Some tools for building web-based groupware

Michel Beaudouin-Lafon

mbl@lisn.fr

The web as platform

- Web standards
  - HTTP(S): protocol between client and server
  - HTML: content description
  - CSS: style
- Browser
  - Represents content as a DOM tree
  - JavaScript to modify the tree, react to events, and communicate with server

The web as platform

- Server
  - Responds to client requests (HTTP)
    - GET, HEAD, POST, PUT, DELETE
    - Not suitable for real-time interaction (The result replaces the current page)
  - Websocket connection for real-time interaction between client and server
    - Event-based (asynchronous) programming

JavaScript

- More powerful than you think
- Functional: functions are first-class objects
  - **function** addmap(array, f) {
    var sum = 0
    for (var item of array)
    sum += f(item)
    return sum
    }
  - addmap([1, 2, 3], **function**(x) { **return** x*x }) // 14

Modern JavaScript

- Anonymous functions can be noted
  - param **=>** expression

    or

    (p1, p2) **=>** { ... body ... **return** expression }
- Previous example:
  - addmap([1, 2, 3], **function**(x) { **return** x*x }) // 14

    becomes

    addmap([1, 2, 3], x **=>** x*x) // 14

Modern JavaScript

- Classes
  - **class** Point {
    constructor() { this.x = 0; this.y = 0 }
    moveBy(dx, dy) { this.x += dx; this.y += dy }
    }
  - **var** p = new Point(); p.moveBy(10, 20)
- Other useful features, including
  - Iterators and generators
  - Arrow functions, Promises

TypeScript

https://www.typescriptlang.org

- JavaScript with types
  - **function** fib(n: number): number { … }
- Powerful type system
  - Generic types:
    function swap<T>(a: T, b:T): T { **var** x = a; a = b; b = x}
- Other powerful features
- Translates to plain JavaScript
- Works with plain JavaScript libraries
  - Through definition files

Node.js

https://nodejs.org/en/

- Use JavaScript to program web app servers
- Module system
  - Modules are installed with **npm**,
    the Node Package Manager: npm install <package>
  - Packages can be local or global (npm –g)
  - A package is imported with require:
    var fs = require('fs'); // access to file system
    fs.fileExists(…)
  - Node has built-in modules to access the OS:
    file system, processes, HTTP protocol, …

Node.js

- Node implements a reactive programming model based on events and event handlers
- Asynchronous event handling
  - **var** Emitter = require('events').EventEmitter
    var chatRoom = new Emitter(), participants = []
  - chatRoom.on('hello', name **=>** {
    console.log(name, 'says Hi!')
    });
  - chatRoom.on('hello', name **=>** {
    participants.push(name)
    });
  - chatRoom.emit('hello', 'Alice')

Node.js

- Using continuations (or callbacks) to program in an asynchronous world
  - Many node modules use callbacks, which are called when the task is actually performed:
    - **var** fs = require('fs')
      var fd = fs.open('myFile', 'w')
      if (fd) fs.write(fd, 'Hello') // wrong: file not opened!
    - fs.open('myFile', 'w', (err, fd) => {
        if (! err) fs.write(fd, 'Hello', (err, fd) => { ...});
      });
      // green function called when **open** is finished
      // blue function called when **write** is finished

Promises

- Facilitate asynchronous programming
- An asynchronous call returns a Promise object instead of taking a callback
- The then function of the promise is used to specify what to do when the call succeeds:
  - ```
    var fs = require('fs'), util = require('util')
    var readdir = util.promisify(fs.readdir)
    readdir(path).then(files => {
        console.log(files.join('\n'))
    });
    ```
    ```

Promises

- Instead of being nested like callbacks, Promises are chained
- open(path, 'w').**then**(fd => {
  write(fd, 'Hello')
  return fd
  }).**then**(fd => close(fd))

async/await

- Syntactic sugar around promises
- **async function** f() {
  return new Promise…
  }
  …
  await f() // wait until promise is resolved
- Previous example:
  fd = **await** open(path, 'w')
  await write(fd, 'Hello')
  await close(fd)

Express HTTP server

https://expressjs.com

- Basis of many web applications
  - **% npm install express**
  - **var** express = require('express'),
    app = express()
    app.get('/', (request, response) **=>** {
    response.send('Hello World')
    })
    app.listen(8080)
  - // serve static files
    app.use(express.static(__dirname + '/public'))

socket.io

https://socket.io

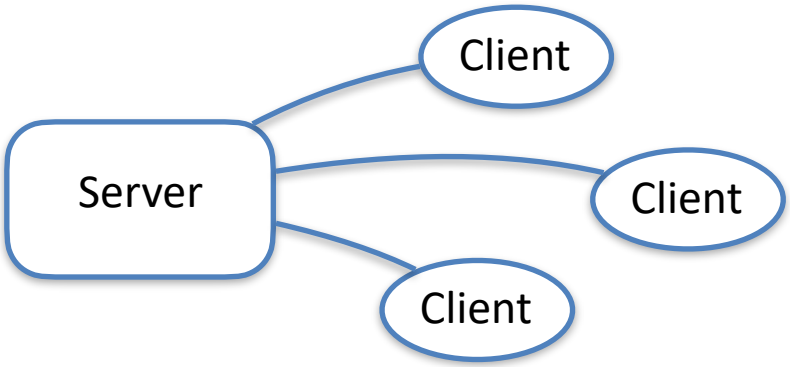
- Communication between web page and server
- Server :
  - **% npm install socket.io**
  - **var** io = require('socket.io').listen(http)

