

**Hassen Ben Rebah**

**Benoît Mariat**

Préfaces de François-Guillaume Ribreau et Marc Buils

# API HTML 5

**Maîtrisez le Web moderne !**

canvas, SVG, géolocalisation,  
Web Storage, Web Workers...



**EYROLLES**

## La nouvelle référence sur HTML 5

Malgré l'importance qu'occupent les nouvelles API JavaScript de HTML 5 au sein des applications web actuelles, il n'existe que peu d'ouvrages qui détaillent et présentent ces nouvelles spécifications. L'objectif de ce livre est justement de devenir une référence sur ces fonctionnalités avancées en répondant à ce manque via la présentation de chacune des API les plus utiles de manière claire, explicite et conforme aux normes du W3C.

Cet ouvrage aborde les principaux sujets suivants : l'API canvas permet de dessiner des graphiques 2D, de manipuler des images et de créer des animations, l'API SVG sert à définir des graphismes vectoriels bidimensionnels, l'API Géolocalisation détermine la position d'un objet sur un plan ou une carte à l'aide de ses coordonnées géographiques, l'API Glisser-Déposer permet de déplacer des éléments graphiques d'une zone vers une autre à l'aide de la souris, l'API Web Storage offre aux applications web la possibilité de stocker des données localement dans une base de données côté client (au niveau du navigateur), l'API Application Cache, de moins en moins utilisée, consiste à rendre les applications web disponibles en cas de coupure de la connexion Internet et enfin l'API Web Workers pour exécuter du code JavaScript en parallèle.

## Des TP's pour pratiquer

Des travaux pratiques présentent une synthèse des concepts traités dans l'ouvrage sous forme d'exercices corrigés. L'objectif de ces travaux pratiques est d'illustrer concrètement l'utilisation conjointe des différentes API HTML 5.

## À qui cet ouvrage s'adresse-t-il ?

- Aux développeurs web, intégrateurs qui souhaitent mettre en œuvre les nouvelles API d'HTML 5 et moderniser leurs bonnes pratiques
- Aux designers web qui souhaitent découvrir toutes les possibilités que leur offre HTML 5

### Au sommaire

**canvas.** API de dessin 2D du canvas • Grille de coordonnées • Formes géométriques simples • Tracés et chemins • **Scalable Vector Graphics.** Dessiner avec SVG • Formes simples • Texte • Animation SVG • **Géolocalisation.** Principe et méthodes • Récupérer les coordonnées d'une position • Gérer les erreurs • Géolocalisation sur une carte • **Glisser-Déposer.** Principe et événements • Attribut draggable • Exemple complet • Glisser-Déposer des fichiers • **Stockage local.** Objets et interface Web Storage • Manipuler des données via l'interface Storage • Exemple complet • **Application web hors ligne.** Structure du fichier Manifest • Gérer un cache • API Application Cache • Mettre à jour le cache • Exemple complet • **Web Workers.** Comprendre la programmation parallèle • Types de Web Workers • Manipulation des Web Workers • **Travaux pratiques.** TP 1 : logiciel de dessin • TP 2 : jeu du Gomoku • TP 3 : gestion d'infrastructure

**Hassen Ben Rebah** est enseignant universitaire à l'Institut supérieur des études technologiques de Mahdia. Diplômé de l'Institut supérieur d'informatique et de multimédia de Sfax-Tunisie depuis 2005, il enseigne essentiellement les modules relatifs à la programmation web (HTML 5, CSS 3, JavaScript, PHP, XML, etc) et la sécurité informatique.

Diplômé de l'ENSEA, ingénieur en informatique spécialisé dans le Web depuis 2005 et passionné par le JavaScript, **Benoît Mariat** intervient sur des forums d'entraide et crée des tutoriels afin de partager sa passion de la programmation et du Web. Après 8 années passées chez SAP à améliorer la performance des applications Web, il intègre Intersec en tant qu'expert JavaScript pour améliorer leurs interfaces front-end.

[www.editions-eyrolles.com](http://www.editions-eyrolles.com)

# API HTML 5

Maîtrisez le Web moderne !

#### DANS LA MÊME COLLECTION

B. Barré. – **Concevez des applications mobiles avec React Native.**  
N°67563, 2018, 220 pages.

R. Rimelé. – **HTML 5 – Une référence pour le développeur web.**  
N°67401, 3<sup>e</sup> édition, 2017, 832 pages.

P. Ficheux. – **Linux embarqué – Mise en place et développement.**  
N°67484, 2017, 220 pages.

M. Bidault. – **Programmation Excel avec VBA.**  
N°67401, 2017, 480 pages.

K. Novak. – **Débuter avec LINUX.**  
N°13793, 2017, 522 pages.

P. Martin, J. Pauli, C. Pierre De Geyer. – **PHP 7 avancé.**  
N°14357, 2016, 732 pages.

R. Goetter. – **CSS 3 Flexbox.**  
N°14363, 2016, 152 pages.

#### SUR LE MÊME THÈME

M. Nebra. – **Réalisez votre site web avec HTML 5 et CSS 3.**  
N°67476, 2<sup>e</sup> édition, 2017, 348 pages.

F. Draillard. – **Premiers pas en CSS 3 et HTML 5.**  
N°67430, 7<sup>e</sup> édition, 2017, 514 pages.

H. Giraudel, R. Goetter. – **CSS 3 : Pratique du design web.**  
N°14023, 2015, 372 pages.

R. RIMELE. – **Mémento HTML 5.**  
N°67514, 3<sup>e</sup> édition, 2017, 14 pages.

J. KEITH, R. ANDREW. – **HTML 5 pour les web designers.**  
N°14437, 2<sup>e</sup> édition, 2016, 104 pages.

Retrouvez nos bundles (livres papier + e-book) et livres numériques sur  
<http://izibook.eyrolles.com>



