```
 __    _____ _____ _____ _____ _____ ____  _____
|  |  |   __|     |   | |  _  | __  |    \|     |
|  |__|   __|  |  | | | |     |    -|  |  |  |  |
|_____|_____|_____|_|___|__|__|__|__|____/|_____|
                                                v. 0.1
```

install dependencies and build it

```
> npm install
```

```
> npm run build
```

try it

```
> npm run serve
```

open http://localhost:4567 (http://localhost:4567) (or at the port set on *src/vars.json* in the `SERVER.PORT`)

test

while the `npm run serve` is running is possible to run `npm test` that will run some tests on some of the pages served (using puppeteer)

use it

First of all in your html include *Leonardo.js* in the <head> tag:

```
<script src="path/to/Leonardo.js"></script>
```

Now create another <script> tag to use *Leonardo.js*:

```
<script>
    var L = Leonardo (300, 200, {id : "target"});
</script>
```

parameters:
width : the width in pixels (required)
height : the height in pixels (required)
attrs : an hash of required attributes for the <svg> tag

for *svg namespaces* is enough just to pass a `ns` element containing an array containing one or more from the following set :

```
['cc', 'dc', 'ev', 'rdf', 'svg', 'xlink']
```

if all are needed is enough to pass '*'.

Now we can create new `Elements` through `L`.

---

tags

Every function listed below creates a `Element` instance, and thus benefits the following instance methods: attrs, styles, add, on ,off, clone, trans, rotate, scale, mirrorO, mirrorV and move. I will describe all them soon.

Once these elements are created at some point they must be added either directly to the root <svg> tag either to a <g> group element (same here for <g>).

```
L.add(as, many, elements, as, needed)
```

<desc>

```
var desc = L.desc('This is the description of my svg')
```

Returns a <desc> tag containing the text passed to it

<image>

```
var image = L.image(x, y, w, h, src)
```

Returns a <image> tag positioned at P{x,y}; about *w* and *h* are meant to be the clearly the sizes but real image size will win on it, in the end the ratio cannot be modified.

<line>

```
var line = L.line(x1,y1, x2,y2)
```

Returns a <line> tag representing a segment starting from P1(x1,y1) and ending in P2(x2,y2). Here could be useful to use the `attrs` function to, for example, style the line:

```
var line = L.line(...).attrs({"stroke-width" : 1.5,"stroke" : 'green'});
```

<polyline>

```
var polyline = L.polyline(x1,y1, x2,y2 [,x3,y3[...]])
```

creates a polyline which can even be opened (does not close it automatically).