```
io.on('connection', client => {
    client.on('msg', data => {
        // send to all other clients
        client.broadcast.emit('msg', data)
    })
})
```

socket.io

- Client (web page)
  - <script src="/socket.io/socket.js"></script>
    <script>
      var client = io()
      client.on('connect', () **=>** client.emit('msg', 'Hello'))
      client.on('msg', data => $('#chat').append(data))
    </script>

socket.io



- Server

```
var express = require('express'),
      app = express()
app.get('/', (request, response) => {
   response.send('Hello World')
})
app.listen(8080)

app.use(express.static(__dirname + '/public'))

// manage live connections
var io = require('socket.io').listen(http)
io.on('connection', client => {
   client.on('msg', data => {
      // send to all other clients
      client.broadcast.emit('msg', data)
   })
})
```

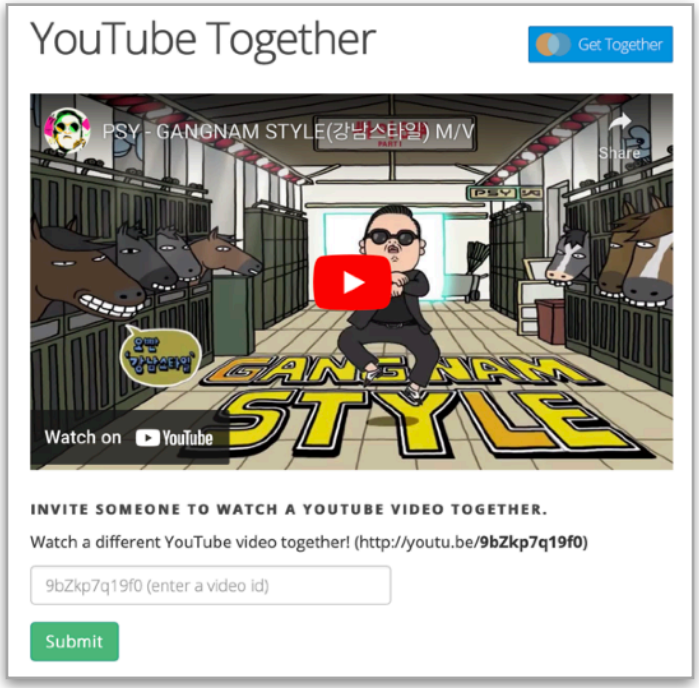
- Client (web page)