**Hassen Ben Rebah**

**Benoît Mariat**

Préfaces de François-Guillaume Ribreau et Marc Buils

# **API HTML 5**

**Maîtrisez le Web moderne !**

**EYROLLES**



ÉDITIONS EYROLLES  
61, bd Saint-Germain  
75240 Paris Cedex 05  
[www.editions-eyrolles.com](http://www.editions-eyrolles.com)

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans l'autorisation de l'Éditeur ou du Centre Français d'exploitation du droit de copie, 20, rue des Grands Augustins, 75006 Paris.

© Groupe Eyrolles, 2018, ISBN : 978-2-212-67554-2

*Je dédie ce modeste travail à ma mère Aïcha, mon père Belgaçem,  
ma femme Faïza et mon fils Rayen.*

Hassen Ben Rebah

*Je remercie Claire qui m'a supporté toutes ces années et qui m'a soutenu  
pendant le projet de ce livre. Je remercie également toute ma famille  
pour leur affection et leur soutien dans les moments très difficiles.*

Benoît Mariat



# Table des matières

---

<b>Avant-propos .....</b>	<b>1</b>
Objectifs et structure de l'ouvrage .....	1
À qui s'adresse cet ouvrage ? .....	2
 <b>CHAPITRE 1</b>	
<b>canvas.....</b>	<b>3</b>
Déclaration d'un canvas .....	4
API de dessin 2D du canvas .....	5
Grille de coordonnées .....	7
Formes géométriques simples .....	7
Tracés et chemins .....	10
Tracer des lignes droites .....	10
Style de contour et de remplissage des lignes .....	14
Tracer des courbes .....	16
<i>La méthode arc()</i> .....	17
<i>La méthode quadraticCurveTo()</i> .....	19
<i>La méthode bezierCurveTo()</i> .....	21
Tracer des chemins .....	22
<i>La méthode arcTo()</i> .....	24
Tracer des rectangles avec la méthode rect() .....	26
Styles et modes de remplissage .....	27
<i>Remplissage par une couleur unie</i> .....	27
<i>Remplissage par un dégradé de couleur</i> .....	29
<i>Remplissage avec un motif</i> .....	31
Manipuler des images .....	33
<i>Dessiner des images</i> .....	33
<i>Créer une image par pixel</i> .....	37
<i>Lire et copier des pixels d'une image</i> .....	38
<i>Appliquer des filtres et modifier des pixels d'une image</i> .....	40
Manipuler du texte .....	42
Quelques effets .....	45
<i>Gérer la transparence</i> .....	45
Prise en charge de l'API canvas .....	56

## CHAPITRE 2

**Scalable Vector Graphics..... 57**

Dessiner avec SVG .....	58
Structure d'un document SVG .....	60
<i>Élément de groupement &lt;g&gt;</i> .....	60
<i>Élément &lt;symbol&gt;</i> .....	61
<i>Élément &lt;defs&gt;</i> .....	62
<i>Élément &lt;use&gt;</i> .....	62
<i>Élément &lt;desc&gt;</i> .....	64
Système de coordonnées .....	64
<i>Viewport</i> .....	64
<i>Attributs viewBox et preserveAspectRatio</i> .....	65
Formes simples .....	70
Chemins .....	72
Remplissage et mise en forme .....	76
Intégrer les propriétés de style .....	77
<i>Sous forme d'attributs à l'intérieur d'une balise</i> .....	77
<i>Sous forme d'une feuille de styles externe</i> .....	78
<i>Sous forme d'une feuille de styles interne</i> .....	79
<i>Sous forme de style intégré dans la balise de forme</i> .....	80
Motifs et dégradés de couleurs .....	80
<i>Motifs</i> .....	80
<i>Dégradés de couleurs</i> .....	82
Texte .....	85
Mettre en forme du texte .....	86
Manipuler du texte .....	87
<i>Texte multiligne</i> .....	87
<i>Glissement de texte</i> .....	88
<i>Rotation de texte</i> .....	89
<i>Texte le long d'une ligne courbe</i> .....	90
Animation SVG .....	91
Élément <animate> .....	92
Élément <animateColor> .....	95
Élément <set> .....	97
Élément <animateTransform> .....	98
Élément <animateMotion> .....	101
Prise en charge de l'API SVG .....	103

## CHAPITRE 3

**Géolocalisation..... 105**

Principe .....	107
Méthodes .....	107
Récupérer les coordonnées d'une position .....	109
Gérer les erreurs .....	111
Options supplémentaires .....	113
Géolocalisation sur une carte .....	114
Prise en charge de l'API de géolocalisation .....	120

