



# Nouveaux Moyens d'Interaction

#FishEye #VR

Luc MARONGIU - 2017

*Tech lead - Air France*  
ls.marongiu@gmail.com



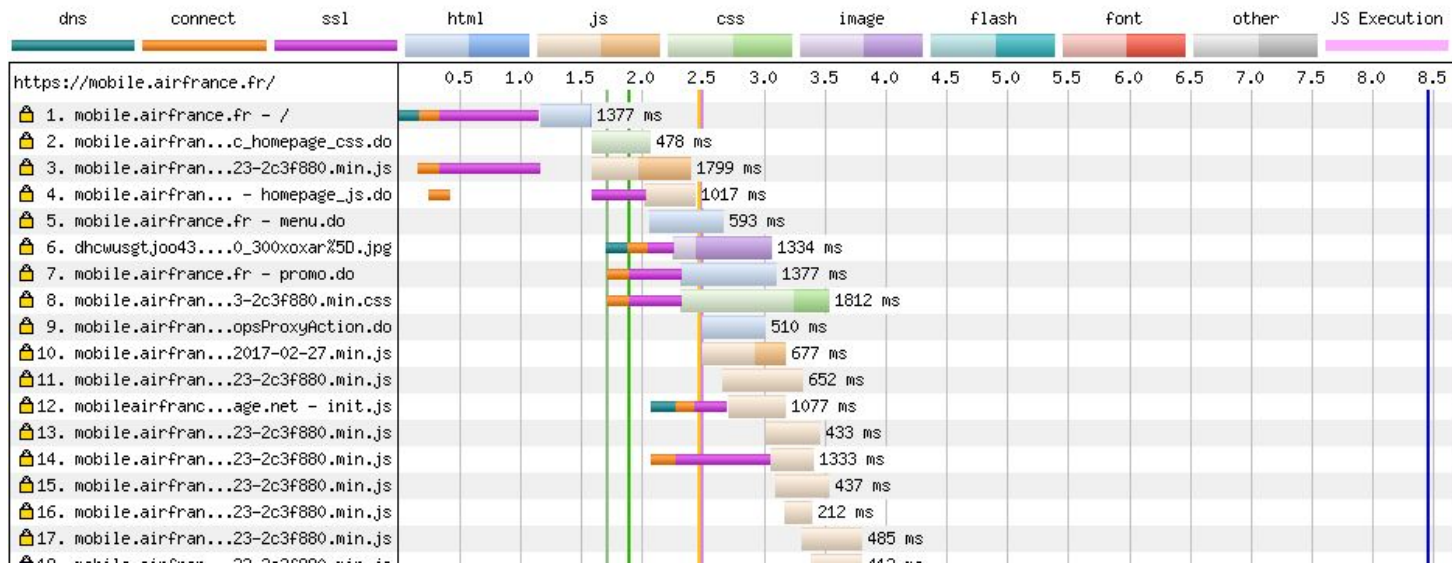
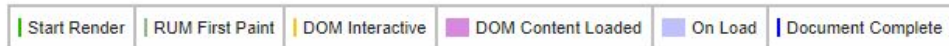
# Previously...

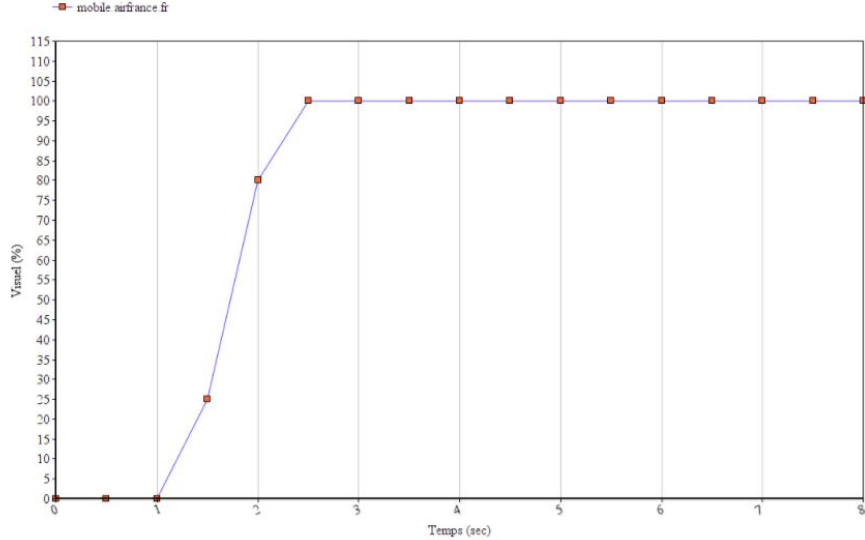
on AMC..

Load Time	First Byte	Start Render	Visually Complete	Speed Index	Result (error code)	Document Complete			Fully Loaded			
						Time	Requests	Bytes In	Time	Requests	Bytes In	Certificates
8.449s	1.567s	1.889s	3.200s	2440	0	8.449s	58	803 KB	8.642s	60	804 KB	55 KB

Colordepth	RUM First Paint	domInteractive	domContentLoaded	loadEvent
24	1.708s	2.453s	2.453s - 2.496s (0.043s)	8.439s - 8.450s (0.011s)