```
<script src="/socket.io/socket.io.js"></
script>
<script>
      var client = io()
      client.on('connect', () =>
         client.emit('msg', 'Hello')
      )
      client.on('msg', data =>
         $('#chat').append(data)
      )
</script>
```

TogetherJS

https://togetherjs.com

- Collaboration features for web applications:
  - Text/audio chat, User focus/presence, Co-browsing, Real-time content sync



ShareDB

https://share.github.io/sharedb/

- Shared text strings and JSON objects synchronized through operational transformation
- Server
  - **var** ShareDB = require('sharedb')
    var share = **new** ShareDB(options)
    …
  - // when a client connects to the server:
    webSocketServer.on('connection', ws => {
        share.listen(**new** WebSocketJSONStream(ws))
    })

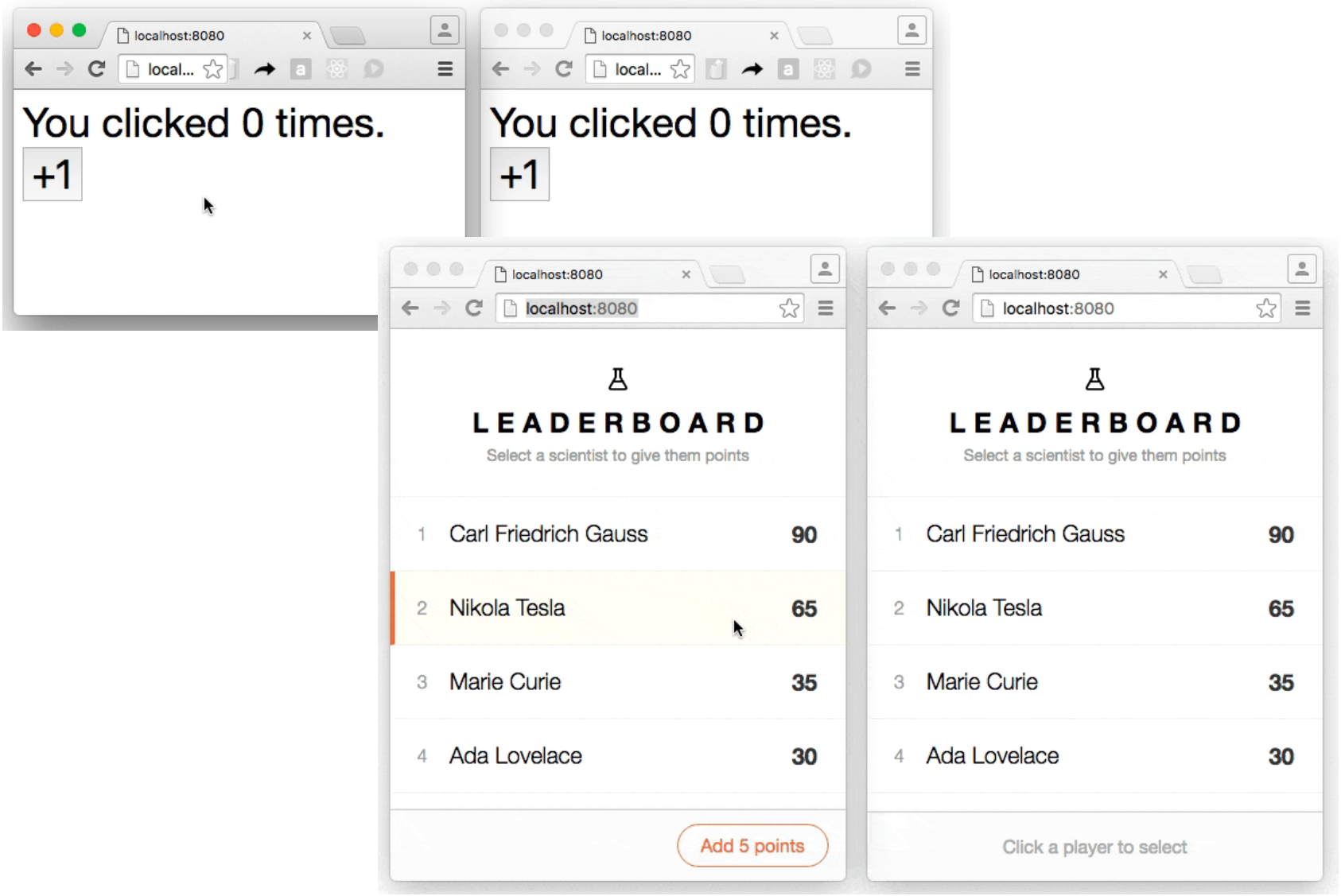
ShareDB

https://share.github.io/sharedb/

- Client that synchronizes text in a textarea
  - **var** ShareDB = require('shared/lib/client')
    var sock = **new** WebSocket('<server address>')
    var connection = **new** ShareDB.Connection(sock)
    …
  - **var** doc = connection.get('mydoc', 'textarea')
    doc.subscribe(err **=>** {
        if (err) **throw** err
        if (! doc.type) doc.create({text: 'hello'})
    }
  - doc.on('op', () **=>** console.log(doc.data.text))

ShareDB

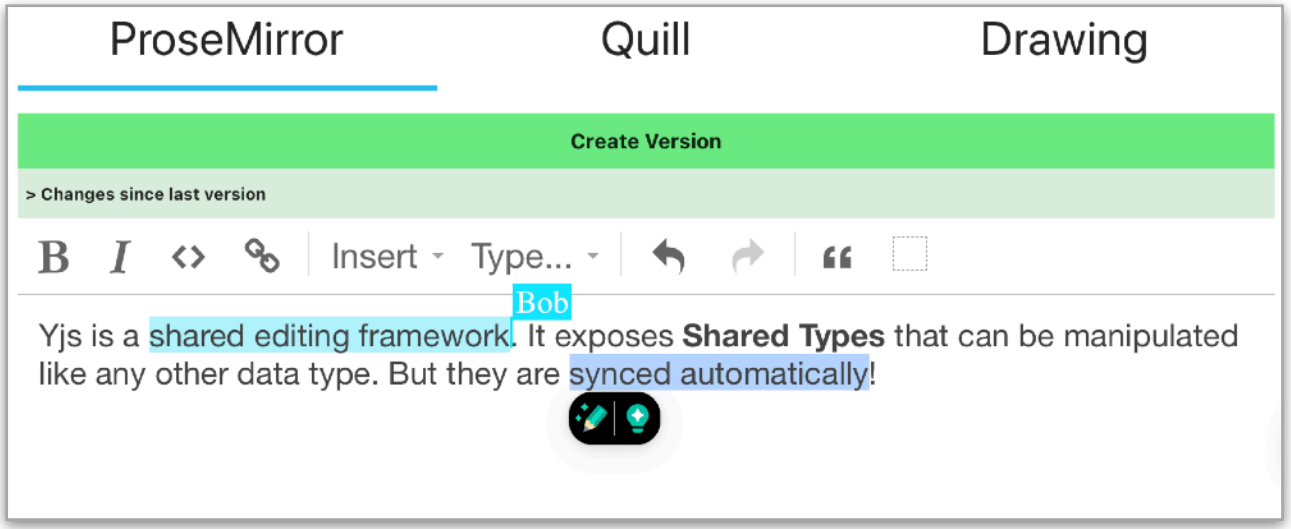
https://share.github.io/sharedb/

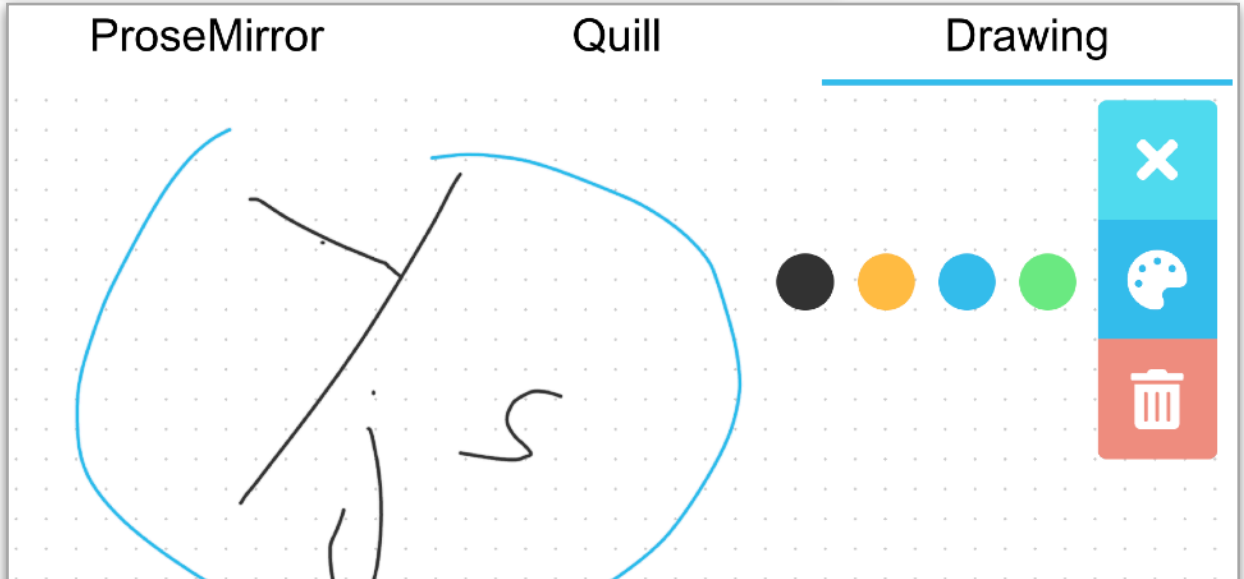


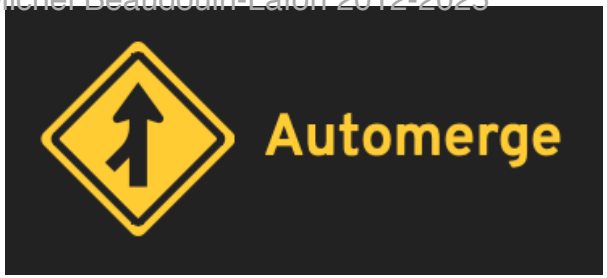
https://yjs.dev

- Shared types that are synced peer-to-peer using Convergent Replicated Data Types (CRDTs)
  - **import** * **as** Y **from** 'yjs'
    import { WebrtcProvider } **from** 'y-webrtc'
  - **var** ydoc = **new** Y.Doc()
    var provider = **new** WebrtcProvider('demo', ydoc)
    var yarray = ydoc.getArray('data')
    yarray.observe(ev => console.log(yarray.toArray()))
  - yarray.push([1])

https://yjs.dev







https://automerge.org

- Share documents across *adapters*: network, local storage

