Welcome to the Study on Evaluating Explicit Programming Strategies

In the following slides, you’ll learn some key concepts that will help you in completing the study, work on a few hands-on exercises to try out your understanding of the concepts, and then begin the two study tasks. As you go through the slides, you should make sure you understand the concepts and ask the experimenter if you are confused.

Background survey

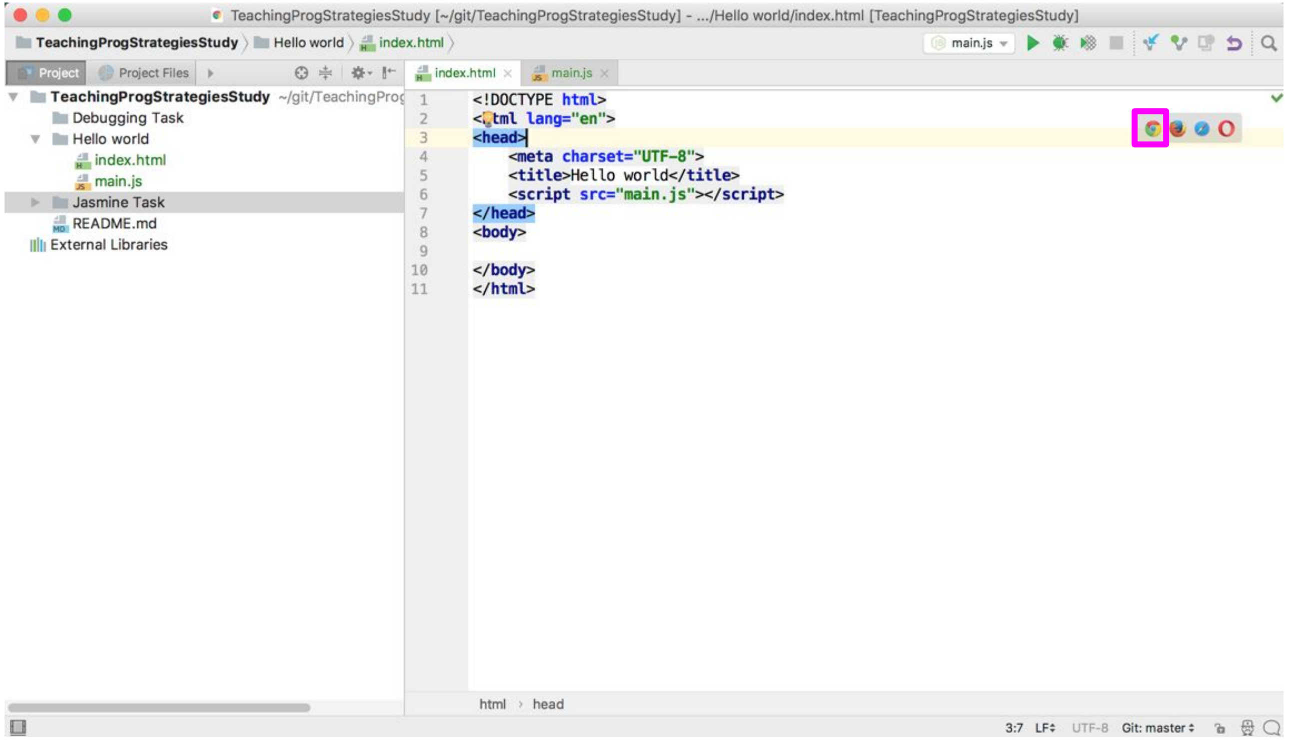
Please complete the following survey:

https://goo.gl/forms/1hfffUdTAf0xDo5f1

WebStorm

WebStorm is an IDE for editing, testing, and debugging code.

To run a project, you should (1) navigate to the **HTML** file in the current directory and (2) click the Chrome browser **button** in the right hand edge of the code window.


