

Cours N° 2

Nouveautés HTML 5

Programmation Web
Master GL - 2019- 2020

HTML 1999

HTML 5 nouvelle version : HTML, XHTML , DOM

HTML 5 : Project de w3c.org

5th Revision of HTML standards [W3C]

Structuration et présentation du contenu pour le



W3C(World Wide Web Consortium)

HTML5: Caractéristiques

- HTML 5 introduit un certain nombre de balises et d'attributs (CSS3 support, Media Integration, Graphics, Local Storage, Local SQL, offline apps etc).
- HTML 5 met l'accent sur l'aspect sémantique des éléments
- Amélioration de communiquer facilement (IHM)
- Meilleure caractéristique pour les traitements des erreurs
- Nouvelles caractéristiques doivent être basées sur : HTML, CSS, DOM et JavaScript.
- Minimiser le besoin à l'élément externe (Flash plugin or Quick Times ou ...)
- HTML5 est indépendant de la machine
- Développement de HTML5 est connu par les développeurs d'internet
- Les APIs: canvas ..

- Firefox, Safari, Chrome, Opera, and mobile browsers **supportent** canvas, video, geolocation, local storage, and more.
- Even Microsoft supports most HTML5 features in Internet Explorer 9.
- **Google** supporte **microdata annotations**(balises sémantiques)
Microdata, Microformats and RDFa



SUPPORT MOBILE

Programmation Web

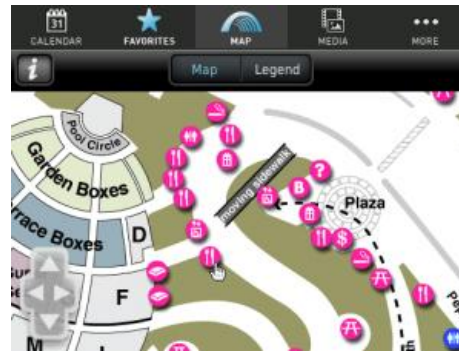


SUPPORT MOBILE

Exemple des application BlackBerry

HTML5/WebWorks

BlackBerry HTML5/WebWorks is
an application platform that enables
developers to create standalone
applications using modern
and standardized web technologies



The Hollywood Bowl for BlackBerry PlayBook



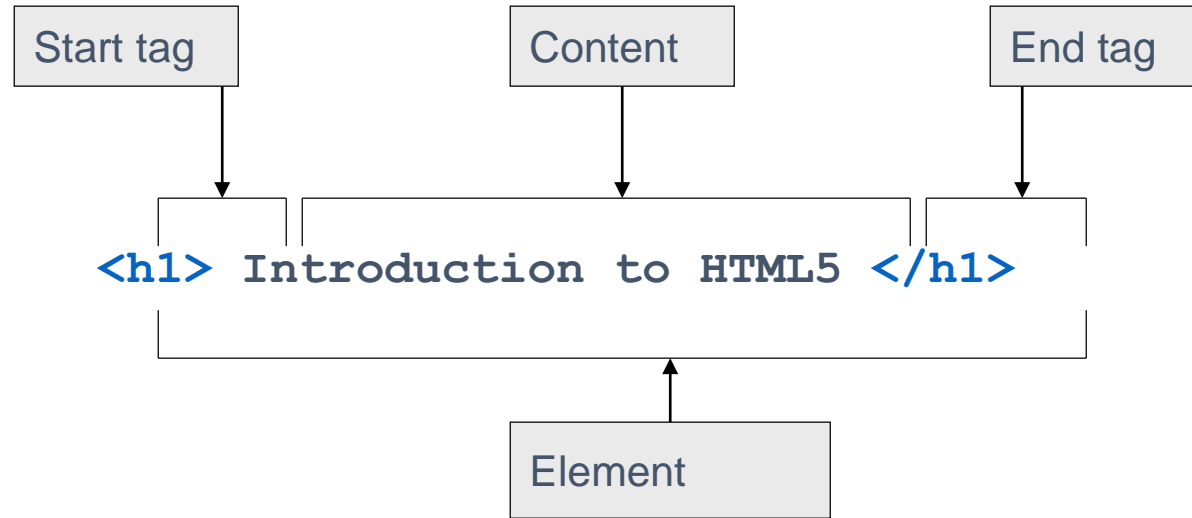
Guitar Chords for the BlackBerry PlayBook



Facebook for the BlackBerry PlayBook



The Hockey News for the BlackBerry PlayBook



- commentaire HTML5 (identique à XHTML et XML):

`<!-- This is a comment -->`

HTML 5 : Les APIs

- **Canvas**
- **<svg>**
- **Web Forms**
- Geolocation:
- Web Storage
 - Local Storage
 - Session Storage
- Offline Applications
- <audio>,
- <video>
- Indexed DB
- Web Workers

HTML 5 : les nouveaux éléments form, input

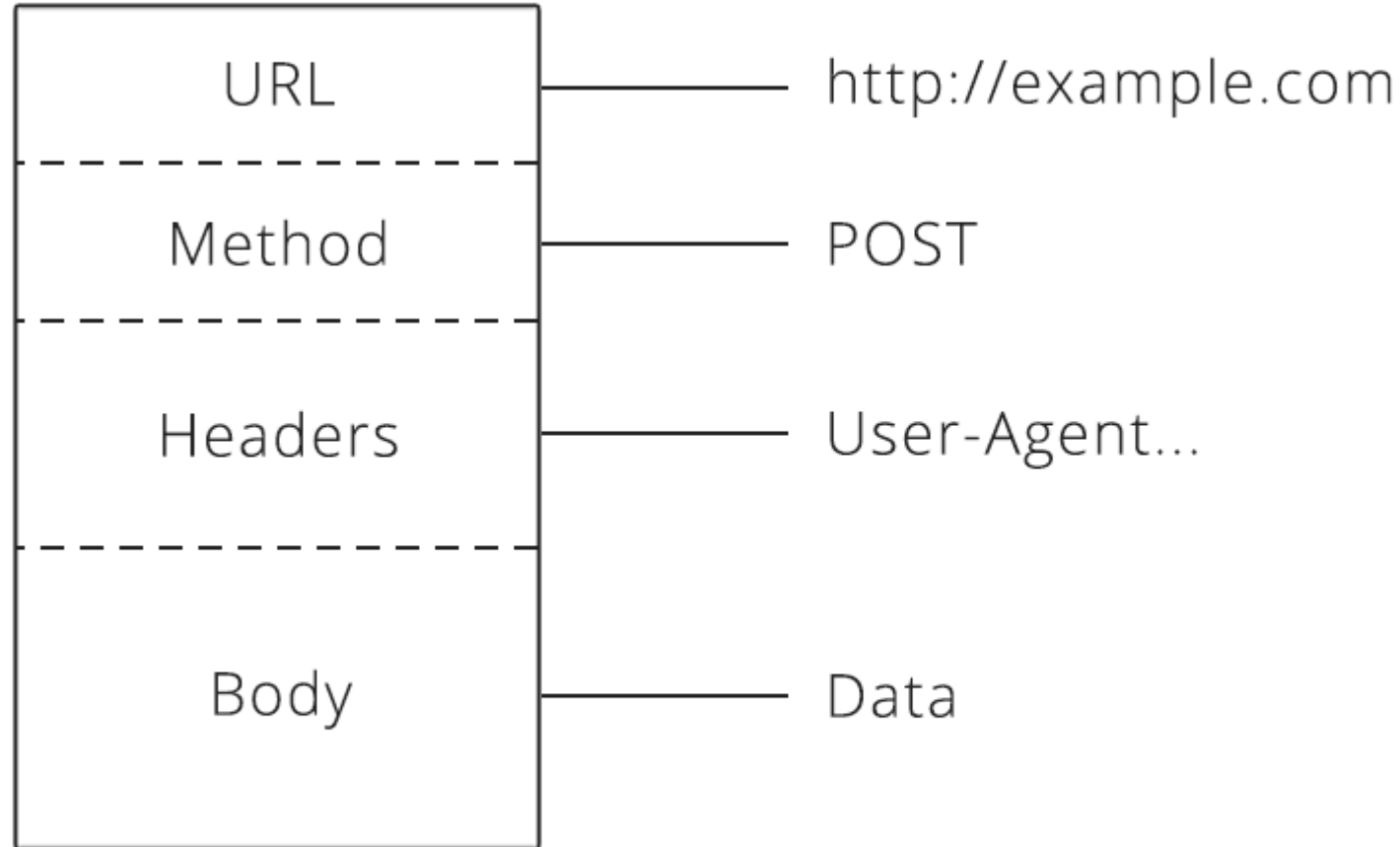
Autre nouveau élément

New

- `<datalist>`
- `<output>`
- `<meter>`
- `<progress>`
- `<keygen>`
- `<fieldset>`
- `<legend>`
- `<textarea>`
- `<label>`
- `<select>`
- `<option>`
- `<optgroup>*`

HTML5 – Les métadonnées

- Une **métadonne** est définie par la méthode **POST** et les **headers** fournies dans la requête.
- Une métadonne est une donnée qui est utilisée pour décrire d'autres données.
- Les métadonnées sont accessibles via l'API **XMLHttpRequest**.
- L'utilisateur peut accéder aux métadonnées d'une requête via l'API **XMLHttpRequest**.