```
const repo = new Repo({
  network: [new BroadcastChannelNetworkAdapter()],
  storage: new IndexedDBStorageAdapter() })
```

  - Create document and make changes

```
const handle = repo.create({foo: "bar"})
handle.change(doc => doc.foo = "baz")
```

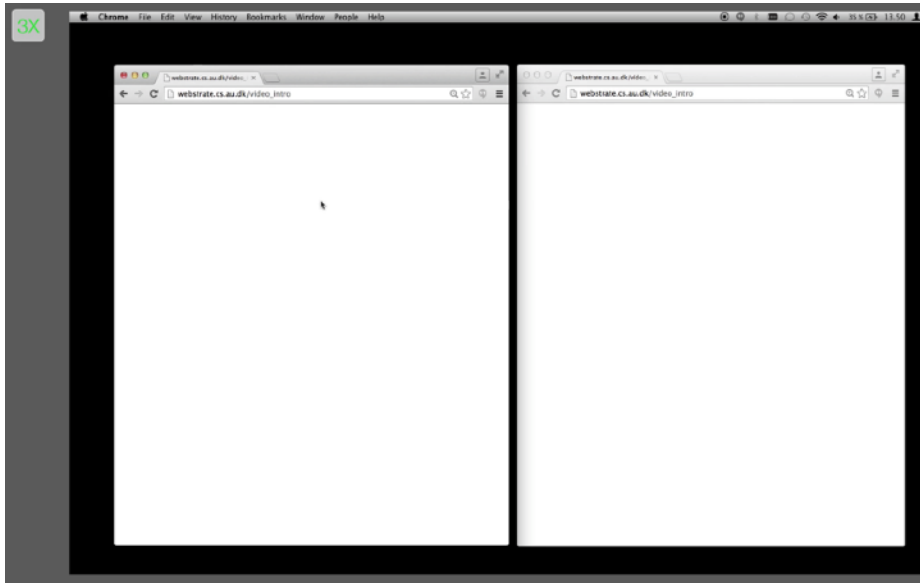
  - Connect to document and receive changes