## CHAPITRE 4

**Glisser-Déposer..... 121**

Principe .....	121
Événements .....	122
Attribut draggable .....	122
Objet DataTransfer .....	123
Propriétés .....	123
<i>effectAllowed</i> .....	123
<i>dropEffect</i> .....	124
Méthodes .....	125
<i>setData()</i> .....	125
<i>getData()</i> .....	125
<i>setDragImage()</i> .....	126
Exemple complet .....	126
Glisser-Déposer des fichiers.....	128
Déposer des fichiers du système d'exploitation vers le navigateur .....	128
Déposer des fichiers du navigateur vers le système d'exploitation .....	129
Objet FileReader .....	131
<i>Méthodes</i> .....	131
<i>Événements</i> .....	131
<i>Propriétés</i> .....	132
Prise en charge de l'API Glisser-Déposer et de l'objet FileReader .....	134

## CHAPITRE 5

**Stockage local..... 135**

Objets Web Storage .....	136
Interface Web Storage .....	136
Manipuler des données via l'interface Storage .....	137
Exemple complet .....	140
Prise en charge de l'API Web Storage .....	141

## CHAPITRE 6

**Application web hors ligne ..... 143**

Structure du fichier Manifest .....	144
CACHE .....	145
NETWORK .....	145
FALLBACK .....	146
Assigner le Manifest au site web .....	147
Gérer un cache .....	149
API Application Cache .....	151
Propriétés .....	151
Événements .....	152
Méthodes .....	153
Mettre à jour le cache .....	154
Exemple complet d'utilisation du cache .....	154
Prise en charge de l'API Application Cache .....	158

## CHAPITRE 7

**Web Workers..... 159**

Comprendre la programmation parallèle .....	160
Programmation séquentielle .....	160
<i>Programmation impérative</i> .....	160
<i>Programmation événementielle</i> .....	161
Programmation parallèle .....	162
<i>Communication par mémoire partagée</i> .....	163
<i>Communication par passage de message</i> .....	163
Types de Web Workers .....	163
Workers dédiés .....	163
<i>Créer un Web Worker</i> .....	164
Gérer la communication d'un Worker : envoyer et recevoir des messages .....	166
Gérer un Worker .....	169
<i>Restriction</i> .....	169
<i>Espace global dans le Worker</i> .....	169
<i>Terminaison d'un Worker</i> .....	169
Workers partagés .....	171
<i>Créer un Web Worker partagé</i> .....	171
<i>Gérer la communication des Workers partagés</i> .....	173
Service Workers .....	176
<i>Cycle de vie</i> .....	176
<i>Objet Promise</i> .....	177
<i>Interfaces</i> .....	179
<i>Compatibilité des objets</i> .....	184
<i>Créer un Service Worker</i> .....	185
Manipulation des Web Workers .....	194
Gestion des erreurs .....	194
Contexte d'exécution du Web Worker .....	196
<i>Objet location</i> .....	196
<i>Objet navigator</i> .....	196
Importation des scripts .....	197
Enchaînement des Web Workers .....	198
Prise en charge de l'API Web Worker .....	200

## CHAPITRE 8

**Travaux pratiques..... 201**

TP 1 : logiciel de dessin .....	202
Exercice 1 : Canvas .....	202
<i>Énoncé</i> .....	202
<i>Solution</i> .....	203
Exercice 2 : localStorage .....	209
<i>Énoncé</i> .....	209
<i>Solution</i> .....	209
Exercice 3 : shared Worker .....	212
<i>Énoncé</i> .....	212
<i>Solution</i> .....	213
TP 2 : jeu du Gomoku .....	223
Exercice 1 : SVG .....	224



<i>Énoncé</i> .....	224
<i>Solution</i> .....	224
Exercice 2 : Web Worker .....	231
<i>Énoncé</i> .....	231
<i>Solution</i> .....	231
Exercice 3 : localStorage .....	242
<i>Énoncé</i> .....	242
<i>Solution</i> .....	243
Exercice 4 : Glisser-Déposer d'un fichier .....	255
<i>Énoncé</i> .....	255
<i>Solution</i> .....	255
TP 3 : gestion d'infrastructure .....	259
Exercice 1 : Glisser-Déposer un élément .....	259
<i>Énoncé</i> .....	259
<i>Solution</i> .....	260
Exercice 2 : localStorage .....	269
<i>Énoncé</i> .....	269
<i>Solution</i> .....	269
Exercice 3 : géolocalisation .....	277
<i>Énoncé</i> .....	277
<i>Solution</i> .....	278

<b>Index</b> .....	<b>291</b>
--------------------	------------



## Préface de François-Guillaume Ribreau

JavaScript est sans aucun doute le langage de programmation le plus déployé et le plus populaire dans le monde.

Rien ne le prédestinait pourtant à le devenir. Créé en 1995 par Brendan Eich pour le compte de Netscape, JavaScript a été initialement pensé pour des besoins de scripting léger côté serveur et côté client. Il est *single-thread*, à typage dynamique, faiblement typé et orienté objet par prototype.

Aujourd'hui, JavaScript est disponible sur n'importe quel smartphone du marché, contrôle des robots et des objets connectés, sert à piloter des drones et permet aux géants du Web d'exposer intégralement leurs produits dans la fenêtre de nos navigateurs. Il était donc plus que primordial de proposer de nouvelles API pour faciliter le développement d'applications Internet riches car les API navigateur initiales étaient loin de répondre aux nouveaux besoins des plates-formes web.

Les tentatives de Flash et de Silverlight pour faciliter le développement d'interactions et d'applications clientes riches ont permis d'identifier les carences des API HTML.

