.any(Function));
  });
});
```

All tests were successful (see image below).



The image is a screenshot of the Jasmine test runner interface. At the top left, it shows the Jasmine logo and version 2.99.0. At the top right, there is an 'Options' button. Below the header, a green bar displays '30 specs, 0 failures' on the left and 'finished in 0.107s' on the right. The main area contains a list of test specifications organized into categories: controller, routing, toggle all, new todo, element removal, remove completed, element complete toggle, and edit item. Each category lists specific test cases that all passed successfully.

```
controller
  should show entries on start-up

routing
  should show all entries without a route
  should show all entries without "all" route
  should show active entries
  should show completed entries

should show the content block when todos exists
should hide the content block when no todos exists
should check the toggle all button, if all todos are completed
should set the "clear completed" button
should highlight "all" filter by default
should highlight "Active" filter when switching to active view

toggle all
  should toggle all todos to completed
  should update the view

new todo
  should add a new todo to the model
  should add a new todo to the view
  should clear the input field when a new todo is added

element removal
  should remove an entry from the model
  should remove an entry from the view
  should update the element count

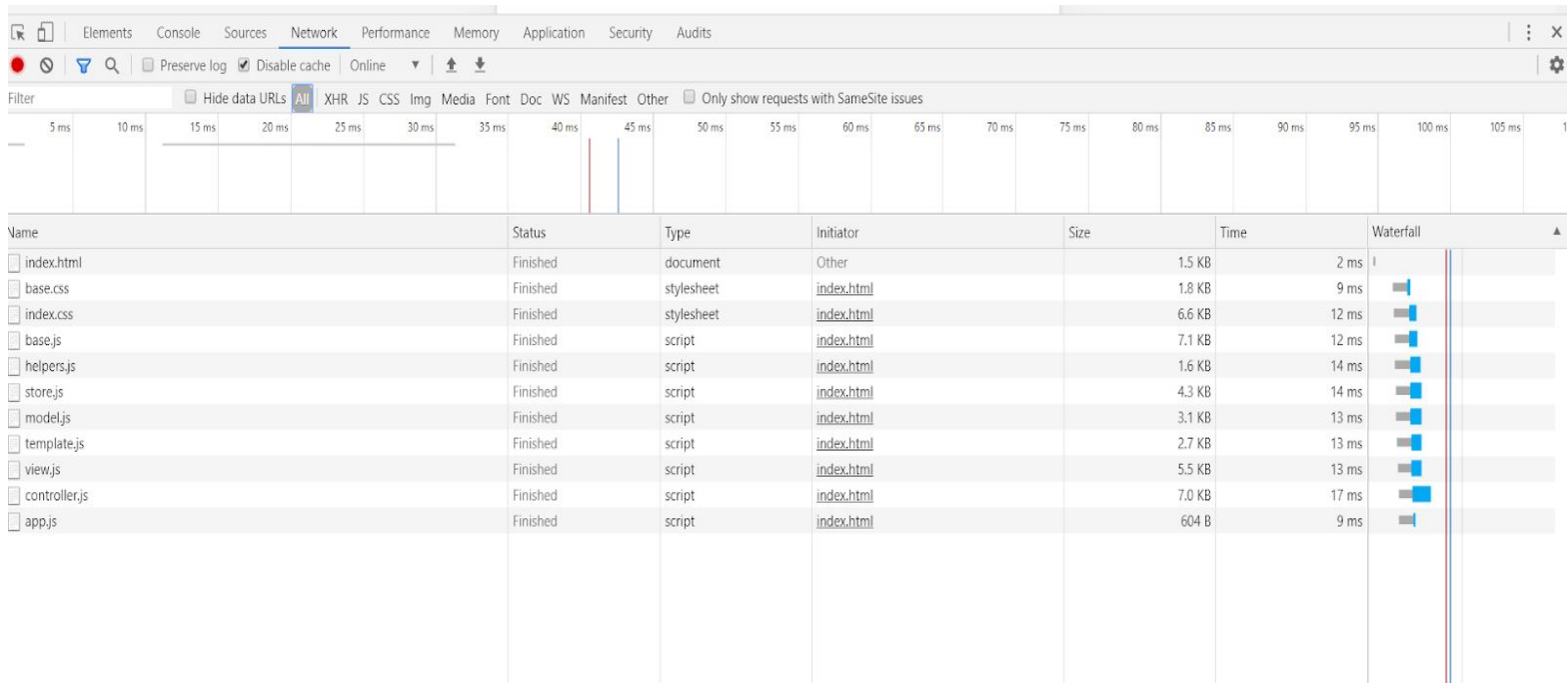
remove completed
  should remove a completed entry from the model
  should remove a completed entry from the view

element complete toggle
  should update the model
  should update the view

edit item
  should switch to edit mode
  should leave edit mode on done
  should persist the changes on done
  should remove the element from the model when persisting an empty title
  should remove the element from the view when persisting an empty title
  should leave edit mode on cancel
  should not persist the changes on cancel
```

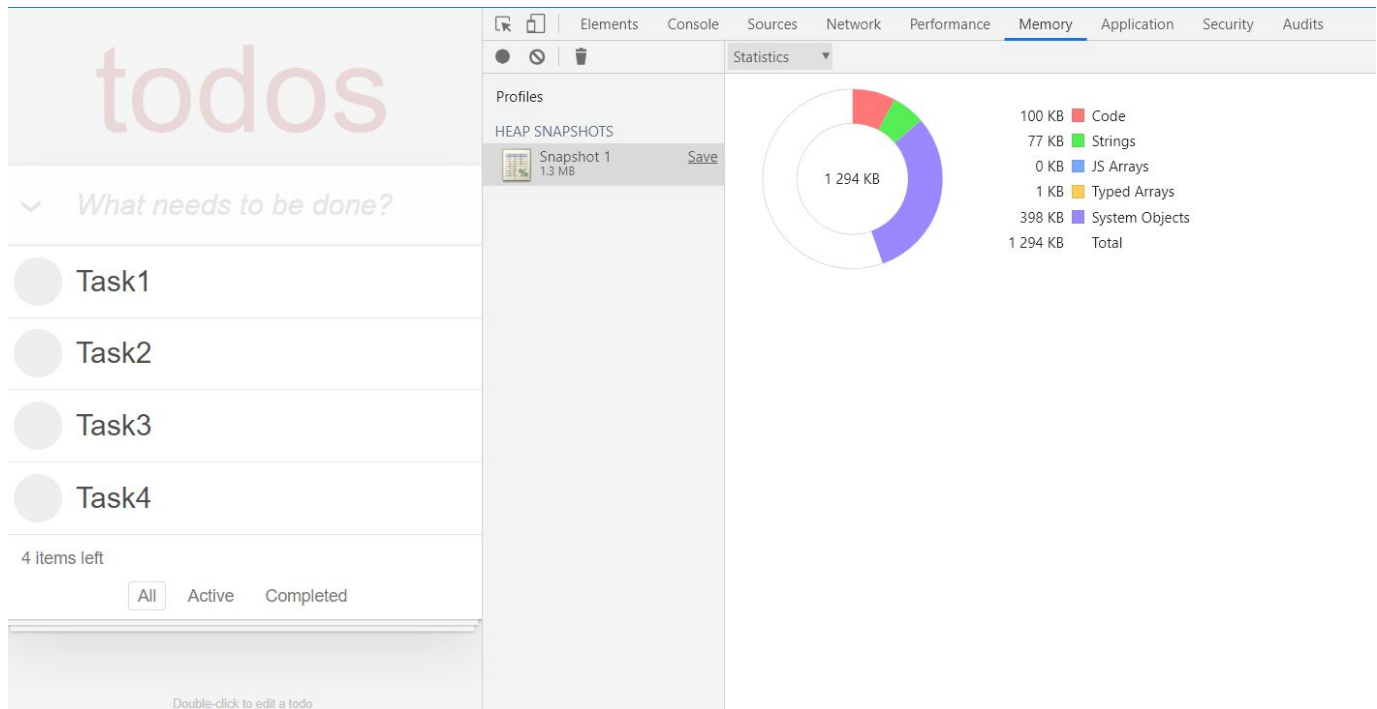
Loading

- PAGE LOADING TIME - 516ms