```
const handle = await repo.find("<url>")
handle.on("change", evt => console.log(evt.doc))
```

Webstrates

https://webstrates.github.io

- Web server that synchronizes the pages it serves as they are edited
- Real-time shared editing “out of the box”
- Transclusion: embed a webstrate in another webstrate
- API for advanced features: permissions, assets, cookies, messaging, transient content, …

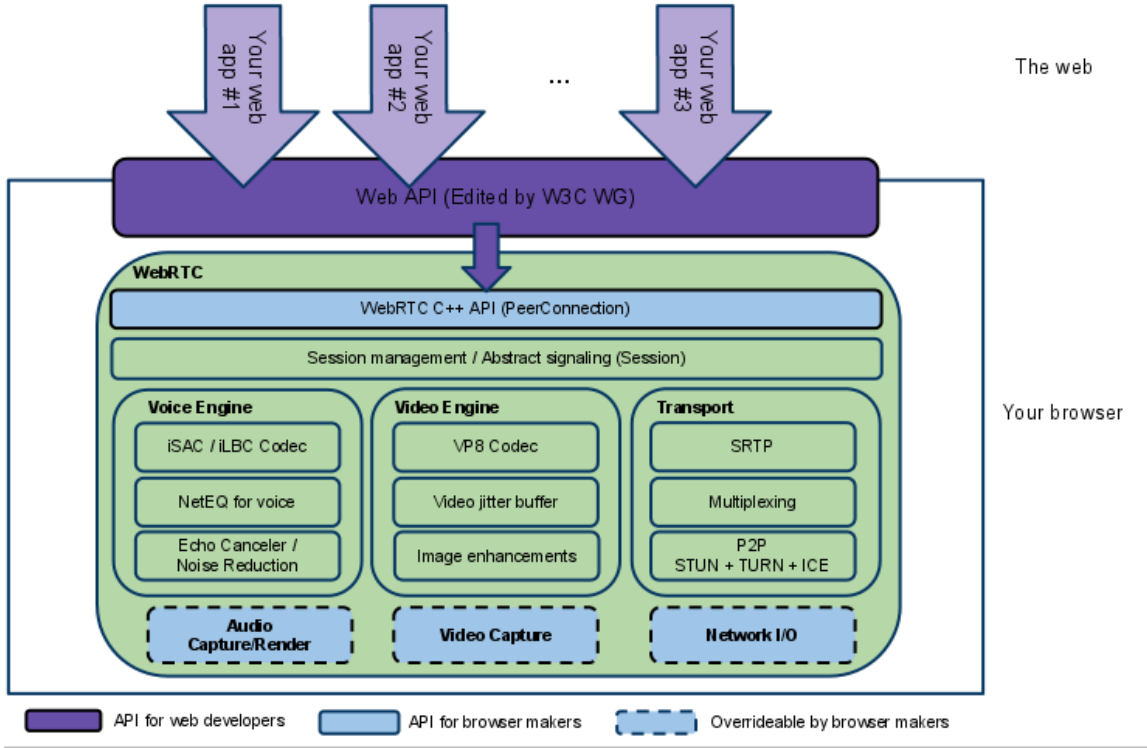


WebRTC

https://webrtc.org

- Set of APIs for real-time communication
- Supports audio, video, data
- Web APIs as well as native APIs

- But:
- Complex
- Inconsistent support across browsers



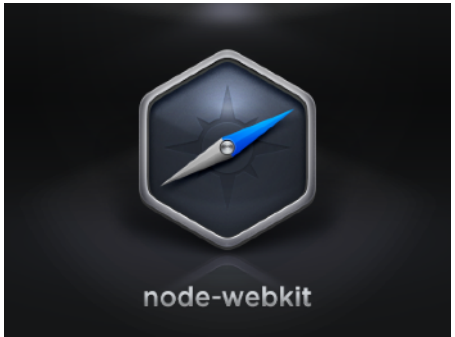
WebRTC

- // Get access to local camera / microphone
  stream = **await** getUserMedia({'video': true, 'audio': true})
  videoElement.srcObject = stream // local playback
- // Connect to peer
  conn = **new** RTCPeerConnection(config)
  stream.getTracks().forEach(t => conn.addTrack(t, stream))
  … // negotiate offer through ICE candidates
- // Connection established: show remote video/audio
  conn.addEventListener('connectionstatechange', ev => {
  if (conn.connectionState === 'connected')
  conn.addEventListener('track', async (ev) => {
  remoteVideo.srcObject = ev.streams[0] })
  })

Node-webkit (nwjs)

https://nwjs.io

- Combines a node.js server and a web browser
- Access to desktop: files, menus, …
- Desktop-based web applications