## Waterfall View

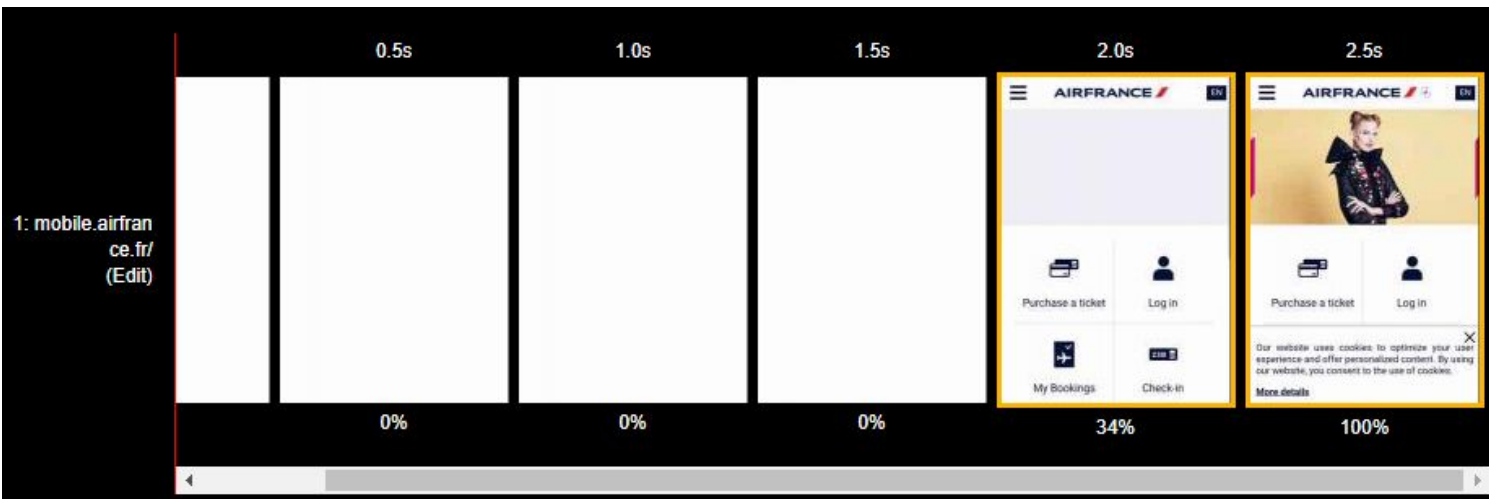




✓ **Règle n°1** : Prioriser les requêtes responsables du visuel : “critical render path”

✓ **Règle n°2** : Diminuer le nombre et poids de ces requêtes (inline, minification, concaténation)

✓ **Règle n°3** : La première requête ne doit pas dépasser 13.8 Ko



Visuel 100% :  
**2.5 sec**

Load :  
**8.5 sec**

# Runtime (en ms) - <http://www.stefankrause.net>

<a href="#">Name</a>	angular-v1.6.3-keyed	angular-v4.1.2-keyed	ember-v2.13.0	react-lite-v0.15.30	react-v15.5.4-keyed	react-v15.5.4-mobX-v3.1.9	react-v15.5.4-redux-v3.6.0	vanillajs-keyed
<a href="#">create rows</a> Duration for creating 1000 rows after the page loaded.	251.88.0 (1.8)	193.17.9 (1.4)	344.613.8 (2.5)	170.09.6 (1.2)	188.910.9 (1.4)	243.99.4 (1.8)	212.214.2 (1.5)	138.55.8 (1.0)
<a href="#">replace all rows</a> Duration for updating all 1000 rows of the table (with 5 warmup iterations).	278.316.7 (1.9)	197.45.3 (1.3)	292.712.1 (2.0)	235.46.8 (1.6)	201.06.4 (1.4)	229.212.2 (1.5)	206.77.3 (1.4)	148.04.5 (1.0)
<a href="#">swap rows</a> Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	14.71.5 (1.0)	13.41.0 (1.0)	16.41.5 (1.0)	30.01.8 (1.9)	14.70.9 (1.0)	18.01.2 (1.1)	17.11.3 (1.1)	11.41.1 (1.0)
<a href="#">create many rows</a> Duration to create 10,000 rows	3108.72162.2 (2.3)	1946.041.8 (1.5)	2569.356.3 (1.9)	2300.951.4 (1.7)	1852.429.0 (1.4)	2217.371.5 (1.7)	1931.735.6 (1.5)	1331.122.2 (1.0)
<a href="#">append rows to large table</a> Duration for adding 1000 rows on a table of 10,000 rows.	454.842.1 (1.5)	324.610.1 (1.1)	524.223.6 (1.8)	2087.565.2 (7.1)	345.610.4 (1.2)	459.847.2 (1.6)	366.410.9 (1.2)	295.312.8 (1.0)
<a href="#">clear rows</a> Duration to clear the table filled with 10,000 rows.	817.637.2 (4.7)	379.911.3 (2.2)	303.974.7 (1.7)	344.126.4 (2.0)	398.48.2 (2.3)	495.128.8 (2.8)	410.99.8 (2.4)	174.84.2 (1.0)
<a href="#">startup time</a> Time for loading, parsing and starting up	118.15.1 (2.9)	84.32.6 (2.1)	245.25.6 (6.0)	44.92.6 (1.1)	70.02.9 (1.7)	87.64.3 (2.2)	93.86.9 (2.3)	40.59.5 (1.0)
<a href="#">slowdown geometric mean</a>	<b>2.08</b>	<b>1.45</b>	<b>2.10</b>	<b>1.92</b>	<b>1.42</b>	<b>1.74</b>	<b>1.56</b>	<b>1.00</b>