ées sont

éciales

elles sont

Request

HTML5 – Quelques métadonnées

Métadonnée	Description	Exemple
author	Donne l'auteur de la page	<code><meta name="author" content="Mostefai Mohammed Amine" /></code>
description	Donne un résumé sur la page	<code><meta name="description" content="Un article sur quelques nouveautés sur HTML5" /></code>
keywords	Mots clé relatifs à la page	<code><meta name="keywords" content="HTML5, developpement web, formation, nouveauté" /></code>
refresh	Rafraîchissement automatique	<code><meta http-equiv="refresh" content="30" /></code>
refresh	Redirection après N secondes	<code><meta http-equiv="refresh" content="30" url="http://www.esi.dz" /></code> <code></head></code>
pragma	Empêche le navigateur de mettre la page en cache	<code><meta http-equiv="pragma" content="no-cache" /></code>
expires	Indique l'expiration et force le navigateur à recharger la page	<code><meta http-equiv="Expires" content="03 Jun 2013 14:30:00"/></code>

Incorporer des elements (HTML5)

La balise ***embed*** est utilisée pour incorporer des éléments externes (par exemple, des animations flash)

L'attribut ***src*** indique l'url de L'élément externe

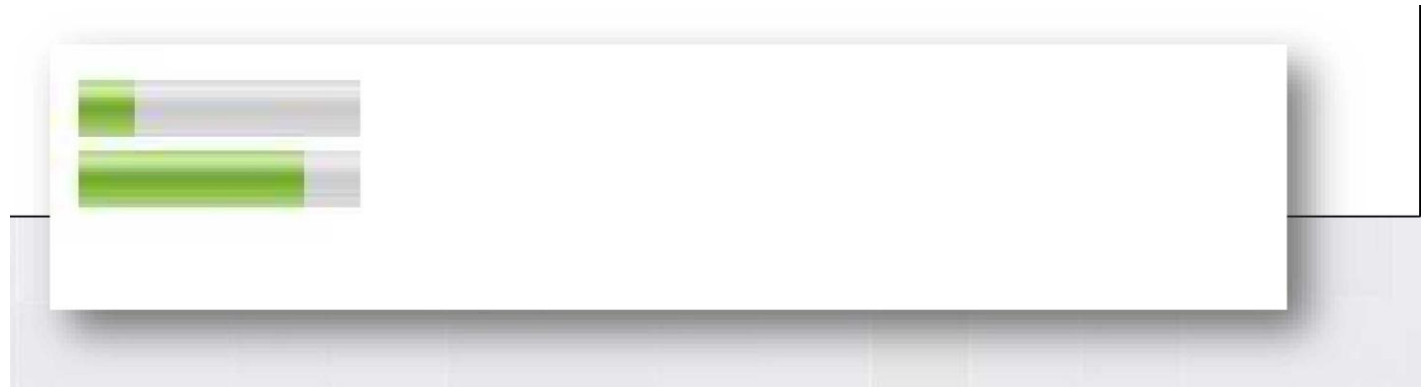
```
<body>  
<embed src="swf/speakers.swf"></embed>  
</body>
```



Montrer la progression (HTML5)

- La balise **meter** utilisee pour montrer la progression des actions
- L'attribut **value** indique la progression en cours, **min** et **max** indique l'intervalle de progression

```
<meter value="2" min="0" max="10">
Vient juste de commencer
</meter>
```



HTML 5: Attributs Input

- o autocomplete .
- o novalidate
- o autofocus .
- o min .
- o max .
- o step .
- o multiple .
- o pattern (regexp) .
- o placeholder .
- o required
- o form .
- o formaction (formoverrides).
- o formenctype (formoverrides).
- o formmethod (formoverrides).
- o formnovalidate (formoverrides).
- o formtarget (formoverrides).
- o height .
- o width .
- o list .
- .

Example

```
<p>  
<label for="inputID">Label: </label>  
<input id="inputID" name="inputName"  
  placeholder="placeholder text"  
  pattern="\w{6,9}"  
  required  
  autofocus  
  type="text" />  
</p>
```

Example

```
<LABEL>
```

```
<p>
```

```
<label for="inputID">Label: </label>
```

```
<input id="inputID" name="inputName"  
  placeholder="placeholder text"  
  pattern="\w{6,9}"  
  required  
  autofocus  
  type="text" />
```

```
</p>
```


PLACEHOLDER ATTRIBUTE

```
<p>  
<label for="inputID">Label: </label>  
<input id="inputID" name="inputName"  
  placeholder="placeholder text"  
  pattern="\w{6,9}"  
  required  
  autofocus  
  type="text" />  
</p>
```

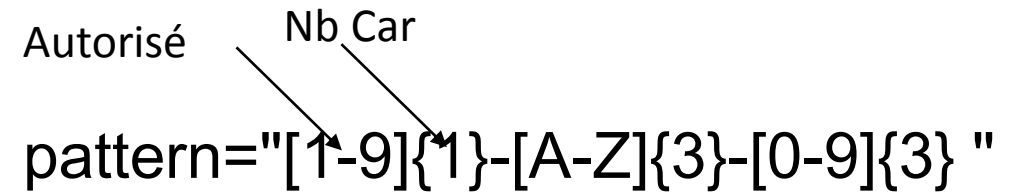
Exemple

PATTERN ATTRIBUTE

```
<p>
<label for="inputID">Label: </label>
<input id="inputID" name="inputName"
  placeholder="placeholder text"
  pattern="\w{6,9}"
  required
  autofocus
  type="text" />
</p>
```

Autorisé Nb Car

pattern="[1-9]{1}-[A-Z]{3}-[0-9]{3} "



Voici quelques exemples d'expressions régulières:

\d représente les chiffres

\w désigne un caractère quelconque

L'* nombre quelconque de caractères

[A-Z]{3}[0-9]*

^ et \$ indiquent respectivement que l'expression doit commencer et finir

REQUIRED ATTRIBUTE

```
<p>  
<label for="inputID">Label: </label>  
<input id="inputID" name="inputName"  
  placeholder="placeholder text"  
  pattern="\w{6,9}"  
  required  
  autofocus  
  type="text" />  
</p>
```

AUTOFOCUS ATTRIBUTE

```
<p>  
<label for="inputID">Label: </label>  
<input id="inputID" name="inputName"  
  placeholder="placeholder text"  
  pattern="\w{6,9}"  
  required  
  autofocus  
  type="text" />  
</p>
```

TYPE ATTRIBUTE

```
<p>  
<label for="inputID">Label: </label>  
<input id="inputID" name="inputName"  
  placeholder="placeholder text"  
  pattern="\w{6,9}"  
  required  
  autofocus  
  type="text" />  
</p>
```

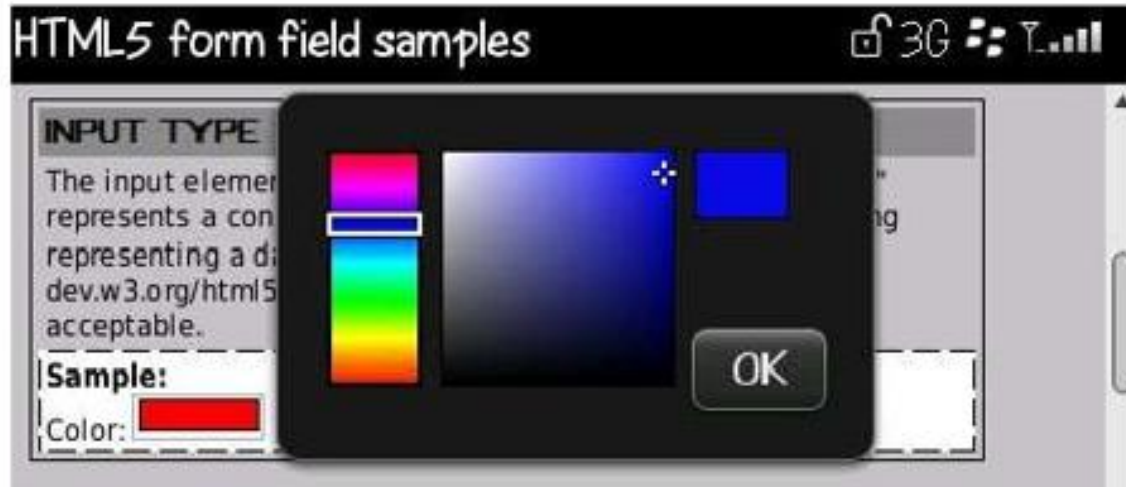

13 NEW <INPUT> TYPES IN HTML5

- color
- url
- tel
- email
- number
- range
- search
- date
- datetime
- datetime-local
- month
- time
- week

