[MS-CANVAS2D]:

Microsoft Edge / Internet Explorer HTML Canvas 2D Context Standards Support Document

---

Revision Summary

Date	Revision History	Revision Class	Comments
3/22/2016	1.0	New	Released new document.
7/19/2016	1.1	Minor	Clarified the meaning of the technical content.
11/2/2016	1.1	None	No changes to the meaning, language, or formatting of the technical content.
3/14/2017	1.1	None	No changes to the meaning, language, or formatting of the technical content.
10/3/2017	1.1	None	No changes to the meaning, language, or formatting of the technical content.
2/22/2018	1.1	None	No changes to the meaning, language, or formatting of the technical content.
3/23/2018	1.1	None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This documented describes the level of supported provided by Microsoft web browsers for the W3C *HTML Canvas 2D Context* specification [W3C-CANVAS2D], published 19 November 2015. The [W3C-CANVAS2D] specification defines the 2D Context for the HTML canvas element.

1.1 Glossary

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[W3C-CANVAS2D] World Wide Web Consortium, "HTML Canvas 2D Context", W3C Recommendation 19 November 2015, https://www.w3.org/TR/2015/REC-2dcontext-20151119/

1.2.2 Informative References

None.

1.3 Microsoft Implementations

The following Microsoft web browser versions implement some portion of the [W3C-CANVAS2D] specification:

- Windows Internet Explorer 9
- Windows Internet Explorer 10
- Internet Explorer 11
- Microsoft Edge

Each browser version may implement multiple document rendering modes. The modes vary from one to another in support of the standard. The following table lists the document modes supported by each browser version.

Browser Version	Document Modes Supported
Internet Explorer 9	Quirks Mode IE7 Mode

Browser Version	Document Modes Supported
	IE8 Mode IE9 Mode
Internet Explorer 10	Quirks Mode IE7 Mode IE8 Mode IE9 Mode IE10 Mode
Internet Explorer 11	Quirks Mode IE7 Mode IE8 Mode IE9 Mode IE10 Mode IE11 Mode
Microsoft Edge	EdgeHTML Mode

For each variation presented in this document there is a list of the document modes and browser versions that exhibit the behavior described by the variation. All combinations of modes and versions that are not listed conform to the specification. For example, the following list for a variation indicates that the variation exists in three document modes in all browser versions that support these modes:

Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)

1.4 Standards Support Requirements

To conform to [W3C-CANVAS2D], a user agent must implement all required portions of the specification. Any optional portions that have been implemented must also be implemented as described by the specification. Normative language is usually used to define both required and optional portions. (For more information, see [RFC2119].)

The following table lists the sections of [W3C-CANVAS2D], and whether they are considered normative or informative.

Sections	Normative/Informative
1	Normative
2	Informative
3-4	Normative
5-7	Informative
8	Normative
9-10	Informative
11	Normative
12-19	Informative

1.5 Notation

The following notations are used in this document to differentiate between notes of clarification, variation from the specification, and points of extensibility.

Notation	Explanation
C####	This identifies a clarification of ambiguity in the target specification. This includes imprecise statements, omitted information, discrepancies, and errata. This does not include data formatting clarifications.
V####	This identifies an intended point of variability in the target specification such as the use of MAY, SHOULD, or RECOMMENDED. (See [RFC2119].) This does not include extensibility points.
E####	Because the use of extensibility points (such as optional implementation-specific data) can impair interoperability, this profile identifies such points in the target specification.

For document mode and browser version notation, see also section 1.3.

2 Standards Support Statements

This section contains all variations and clarifications for the Microsoft implementation of [W3C-CANVAS2D].

- Section 2.1 describes normative variations from the MUST requirements of the specification.
- Section 2.2 describes clarifications of the MAY and SHOULD requirements.
- Section 2.3 considers error handling aspects of the implementation.
- Section 2.4 considers security aspects of the implementation.

2.1 Normative Variations

The following subsections describe normative variations from the MUST requirements of [W3C-CANVAS2D].

2.1.1 [W3C-CANVAS2D] Section 1 Conformance requirements

V0001: Some CanvasRenderingContext2D methods do not take the proper argument types

The specification states:

```
1 Conformance requirements
...
This specification defines the 2d context type, whose API is implemented using the
CanvasRenderingContext2D interface.
...
interface CanvasRenderingContext2D {
    ...
    // transformations (default: transform is the identity matrix)
    void scale(unrestricted double x, unrestricted double y);
    void rotate(unrestricted double angle);
    void translate(unrestricted double x, unrestricted double y);
    void transform(unrestricted double a, unrestricted double b, unrestricted double c,
      unrestricted double d, unrestricted double e, unrestricted double f);
    void setTransform(unrestricted double a, unrestricted double b, unrestricted double c,
      unrestricted double d, unrestricted double e, unrestricted double f);
    ...
    // colors and styles (see also the CanvasDrawingStyles interface)
    ...
    CanvasGradient createLinearGradient(double x0, double y0, double x1, double y1);
    CanvasGradient createRadialGradient(double x0, double y0, double r0, double x1,
      double y1, double r1);
    ...
    // rects
    void clearRect(unrestricted double x, unrestricted double y, unrestricted double w,
      unrestricted double h);
    void fillRect(unrestricted double x, unrestricted double y, unrestricted double w,
      unrestricted double h);
    void strokeRect(unrestricted double x, unrestricted double y, unrestricted double w,
      unrestricted double h);
    ...
};
```