Les nouvelles API HTML 5 permettent de contourner la contrainte du single-threading de JavaScript (*Web Workers*), de proposer un mode de fonctionnement hors ligne pour les applications (*Service Workers*, *offline manifest*), de stocker des informations côté client de manière indéterminée ou le temps d'une session (*localStorage*, *sessionStorage*), de géolocaliser les utilisateurs (*Geolocation API*), de bénéficier d'une gestion native du glisser-déposer ou encore de réaliser des dessins vectoriels ou bitmap (SVG, Canvas), prouesse jusqu'alors réservée aux plug-ins tels que Flash ou Silverlight.

Composé d'exemples simples, ce livre sera le compagnon idéal pour tout débutant souhaitant découvrir certaines de ces nouvelles API JavaScript proposées dans la spécification HTML 5.

François-Guillaume Ribreau

CTO @Redsmin @ImageCharts et @KillBugApp

Architecte et responsable développements numériques chez @OuestFrance

## Préface de Marc Buils

En quelques années, JavaScript, langage de programmation d'abord peu apprécié par les développeurs, est devenu le plus populaire d'entre eux. Aujourd'hui, il bénéficie de l'écosystème de développement le plus large. On le retrouve sur la plupart des plates-formes ; pour la création de pages web interactives bien sûr, mais également pour développer un serveur d'API, piloter un robot ou créer des applications mobiles, pour ne citer que ces exemples. JavaScript est devenu l'un des langages de programmation les plus polyvalents du marché.

Que semble loin le temps où JavaScript n'était utilisé que pour ajouter une traînée d'étoiles au curseur ou faire défiler des flocons sur une page statique... On le voit maintenant de plus en plus souvent utilisé pour d'importants projets dont certains peuvent s'étendre sur plusieurs années. Les exigences qui en découlent en termes de maintenance sur le long terme requièrent dès lors de bien en comprendre le fonctionnement.

L'engouement autour de JavaScript et de HTML 5 peut aisément conduire à sauter quelques étapes en apprenant à utiliser un framework complet avant même de connaître JavaScript lui-même. Les ouvrages dédiés à JavaScript et au HTML 5 se font d'ailleurs bien trop rares quand on voit combien de livres sont consacrés à Angular, jQuery, React ou Node.js.

Tous ces outils ont pourtant un point commun : ils requièrent de connaître JavaScript avant de pouvoir être utilisés.

Lorsque Hassen et Benoît m'ont annoncé qu'ils écrivaient un livre sur le sujet, je les ai immédiatement félicités pour cet excellent projet et j'ai accepté avec joie d'en écrire la préface. Cela fait longtemps que j'attendais qu'un livre, contrairement à ceux qui cèdent à la facilité d'impressionner le lecteur en leur promettant de réaliser mille effets, explore réellement les dessous des API JavaScript pour mieux comprendre ce qu'on développe. Voilà précisément ce que ce livre vous propose.

Marc Buils

CEO chez CestDansLeCoin-France

Consultant JavaScript depuis 2006

# Avant-propos

---

## Objectifs et structure de l'ouvrage

Malgré l'importance qu'occupent les nouvelles API JavaScript de HTML 5 au sein des applications web actuelles, il n'existe que peu d'ouvrages qui détaillent et présentent ces nouvelles spécifications. *API HTML 5 : Maîtrisez le Web moderne* constitue une référence sur ces fonctionnalités avancées et répond à ce manque en présentant ces API de manière claire, explicite et conforme aux normes du W3C.

La maîtrise du contenu de cet ouvrage nécessite des connaissances de base en matière de programmation (déclaration de variables, utilisation des boucles, etc), de langage HTML (structure du document, balises principales, etc) et de JavaScript (notion d'objet, méthode, propriété, DOM, etc..)

Cet ouvrage s'articule en deux volets : dans le premier, les auteurs présentent les concepts de base relatifs aux API JavaScript de HTML 5 et les illustrent par des exemples commentés. Dans le second, ils proposent des travaux pratiques corrigés qui permettent aux lecteurs d'assimiler les techniques d'intégration et de manipulation de ces API au sein des pages ou des applications web.

Cet ouvrage aborde les principaux sujets suivants : l'API **Canvas** qui permet de dessiner des graphiques 2D, de manipuler des images et de créer des animations, l'API **SVG** qui permet de définir des graphismes vectoriels bidimensionnels, l'API **Géolocalisation** qui détermine la position d'un objet sur un plan ou une carte à l'aide de ses coordonnées géographiques, l'API **Glisser-Déposer** qui permet de déplacer des éléments graphiques d'une zone vers une autre à l'aide de la souris, l'API **Web Storage** qui offre aux applications web la possibilité de stocker des données localement dans une base de données côté client (au niveau du navigateur), l'API **Application Cache** qui tend à disparaître et qui consiste à rendre les applications web disponibles en cas de coupure de la connexion Internet, l'API **Web Workers** qui permet d'exécuter du code JavaScript en parallèle. Pour finir, un chapitre de **travaux pratiques** présente une synthèse des concepts traités dans les chapitres précédents sous forme d'exercices corrigés. L'objectif de ces travaux pratiques est d'illustrer concrètement l'utilisation conjointe des différentes API HTML 5.

Tous les codes sources des exemples et des travaux pratiques présentés dans ce livre sont testés et accessibles en téléchargement libre sur le site des éditions Eyrolles (<http://www.editions-eyrolles.com/dl/0067554>).