Input de HTML5

COLOR ON BLACKBERRY

Color: `<input type="color"
id="txtColor" value="#FF0000"/>`



URL

```
<input id="url" name="url"  
  type="url"  
  placeholder="http://www.x.com"  
  pattern="^http(s)?://.*"  
  required />
```

datalist

```
<p>  
<label for="url">Web Address: </label>  
<input id="url" name="url"  
      type="url"  
      placeholder="http://www.domain.com"  
      required  
      list="mydatalist"/ >  
</p>  
<datalist id="mydatalist">  
  <option value="http://www.standardista.com"/>  
    <option value="http://www.apress.com" />  
    <option value="http://www.evotech.net" />  
</datalist>
```

datalist

```
<datalist id="mydatalist">  
<option value="http://www.standardista.com"  
  label="standardista" />  
  <option value="http://www.apress.com"  
    label="apress" />  
    <option value="http://www.evotech.net"  
      label="Evotech"/>  
</datalist>
```

Web Address:

http://www.standardista.com	standardista
http://www.apress.com	apress
http://www.evotech.net	Evotech

PHONE NUMBERS

```
<label for="tel">Telephone: </label>  
<input id="tel" type="tel" name="tel"  
  placeholder="XXX-XXX-XXXX"  
  pattern="[0-9]{3}-[0-9]{3}-[0-9]{4}"  
  required />
```

EMAIL ADDRESSES

```
<label for="email">Email: </label>
```

```
<input id="email" name="email"  
  type="email"  
  placeholder="you@domain.com"
```

multiple

```
required />
```

NUMBER

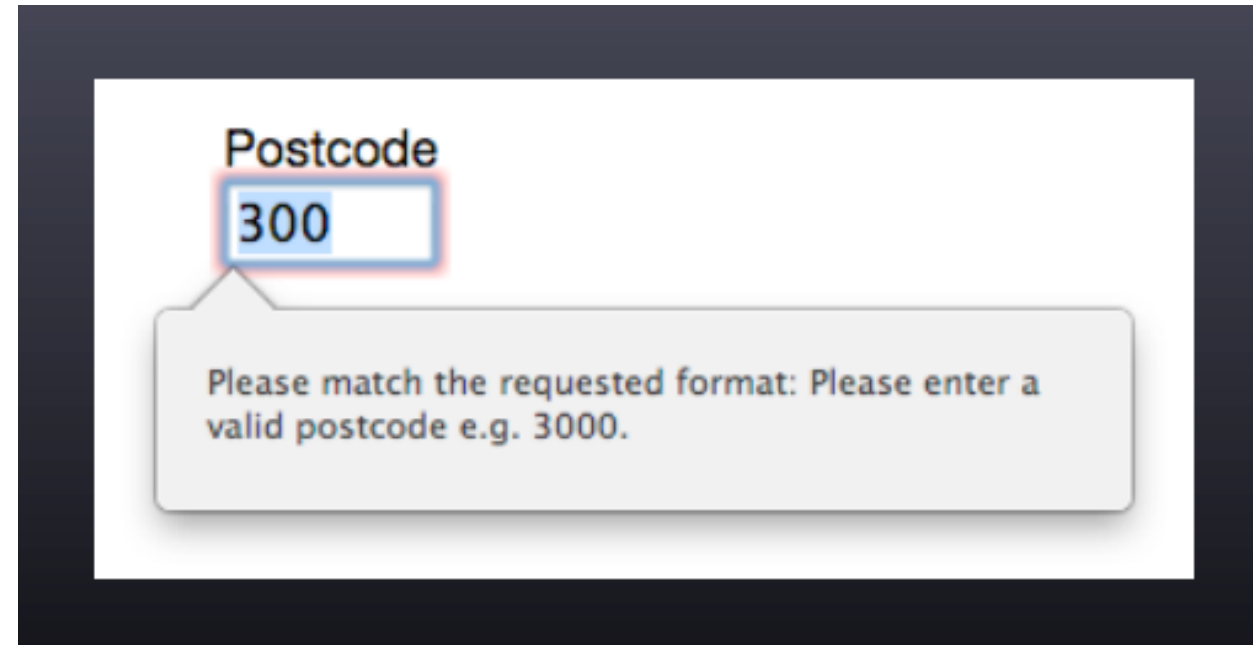
```
<input name="n" type="number"/>
```

- min
- max
- step

```
<input id="nickels" name="nickels"
type="number"
placeholder="0, 5, 10 &hellip;"
pattern="[0-9]*[05]"
min="0"
max="1000"
step="5"
required />
```

MESSAGES D'ERREUR

```
<input pattern="\d{4}"  
title="Please enter a  
valid postcode e.g.  
3000">
```



Mot de passe

Deviner le mot de passe

- Fouille exhaustive
 - Explorer toutes les combinaisons possibles. Par exemple, si un mot de passe est codé sur deux chiffres, on utilise les 100 combinaisons possibles
- Fouille intelligente
 - Exploiter toute information disponible.
 - Cibler la recherche

Pourquoi les mots de passes
doivent être longs?

Fouille exhaustive

Il existe des outils qui peuvent “essayer” approximativement 500.000 mots de passe à la seconde

Fouille exhaustive

- On peut tester 500.000 mots de passe à la seconde
- 4 chiffres :
 - 10.000 possibilités (donc moins d'une seconde),
- 6 chiffres :
 - cela prend 2 secondes
- 8 chiffres :
 - cela prend 200 secondes (toujours moins d'une heure)
- 16 chiffres :
 - 634 années,

Fouille exhaustive

- On peut tester 500.000 mots de passe à la seconde
- 4 chiffres :
 - 10.000 possibilités (donc moins d'une seconde),
- 6 chiffres :
 - cela prend 2 secondes
- 8 chiffres :
 - cela prend 200 secondes (toujours moins d'une heure)
- 16 chiffres :
 - 634 années,

Mot de passe – sécurité

Avec une taille de mots de passe = 8

- Caractères alphabétiques : 5 jours
- Chiffres et caractères alphabétiques : 2 mois (65 jours)
- Chiffres, caractères alpha et spéciaux : quelques années
- Chiffres, caractères majuscules et minuscules : quelques centaines d'années

Fouille intelligente

- Dictionnaires :
 - En général
 - Des villes
 - Des prénoms
 - Etc
- Deviner des lettres
 - on connaît l'utilisateur,
 - Statistique. Le chiffre « 0 » existe probablement

Conseils donnés aux utilisateurs

Toujours mettre un mot de passe

Un mot de passe devrait:

- avoir une longueur raisonnable (généralement, plus de 8 caractères)

- être constitué de chiffres, de lettres, de ponctuations ou caractères spéciaux

- ne pas être un mot existant dans le dictionnaire

- être modifié régulièrement

- être différent pour chaque système

Conseils donnés aux utilisateurs

Changer les mots de passe par défaut laissés à l'installation

Éviter les mots de passe trop évidents

(ex: noms de villes, prénoms)

Faire intervenir le système

- Forcer les utilisateurs à utiliser un mot de passe généré par le système
- Temps de validité des mots de passe pour forcer un changement fréquent
- Limiter le nombre d'essais ratés

API : SVG PROGRAMMING

SVG: SVG, pour **Scalable Vector Graphics**, est un langage XML pour faire des dessins vectoriels en 2D.