All document modes (All versions)

Some `CanvasRenderingContext2D` methods do not take the proper argument types. The methods and their argument types as implemented are:

void scale(float x, float y);

void rotate(float angle);

void translate(float x, float y);

void transform(float a, float b, float c, float d, float e, float f);

void setTransform(float a, float b, float c, float d, float e, float f);

CanvasGradient createLinearGradient(float x0, float y0, float x1, float y1);

CanvasGradient createRadialGradient(float x0, float y0, float r0, float x1, float y1, float r1);

void clearRect(float x, float y, float w, float h);

void fillRect(float x, float y, float w, float h);

void strokeRect(float x, float y, float w, float h);

V0002: The globalAlpha attribute returns a float

The specification states:

```
1 Conformance requirements
...
This specification defines the 2d context type, whose API is implemented using the
CanvasRenderingContext2D interface.
...
interface CanvasRenderingContext2D {
    ...
    // compositing
        attribute unrestricted double globalAlpha; // (default: 1.0)
    ...
};
```

All document modes (All versions)

The `globalAlpha` attribute returns a `float` instead of an `unrestricted double`:

attribute float globalAlpha;

V0003: The drawFocusIfNeeded method is not supported

The specification states:

```
1 Conformance requirements
...
This specification defines the 2d context type, whose API is implemented using the
CanvasRenderingContext2D interface.
...
interface CanvasRenderingContext2D {
    ...
    // path API (see also CanvasPathMethods)
    ...
    void drawFocusIfNeeded(Element element);
    ...
};
```

IE11 Mode, IE10 Mode, and IE9 Mode (All versions)

The `drawFocusIfNeeded` method is not supported.

V0004: The lineWidth and miterLimit attributes return a float

The specification states:

```
1 Conformance requirements
...
This specification defines the 2d context type, whose API is implemented using the
CanvasRenderingContext2D interface.
...
interface CanvasDrawingStyles {
    // line caps/joins
        attribute unrestricted double lineWidth; // (default: 1)
        ...
        attribute unrestricted double miterLimit; // (default: 10)
        ...
   };
```

All document modes (All versions)

The `lineWidth` and `miterLimit` attributes return a `float` instead of an `unrestricted double`:

attribute float lineWidth;

attribute float miterLimit;

V0005: The methods of CanvasPathMethods have arguments of type float

The specification states:

```
1 Conformance requirements
...
This specification defines the 2d context type, whose API is implemented using the
CanvasRenderingContext2D interface.
...
interface CanvasPathMethods {
    // shared path API methods
    void closePath();
    void moveTo(unrestricted double x, unrestricted double y);
    void lineTo(unrestricted double x, unrestricted double y);
    void quadraticCurveTo(unrestricted double cpx, unrestricted double cpy,
      unrestricted double x, unrestricted double y);
    void bezierCurveTo(unrestricted double cp1x, unrestricted double cp1y,
      unrestricted double cp2x, unrestricted double cp2y, unrestricted double x,
      unrestricted double y);
    void arcTo(unrestricted double x1, unrestricted double y1, unrestricted double x2,
      unrestricted double y2, unrestricted double radius);
    void rect(unrestricted double x, unrestricted double y, unrestricted double w,
      unrestricted double h);
    void arc(unrestricted double x, unrestricted double y, unrestricted double radius,
      unrestricted double startAngle, unrestricted double endAngle, optional boolean
      counterclockwise = false);
};
```

All document modes (All versions)

The methods of `CanvasPathMethods` have arguments of type `float` instead of `unrestricted double`:

```
void moveTo(float x, float y);
void lineTo(float x, float y);
void quadraticCurveTo(float cpx, float cpy, float x, float y);
void bezierCurveTo(float cp1x, float cp1y, float cp2x, float cp2y, float x, float y);
void arcTo(float x1, float y1, float x2, float y2, float radius);
void rect(float x, float y, float w, float h);
void arc(float x, float y, float radius, float startAngle, float endAngle,
  optional boolean counterclockwise = false);
```

V0006: The addColorStop method defines the offset argument as type float

The specification states:

```
1 Conformance requirements
...
This specification defines the 2d context type, whose API is implemented using the
CanvasRenderingContext2D interface.
...
interface CanvasGradient {
    // opaque object
    void addColorStop(double offset, DOMString color);
};
```

All document modes (All versions)

The `addColorStop` method defines the `offset` argument as type `float` instead of type `double`:

```
void addColorStop(float offset, DOMString color);
```