## COÛT DEV

→ Apprendre

→ Ré-apprendre

→ Debug



## COÛT UTILISATEUR

→ Temps

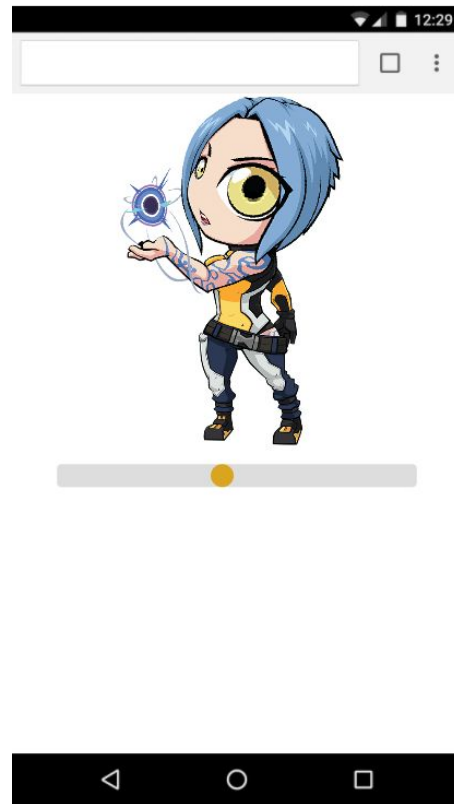
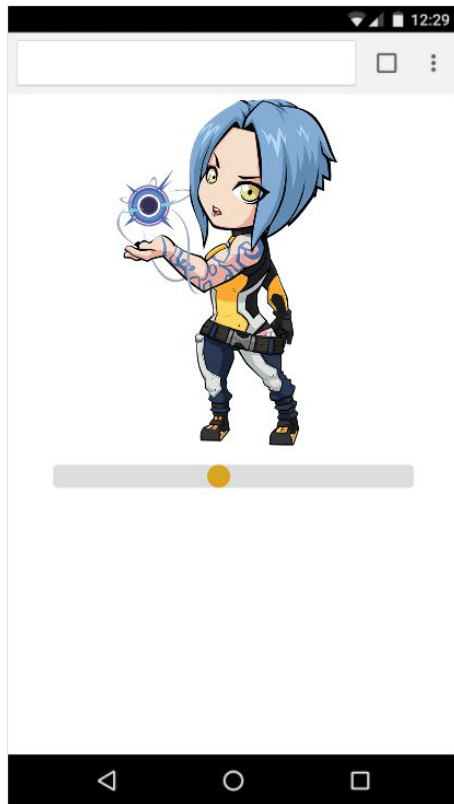
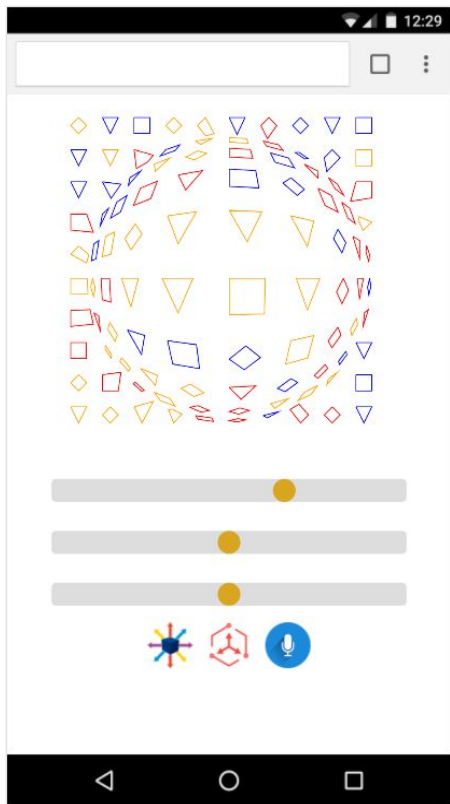
→ Bandwidth

→ Utilisation CPU (batterie)

→ Frame rate

→ Utilisation mémoire

# TD : Fisheye effect



## NATIF

→ Performant

→ Plus de possibilité



## NATIF

- Performant
- Plus de possibilité

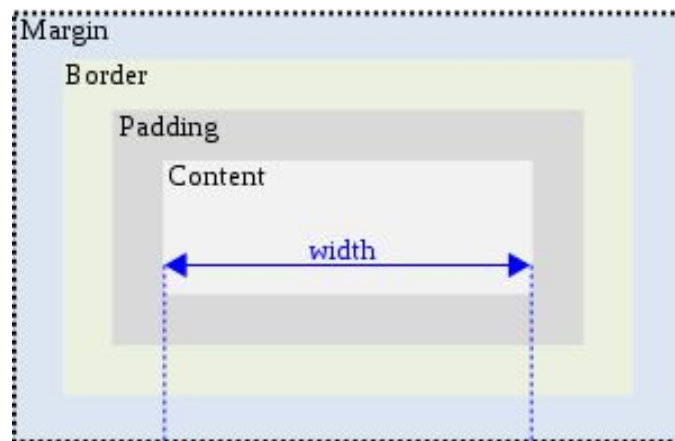
## WEB

- Accessible
- Universel
- Moins de code
- Plus facile

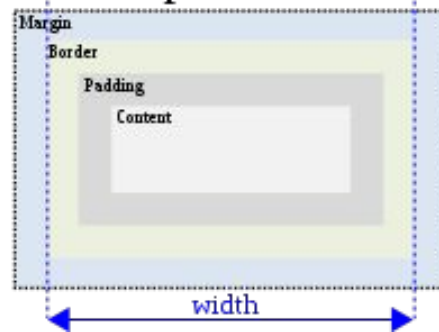
Il fut un temps....



## W3C box model



## Internet Explorer box model



```
currentBtn.attachEvent("onmouseover", showHint);
```

This is the method that adds the event handler in Internet Explorer.

This time you keep the "on" at the beginning of the event name...

You still give the function the name of the handler to run when the event occurs.

That mysterious "false" disappears in attachEvent().

Name	Description	Argument type	Argument name
attachEvent	Similar to W3C's addEventListener method.	String	sEvent
		Pointer	fpNotify
detachEvent	Similar to W3C's removeEventListener method.	String	sEvent
		Pointer	fpNotify
fireEvent	Similar to W3C's dispatchEvent method.	String	sEvent
		Event	oEventObject