- une recommandation du W3C.
- javascript pour les canvas vs XML pour SVG

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1"
height="190">
  <polygon points="100,10 40,180 190,60 10,60
160,180"
  style="fill:lime;stroke:purple;stroke-width:5;fill-
rule:evenodd;">
</svg>
```



Avantages de l'utilisation SVG sur les autres formats d'image (JPEG ou GIF) sont:

- Images SVG peuvent être créés et édités avec ne importe quel éditeur de text
- Images SVG peuvent être recherchés, indexés, scriptés, et compressés
- Images SVG sont évolutive
- Images SVG peuvent être imprimés de haute qualité à toute résolution
- Les images SVG gardent la même qualité dans un zoom

HTML 5 : Canvas

- **CANVAS** : L'avenir des graphiques sur le Web



- **CANVAS** : L'avenir des graphiques sur le Web

- Créé par Apple pour les « [dashboard widgets](#) »
- plateforme de dessin 2D dans le navigateur

Permet de dessiner au sein de la page web : HTML `<canvas>`

- Extensible via une API JavaScript

Dessiner sur l'élément CANVAS avec JavaScript

Utilise que JavaScript et HTML - pas de plugins

- Vous pouvez avoir plusieurs `<canvas>` éléments sur une seule page HTML.
- maintenant développé comme une spécification du W3C

- Exemple dashboard widgets



- **CANVAS :Méthode**

- Context methods
- beginPath()
- moveTo()
- lineTo()
- fill()
- fillRect()
- arc()
- addColorStop()
- drawImage()
- creatRadialGradient

- **CANVAS :Méthode**

Propriété	Description
fillStyle	Définit ou retourne la couleur, le dégradé ou le motif utilisé pour remplir le dessin (ex. : <code>ctx.fillStyle="#FF0000";</code>)
strokeStyle	Définit ou retourne la couleur, dégradé ou motif utilisé pour les chemins (strokes ou paths)
shadowColor	Définit ou retourne la couleur à utiliser pour les ombres
shadowBlur	Définit ou retourne le niveau de flou à utiliser pour les ombres
shadowOffsetX	Définit ou retourne la distance horizontale entre l'ombre et la forme
shadowOffsetY	Définit ou retourne la distance verticale entre l'ombre et la forme

- Création:

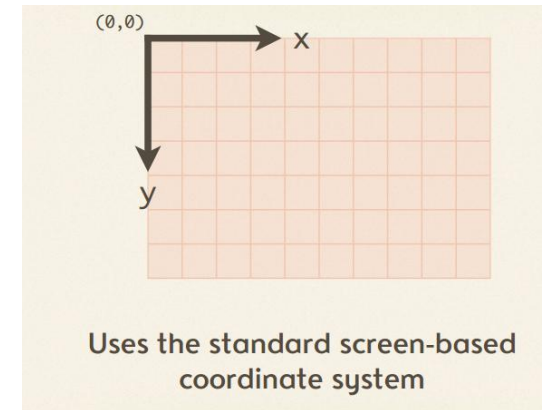
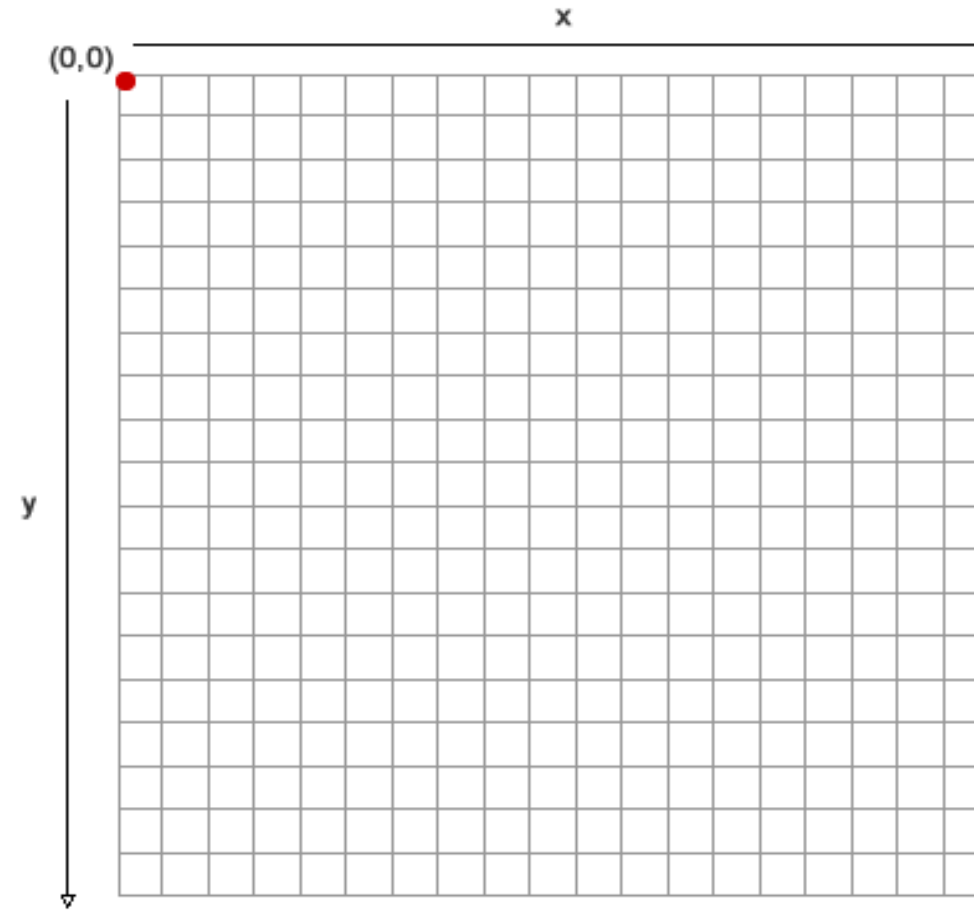
```
<canvas height="600" width="800"></canvas>
```



HEIGHT AND WIDTH NEED TO BE SET EXPLICITLY

HTML 5 – Les Canvas

- **2D DRAWING:**



HTML 5 – Les Canvas

grab the canvas
element

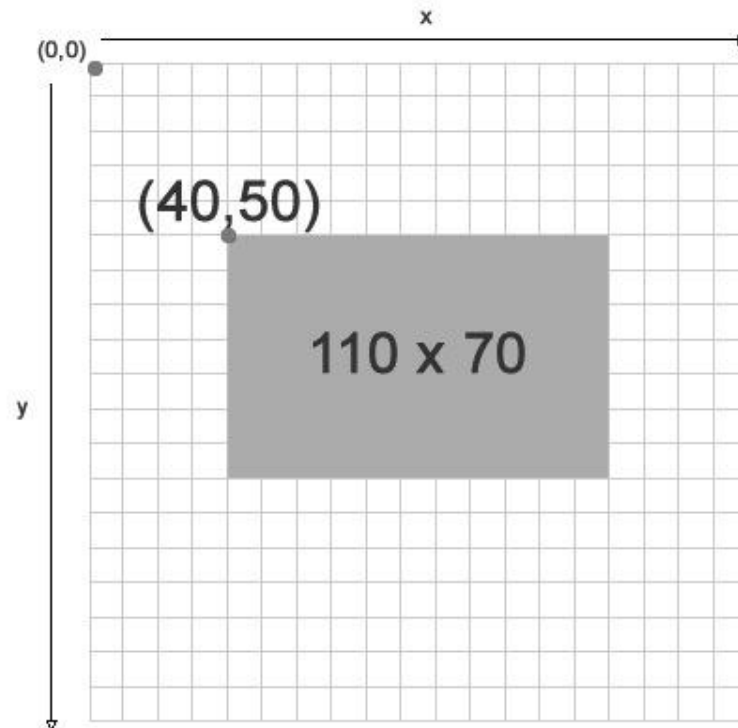
```
var mycanvas = document.getElementById("the_canvas");  
  
var context = mycanvas.getContext("2d");
```

set up a 2D context

Dessinant un rectangle

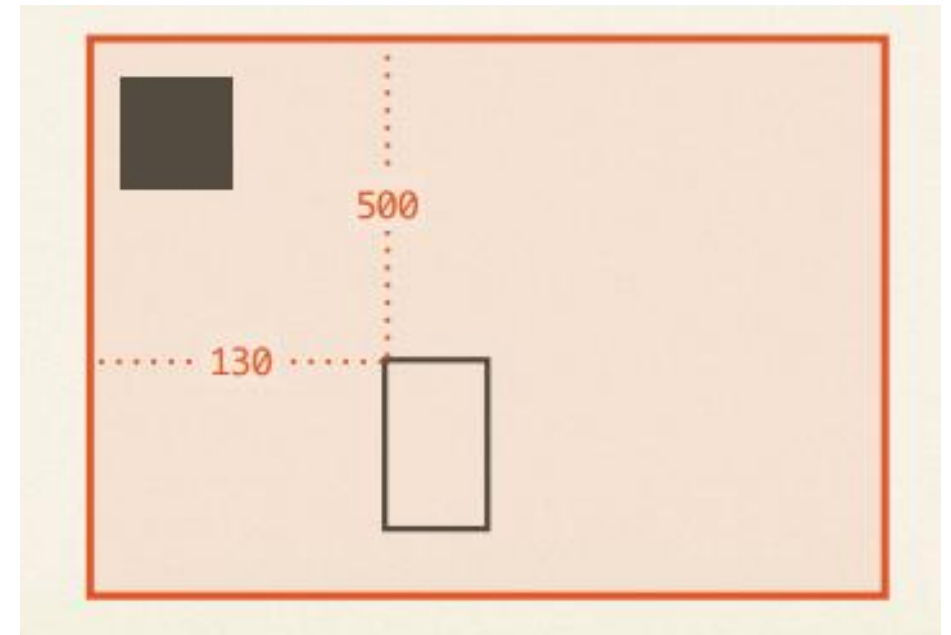
```
context.fillStyle = "rgba(0, 204, 204, 1)";
```