- DOM CONTENT LOADED TIME - 227 ms
- NUMBER OF REQUESTS SENT - 55
- AMOUNT OF DATA TRANSFERRED - 432 KB



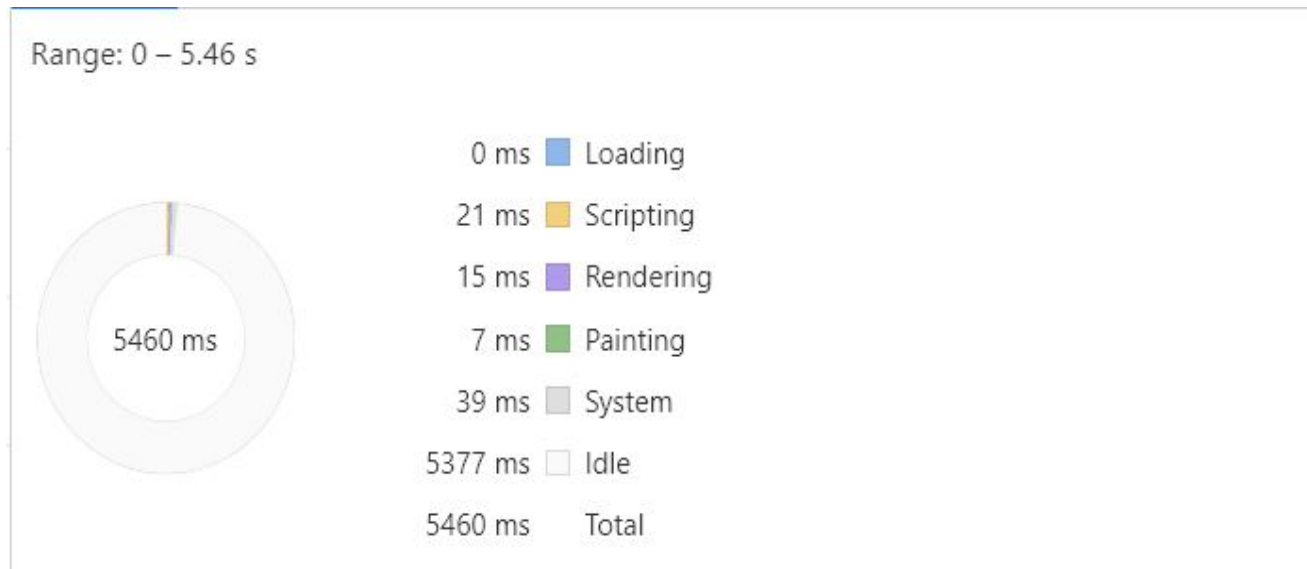
Memory

Total Amount of Memory : 1.294MB

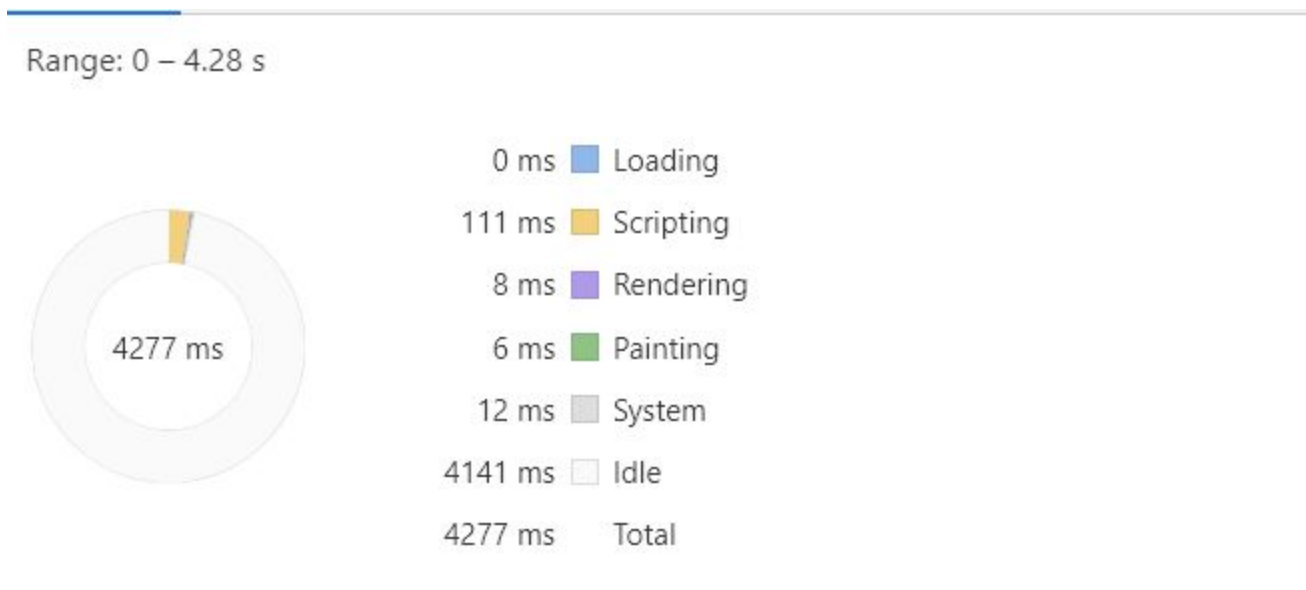


User actions

Adding new task to the list - 5 s

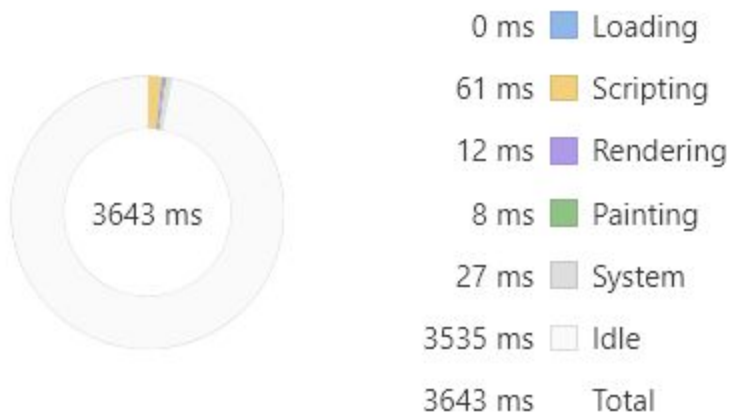


Removing Task from List - 4s



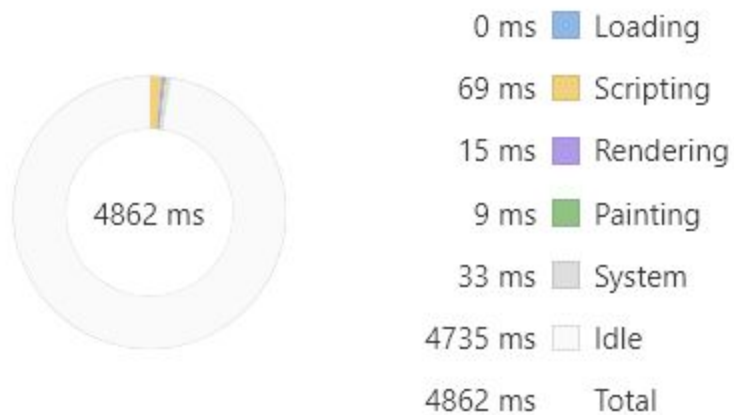
Marking one task as completed - 3s

Range: 0 – 3.64 s

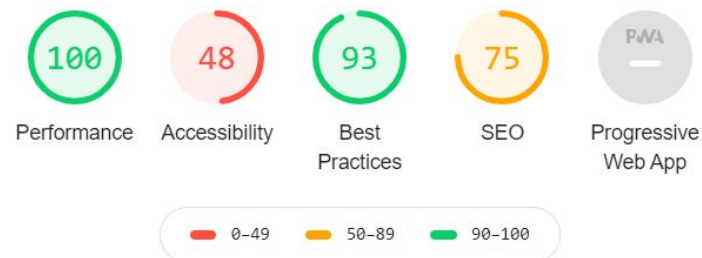


Marking All tasks as completed 4.8s

Range: 0 – 4.86 s



Audit results



Improvements that should be made :

Performance 100

Accessibility 48

- Background and foreground colors do not have a sufficient contrast ratio.