TeachingProgStrategiesStudy [~/git/TeachingProgStrategiesStudy] - .../Hello world/index.html [TeachingProgStrategiesStudy]
TeachingProgStrategiesStudy
Hello world
index.html
main.js
Project
Project Files
TeachingProgStrategiesStudy ~/git/TeachingProg
Debugging Task
Hello world
index.html
main.js
Jasmine Task
README.md
External Libraries
index.html
main.js
html
head
3:7
LF
UTF-8
Git: master

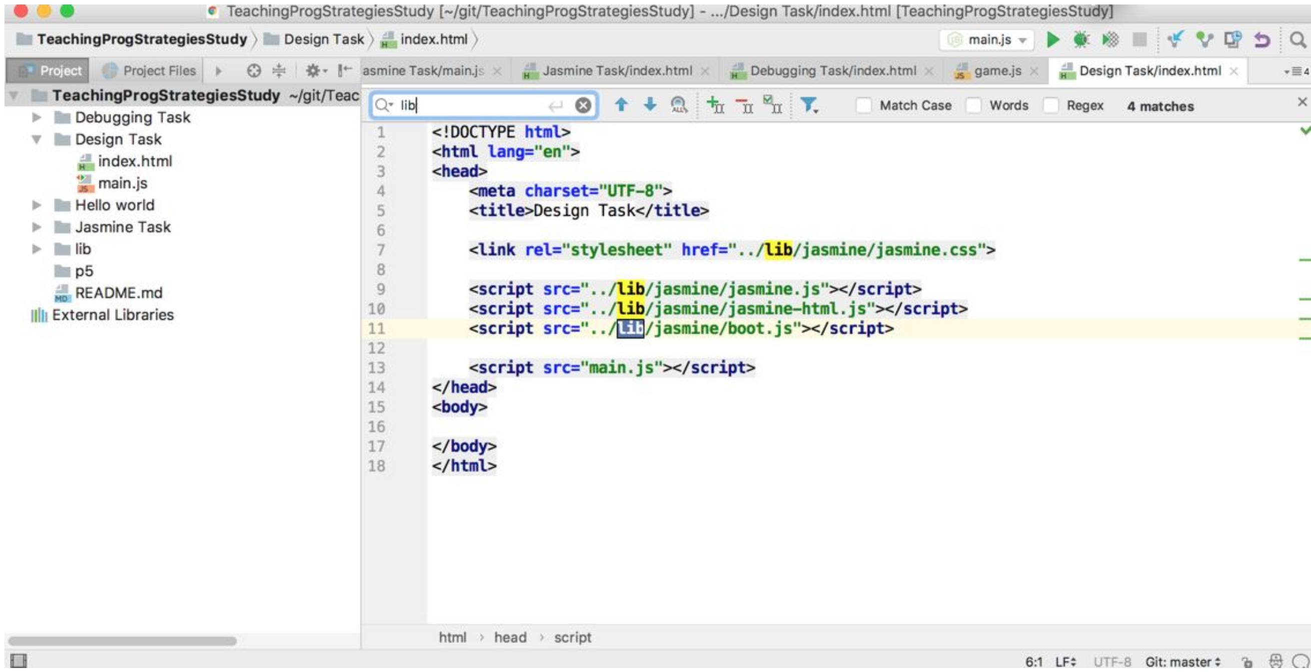
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello world</title>
    <script src="main.js"></script>
</head>
<body>