```
context.fillRect(40,50,110,70);
```



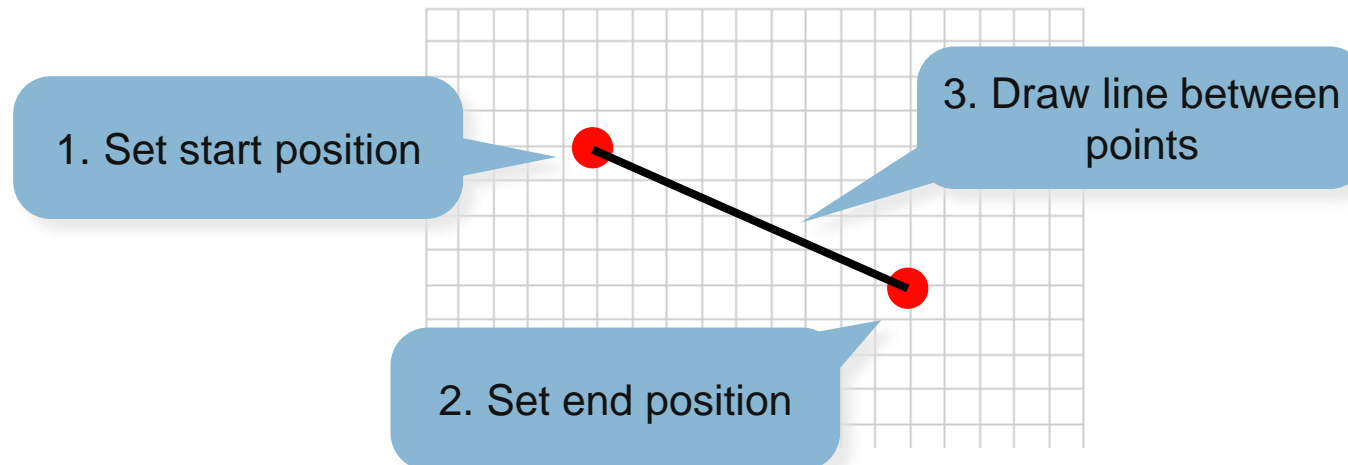
HTML 5 – Les Canvas

- `ctx.fillStyle = 'rgb(65, 60, 50)';`
- `ctx.fillRect(25, 50, 100, 100);`
- `ctx.strokeStyle = 'rgb(65, 60, 50)';`
- `ctx.strokeRect(130, 500, 40, 70);`

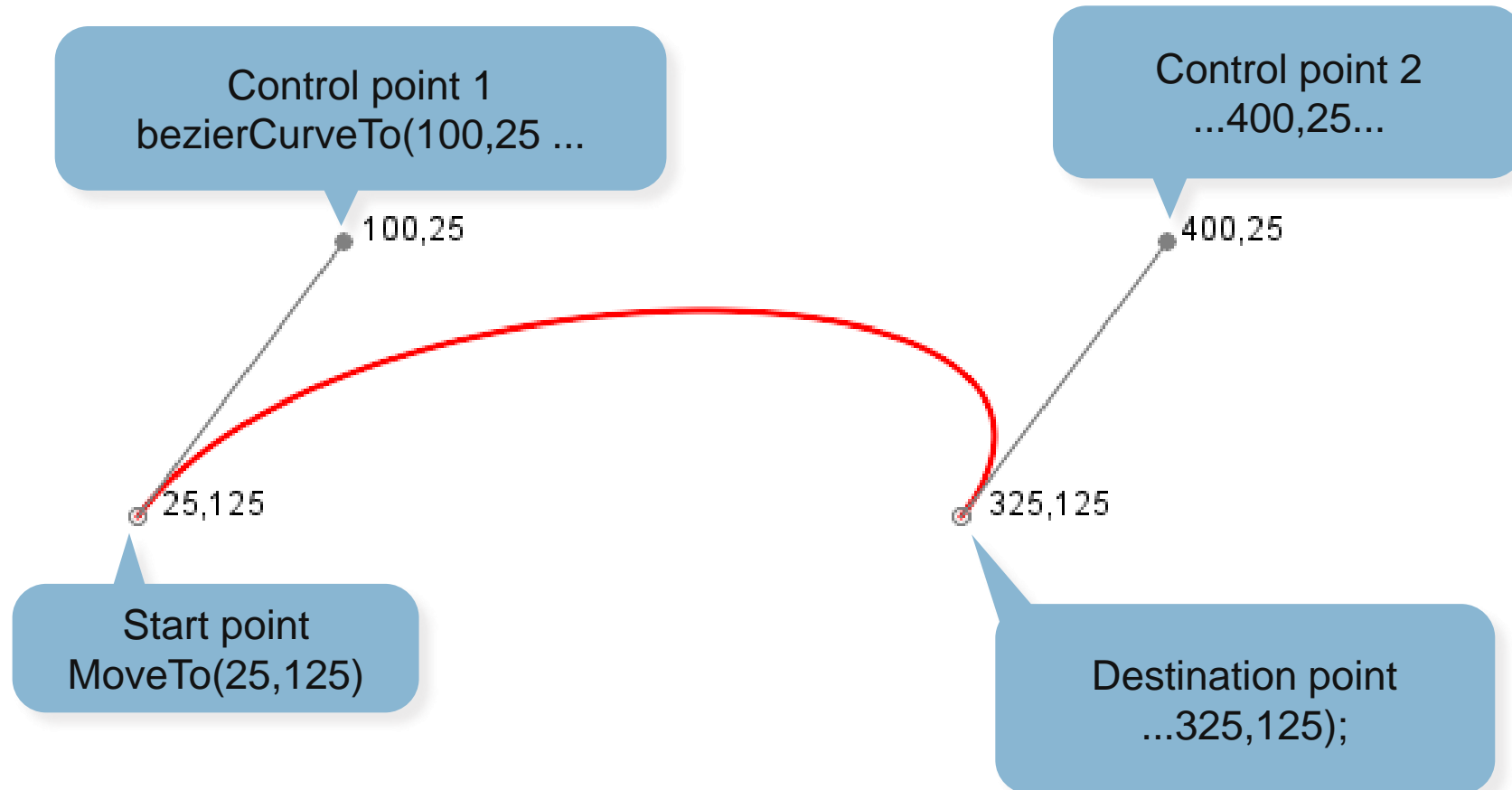


Dessin de lignes

```
context.beginPath(); //set up to draw a path  
context.moveTo(x,y); //move to the start position  
context.lineTo(x,y); //set the end point  
context.stroke(); //draw the line
```



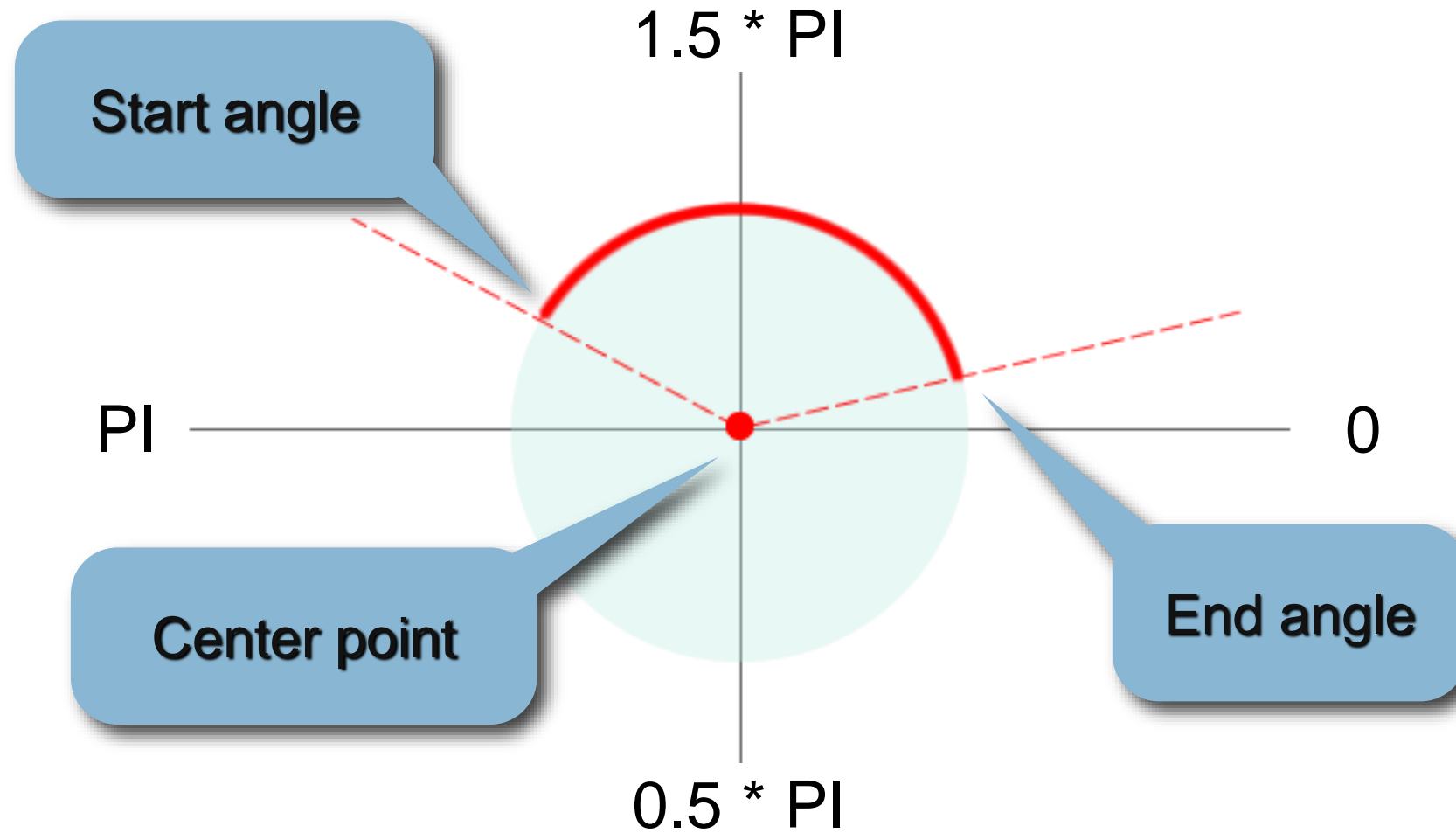
Dessiner une courbe



Codage des courbes