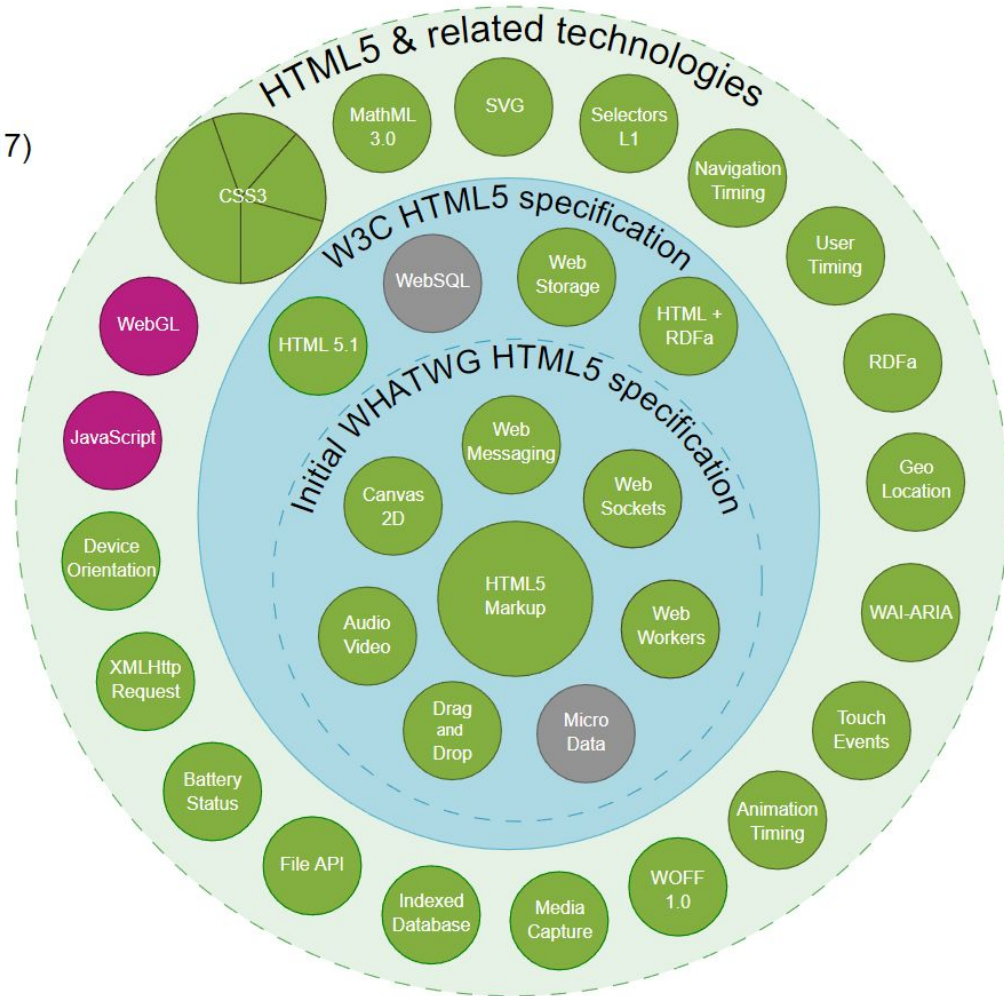
Mais  
aujourd'hui !



# HTML5.1

Taxonomy & Status (September 2017)

-  Recommendation/Proposed
-  Candidate Recommendation
-  Last Call
-  Working Draft
-  Non-W3C Specifications
-  Depreceted or inactive





→ ES6 ou EcmaScript 2015

→ *class, extends, super, getter/setter, static, Map, Set, Promise ...*

→ **Couverture : 97%**

→ IE 11 → 2013

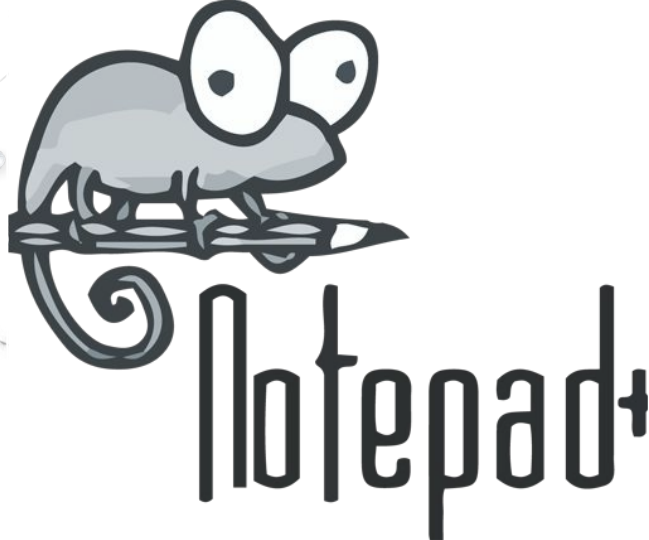
Feature name	Desktop browsers							
	11%	93%	96%	94%	97%	97%	99%	99%
	IE.11	Edge.14	Edge.15	FF.52 ESR	FF.55	CH.60, OP.47 <sup>[1]</sup>	SF.10	SF.10.1
• <a href="#">default function parameters</a>	0/7	7/7	7/7	6/7	7/7	7/7	7/7	7/7
• <a href="#">rest parameters</a>	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5
• <a href="#">spread (...) operator</a>	0/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15
• <a href="#">object literal extensions</a>	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6
• <a href="#">for...of loops</a>	0/9	7/9	9/9	7/9	9/9	9/9	9/9	9/9
• <a href="#">octal and binary literals</a>	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4
• <a href="#">template literals</a>	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5
• <a href="#">RegExp "v" and "u" flags</a>	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5
• <a href="#">destructuring declarations</a>	0/22	21/22	22/22	21/22	22/22	22/22	22/22	22/22
• <a href="#">destructuring assignment</a>	0/24	23/24	24/24	23/24	24/24	24/24	24/24	24/24
• <a href="#">destructuring parameters</a>	0/24	22/24	23/24	21/24	24/24	24/24	24/24	24/24
• <a href="#">Unicode code point escapes</a>	0/2	2/2	2/2	1/2	2/2	2/2	2/2	2/2
• <a href="#">new.target</a>	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
<b>Bindings</b>								
• <a href="#">const</a>	12/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16
• <a href="#">let</a>	10/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12
• <a href="#">block-level function declaration<sup>[1]</sup></a>	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
<b>Functions</b>								
• <a href="#">arrow functions</a>	0/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13
• <a href="#">class</a>	0/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24
• <a href="#">super</a>	0/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8
• <a href="#">generators</a>	0/27	27/27	27/27	25/27	27/27	27/27	27/27	27/27
<b>Built-ins</b>								
• <a href="#">typed arrays</a>	16/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46
• <a href="#">Map</a>	8/19	18/19	19/19	18/19	19/19	19/19	19/19	19/19
• <a href="#">Set</a>	8/19	18/19	19/19	18/19	19/19	19/19	19/19	19/19

Source : <http://kangax.github.io/compat-table/es6/>



Osons

**Javascript** Vanilla !



→ **IDE** (*notepad++ ou autre*)

→ **Navigateur** (*chrome*)

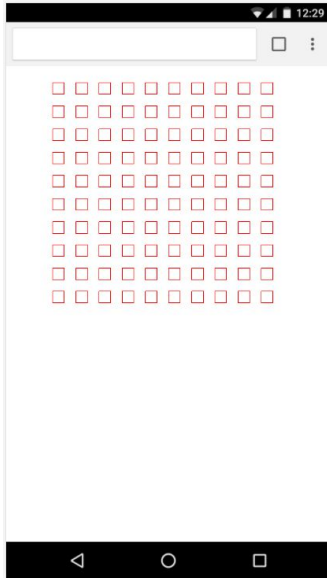
→ **Serveur web** (*wamp, xamp, ...*)

*“ Par où commencer ? ”*

Vous dans 10 minutes...

