Настройка окружения

1. Устанавливаем Visual Studio Code.

VS Code – это редактор кода (code editor). До этого мы работали с Visual Studio, которая является средой разработки (IDE).

IDE – тяжеловесная функциональная программа, с миллионом встроенных функций, и обычно нацеленная на какие-то определённые языки программирования или стеки технологий. Visual Studio, например, в первую очередь рассчитана на разработку Windows приложений и .NET. Intellij IDEA – для Java. Android Studio – только для разработки под Android. Pycharm – для Python и т.д.

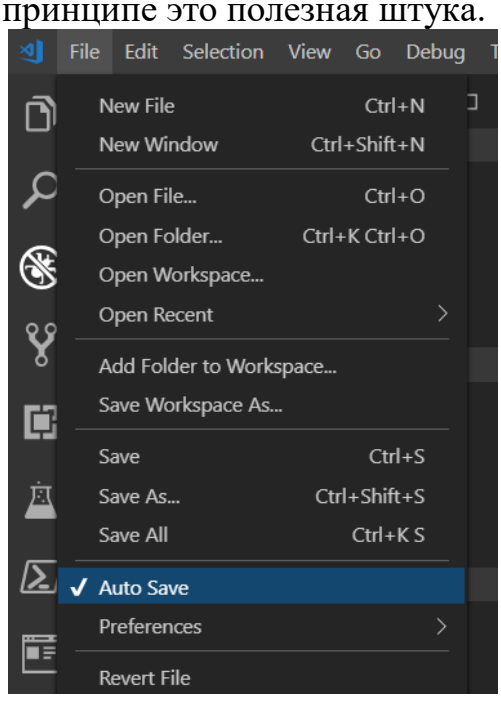
Редактор кода – это, в своей основе, просто легковесный текстовый редактор (как блокнот). Главная особенность редакторов в их модульности. Существуют миллионы расширений, которые можно подключить к редактору, чтобы наделить его какой-либо функциональностью. Хотите подсветку Java синтаксиса – установите соответствующее расширение. Хотите отладку Python – установите соответствующее расширение. Хотите китайский язык – устанавливайте расширение. Хотите встроенное редактирование изображений – расширение. И так для чего угодно. Редакторы кода – это, своего рода, конструктор, который собираете вы сами, под ваши нужды.

2. Русский язык

Если вы хотите русскоязычный интерфейс, то нужно установить расширение **Russian Language Pack for Visual Studio Code** (но лучше научиться на английском);

3. Автосохранение

Сразу же нужно включить автосохранение. Для некоторых расширений это необходимо, да и в принципе это полезная штука.



4. Расширения

Рекомендуемые расширения для работы во фронтенде и просто в VS Code:

- **Auto Close Tag**;

В HTML при открытии тега автоматически будет появляться его закрывающая часть.

- **Auto Rename Tag**;

При изменении имени тега в HTML автоматически будет изменяться имя и у закрывающего тега.

- **Beautify**;

Добавляет возможность форматировать (выравнивать отступы, скобочки и т.д.) HTML, CSS и JS файлы (Shift+Alt+F).

- **CSS Peek**;

С этим расширением можно просматривать и редактировать выбранные CSS стили прямо в HTML коде, а также переходить к их определению (стандартные команды Peek Definition Alt+F12 и Go to Definition F12).

- **Debugger for Chrome**;

Благодаря этому расширению можно выполнять отладку JS кода не только в браузере, но и прямо в редакторе.

- **ESLint**;

Линтер для JS кода. Добавляет автоматическую проверку JS кода с выводом сообщений об ошибках и предупреждениях касающихся использующегося синтаксиса.

- **Git History**;

Позволяет просматривать историю изменений коммитов прямо в редакторе.

- **IntelliSense for CSS class names in HTML**;

Автодополнение имён CSS классов при их вводе в HTML.

- **Live Server**;

Live Server запускает в редакторе встроенный локальный веб сервер, чтобы можно было просматривать веб страницы, как если бы они находились на реальном сервере. А также включает hot-reloading, т.е. автоматическое обновление страницы в браузере при изменении исходного файла.