```
var controlPt1 = {x:110,y:30};  
var controlPt2 = {x:130,y:80};  
var startPt = {x:75,y:140};  
ctx.beginPath(); //prepare path  
ctx.moveTo(startPt.x,startPt.y);  
ctx.bezierCurveTo(  
    controlPt1.x,controlPt1.y,  
    controlPt2.x,controlPt2.y,  
    startPt.x,startPt.y  
);  
ctx.stroke();
```

Dessiner Arc



Dessiner ARCS & CIRCLES

- Les cercles sont des types d'arcs
- Angles sont en radians (besoin de calculer entre les degrés et radians)

```
ctx.beginPath();ctx.arc(x, y, radius, 0, Math.PI*2,  
true);ctx.closePath();ctx.fill();
```

Start angle

End angle

TEXT

- Text is "drawn" to the canvas

```
context.fillText("Hello world", 10, 50);
```

- Style text in CSS syntax with **.font** property

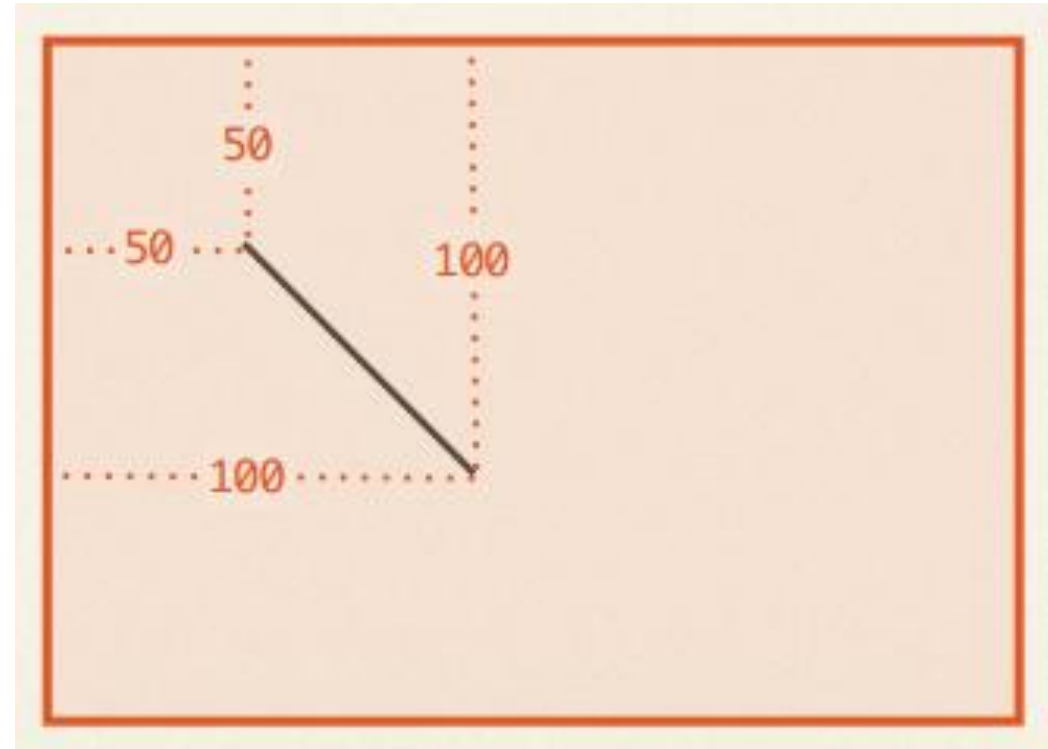
```
context.font = "20pt Arial";
```

- Get the dimensions of a text area

```
textObj = ctx.measureText(d);  
width = textObj.width;
```

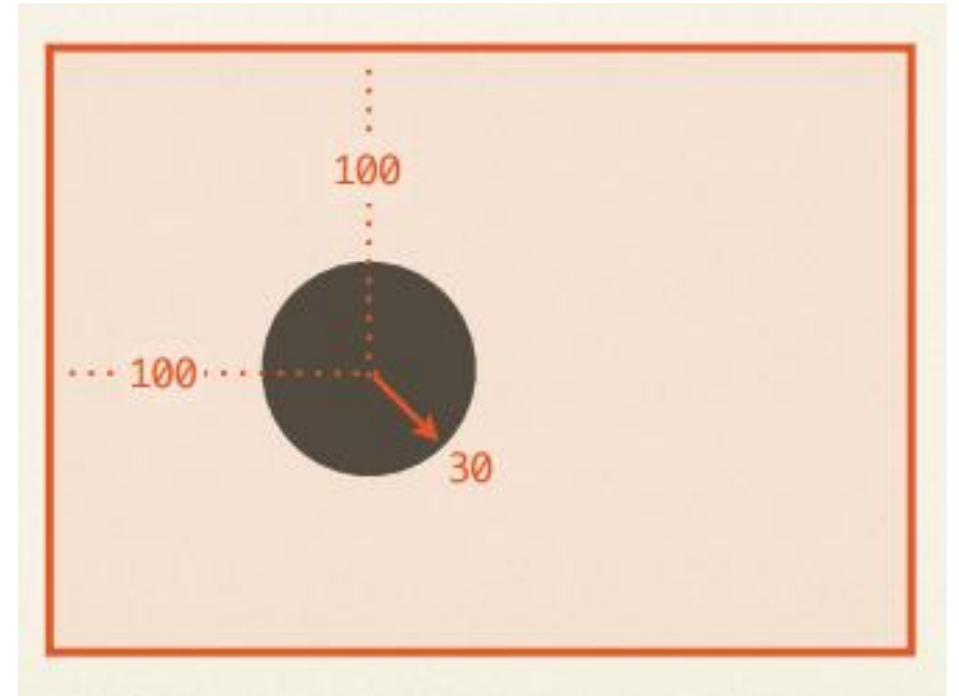
- Exemple

```
ctx.strokeStyle = 'rgb(65, 60, 50)';  
ctx.beginPath();  
ctx.moveTo(50, 50);  
ctx.lineTo(100, 100);  
ctx.stroke();
```



- Exemple

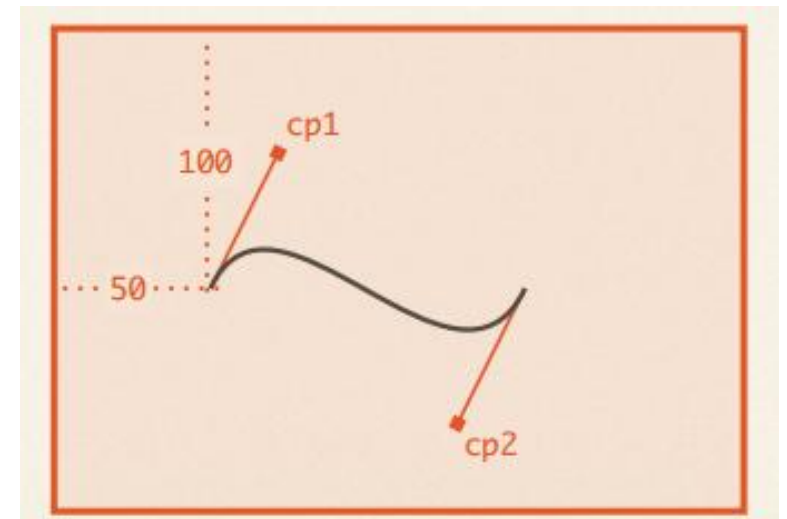
```
ctx.fillStyle = 'rgb(65, 60, 50)';  
ctx.beginPath();  
ctx.arc(100, 100, 30, 0, Math.PI*2, true);  
ctx.fill();
```



HTML 5 – Les Canvas

- Exemple