1

- Point & Polygon
- generatePolygons()



```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Hello World</title>
6     <meta content='user-scalable=0' name='viewport' />
7     <style>
8       * {
9         padding: 0;
10        margin: 0;
11      }
12    </style>
13  </head>
14  <body>
15    <canvas id="myCanvas" width="500" height="500"></canvas>
16    <script>"use strict";
17      /**
18       * Entry point
19       */
20    window.addEventListener("DOMContentLoaded", function() {
21      const canvas = document.getElementById("myCanvas");
22      const ctx = canvas.getContext("2d");
23      ...
24    });
25
26    /**
27     * Javascript Model
28     */
29    class Polygon {
30      ...
31    }
32  </script>
33 </body>
34 </html>
```



**NOT IMPRESSED**

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Hello World</title>
6     <meta content='user-scalable=0' name='viewport' />
7     <style>
8       * {
9         padding: 0;
10        margin: 0;
11      }
12    </style>
13  </head>
14  <body>
15    <canvas id="myCanvas" width="500" height="500"></canvas>
16    <script>"use strict";
17    /**
18     * Entry point
19     */
20    window.addEventListener("DOMContentLoaded", function() {
21      const canvas = document.getElementById("myCanvas");
22      const ctx = canvas.getContext("2d");
23      ...
24    });
25
26    /**
27     * Javascript Model
28     */
29    class Polygon {
30      ...
31    }
32  </script>
33 </body>
34 </html>
```

## CSS : File or Inline ?

- Inline si poids page < 13.8 Ko
- Inline si **style critique**
- Sinon file **async** (link rel dans body)



**NOT IMPRESSED**

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Hello World</title>
6     <meta content='user-scalable=0' name='viewport' />
7     <style>
8       * {
9         padding: 0;
10        margin: 0;
11      }
12    </style>
13  </head>
14  <body>
15    <canvas id="myCanvas" width="500" height="500"></canvas>
16    <script>"use strict";
17      /**
18       * Entry point
19       */
20      window.addEventListener("DOMContentLoaded", function() {
21        const canvas = document.getElementById("myCanvas");
22        const ctx = canvas.getContext("2d");
23        ...
24      });
25
26      /**
27       * Javascript Model
28       */
29      class Polygon {
30        ...
31      }
32    </script>
33  </body>
34 </html>
```

```
1 "use strict";
2 varialeMalDéclarée = 17; // lève une ReferenceError
```

JS :

- Rend **explicite erreur** normalement silencieuse
- Exécution **plus rapide** du js
- **Interdit** Next ES keyword



**NOT IMPRESSED**

[https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Strict\\_mode](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Strict_mode)

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Hello World</title>
6     <meta content='user-scalable=0' name='viewport' />
7     <style>
8       * {
9         padding: 0;
10        margin: 0;
11      }
12    </style>
13  </head>
14  <body>
15    <canvas id="myCanvas" width="500" height="500"></canvas>
16    <script>"use strict";
17      /**
18       * Entry point
19       */
20    window.addEventListener("DOMContentLoaded", function() {
21      const canvas = document.getElementById("myCanvas");
22      const ctx = canvas.getContext("2d");
23      ...
24    });
25
26    /**
27     * Javascript Model
28     */
29    class Polygon {
30      ...
31    }
32  </script>
33 </body>
34 </html>
```

JS :

- Equivalent “*main*” java
- **Fin** du <body> !
- “**DomContentLoaded**” et pas “*load*” !



**NOT IMPRESSED**



# DOMContentLoaded

🌐 Cette traduction est incomplète. Aidez à traduire cet article depuis l'anglais.

L'événement `DOMContentLoaded` est déclenché lorsque le document HTML initial a été complètement chargé et analysé, sans attendre les feuilles de style, images et sous-trames pour terminer le chargement. Un événement très différente - `load` - doit être utilisé seulement pour détecter une page entièrement chargée. **C'est une erreur très répandue d'utiliser `load` quand `DOMContentLoaded` serait beaucoup plus approprié, alors soyez prudents.**

```
<script>
"use strict";
/**
 * Entry point, initialize PolygonManager and listeners
 */
window.addEventListener("DOMContentLoaded", function () {
  // Get canvas and draw polygons
  const canvas = document.getElementById("fisheye");
  const ctx = canvas.getContext("2d");
  ...
});
</script>
```

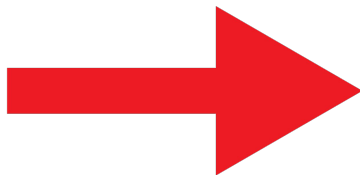
```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Hello World</title>
6     <meta content='user-scalable=0' name='viewport' />
7     <style>
8       * {
9         padding: 0;
10        margin: 0;
11      }
12    </style>
13  </head>
14  <body>
15    <canvas id="myCanvas" width="500" height="500"></canvas>
16    <script>
17      /**
18       * Entry point
19       */
20      window.addEventListener("DOMContentLoaded", function() {
21        const canvas = document.getElementById("myCanvas");
22        const ctx = canvas.getContext("2d");
23        ...
24      });
25
26      /**
27       * Javascript Model
28       */
29      class Polygon {
30        ...
31      }
32    </script>
33  </body>
34 </html>
```

## JS : File or Inline ?

- Inline si poids page < 13.8 Ko
- Inline si **js critique**
- Sinon file

# Javascript ES6

```
/**
 * Point
 * @class
 */
class Point {
  /**
   * @param {Number} x
   * @param {Number} y
   */
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }
  /**
   * @param {Number} a
   * @param {Number} b
   */
  static distance(a, b) {
    const dx = a.x - b.x;
    const dy = a.y - b.y;
    return Math.hypot(dx, dy);
  }
}
```