- Avoids web protocol issues (same-origin, …)
- Global namespace shared between node server and every web window

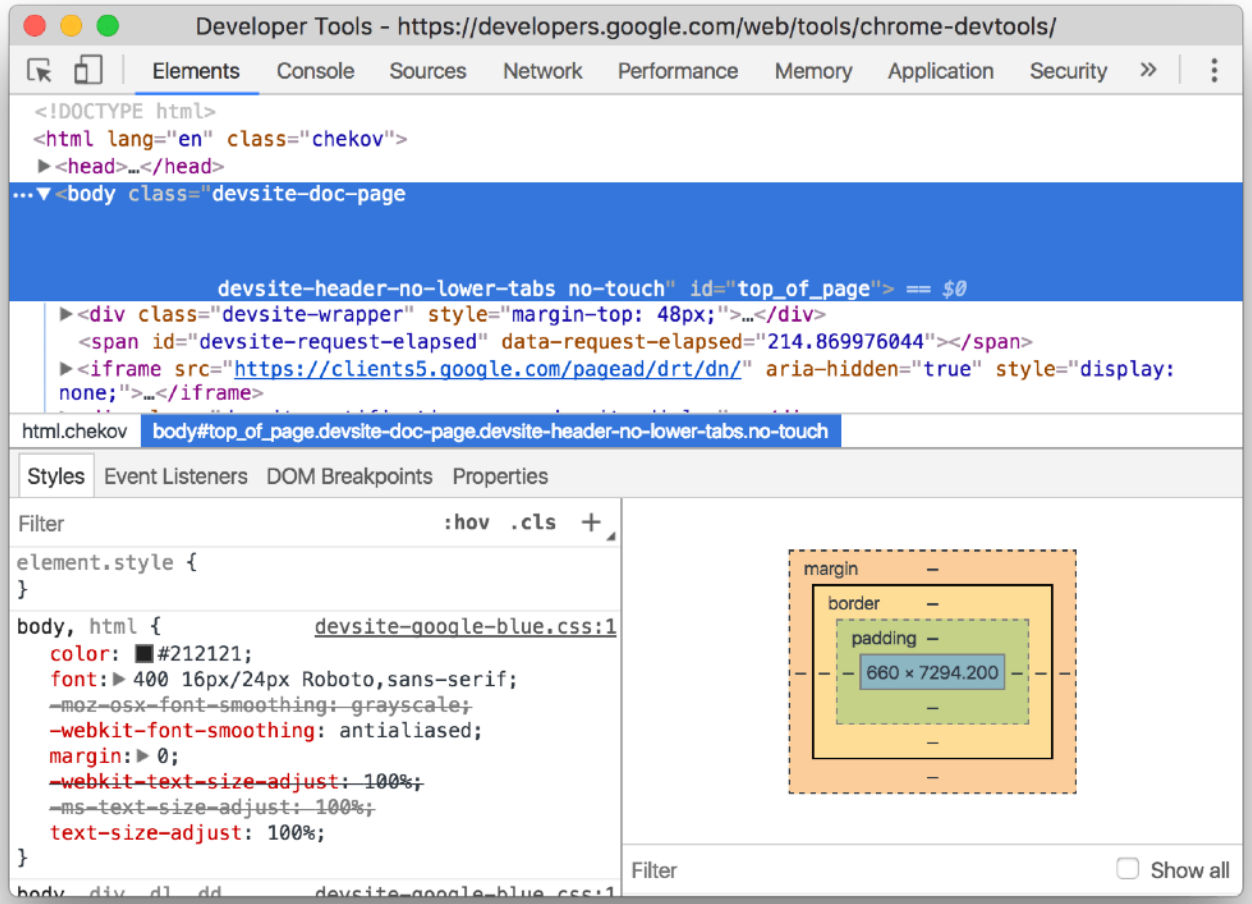


Node-webkit

- A web page that lists the files in the current directory
  - <script>
    var fs = require('fs') // node module in HTML file!
    fs.readdir('.', **function**(err, files) {
      files.map(**function**(filename) {
        $('#list').append('
'+filename+'
');
      });
    });
    </script>
    <ul id='list'>

Debugging

- In the browser: developer tools



Debugging

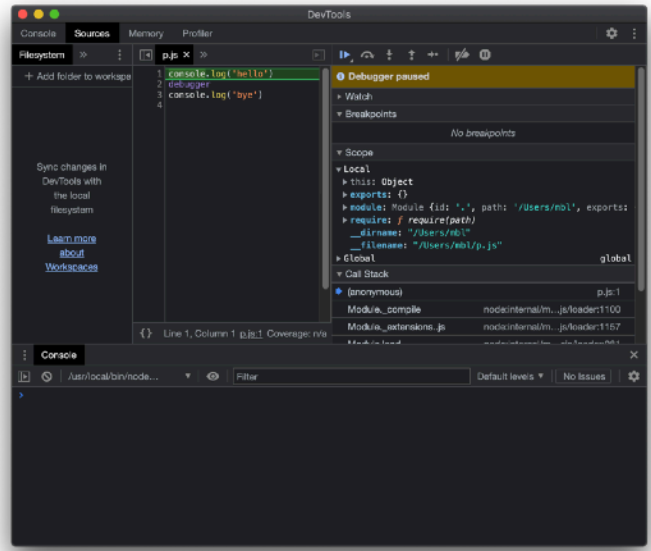
- Node.js
  - Built-in debugger:

    % node inspect myscript.js
    % node debug myscript.js
    (you need to add calls to debugger in the script)
  - Remote debugging with web inspector
    - Run the script in debug mode

      % node inspect myscript.js

      or

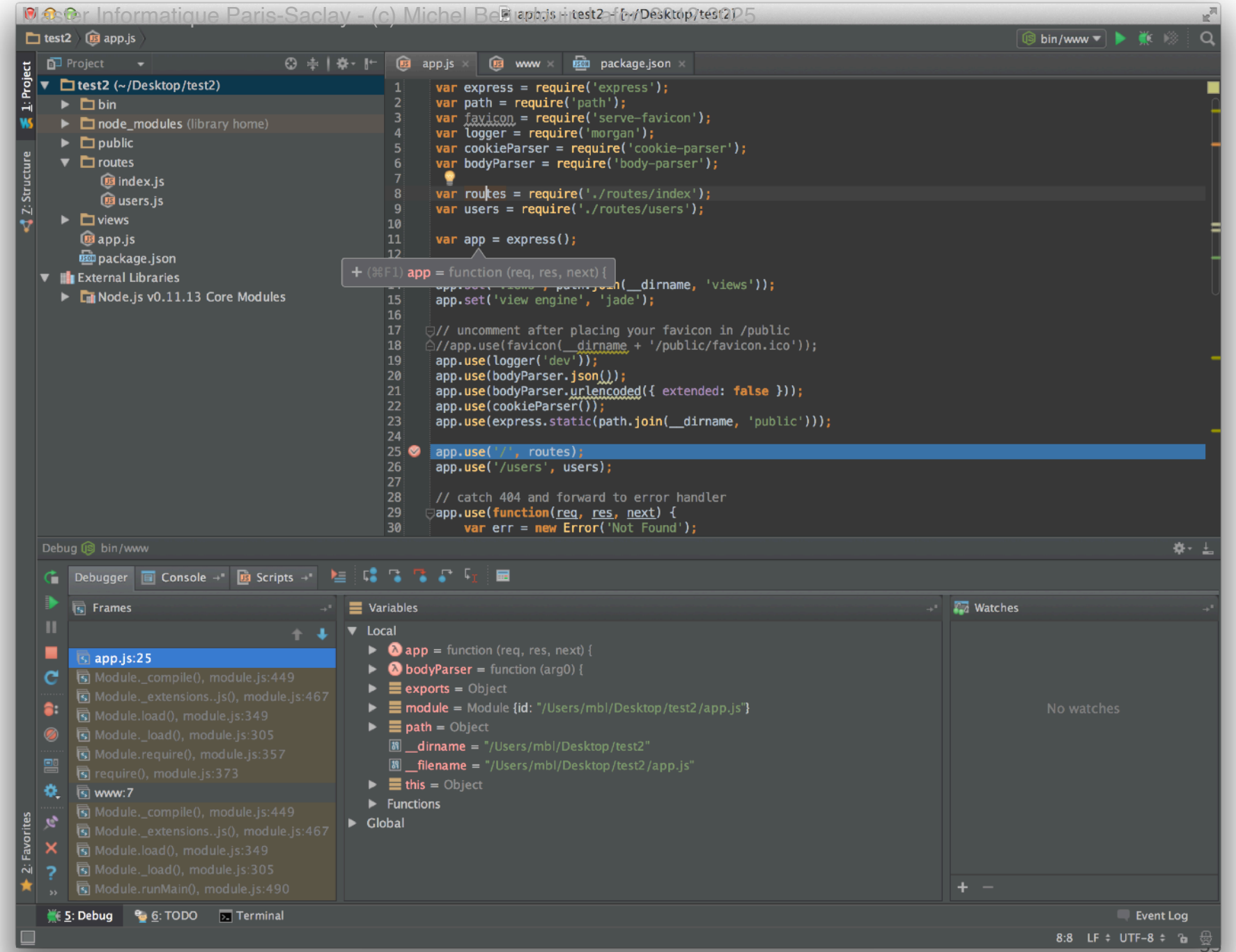
      % node --inspect myscript.js
    - On Chrome, goto chrome://inspect and configure network targets



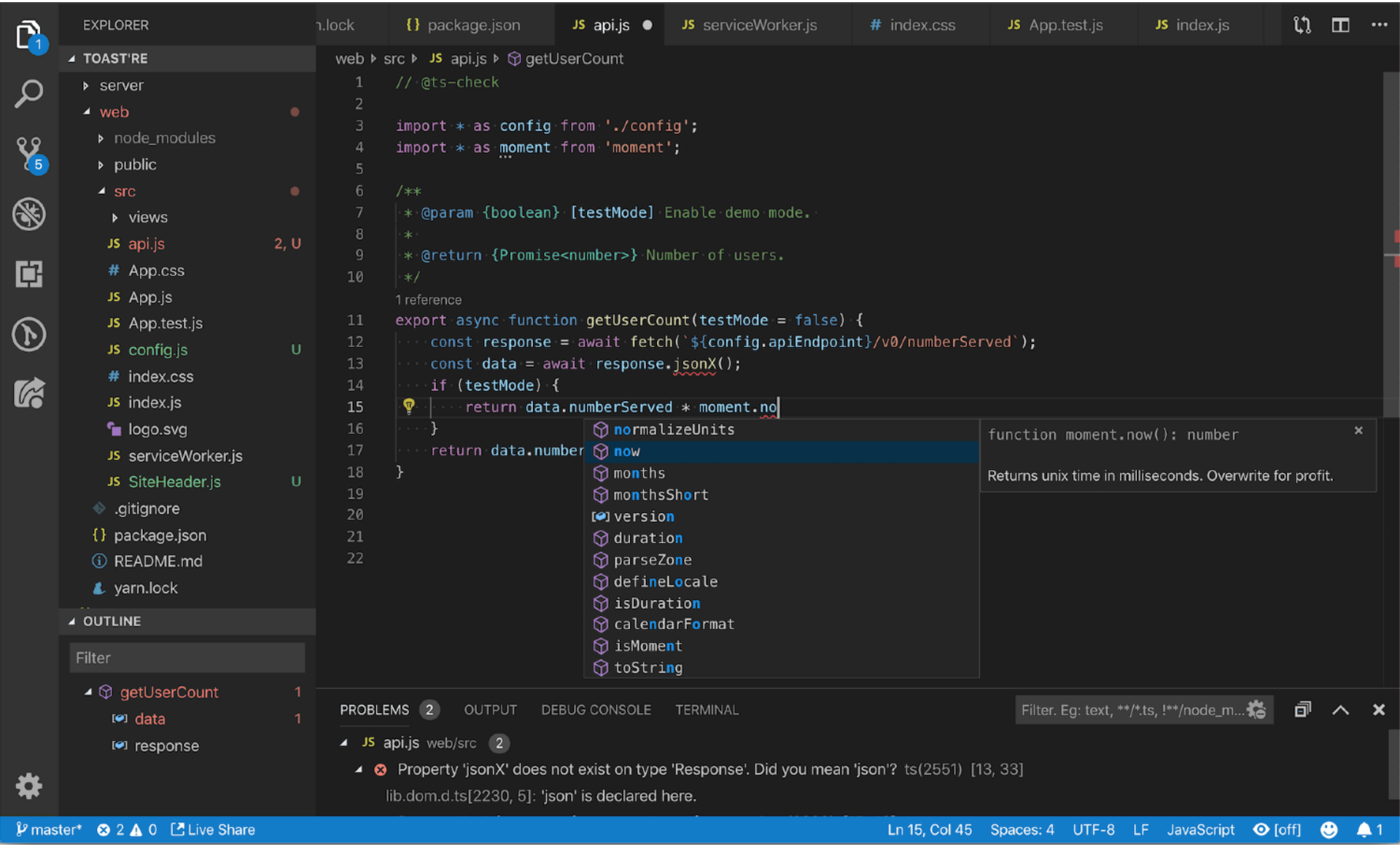
WebStorm (IntelliJ), VSCode (Microsoft)

https://www.jetbrains.com/webstorm/
https://code.visualstudio.com

- Integrated Development Environment (IDE)
- Project manager
- Editor
  - Multiple cursors
  - Code completion
  - Refactoring
  - Linting
- Debugger
- Support Typescript


app.js – test2 – [~/Desktop/test2]
test2 app.js
bin/www
Project
app.js
www
package.json