- Form elements do not have associated labels

Best Practices 93

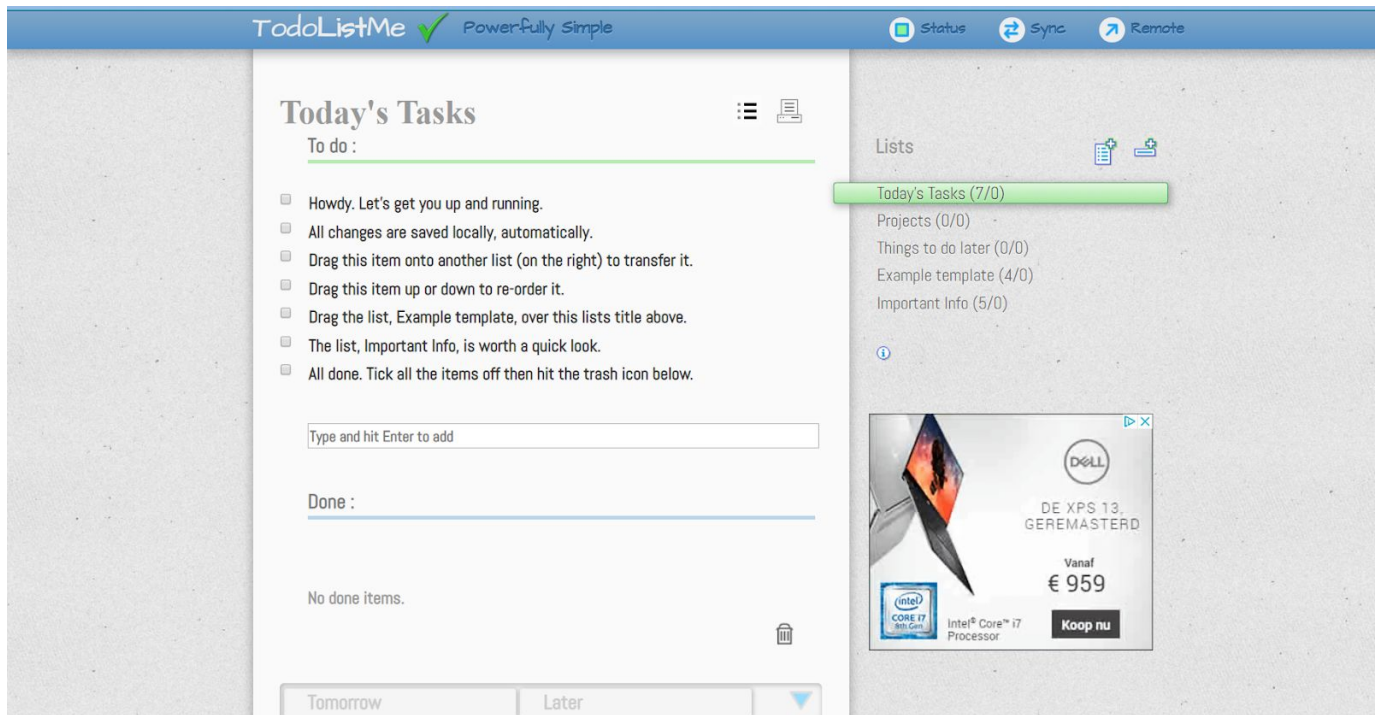
- Browser errors were logged to the console

SEO (search engine optimization)

- Does not have a `<meta name="viewport">` tag with width or initial-scaleNo `<meta name="viewport">` tag found
- Document does not have a meta description

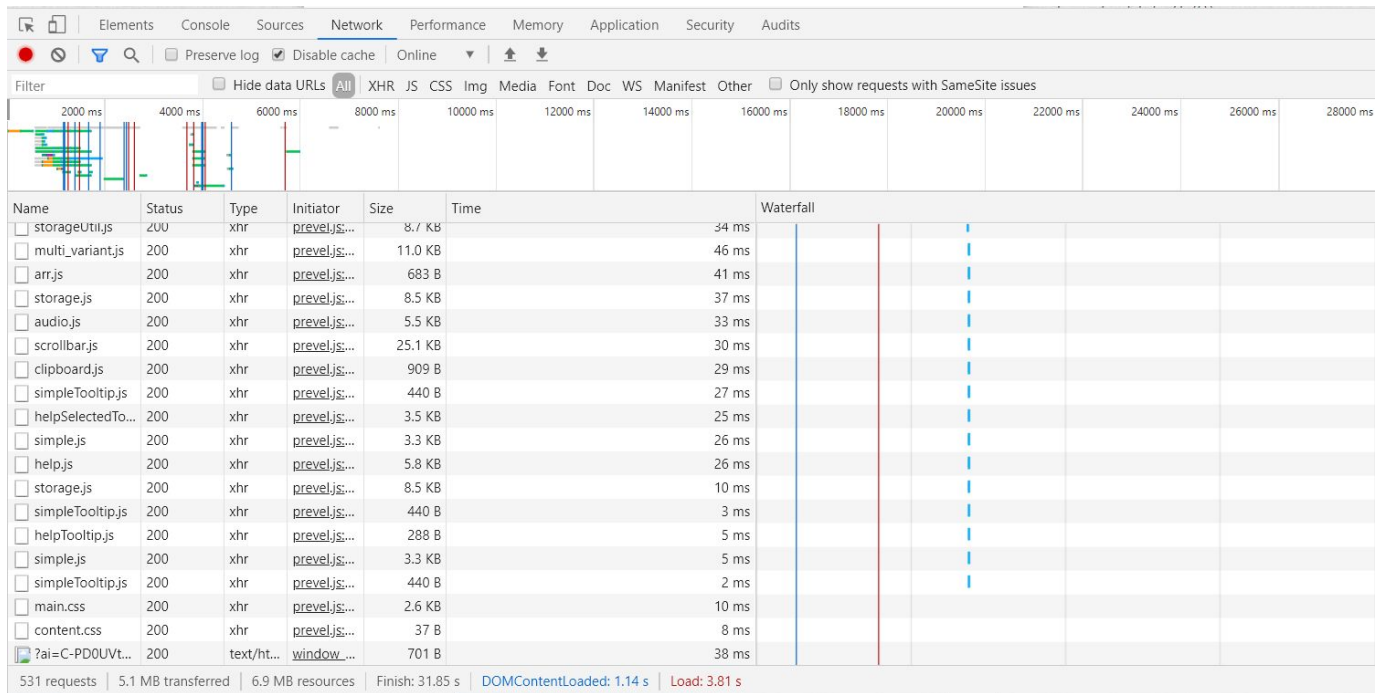
Competitors application is hosted under this link :

<http://todolistme.net/>

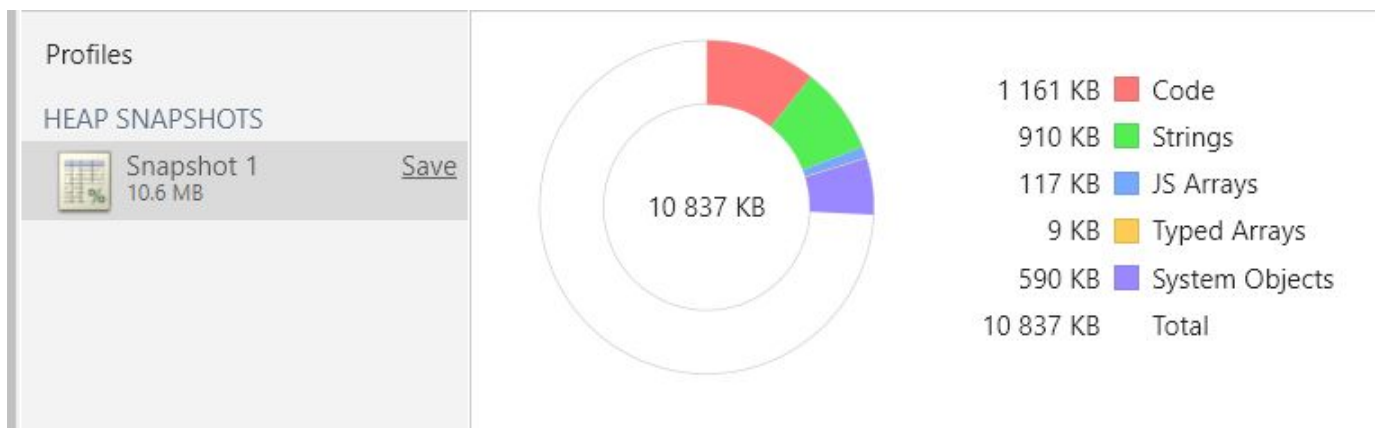


Loading

- PAGE LOADING TIME - 2.56 s
- DOM CONTENT LOADED TIME - 821 ms
- NUMBER OF REQUESTS SENT - 36
- AMOUNT OF DATA TRANSFERRED - 525 KB

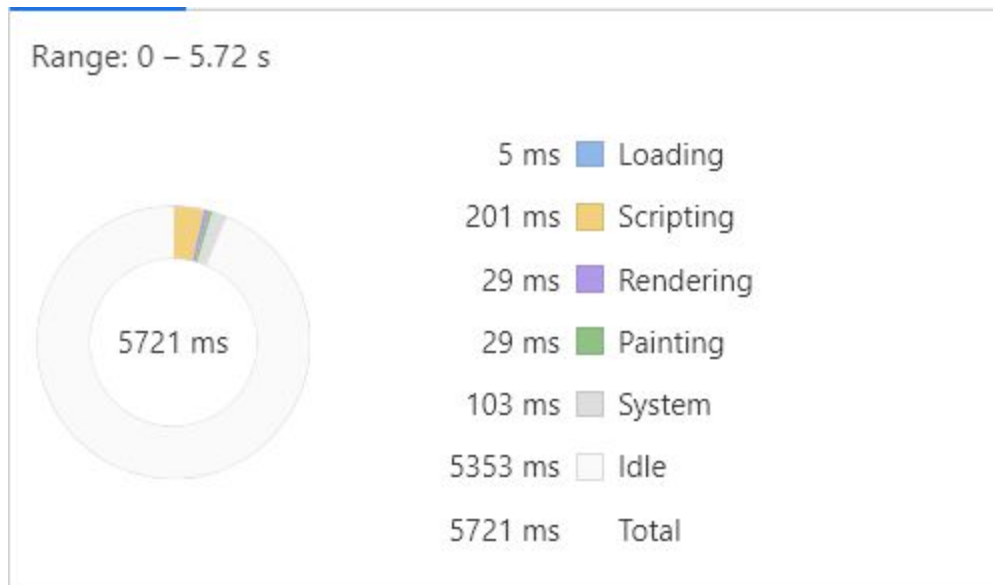


Memory:

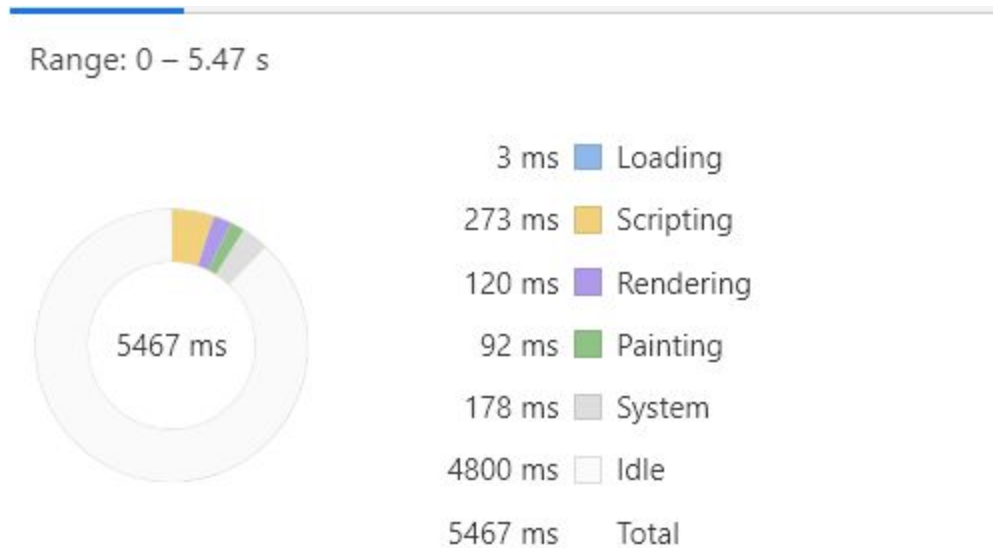


User actions

Adding new task to the list - 5 s

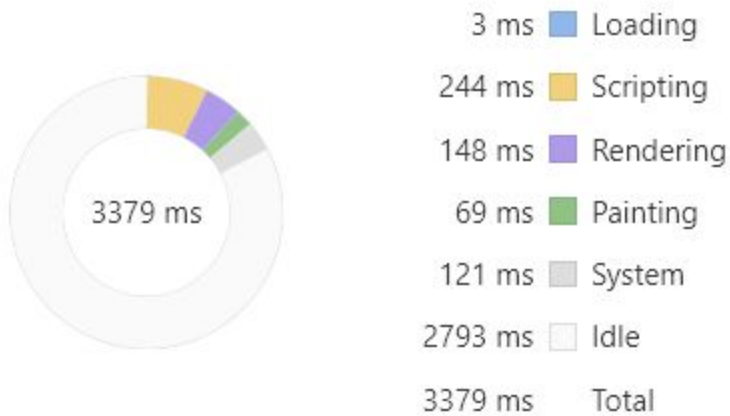


Removing Task from List - 5s

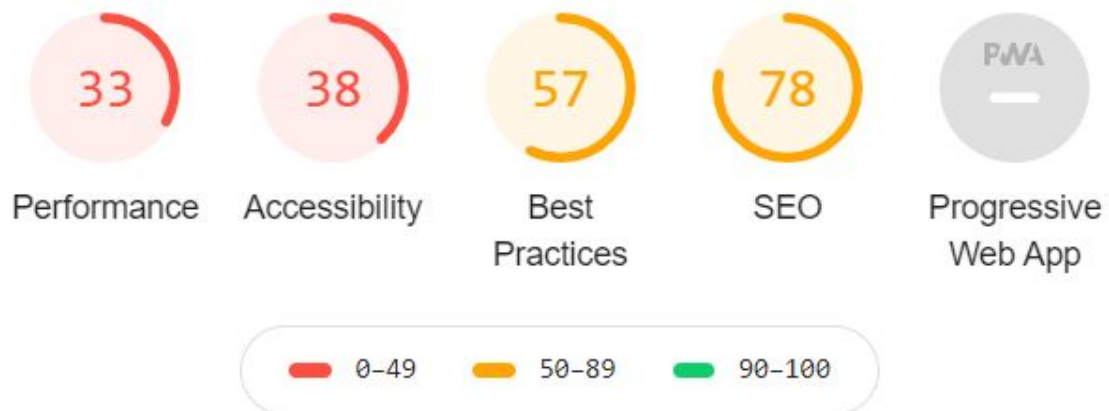


Marking one task completed - 3s

Range: 0 – 3.38 s



Audit results



Improvements that should be made :

- Background and foreground colors do not have a sufficient contrast ratio. Low-contrast text is difficult or impossible for many users to read.