```
let p1 = new Point(10,10),
    p2 = new Point(20,20);

p1.x = p2.x;

const dist = Point.distance(p1,p2);
```

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Hello World</title>
6     <meta content='user-scalable=0' name='viewport' />
7     <style>
8       * {
9         padding: 0;
10        margin: 0;
11      }
12    </style>
13  </head>
14  <body>
15    <canvas id="myCanvas" width="500" height="500"></canvas>
16    <script>
17      /**
18       * Entry point
19       */
20      window.addEventListener("DOMContentLoaded", function() {
21        const canvas = document.getElementById("myCanvas");
22        const ctx = canvas.getContext("2d");
23        ...
24      });
25
26      /**
27       * Javascript Model
28       */
29      class Polygon {
30        ...
31      }
32    </script>
33  </body>
34 </html>
```

Canvas : zone de dessin

```
1 <canvas id="canvas" width="100" height="100"></canvas>
2
3 <input id="r" type="range" value="250" min="40" max="500">
4
5 <canvas id="canvas2" width="300" height="300"></canvas>
```

HTML ⚙

```
1 input {display:block}
```

```
1 const canvas = document.getElementById("canvas");
2 const ctx = canvas.getContext("2d");
3
4 ctx.rect(35,35,50,50); // (x,y,width,height)
5 ctx.stroke();
6
7 // draw line
8 ctx.beginPath();
9 ctx.moveTo(35,35); // start to (x,y)
10 ctx.lineTo(85,85); // go to (x,y)
11 ctx.lineWidth = 2;
12 ctx.strokeStyle = "red";
13 ctx.stroke();
14
15 const canvas2 = document.getElementById("canvas2");
16 const ctx2 = canvas2.getContext("2d");
17
18 // load image in canvas
19 let img = new Image();
20 img.src = "https://www.gstatic.com/webp/gallery3/1_webp_a.png";
21 img.onload = function () {
22   ctx2.drawImage(img, 0, 0);
23 };
24 // Get pixels from canvas
25 let data = ctx2.getImageData(0,0,canvas2.width,canvas2.height); // (x,y,width,height)
26 // Replace them
27 ctx2.putImageData(data, 0, 0); // (ImageData,x,y)
```

JAVASCRIPT ⚙



```
1 <canvas id="canvas" width="100" height="100"></canvas>
2
3 <input id="r" type="range" value="250" min="40" max="500">
4
5 <canvas id="canvas2" width="300" height="300"></canvas>
```

HTML ⚙

```
1 input {display:block}
```

```
1 const canvas = document.getElementById("canvas");
2 const ctx = canvas.getContext("2d");
3
4 ctx.rect(35,35,50,50); // (x,y,width,height)
5 ctx.stroke();
6
7 // draw line
8 ctx.beginPath();
9 ctx.moveTo(35,35); // start to (x,y)
10 ctx.lineTo(85,85); // go to (x,y)
11 ctx.lineWidth = 2;
12 ctx.strokeStyle = "red";
13 ctx.stroke();
14
15 const canvas2 = document.getElementById("canvas2");
16 const ctx2 = canvas2.getContext("2d");
17
18 // load image in canvas
19 let img = new Image();
20 img.src = "https://www.gstatic.com/webp/gallery3/1_webp_a.png";
21 img.onload = function () {
22   ctx2.drawImage(img, 0, 0);
23 };
24 // Get pixels from canvas
25 let data = ctx2.getImageData(0,0,canvas2.width,canvas2.height); // (x,y,width,height)
26 // Replace them
27 ctx2.putImageData(data, 0, 0); // (ImageData,x,y)
```

JAVASCRIPT ⚙



```
1 <canvas id="canvas" width="100" height="100"></canvas>
2
3 <input id="r" type="range" value="250" min="40" max="500">
4
5 <canvas id="canvas2" width="300" height="300"></canvas>
```

HTML ⚙

```
1 input {display:block}
```

```
1 const canvas = document.getElementById("canvas");
2 const ctx = canvas.getContext("2d");
3
4 ctx.rect(35,35,50,50); // (x,y,width,height)
5 ctx.stroke();
6
7 // draw line
8 ctx.beginPath();
9 ctx.moveTo(35,35); // start to (x,y)
10 ctx.lineTo(85,85); // go to (x,y)
11 ctx.lineWidth = 2;
12 ctx.strokeStyle = "red";
13 ctx.stroke();
14
15 const canvas2 = document.getElementById("canvas2");
16 const ctx2 = canvas2.getContext("2d");
17
18 // load image in canvas
19 let img = new Image();
20 img.src = "https://www.gstatic.com/webp/gallery3/1_webp_a.png";
21 img.onload = function () {
22   ctx2.drawImage(img, 0, 0);
23 };
24 // Get pixels from canvas
25 let data = ctx2.getImageData(0,0,canvas2.width,canvas2.height); // (x,y,width,height)
26 // Replace them
27 ctx2.putImageData(data, 0, 0); // (ImageData,x,y)
```

JAVASCRIPT ⚙



```
1 <canvas id="canvas" width="100" height="100"></canvas>
2
3 <input id="r" type="range" value="250" min="40" max="500">
4
5 <canvas id="canvas2" width="300" height="300"></canvas>
```

HTML ⚙

```
1 input {display:block}
```

```
1 const canvas = document.getElementById("canvas");
2 const ctx = canvas.getContext("2d");
3
4 ctx.rect(35,35,50,50); // (x,y,width,height)
5 ctx.stroke();
6
7 // draw line
8 ctx.beginPath();
9 ctx.moveTo(35,35); // start to (x,y)
10 ctx.lineTo(85,85); // go to (x,y)
11 ctx.lineWidth = 2;
12 ctx.strokeStyle = "red";
13 ctx.stroke();
14
15 const canvas2 = document.getElementById("canvas2");
16 const ctx2 = canvas2.getContext("2d");
17
18 // load image in canvas
19 let img = new Image();
20 img.src = "https://www.gstatic.com/webp/gallery3/1_webp_a.png";
21 img.onload = function () {
22   ctx2.drawImage(img, 0, 0);
23 };
24 // Get pixels from canvas
25 let data = ctx2.getImageData(0,0,canvas2.width,canvas2.height); // (x,y,width,height)
26 // Replace them
27 ctx2.putImageData(data, 0, 0); // (ImageData,x,y)
```