</body>
</html>
```

Find in file

Press COMMAND F to search within a file.

Each match is highlighted with a yellow background.

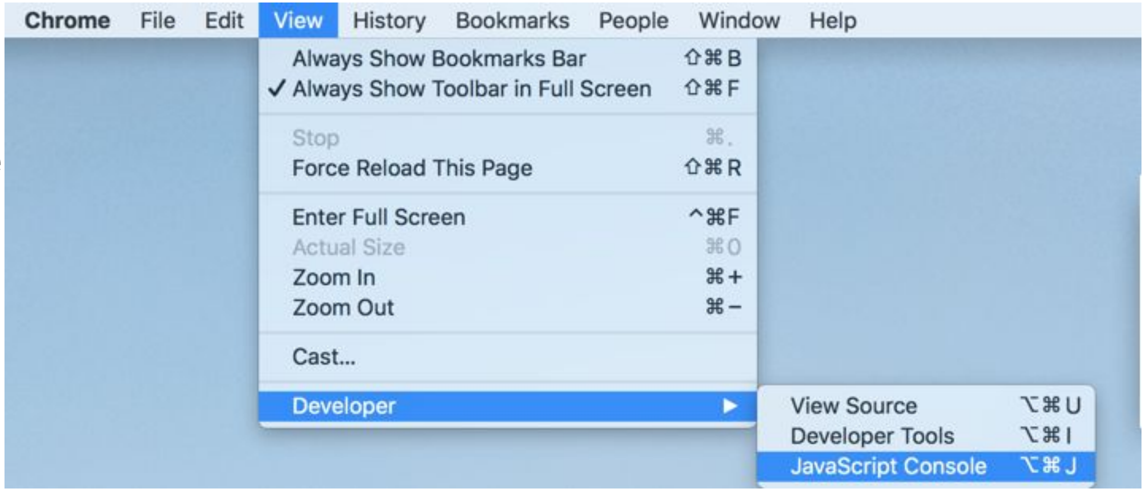
Press ENTER to move between matches.

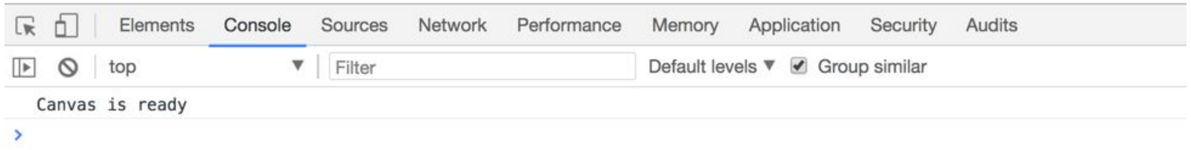


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Design Task</title>

    <link rel="stylesheet" href="../lib/jasmine/jasmine.css">

    <script src="../lib/jasmine/jasmine.js"></script>
    <script src="../lib/jasmine/jasmine-html.js"></script>
    <script src="../lib/jasmine/boot.js"></script>

    <script src="main.js"></script>
</head>
<body>

</body>
</html>
```

Open the Console in Chrome

To see output to the Console in Chrome, go to View / Developer / JavaScript Console.





```
Canvas is ready
>
```

Hello world

Write a program that prints ‘Hello world’ to the console. Verify that the program works correctly by opening the console.

Unit testing with Jasmine

In this study, you will use the Jasmine unit testing framework to write unit tests.

Here's a simple example of a Jasmine unit test:

```
describe("A suite is just a function", function() {
  var a;

  it("and so is a spec", function() {
    a = true;

    expect(a).toEqual(true);
  });
});
```

Declare a set of Jasmine tests

Declare a specific test

Test that the variable a is equal to true.

Jasmine Exercise

Write a test for the following add function

```
function add(a, b)
{

    return a + b;

}
```

Task 1: Debugging task   (30 minutes)

In this task, you’ll fix a bug in a simple web-based game.

The way the game is supposed to work is that the snake moves up, down, left, and right (using the keyboard). Every time the snake eats a dot, it grows in length by one. If the snake collides with itself, the game is over.

As you'll see when you play the game, the snake does not move up, down, left, and right. It just seems to move diagonally, and when you press the arrow keys in certain directions, the game ends.

You have up to 30 minutes. When you believe you’ve found a fix, tell the experimenter which text on which line of code you changed (e.g., line 27, replace "hello" with "goodbye").

Task 1: Debriefing Questions

Task 2: Design Task (30 minutes)

In this task, you’ll design and implement a simple autocomplete feature.

Whenever the user begins typing a new word, your autocomplete will recommend possible completions. Your autocomplete will generate completions based on the words that the user has already entered in the text area, ranking valid possible completions from most to least likely based on the frequency of the word in existing text. If there are no possible completions based on these words, your system should generate an empty list of completions.

Your goal is to build a working implementation and to craft a clear and easy to maintain design. You have up to 30 minutes. Notify the experimenter when you're done.

Task 2: Debriefing Questions

Study Debriefing