```
ctx.strokeStyle = 'rgb(65, 60, 50)';  
ctx.beginPath();  
ctx.moveTo(50, 100);  
ctx.bezierCurveTo(70, 50, 130, 150, 150, 100);  
ctx.stroke();
```



- Programmation POO permet beaucoup à faire à travers la canevas
- Il est souple et puissant
 - moteurs d'animation
 - Pseudo 3D graphics
- La lecture des valeurs de pixels ouvre beaucoup de portes
- Intégration avec d'autres éléments HTML5

Canvas	SVG
<ul style="list-style-type: none">• dépendante de résolution• Pas de support pour les gestionnaires d'événements• Pauvres capacités de rendu de texte• Vous pouvez enregistrer l'image résultante comme .png ou .jpg• Bien adapté pour les jeux graphique-intensifs	<ul style="list-style-type: none">• Résolution indépendante• Support aux gestionnaires d'événements• Le mieux adapté pour les applications avec de grandes zones de rendu (Google Maps, SIG) Rendu lent si complexe (tout ce qui utilise le DOM beaucoup sera lente)• inadaptés pour des applications Jeux• Chaque objet manipulé via le DOM• Pourrait avoir des problèmes de performances si vous utilisez de nombreux objets

Question !!

- Canvas vs. Flash

- **CANVAS** : Rectangles

Méthode	Description
rect()	Crée un rectangle (ex. : <code>ctx.rect(40, 30, 200, 125);</code>)
fillRect()	Dessine un rectangle "plein" ou "rempli" (ex. : <code>ctx.fillRect(15,30,150,75);</code>)
strokeRect()	Dessine un rectangle (non rempli) (ex. : <code>ctx.strokeRect(20,20,150,100);</code>)
clearRect()	Efface les pixels spécifiés dans un rectangle donné (ex. : <code>ctx.fillStyle="#00dd00";</code> <code>ctx.fillRect(0,0,350,180);</code> <code>ctx.clearRect(20,20,120,70);</code>)

HTML 5 – Les APIs

- CANVAS :Chemins (Path)

Méthode	Description
fill()	Remplit le dessin courant (chemin, path) (ex. : ctx.rect(20,20,150,100); ctx.fillStyle="red"; ctx.fill();)
stroke()	Dessine le chemin (path) préalablement défini (ex. : ctx.beginPath(); ctx.moveTo(50,30); ctx.lineTo(150,150); ctx.lineTo(150,300); ctx.strokeStyle="#aa9933"; ctx.stroke();)
beginPath()	Commence un chemin, ou réinitialise le chemin courant (ex. : ctx.beginPath(); ctx.moveTo(50,30); ctx.lineTo(150,150); ctx.lineTo(150,300); ctx.strokeStyle="#aa9933"; ctx.stroke();)

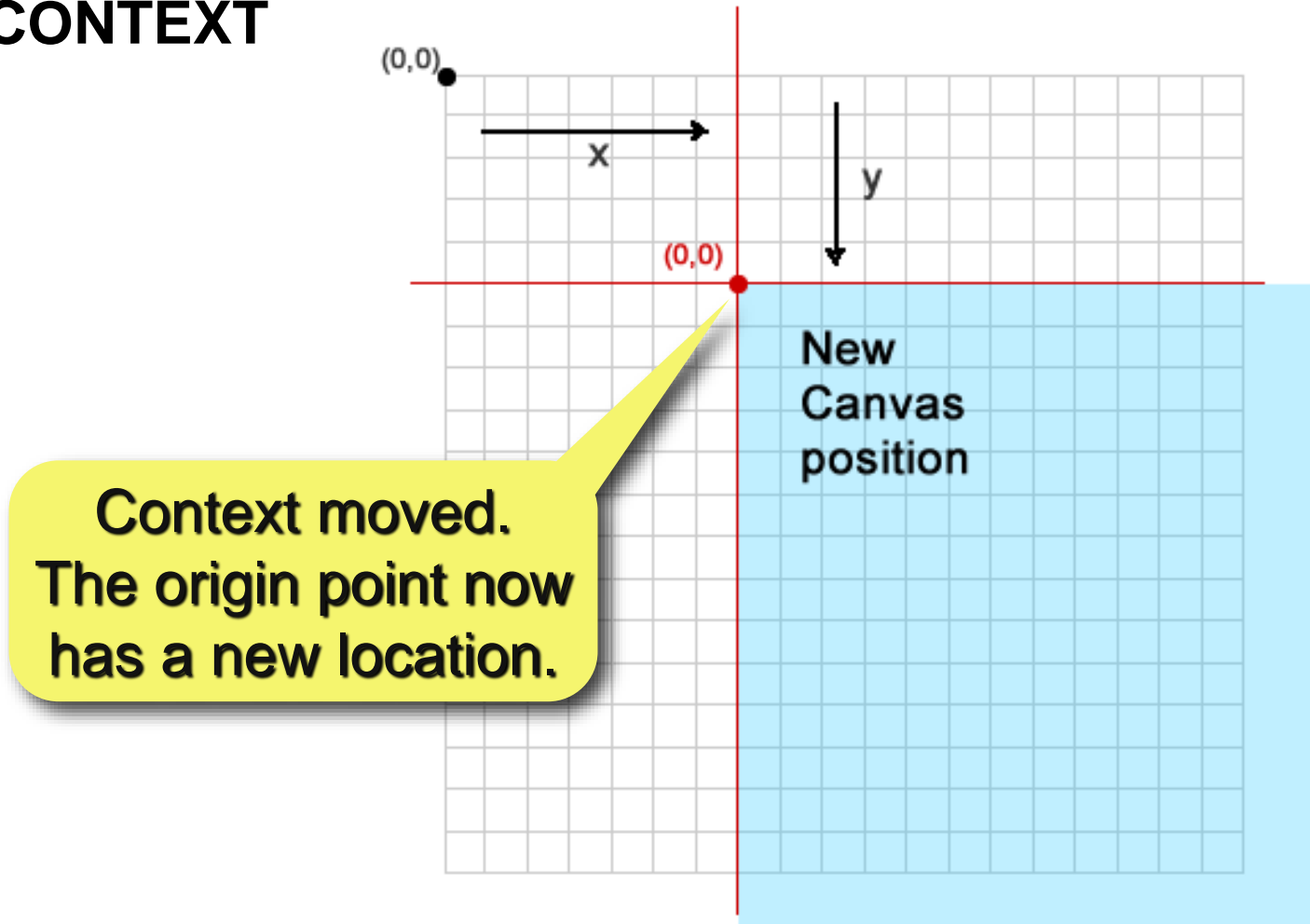
TRANSFORMATIONS

Pour changer l'orientation d'un élément sur le canvas , vous devez passer toute le canvas

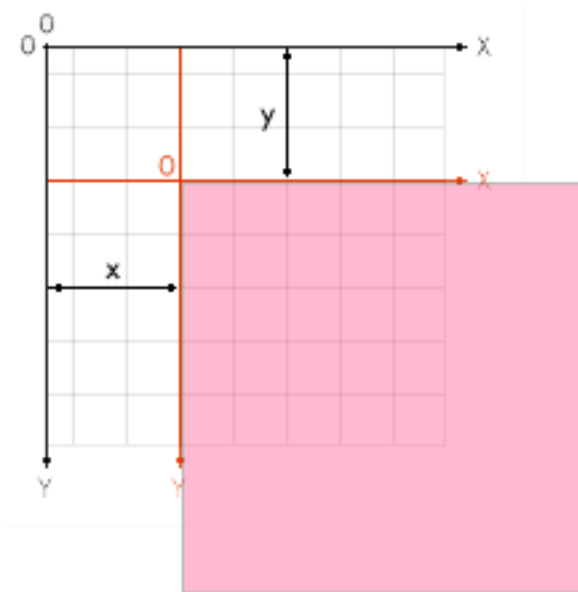
Translation: déplacer la toile et son origine à un point différent dans la grille

Tournez: tourner le canvas autour de l'origine actuelle

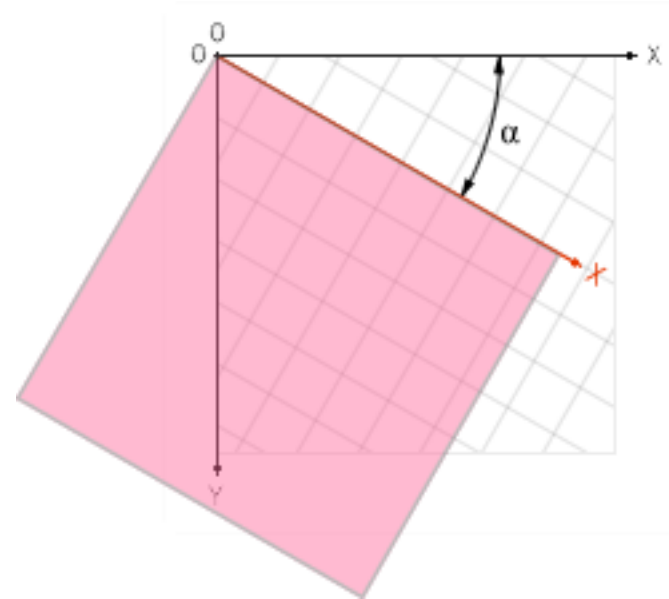
TRANSLATE CONTEXT



TRANSLATE & ROTATE

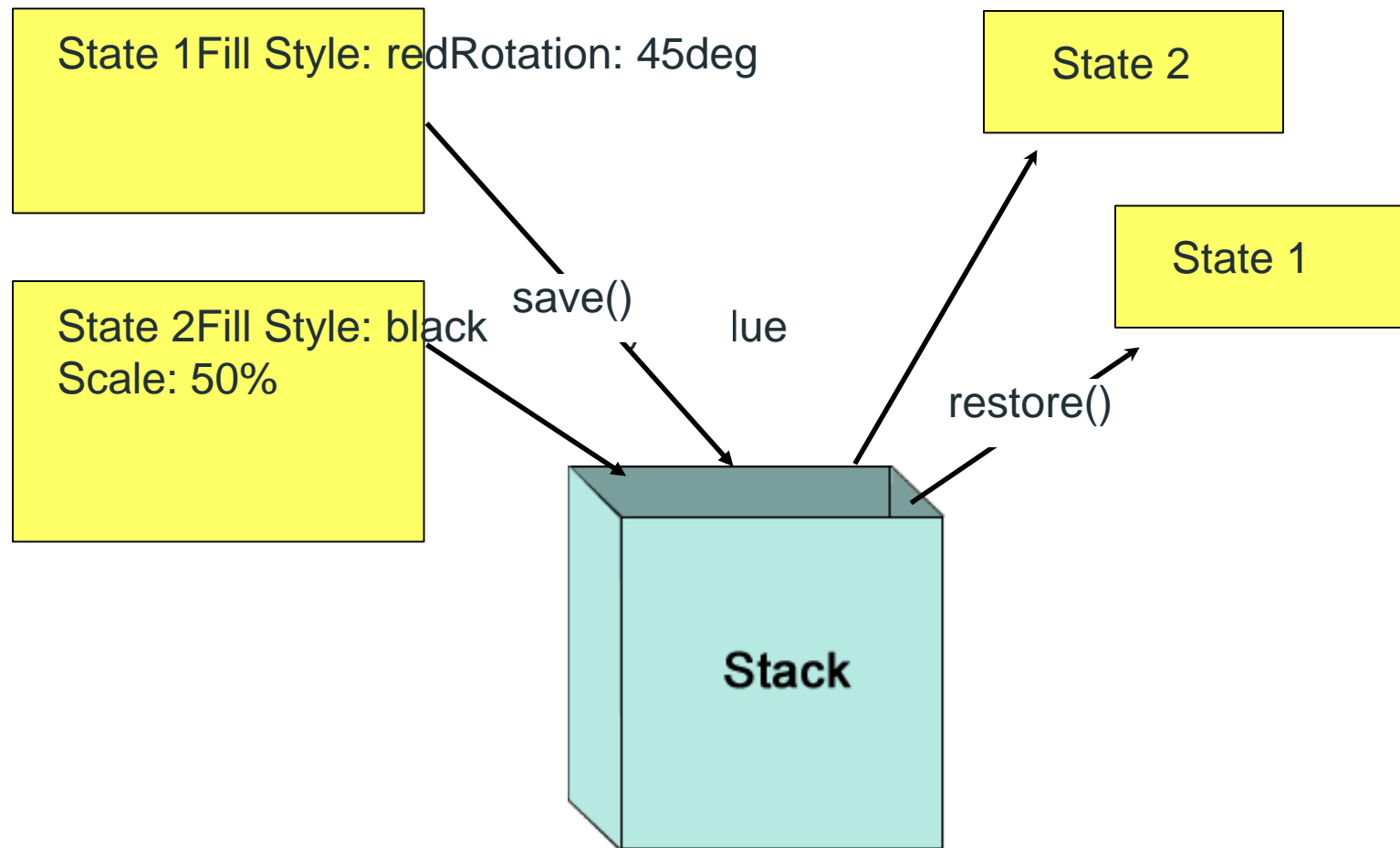


Translate
`translate(x, y)`



Rotate
`rotate(angle)`

THE **STATE** STACK



SAVE & RESTORE