- **Quokka.js**;

Позволяет запускать интерпретатор JS прямо в редакторе. JS файл запускается в специальном режиме, и работа с кодом происходит так же, как если бы вы писали его в консоли браузера (Ctrl+K, Q).

- Рыбный текст;

Это рыбный текст!

- **stylelint**;

Линтер для CSS. Добавляет проверку синтаксиса CSS с выводом сообщений об ошибках и предупреждениях.

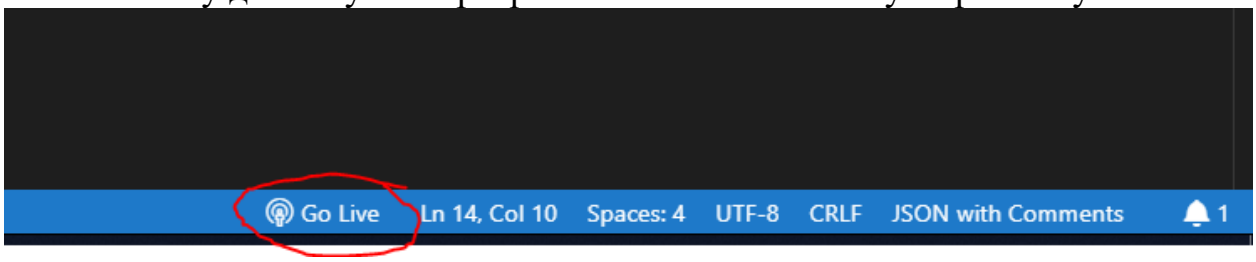
- **Visual Studio IntelliCode – Preview**;
  Просто улучшает некоторые аспекты автодополнения кода.
- **vscode-icons**;
  Набор разнообразных классных иконок для различных типов файлов в проводнике редактора.

5. Выполнение JS кода

Если вы просто пишете JS код без привязки к HTML странице, и хотите посмотреть, как он работает, можно, в принципе, делать это в консоли браузера. Но ещё удобнее будет использовать расширение **Quokka.js**. Нужно лишь перевести файл в Quokka-режим (хоткей Ctrl+K, Q) и весь написанный код будет автоматически интерпретироваться. Консольный вывод будет отображаться снизу на вкладке Output.

Но это, конечно, редкий сценарий работы. Гораздо чаще разработка на JS идёт в связке с HTML страницами. Вы открываете HTML страницу в браузере – и JS код исполняется. Для того, чтобы работать со своими страницами в браузере, по-хорошему, нужен веб сервер. И самый простой вариант его получить – использовать расширение **Live Server**.

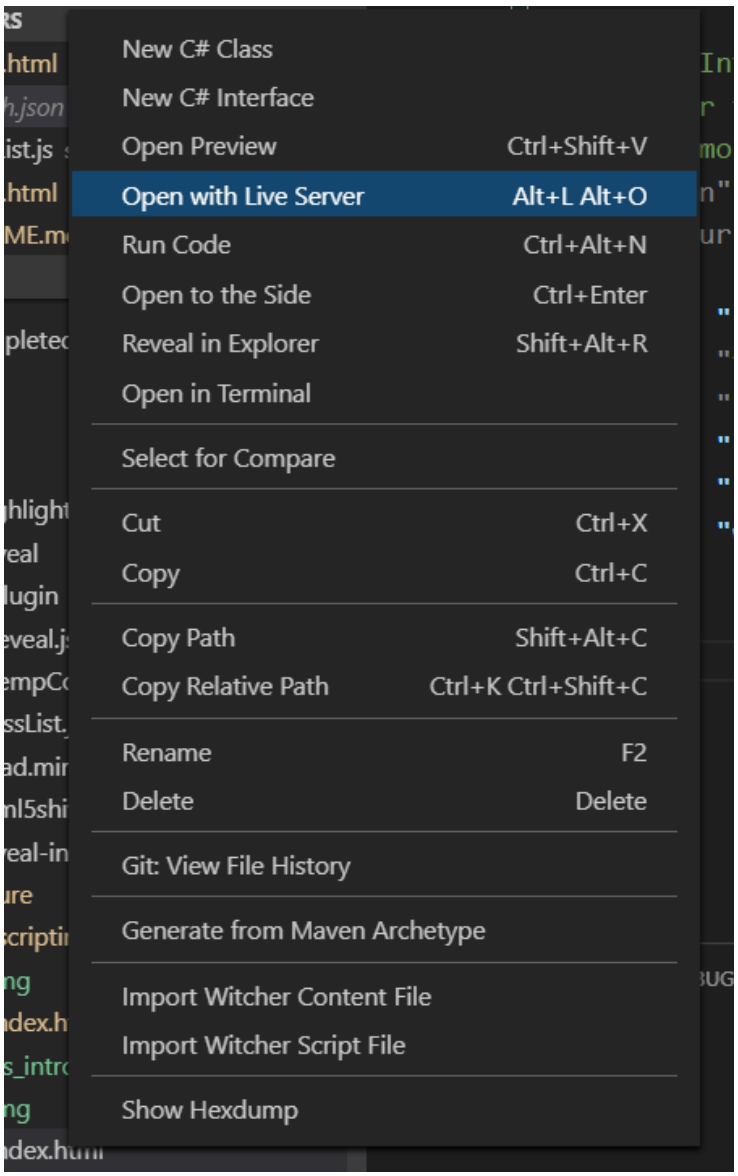
Кнопку для запуска сервера можно найти на статусбаре внизу окна:



Это действие запустит локальный веб сервер на порту 5500 (по умолчанию), отслеживающий корневую директорию вашего проекта.

И уже дальше, в браузере, вы можете открывать, какие вам нужно, ваши страницы.

Можно также открывать конкретный файл через VS Code. Во встроенном проводнике, в контекстном меню по нажатию на нужный файл необходимо выбрать пункт Open with Live Server (а можно использовать хоткей Alt+L Alt+O).



Запускать код в браузере не составляет труда, но что насчёт отладки? Вот здесь уже есть варианты.

6. Отладка

В общем, есть 4 пути:

- Не использовать отладку;
- Страница в браузере, отладка в браузере;
- Страница в браузере, отладка в редакторе;
- Страница в редакторе, отладка в редакторе.

I. «Йа у мамы программист»

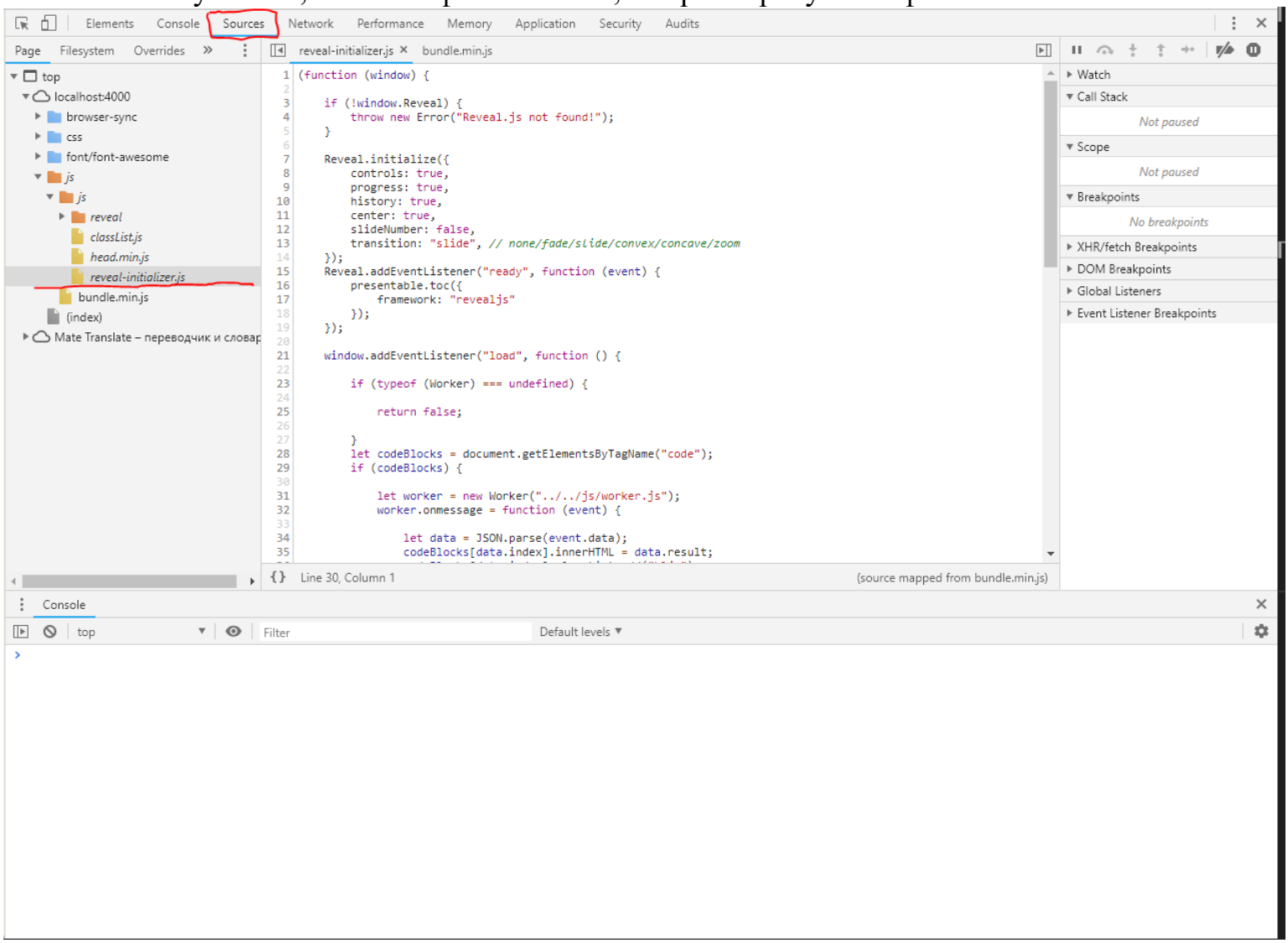
Кому вообще нужна отладка? Можно просто везде понатыкать `console.log` и смотреть на вывод в консоли.