V0007: The width attribute returns a float

The specification states:

```
1 Conformance requirements
...
This specification defines the 2d context type, whose API is implemented using the
CanvasRenderingContext2D interface.
...
interface TextMetrics {
    readonly attribute double width;
};
```

All document modes (All versions)

The `width` attribute returns a `float` instead of a `double`:

```
readonly attribute float width;
```

V0008: The addHitRegion, removeHitRegion, and clearHitRegion methods are not supported

The specification states:

```
1 Conformance requirements
...
This specification defines the 2d context type, whose API is implemented using the
CanvasRenderingContext2D interface.
...
interface CanvasRenderingContext2D {
    ...
    // hit regions
    void addHitRegion(HitRegionOptions options);
    void removeHitRegion(DOMString id);
    void clearHitRegions();
    ...
};
```

All document modes (All versions)

The `addHitRegion`, `removeHitRegion`, and `clearHitRegion` methods are not supported.

2.1.2 [W3C-CANVAS2D] Section 3 Line styles

V0009: The lineCap and lineJoin attributes accept invalid values and try to use them

The specification states:

```
3 Line styles
...
The lineCap attribute defines the type of endings that UAs will place on the end of
lines. The three valid values are "butt", "round", and "square".
....
The lineJoin attribute defines the type of corners that UAs will place where two lines
meet. The three valid values are "bevel", "round", and "miter".
```

All document modes (All versions)

The `lineCap` and `lineJoin` attributes accept invalid values and try to use them.

2.1.3 [W3C-CANVAS2D] Section 4 Text styles

V0010: The 'inherit' keyword is supported but should not be

The specification states:

```
4 Text styles
...
The font IDL attribute, on setting, must be parsed the same way as the 'font' property of
CSS (but without supporting property-independent style sheet syntax like 'inherit'), ...
```

All document modes (All versions)

The `'inherit'` keyword is supported but should not be.

V0011: When the textAlign IDL attribute is set to a valid value followed by a null character the value is set

The specification states:

```
4 Text styles
...
The textAlign IDL attribute, on getting, must return the current value. On setting, if
the value is one of "start", "end", "left", "right", or "center", then the value must be
changed to the new value. Otherwise, the new value must be ignored. When the context is
created, the textAlign attribute must initially have the value "start".
```

All document modes (All versions)

When the textAlign IDL attribute is set to a valid value followed by a null character (e.g., `"end\0"`) the value is set instead of ignored.

V0012: The textBaseline attribute does not properly align the text

The specification states:

```
4 Text styles
...
The textBaseline IDL attribute, on getting, must return the current value. On setting, if
the value is one of "top", "hanging", "middle", "alphabetic", "ideographic", or "bottom",
then the value must be changed to the new value. Otherwise, the new value must be
ignored. When the object implementing the CanvasDrawingStyles interface is created, the
textBaseline attribute must initially have the value "alphabetic".
```

All document modes (All versions)

The `textBaseline` attribute, when set to any of the allowed keyword values, does not properly align the text.

2.1.4 [W3C-CANVAS2D] Section 8 Fill and stroke styles

V0013: The fillStyle attribute allows invalid values

The specification states:

```
8 Fill and stroke styles
...
The fillStyle attribute represents the color or style to use inside shapes, and the
strokeStyle attribute represents the color or style to use for the lines around the
shapes.

Both attributes can be either strings, CanvasGradients, or CanvasPatterns. On setting,
strings must be parsed as CSS <color> values and the color assigned, and CanvasGradient
and CanvasPattern objects must be assigned themselves [CSSCOLOR]. ...
```

All document modes (All versions)

The `fillStyle` attribute allows invalid values and parses them and maps them to valid values.

2.1.5 [W3C-CANVAS2D] Section 11 Drawing paths to the canvas

V0014: The drawFocusIfNeeded element is not supported

The specification states:

```
11 Drawing paths to the canvas

The context always has a current default path. There is only one current path, it is not
part of the drawing state. The current path is a path, as described above.
...
context . drawFocusIfNeeded(element)
    Informs the user of the canvas location for the fallback element, based on the
    current path. If the given element has focus, draws a focus outline around the
    current path following the platform or user agent conventions for focus outlines as
    defined by the user agent.
```

All document modes (Internet Explorer 11)

The `drawFocusIfNeeded` element is not supported.

2.2 Clarifications

The following subsections describe clarifications of the MAY and SHOULD requirements of [W3C-CANVAS2D].

2.2.1 [W3C-CANVAS2D] Section 4 Text styles

C0001: Bitmap fonts can be specified and are rendered by the browser

The specification states:

```
4 Text styles
...
Only vector fonts should be used by the user agent; if a user agent were to use bitmap
fonts then transformations would likely make the font look very ugly.
```

All document modes (All versions)

Bitmap fonts can be specified and are rendered by the browser.

2.3 Error Handling

There are no additional error handling considerations.

2.4 Security

There are no additional security considerations.

3 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

4 Index