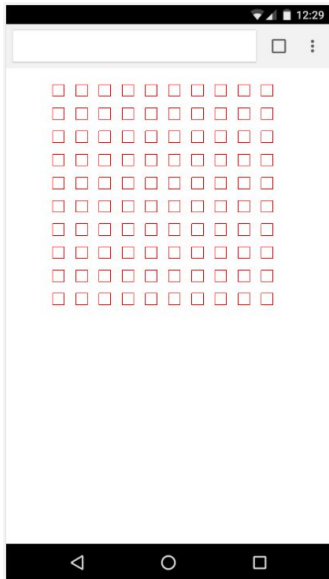
JAVASCRIPT ⚙





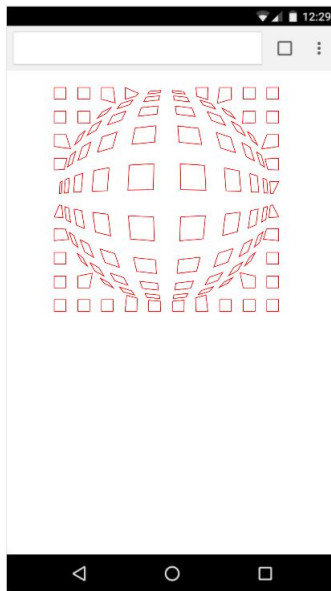
1

- Point & Polygon
- generatePolygons()



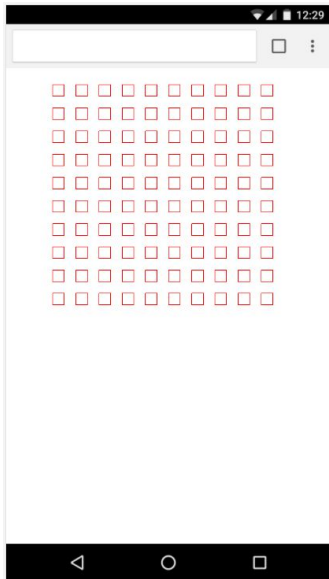
2

Fisheye deformation



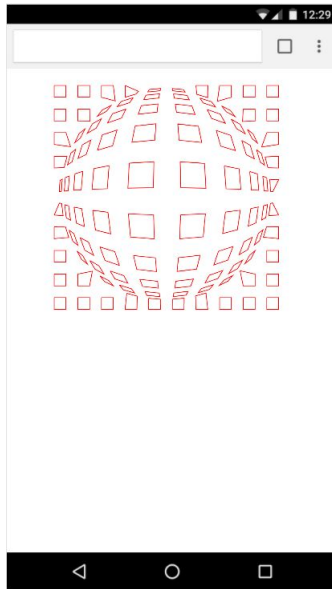
1

- Point & Polygon
- generatePolygons()



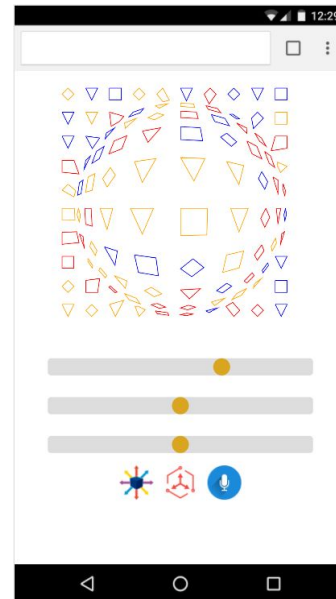
2

Fisheye deformation



3

Listeners



One more thing...

# Devtools : Debug

Nexus 5X 412 x 732 100% DPR: 2.6

Elements Console Sources Network Performance Memory Application Security Audits ADBlock

Sources >> index.html x include.preload.js

Paused on breakpoint

Watch

Call Stack

- Point index.html:68
- setDeformCenter index.html:138
- mobileTouchListener index.html:42

Scope

Local

- this: Point
  - x: 310.67999267578125
  - y: 192.6699981689453

Script

- Global Window

Breakpoints

- index.html:68
  - this.x = x;
- XHR Breakpoints
- DOM Breakpoints
- Global Listeners
- Event Listener Breakpoints

```
54 canvas.addEventListener("mousemove", desktopClickListener,{passive: true});
55 document.getElementById("r").addEventListener("input", rRangeListener,{passive: true});
56 });
57
58 /**
59  * Point
60  * @class
61  */
62 class Point {
63     /**
64      * @param {Number} x
65      * @param {Number} y
66      */
67     constructor(x, y) { x = 310.67999267578125, y = 192.6699981689453
68         this.x = x;
69         this.y = y;
70     }
71     /**
72      * @param {Number} a
73      * @param {Number} b
74      */
75     static distance(a, b) {
76         const dx = a.x - b.x;
77         const dy = a.y - b.y;
78         return Math.hypot(dx, dy);
79     }
80 }
81
82 class ImageManager {
83     /**
84
```

Line 68, Column 5

Console What's New Search Sensors

top Filter All levels

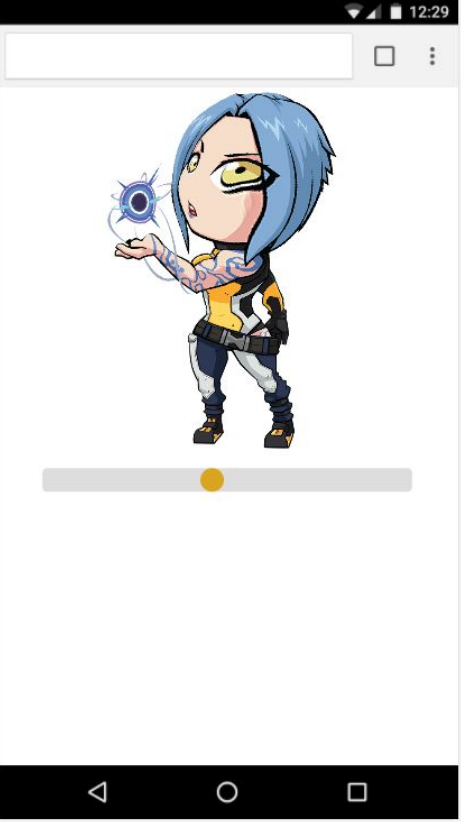
- Hide network
- Preserve log
- Selected context only
- User messages only
- Log XMLHttpRequests
- Show timestamps
- Autocomplete from history