II. Классический вариант

Для этого способа ничего дополнительного устанавливать и настраивать не требуется – всё необходимое встроено в браузер. Вы просто

открываете свою HTML страницу с подключенным JS файлом (со включенным Live Server-ом, разумеется). При загрузке страницы этот JS код выполняется.

Чтобы отлаживать этот код в браузере, нужно открыть инструменты разработчика (F12 или Ctrl+Shift+I) и зайти на вкладку Sources (или Debug). И уже там, в списке файлов слева, выбрать требуемый файл.



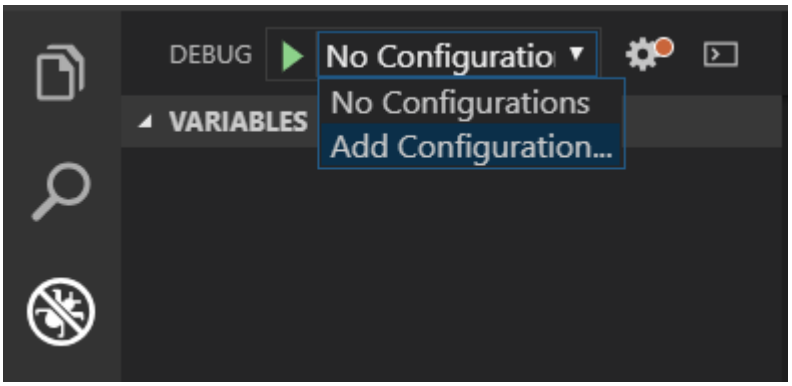
Вся работа с отладкой в браузере происходит в этом окне. Сам процесс отладки везде одинаковый: ставим точку останова, выполняем код в пошаговом режиме и т.д. Только в качестве перезапуска программы – перезагрузка страницы.

III. Профессиональный вариант

Суть этого вариант в том, чтобы перенести отладку из браузера в редактор. Для этого уже потребуются кое-какие телодвижения и расширение **Debugger for Chrome**.

Для этого нужно кое-как настроить VS Code. А именно, добавить конфигурацию запуска отладки с подключением к отладчику Chrome. Конфигурации запуска описываются в специальном файле launch.json. Изначально не должно быть ни одной конфигурации запуска и,

соответственно, самого файла launch.json. Один из способов, как добавить новую конфигурацию и создать этот файл – на вкладке Отладки нажать кнопку Add configuration.