test2 (~/Desktop/test2)
bin
node_modules (library home)
public
routes
index.js
users.js
views
app.js
package.json
External Libraries
Node.js v0.11.13 Core Modules
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var routes = require('./routes/index');
var users = require('./routes/users');
var app = express();
(⌘F1) app = function (req, res, next) {
app.set('view engine', 'jade');
// uncomment after placing your favicon in /public
//app.use(favicon(__dirname + '/public/favicon.ico'));
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
app.use('/', routes);
app.use('/users', users);
// catch 404 and forward to error handler
app.use(function(req, res, next) {
var err = new Error('Not Found');
Debug bin/www
Debugger
Console
Scripts
Frames
app.js:25
Module._compile(), module.js:449
Module._extensions..js(), module.js:467
Module.load(), module.js:349
Module._load(), module.js:305
Module.require(), module.js:357
require(), module.js:373
www:7
Module._compile(), module.js:449
Module._extensions..js(), module.js:467
Module.load(), module.js:349
Module._load(), module.js:305
Module.runMain(), module.js:490
Variables
Local
app = function (req, res, next) {
bodyParser = function (arg0) {
exports = Object
module = Module {id: "/Users/mbl/Desktop/test2/app.js"}
path = Object
__dirname = "/Users/mbl/Desktop/test2"
__filename = "/Users/mbl/Desktop/test2/app.js"
this = Object
Functions
Global
Watches
No watches