## À qui s'adresse cet ouvrage ?

Cet ouvrage est destiné à tous ceux qui désirent tirer parti et comprendre les bases des API JavaScript de HTML 5. Il s'adresse à la fois :

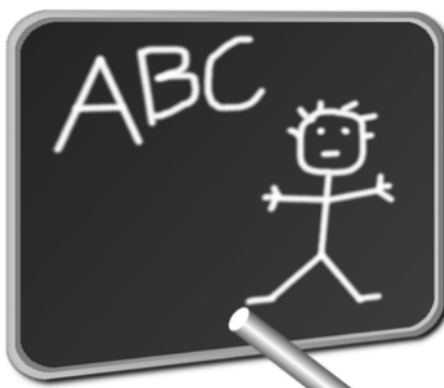
- aux étudiants et universitaires qui recherchent une référence sur ce type d'API ;
- aux développeurs, qui désirent améliorer le développement de leurs applications web.

Hassen Ben Rebah et Benoît Mariat

# 1

## canvas

---



`<canvas>` est un nouvel élément HTML 5 qui peut être utilisé pour créer des graphismes 2D, manipuler des images ou réaliser des animations. Il peut être exploité grâce à une API riche et puissante pour créer des jeux en ligne ou des dispositifs interactifs à base de script d'actions (notamment en JavaScript).

`<canvas>` a initialement été introduit par Apple en 2004 pour être utilisé à l'intérieur de son propre composant macOS WebKit alimentant des applications comme le Widgets Dashboard et le navigateur Safari. Plus tard, en 2005, il a été adopté dans la version 1.8 des navigateurs Gecko (notamment Mozilla Firefox) puis Opera en 2006. Il a par la suite été norma-

lisé par le groupe de travail sur les nouvelles spécifications proposées pour les technologies web de nouvelle génération (WHATWG).

La balise `<canvas>` est un conteneur qui définit une zone dans laquelle peuvent être placés des dessins et des graphismes. Cette zone possède la liste des attributs présentés dans le tableau 1-1.

**Tableau 1-1** Attributs de l'élément `<canvas>`

Attribut	Valeur	Description
id	id	Identifiant de la zone
width	pixel	Largeur de la zone (par défaut 300 pixels)
height	pixel	Hauteur de la zone (par défaut 200 pixels)

## Déclaration d'un canvas

L'élément `<canvas>` se définit avec la syntaxe suivante :

```
<canvas id="identifiant" width="largeur" height="hauteur">  
  Ce texte sera affiché lorsque le navigateur ne prend pas  
  en charge l'élément canvas  
</canvas>
```

Une fois défini, l'élément `<canvas>` peut être intégré au sein d'une page web de la manière suivante.

### Exemple : création d'un canvas

```
<!DOCTYPE html>  
<html>  
  <head>  
    <meta charset="utf-8">  
    <title>Création d'un canvas</title>  
    <!-- Style appliqué à l'élément canvas identifié par l'attribut id=can -->  
    <style type="text/css">  
      #can{  
        border-style:solid;  
        border-width:1px;  
        border-color:red;  
      }  
    </style>  
  </head>  
  <body>  
    <!-- Message destiné aux anciens navigateurs qui ne prennent pas en charge  
    l'élément canvas -->
```



```
<canvas id="can" width="300" height="300">
Désolé, votre navigateur ne prend pas en charge l'élément canvas.
</canvas>
</body>
</html>
```

La figure 1-1 présente le résultat obtenu.

**Figure 1-1**  
Contour de la zone de dessin  
<canvas>



#### Note

Par défaut, un canvas est affiché sans bordure ni contenu initial.

L'élément `<canvas>` peut être stylisé comme n'importe quelle image (marge, contours, arrière-plan, etc.). Si aucun style n'est défini, le canvas est transparent par défaut. Ces règles de style n'affectent pas l'acte de dessiner sur le canvas.

Une même page peut inclure plusieurs `<canvas>`.

L'identifiant **id** est un moyen utile pour accéder aux différents éléments `<canvas>` créés au sein d'une page web et les manipuler.

## API de dessin 2D du canvas

L'élément `<canvas>` de HTML 5 est une API de dessin 2D pour le Web à la fois puissante et simple à utiliser. Elle offre la possibilité de :

- dessiner des formes ;
- transformer des objets ;
- composer plusieurs images ;
- créer des lignes ;
- remplir des dessins avec des couleurs, des styles ou des ombres ;
- créer du texte ;
- manipuler des pixels.

Pour dessiner avec **canvas**, quatre étapes sont requises.

- 1 Créer le conteneur qui englobera les objets à manipuler ou à dessiner grâce à l'élément `<canvas></canvas>`.

```
<canvas id="dessin" width="320" height="250">
Désolé, votre navigateur ne prend pas en charge l'élément canvas.
</canvas>
```

- 2 Établir une référence au **canvas** précédemment créé en 1) grâce à la méthode `getElementById()`. Cette méthode permet d'accéder à l'espace de dessin.

```
var canvas = document.getElementById("dessin");
```

- 3 Définir l'objet contexte grâce à la méthode `getContext("2d")` qui sélectionne l'API de dessin 2D pour l'appliquer à la variable **canvas**.

```
var contexte = canvas.getContext("2d");
```

- 4 Appliquer les instructions de dessin en fonction des besoins.

L'exemple suivant montre comment combiner ces quatre étapes.