Далее в появившемся файле нужно прописать параметры новой конфигурации запуска:

```
"version": "0.2.0",
"configurations": [
    {
        "type": "chrome",
        "request": "launch",
        "name": "Launch Chrome",
        "url": "http://localhost:5500/dist/index.html",
        "webRoot": "${workspaceFolder}"
    }
]
```

В свойстве name можно вводить что угодно – это просто отображаемое имя конфигурации. В свойстве webRoot указывается, где находится корневая директория, отслеживаемая веб сервером. Значение ${workspaceFolder} означает корневую папку проекта, и при настройке Live Server-а по умолчанию, это значение менять не придётся. Самое важное и индивидуальное свойство – это url. Здесь указывается URL-адрес страницы, которая будет открываться при старте отладки. Отлаживать можно будет все страницы проекта – этим параметром просто задаётся точка входа. У каждого она должна быть своя, в зависимости от вашей структуры проекта.

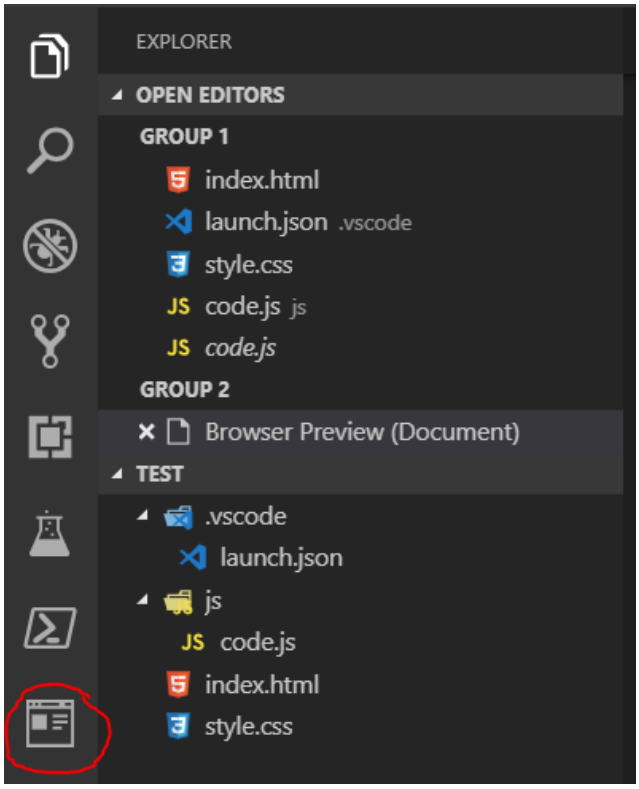
Когда всё это будет настроено, можно будет отлаживать JS код прямо в VS Code, как это происходило в обычной Visual Studio.

IV. Современный вариант (бета-версия)

В этом варианте не только отладка будет перенесена в редактор, но и само отображение страницы будет происходить прямо в VS Code, так что браузер вообще не понадобится. И осуществить такое нам поможет расширение **Browser Preview**. Стоит отметить, что этот способ совсем новый, и расширение находится на стадии разработки и доступно в виде

бета-версии. Поэтому могут встречаться какие-нибудь баги и ошибки отображения.

Функция этого расширения – открывать браузер внутри редактора. После установки расширения, слева на панели должна появится иконка браузера:



По нажатию на эту кнопку экран разделится на 2 секции, и в левой части появится встроенное окно браузера.