> this

< Point {x: 317.81500244140625}

- x: 317.81500244140625
- \_\_proto\_\_: Object

[Violation] 'touchstart' handler took 161973ms index.html:36



# Devtools : Performance

The screenshot displays the Chrome DevTools Performance tab. On the left, a mobile emulator shows a character with blue hair and a yellow and black outfit, holding a blue orb. Below the character are three horizontal bars with yellow dots. The right side of the image shows the Performance tab interface. The top bar includes navigation options like Elements, Console, Sources, Network, Performance, Memory, Application, Security, Audits, and AdBlock. The Performance tab is active, showing a timeline with a total duration of 1.04 s to 2.09 s. A donut chart indicates a total of 1046 ms, broken down into Scripting (1001.3 ms), Rendering (0.1 ms), Painting (0.1 ms), Other (2.5 ms), and Idle (42.4 ms). Below the chart, the Console shows several [Violation] messages for touchstart and touchmove handlers, with durations ranging from 828ms to 1444ms. The bottom of the interface includes a search bar, a filter dropdown set to 'top', and various settings like 'Hide network', 'Preserve log', 'Selected context only', 'User messages only', 'Log XMLHttpRequests', 'Show timestamps', and 'Autocomplete from history'.

Summary Bottom-Up Call Tree Event Log

Range: 1.04 s – 2.09 s

1046 ms

- 1001.3 ms Scripting
- 0.1 ms Rendering
- 0.1 ms Painting
- 2.5 ms Other
- 42.4 ms Idle

Console What's New Search

top Filter All levels

- Hide network
- Preserve log
- Selected context only
- User messages only
- Log XMLHttpRequests
- Show timestamps
- Autocomplete from history

```
[Violation] 'touchstart' handler took 992ms
[Violation] 'touchstart' handler took 952ms
[Violation] 'touchmove' handler took 1059ms
[Violation] 'touchmove' handler took 852ms
[Violation] 'touchmove' handler took 828ms
[Violation] 'touchstart' handler took 1444ms
[Violation] 'touchstart' handler took 1224ms
```

# Devtools : Performance

The image shows the Chrome DevTools Performance tab. On the left is a mobile app interface with a character and a slider. On the right is the performance analysis panel.

**Performance Panel:**

- Buttons: Screenshots, Memory
- Settings:  Disable JavaScript Samples,  Enable advanced paint instrumentation (slow)
- Network: No throttling
- CPU: 5x slowdown
- Timeline: Shows a red bar for a performance violation between 1440ms and 1520ms.
- Legend: JS Heap [8.6 MB - 8.7 MB], Documents [2 - 2], Nodes [79 - 79], Listeners [17 - 17], GPU Memory

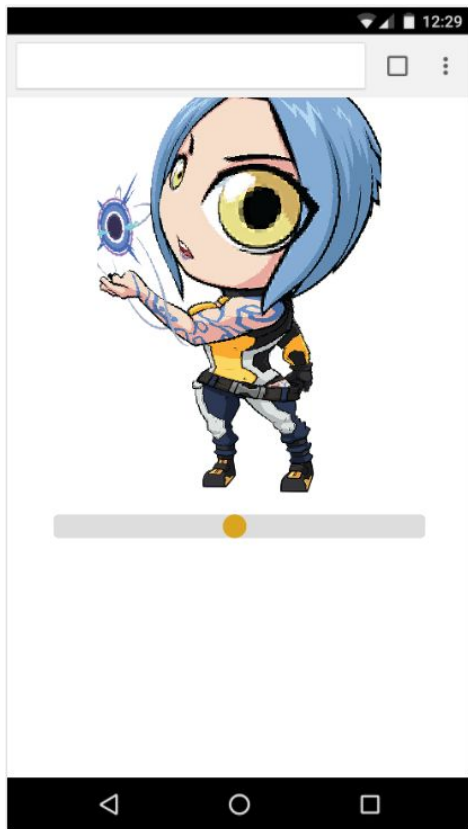
**Summary:**

- Range: 1.41 s - 1.61 s
- Donut chart showing: 196 ms total
- Breakdown: Scripting (173.4 ms), Rendering (0.2 ms), Painting (0.2 ms), Other (2.3 ms), Idle (20.1 ms)

**Console:**

- Filters: top, Filter, All levels
- Options:  Hide network,  Preserve log,  Selected context only,  User messages only,  Log XMLHttpRequests,  Show timestamps,  Autocomplete from history
- Log entries (all violations):
  - [Violation] 'touchstart' handler took 173ms
  - [Violation] 'touchstart' handler took 178ms
  - [Violation] 'touchmove' handler took 193ms
  - [Violation] 'touchmove' handler took 156ms
  - [Violation] 'touchmove' handler took 162ms

# Fisheye image (performance)



→ Utilisation de “*shaders*” mais pour le TD on reste en canvas ‘2d’

```
/**
 * Deform Image with Fisheye formula
 */
fishEyeDeform () {
  const data = this.img.data;
  const width = this.canvas.width;
  const R = ImageManager.r;
  const deformCenterX = this.deformCenter.x;
  const deformCenterY = this.deformCenter.y;
  // Iterate on pixels array
  let i = data.length;
  while (i-=4) {
    const x = (i / 4) % width;
    const y = Math.floor((i / 4) / width);
    const dx = deformCenterX - x;
    const dy = deformCenterY - y;
    const dist = Math.hypot(dx, dy);
    // More perf
    if (dist < R) {
      // Compute new X and Y
      const scale = Math.sin(dist/R*Math.PI/2);
      const x2 = Math.round(deformCenterX - dx * scale);
      const y2 = Math.round(deformCenterY - dy * scale);
      const i2 = (y2 * width + x2) * 4;
      this.deformImg.data[i ] = data[i2];
      this.deformImg.data[i+1] = data[i2+1];
      this.deformImg.data[i+2] = data[i2+2];
      this.deformImg.data[i+3] = data[i2+3];
    }
  }
}
```

## Règle n°1 :

Optimisation ssi nécessaire !

## Règle n°2 :

Ne sacrifiez pas la lisibilité du code !

## Règle n°3 :

Factoriser l'algorithme avant le langage !

## Optimisations faites :

pixels / 2 => 30%

formule => 30%

inline => 20%

while => 10%