#### Exemple : dessiner avec canvas

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Dessiner avec canvas</title>
<style type="text/css">
#dessin {
  border-style:solid;
  border-width: 1px;
  border-color: red;
}
</style>
</head>
<body>
<canvas id="dessin" width="300" height="300">
  Désolé, votre navigateur ne prend pas en charge l'élément canvas.
</canvas>
<script>
var canvas=document.getElementById("dessin");
var contexte=canvas.getContext("2d");
// Suite du script de dessin
</script>
</body>
</html>
```

### Note

Les actions de script JavaScript doivent être placées entre les balises `<script></script>`.  
Les scripts JavaScript peuvent être placés dans la partie `<head></head>` ou la partie `<body></body>`.  
HTML 5 propose aussi une API de dessin en 3D. L'appel de cette API peut être effectué grâce à l'instruction suivante :

```
canvas.getContext("webgl");
```

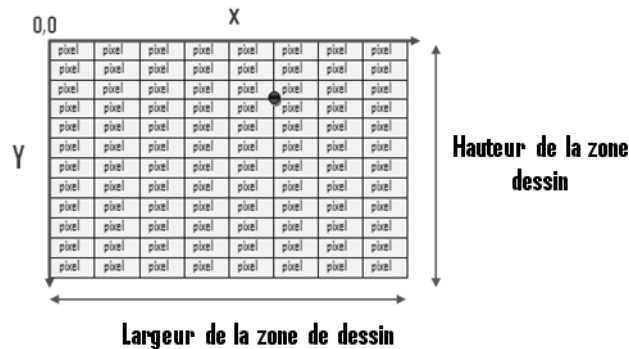
## Grille de coordonnées

Le dessin avec **canvas** s'opère à l'aide de coordonnées. Ce système de coordonnées est structuré de la façon suivante :

- l'origine (0,0) est située au coin supérieur gauche de la fenêtre du navigateur ;
- l'axe des abscisses (x) représente la première coordonnée ;
- l'axe des ordonnées (y) représente la deuxième coordonnée ;
- ces valeurs représentent la grille qui entoure les pixels, et non les pixels eux-mêmes.

Par exemple, un point de coordonnées (5,3) sera identifié comme étant l'intersection d'une projection verticale du pixel numéro 5 situé sur l'axe des (x) et d'une projection horizontale du pixel numéro 3 situé sur l'axe des (y).

**Figure 1-2**  
Grille de coordonnées  
de l'élément `<canvas>`



## Formes géométriques simples

`<canvas>` n'a qu'une seule forme géométrique simple : le rectangle. Toute autre forme (triangle, cercle, etc.) doit être composée en combinant un ou plusieurs trajets ou chemins.

Les méthodes élémentaires de `<canvas>` sont présentées dans le tableau 1-2.

Tableau 1-2 Méthodes élémentaires de canvas

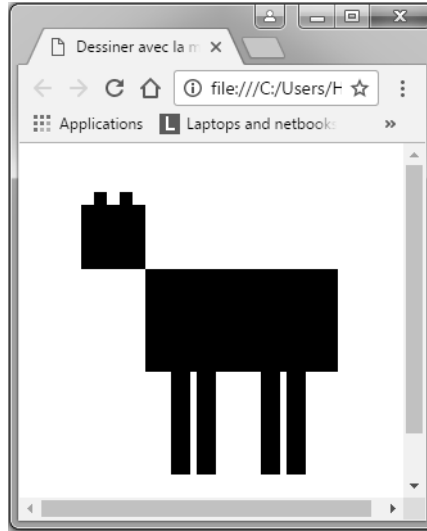
Méthode	Rôle
<code>fillRect(x,y,largeur,hauteur)</code>	Dessine un rectangle plein.
<code>strokeRect(x,y,largeur,hauteur)</code>	Dessine le contour d'un rectangle.
<code>clearRect(x,y,largeur,hauteur)</code>	Dessine un rectangle transparent (efface l'espace interne).
Avec : <b>x</b> et <b>y</b> : la position du coin supérieur gauche du rectangle sur le canvas. <b>largeur</b> et <b>hauteur</b> : la taille du rectangle.	

L'exemple suivant montre comment créer un motif d'animal grâce à la méthode `fillRect()` (figure 1-3).

#### Exemple : dessiner avec la méthode `fillRect()`

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Dessiner avec la méthode fillRect()</title>
</head>
<body>
<canvas id="dessin" width="300" height="300">
  Désolé,votre navigateur ne prend pas en charge l'élément canvas.
</canvas>
<script>
var canvas = document.getElementById("dessin");
var contexte = canvas.getContext("2d");
// Création de la première oreille à gauche
contexte.fillRect(50,30,10,10);
// Création de la deuxième oreille à droite
contexte.fillRect(70,30,10,10);
// Création de la tête
contexte.fillRect(40,40,50,50);
// Création du corps
contexte.fillRect(90,90,150,80);
// Création du premier pied avant
contexte.fillRect(110,170,15,80);
// Création du deuxième pied avant
contexte.fillRect(130,170,15,80);
// Création du premier pied arrière
contexte.fillRect(180,170,15,80);
// Création du deuxième pied arrière
contexte.fillRect(200,170,15,80);
</script>
</body>
</html>
```

**Figure 1-3**  
Dessiner avec fillRect()



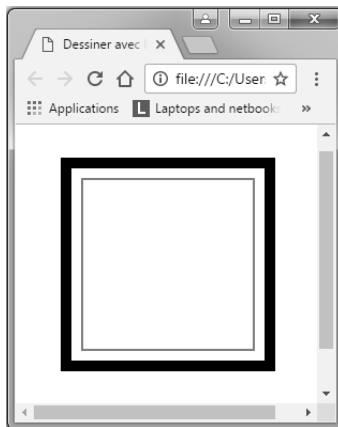
Le deuxième exemple ci-après montre comment créer un rectangle transparent en appelant la méthode `clearRect()`, puis comment placer à l'intérieur le contour d'un second rectangle avec la méthode `strokeRect()`.