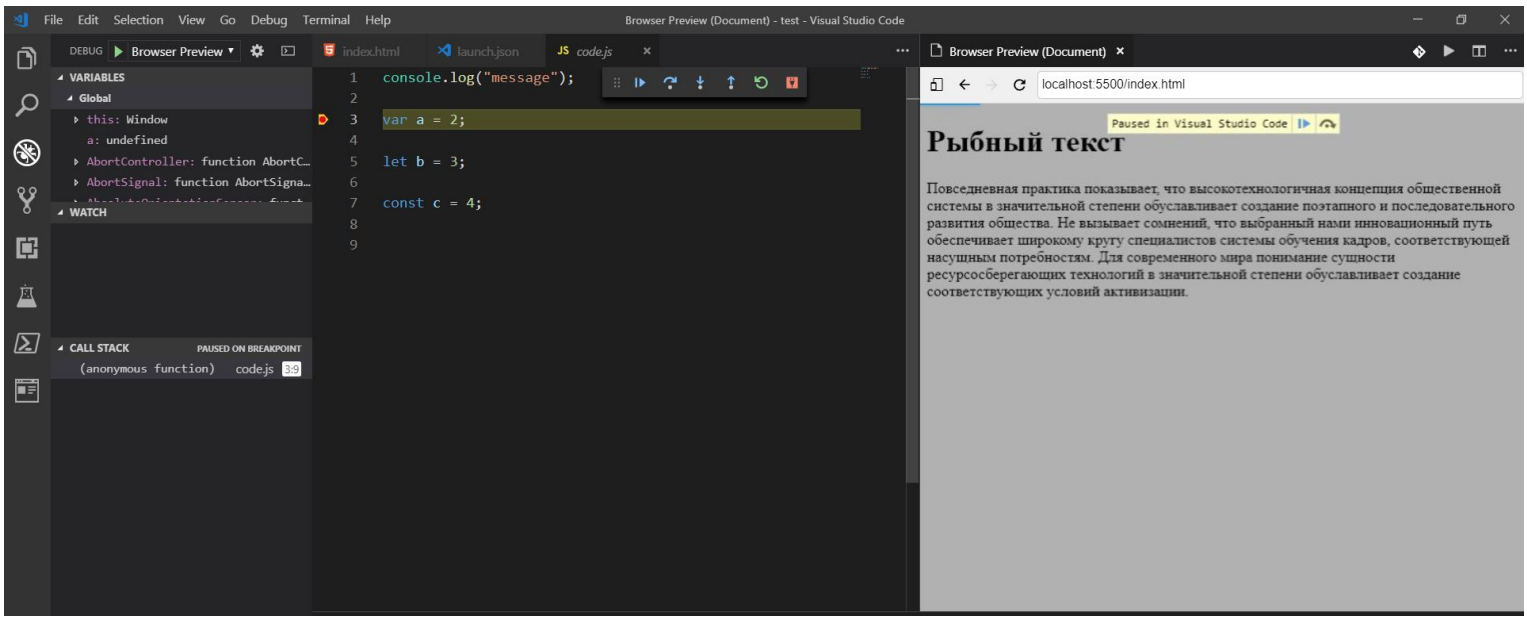
Для того чтобы открывать страницы из вашего проекта понадобится рассмотренное ранее расширение Live Server, а для отладки – Debugger for Chrome.

Сначала нужно активировать Live Server, а затем запустить отладку в подобной конфигурации:

```
{
    "type": "browser-preview",
    "request": "launch",
    "name": "Browser Preview: Launch",
    "url": "http://localhost:5500/index.html",
    "webRoot": "${workspaceFolder}"
}
```

Разумеется, URL нужно указать свой.

В результате всё должно работать: справа открываться окно браузера, а в файлах исходного кода срабатывать точки останова и пошаговое выполнение. Перезапуск/перезагрузка страницы выполняется с помощью кнопки с зелёной закрученной стрелкой на всплывающей панели отладки.


File Edit Selection View Go Debug Terminal Help
Browser Preview (Document) - test - Visual Studio Code
DEBUG
Browser Preview
index.html
launch.json
code.js
VARIABLES
Global
this: Window
a: undefined
AbortController: function AbortC…
AbortSignal: function AbortSigna…
WATCH
CALL STACK
PAUSED ON BREAKPOINT
(anonymous function)
code.js 3:9
1 console.log("message");
3 var a = 2;
5 let b = 3;
7 const c = 4;
Browser Preview (Document)
localhost:5500/index.html
Paused in Visual Studio Code
Рыбный текст
Повседневная практика показывает, что высокотехнологичная концепция общественной системы в значительной степени обуславливает создание поэтапного и последовательного развития общества. Не вызывает сомнений, что выбранный нами инновационный путь обеспечивает широкому кругу специалистов системы обучения кадров, соответствующей насущным потребностям. Для современного мира понимание сущности ресурсосберегающих технологий в значительной степени обуславливает создание соответствующих условий активизации.