- [id] attributes on the page are not unique. Failing Elements - span#later_number.
- <frame> or <iframe> elements do not have a title. Screen reader users rely on frame titles to describe the contents of frames.
- Image elements do not have [alt] attributes. Informative elements should aim for short, descriptive alternate text.
- Form elements do not have associated labels. Labels ensure that form controls are announced properly by assistive technologies, like screen readers. Failing Elements - input#newtodo.newtodonormal and other 7 inputs.
- <html> element does not have a [lang] attribute. If a page doesn't specify a lang attribute, a screen reader assumes that the page is in the default language that the user chose when setting up the screen reader. If the page isn't actually in the default language, then the screen reader might not announce the page's text correctly.
- Does not use HTTPS. All sites should be protected with HTTPS, even ones that don't handle sensitive data. HTTPS prevents intruders from tampering with or passively listening in on the communications between your app and your users.
- Uses document.write(). For users on slow connections, external scripts dynamically injected via document.write() can delay page load by tens of seconds.
- Includes front-end JavaScript libraries with known security vulnerabilities. Some third-party scripts may contain known security vulnerabilities that are easily identified and exploited by attackers. Library Version - jQuery@2.2.4 / Vulnerability Count - 2.
- Does not have a <meta name="viewport"> tag with width or initial-scale. Add a viewport meta tag to optimize your app for mobile screens.
- Document doesn't use legible font sizes. Font sizes less than 12px are too small to be legible and require mobile visitors to "pinch to zoom" in order to read.
- Tap targets are not sized appropriately. Tap targets are too small because there's no viewport meta tag optimized for mobile screens. Interactive elements like buttons and links should be large enough (48x48px) and have enough space around them to be easy enough to tap without overlapping onto other elements.
- Image texture.png used for background is definitely too big (size). Time to load it takes 562 ms (over half second). It needs to be changed for a smaller image and set in css as background-image with repetition to fill up the entire page.
- Some of the advertisements content is loaded as images. It takes a lot of time.
- Too many inline styles have been found in the code. They should be moved into an external css file.



Our Todo List App

Advantages (+)

- Short loading time
- Low memory usage
- Short time of performing functionalities
- Clean and readable code
- Perfect commented code
- Simple design

Disadvantages (-)

- Limited functionality
 - Only local storage is used to save data (Data is lost after cookies are being cleaned).
-

Competitors Todo List Me App

Advantages (+)

- Ability to sort tasks (Normal, Alphabetical, Random and Top 3)
- Ability to print lists
- Ability to drag and drop tasks
- User can create multiple lists
- User can create categories
- Data can be saved on a remote

Disadvantages (-)

- Long loading time
- High memory usage
- Long time of performing functionalities