**Exemple : dessiner avec les méthodes `clearRect()` et `strokeRect()`**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Dessiner avec clearRect() et strokeRect()</title>
</head>
<body>
<canvas id="dessin" width="300" height="300">
Désolé, votre navigateur ne prend pas en charge l'élément canvas.
</canvas>
<script>
var canvas=document.getElementById("dessin");
var contexte=canvas.getContext("2d");
contexte.fillRect(50,50,200,200);
// La méthode clearRect() efface, à partir du pixel de coordonnées
// (60,60), 180 x 180 pixels du carré plein
contexte.clearRect(60,60,180,180);
// La méthode strokeRect() ajoute un contour à l'intérieur du
// carré transparent.
contexte.strokeRect(70,70,160,160);
</script>
</body>
</html>
```

La figure 1-4 présente le résultat obtenu.

**Figure 1-4**  
Dessiner avec `clearRect()`  
et `strokeRect()`



## Tracés et chemins

Pour créer des formes géométriques avec canvas en utilisant des trajets ou des chemins, il faut généralement suivre plusieurs étapes :

- initialiser le trajet ;
- appeler des fonctions de dessin prédéfinies pour spécifier la forme souhaitée du dessin ;
- fermer le trajet (étape optionnelle).

Les méthodes élémentaires de dessin des trajets définies par `<canvas>` sont résumées dans le tableau 1-3.

**Tableau 1-3** Méthodes élémentaires de dessin des trajets

Méthode	Rôle
<code>beginPath()</code>	Déclare le début d'un nouveau chemin.
<code>closePath()</code>	Ferme un chemin.
<code>stroke()</code>	Dessine le chemin en marquant son contour.
<code>fill()</code>	Dessine une forme (tracé) pleine.
<code>moveTo(x,y)</code>	Début un chemin à partir d'un point de coordonnée x et y.
<code>lineTo(x,y)</code>	Dessine une ligne à partir de la position actuelle du dessin vers la position indiquée par le nouveau point de coordonnées x et y.

## Tracer des lignes droites

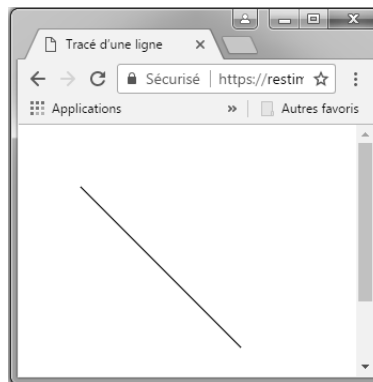
canvas offre la possibilité de dessiner des lignes droites selon le principe illustré par l'exemple suivant.

### Exemple : tracé d'une ligne droite

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Tracé d'une ligne</title>
</head>
<body>
<canvas id="dessin" width="300" height="300">
Désolé, votre navigateur ne prend pas en charge l'élément canvas.
</canvas>
<script>
var canvas=document.getElementById("dessin");
var contexte=canvas.getContext("2d");
contexte.beginPath();
contexte.moveTo(50,50);
contexte.lineTo(200,200);
contexte.stroke();
</script>
</body>
</html>
```

La figure 1-5 montre le résultat obtenu.

**Figure 1-5**  
Tracé d'une ligne



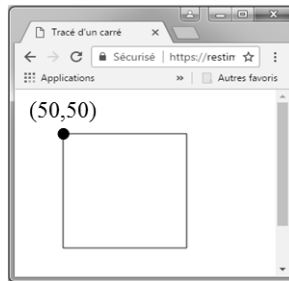
**canvas** permet aussi de combiner plusieurs lignes droites pour dessiner diverses formes géométriques comme des triangles, des droites parallèles, des droites perpendiculaires ou croisées, etc.

L'exemple ci-après montre comment tracer un carré en combinant quatre lignes droites (figure 1-6).