```
ctx.fillRect(0,0,150,150); //Draw with default settings  
ctx.save(); // Save the default state  
ctx.fillStyle = '#09F'; //Change the fill color  
ctx.fillRect(15,15,120,120); // Draw with new settings  
ctx.save(); //Save the current state
```

state 1

```
ctx.fillStyle = '#FFF'; //Change fill again  
ctx.fillRect(15,15,120,120); // Draw with new settings
```

state 2

```
ctx.restore(); //Restore to last saved state  
ctx.fillRect(45,45,60,60); //Draw with restored settings
```

back to state 2

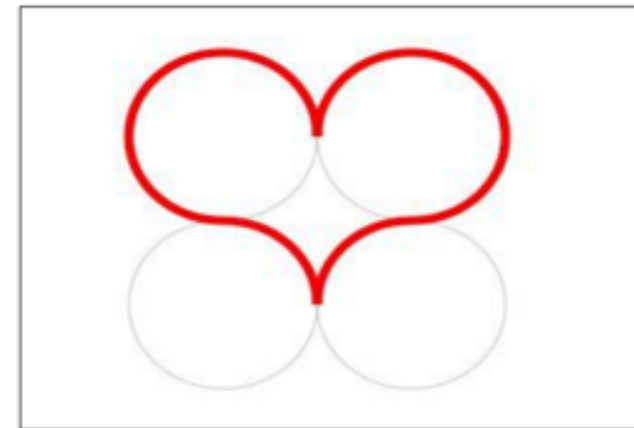
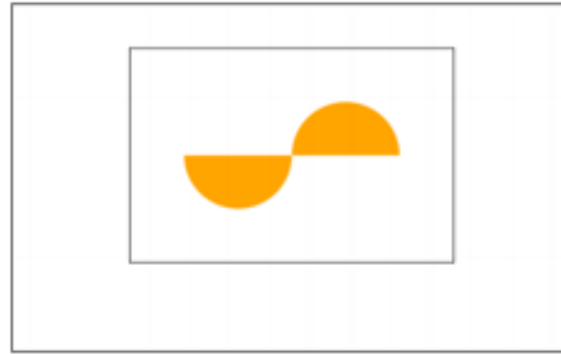
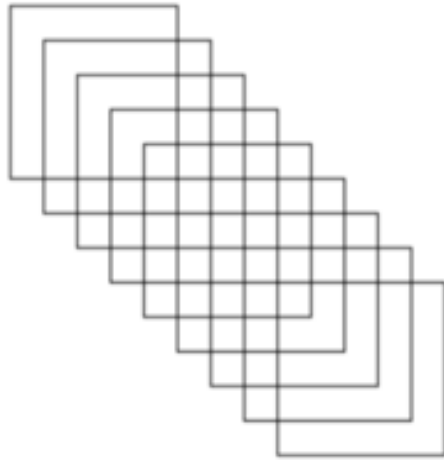
Méthode	Description
scale()	<p>Modifie l'échelle du dessin courant (en plus grand ou plus petit) (ex. : <code>ctx.strokeRect(5,5,25,15);</code> <code>ctx.scale(2,2.5);</code> <code>ctx.strokeRect(5,5,25,15);</code> <code>ctx.scale(2,1.5);</code> <code>ctx.strokeRect(5,5,25,15);</code> <code>ctx.scale(2,3.1);</code> <code>ctx.strokeRect(5,5,25,15);</code>);</p>
rotate()	<p>Fait pivoter le dessin courant dans le sens horloger (négatif trigonométrique, voir remarque séquence précédente) La rotation ne s'applique que sur les dessins demandés après l'instruction <code>rotate()</code> (ex. : <code>ctx.rotate(20*Math.PI/180);</code> <code>ctx.fillRect(50,20,100,50);</code>);</p>

translate()	Remappe ou repositionne la position (0,0) sur la toile Effectue un glissement ou translation (de 80 vers la droite et 100 vers le bas) (ex. : <code>ctx.fillRect(20,10,100,50);</code> <code>ctx.translate(80,100);</code> <code>ctx.fillRect(20,10,100,50);</code>)
transform()	Remplace la matrice de transformation courante pour le dessin (ex. : <code>vv</code>)
setTransform()	Remet à zéro le courant de transformer la matrice d'identité. Exécute ensuite <code>transform ()</code> (ex. : <code>vv</code>)

- ANIMATION
- CANVAS GAMES
- 3D DRAWING

Exercice 01 : Analyse Du Problème

Programmation Web



Exercice 02: Horloge

Programmation Web



Problème :

Comment savoir que le point P n'est pas dans le rectangle ?