5: Debug
6: TODO
Terminal
Event Log
8:8
LF
UTF-8
1: Project
2: Favorites
7: Structure

Webstorm


EXPLORER
TOAST'RE
server
web
node_modules
public
src
views
api.js
App.css
App.js
App.test.js
config.js
index.css
index.js
logo.svg
serviceWorker.js
SiteHeader.js
.gitignore
package.json
README.md
yarn.lock
OUTLINE
Filter
getUserCount
data
response
package.json
api.js
serviceWorker.js
index.css
App.test.js
index.js
web ▸ src ▸ api.js ▸ getUserCount
// @ts-check
import * as config from './config';
import * as moment from 'moment';
/**
* @param {boolean} [testMode] Enable demo mode.
*
* @return {Promise<number>} Number of users.
*/
1 reference
export async function getUserCount(testMode = false) {
const response = await fetch(`${config.apiEndpoint}/v0/numberServed`);
const data = await response.jsonX();
if (testMode) {
return data.numberServed * moment.no
}
return data.number
}
normalizeUnits
now
months
monthsShort
version
duration
parseZone
defineLocale
isDuration
calendarFormat
isMoment
toString
function moment.now(): number
Returns unix time in milliseconds. Overwrite for profit.
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
Filter. Eg: text, **/*.ts, !**/node_m...
api.js web/src 2
Property 'jsonX' does not exist on type 'Response'. Did you mean 'json'? ts(2551) [13, 33]
lib.dom.d.ts[2230, 5]: 'json' is declared here.
master*
2 0
Live Share
Ln 15, Col 45
Spaces: 4
UTF-8
LF
JavaScript
[off]
1

VSCode