**Exemple : tracé d'un carré (combinaison de plusieurs lignes droites)**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Tracé d'un carré</title>
</head>
<body>
<canvas id="dessin" width="300" height="300">
Désolé, votre navigateur ne prend pas en charge l'élément canvas.
</canvas>
<script>
var canvas = document.getElementById("dessin");
var contexte = canvas.getContext("2d");
contexte.beginPath();
contexte.moveTo(50,50);
contexte.lineTo(50,200);
contexte.lineTo(200,200);
contexte.lineTo(200,50);
contexte.lineTo(50,50);
contexte.stroke();
</script>
</body>
</html>
```

**Figure 1-6**  
Tracé d'un carré sans closePath()

**Note**

La première méthode à appeler pour dessiner une forme géométrique en trajets est `beginPath()`. L'appel de cette méthode entraîne la remise à zéro de la liste des trajets, ce qui permet de commencer une nouvelle forme.

La méthode `closePath()` entraîne la fermeture de la forme géométrique en dessinant une ligne droite à partir de la position courante jusqu'au début du trajet. Si la forme a déjà été fermée (cas de **l'exemple de tracé du carré précédent**), cette méthode n'a aucun effet.

La méthode `fill()` permet de fermer et remplir automatiquement les formes géométriques ouvertes (les méthodes `closePath()` et `stroke()` deviennent alors inutiles pour le remplissage).



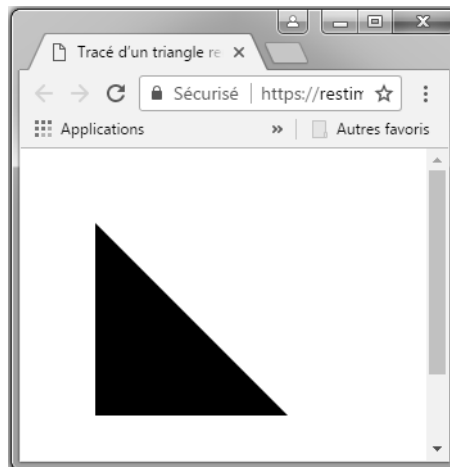
**canvas** offre également la possibilité de dessiner des formes géométriques pleines avec la méthode `fill()`. L'exemple suivant montre comment tracer un triangle droit avec la méthode `fill()`.

**Exemple : tracé d'un triangle rectangle avec la méthode `fill()`**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Tracé d'un triangle rectangle avec la méthode fill()</title>
</head>
<body>
<canvas id="dessin" width="300" height="300">
Désolé, votre navigateur ne prend pas en charge l'élément canvas.
</canvas>
<script>
var canvas=document.getElementById("dessin");
var contexte=canvas.getContext("2d");
contexte.beginPath();
contexte.moveTo(50,50);
contexte.lineTo(50,200);
contexte.lineTo(200,200);
contexte.fill();
</script>
</body>
</html>
```

La figure 1-7 présente le résultat obtenu.

**Figure 1-7**  
Tracé d'un triangle  
avec la méthode `fill()`



## Style de contour et de remplissage des lignes

En plus des fonctions prédéfinies de dessin, canvas offre d'autres options permettant de modifier les couleurs et les styles des lignes. Ces propriétés de style sont résumées dans le tableau 1-4.

Tableau 1-4 Propriétés de style

Propriété	Rôle
lineWidth	Détermine la largeur d'une ligne. Elle reçoit une valeur positive. <b>Exemple :</b> contexte.lineWidth=10;
strokeStyle	Détermine la couleur d'une ligne. Elle reçoit une valeur qui correspond à un code de couleur CSS, un dégradé ou un motif. <b>Exemple :</b> contexte.strokeStyle="red"; contexte.strokeStyle="#800080";
lineCap	Détermine la forme de fin d'une ligne. Elle reçoit l'une des valeurs suivantes : <b>butt</b> : valeur par défaut. <b>round</b> : ajoute un demi-cercle à l'extrémité de la ligne. <b>square</b> : ajoute une boîte dont la largeur est égale à la largeur de la ligne et dont la hauteur est égale à la moitié de son épaisseur. <b>Exemple :</b> contexte.lineCap="round";

L'exemple suivant montre comment utiliser les propriétés de style `lineWidth` et `strokeStyle` pour modifier la largeur et la couleur d'une ligne droite (figure 1-8).

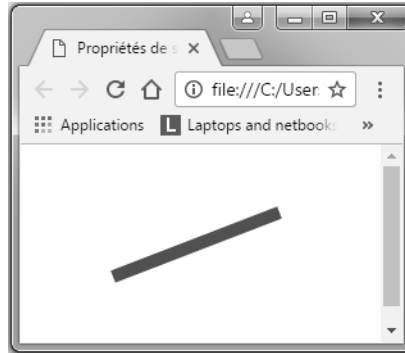
**Exemple : manipulation des propriétés de style (lineWidth et strokeStyle)**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Propriétés de style</title>
</head>
<body>
<canvas id="dessin" width="200" height="150">
Désolé, votre navigateur ne prend pas en charge l'élément canvas.
</canvas>
<script>
var canvas=document.getElementById("dessin");
var contexte=canvas.getContext("2d");
contexte.beginPath();
contexte.moveTo(20,100);
contexte.lineTo(150,50);
contexte.lineWidth=10;
contexte.strokeStyle="rgb(0,100,200)";
```

```
contexte.stroke();
</script>
</body>
</html>
```

**Figure 1-8**

Ligne droite avec une largeur de 10 pixels et de couleur bleue



Il est également possible de modifier la forme de fin d'une ligne en utilisant la propriété de style `lineCap` comme le montre l'exemple suivant.

#### Exemple : utilisation de la propriété de style `lineCap`

```
<!DOCTYPE html>
<html>
<meta charset="utf-8">
<head>
<title>Propriété de style lineCap</title>
</head>
<body>
<canvas id="dessin" width="400" height="400">
Désolé, votre navigateur ne prend pas en charge l'élément canvas.
</canvas>
<script>
var canvas=document.getElementById("dessin");
var contexte=canvas.getContext("2d");
// Propriété linecap=butt
contexte.beginPath();
contexte.moveTo(30,50);
contexte.lineTo(140,250);
contexte.lineWidth=20;
contexte.strokeStyle="rgb(255,0,255)";
contexte.lineCap="butt";
contexte.stroke();
// Propriété linecap=round
contexte.beginPath();
contexte.moveTo(90,50);
contexte.lineTo(200,250);
```