Sprite

The Sprite object is one of the most important objects in PixiJS. It is a drawing item that can be added to a scene and rendered to the screen.

A sprite can be created directly from an image like this:

```
import { Sprite } from 'pixi.js';

const sprite = Sprite.from('assets/image.png');
```

The more efficient way to create sprites is using a Spritesheet, as swapping base textures when rendering to the screen is inefficient.

```
import { Assets, Sprite } from 'pixi.js';

const sheet = await Assets.load('assets/spritesheet.json');
const sprite = new Sprite(sheet.textures['image.png']);
```

new Sprite (options)

ts:85

Name	Type	Description
`options`	SpriteOptions | Texture	The options for creating the sprite.

Extends

- Container

Members

anchor ObservablePoint

The anchor sets the origin point of the sprite. The default value is taken from the Texture and passed to the constructor.

The default is `(0,0)`, this means the sprite's origin is the top left.

Setting the anchor to `(0.5,0.5)` means the sprite's origin is centered.

Setting the anchor to `(1,1)` would mean the sprite's origin point will be the bottom right corner.

If you pass only single parameter, it will set both x and y to the same value as shown in the example below.

Example

```
import { Sprite } from 'pixi.js';

const sprite = new Sprite({texture: Texture.WHITE});
sprite.anchor.set(0.5); // This will set the origin to center. (0.5) is same as (0.5, 0.5).
```

height number overrides

The height of the sprite, setting this will actually modify the scale to achieve the value set.

~~sourceBounds~~ Deprecated

texture

The texture that the sprite is using.

visualBounds Bounds

The bounds of the sprite, taking the texture's trim into account.

width number overrides

The width of the sprite, setting this will actually modify the scale to achieve the value set.

Methods

from (source, skipCache)Sprite static

ts:55

Helper function that creates a new sprite based on the source you provide. The source can be - frame id, image, video, canvas element, video element, texture

Name	Type	Attributes	Default	Description
`source`	Texture | TextureSourceLike			Source to create texture from
`skipCache`	boolean	<optional>	false	Whether to skip the cache or not

Returns:

Type	Description
Sprite	The newly created sprite

destroy (options)overrides

ts:203

Destroys this sprite renderable and optionally its texture.

Name	Type	Attributes	Default	Description
`options`	DestroyOptions		false	Options parameter. A boolean will act as if all options have been set to that value
`options.texture`	boolean	<optional>	false	Should it destroy the current texture of the renderable as well
`options.textureSource`	boolean	<optional>	false	Should it destroy the textureSource of the renderable as well

getSize (out)Size overrides

ts:280

Retrieves the size of the Sprite as a Size object. This is faster than get the width and height separately.

Name	Type	Attributes	Description
`out`	Size	<optional>	Optional object to store the size in.

Returns:

Type	Description
Size	• The size of the Sprite.

setSize (value, height)overrides

ts:295

Sets the size of the Sprite to the specified width and height. This is faster than setting the width and height separately.

Name	Type	Attributes	Description
`value`	number | Optional<Size, "height">		This can be either a number or a Size object.
`height`	number	<optional>	The height to set. Defaults to the value of `width` if not provided.

Inherited Properties

From class Container

accessible boolean inherited

Flag for if the object is accessible. If true AccessibilityManager will overlay a shadow div with attributes set

Default Value:

- false

accessibleChildren boolean inherited

Setting to false will prevent any children inside this container to be accessible. Defaults to true.

Default Value:

- true

accessibleHint string inherited

Sets the aria-label attribute of the shadow div

Default Value:

- undefined

accessiblePointerEvents PointerEvents inherited

Specify the pointer-events the accessible div will use Defaults to auto.

Default Value:

- 'auto'

accessibleText string inherited

Sets the text content of the shadow div

Default Value:

- undefined

accessibleTitle string inherited

Sets the title attribute of the shadow div If accessibleTitle AND accessibleHint has not been this will default to 'container [tabIndex]'

Default Value:

- undefined

accessibleType string inherited

Specify the type of div the accessible layer is. Screen readers treat the element differently depending on this type. Defaults to button.

Default Value:

- 'button'

alpha number inherited

The opacity of the object.

angle number inherited

The angle of the object in degrees. 'rotation' and 'angle' have the same effect on a display object; rotation is in radians, angle is in degrees.

blendMode BLEND_MODES inherited

The blend mode to be applied to the sprite. Apply a value of `'normal'` to reset the blend mode.

Default Value:

- 'normal'

boundsArea Rectangle inherited

An optional bounds area for this container. Setting this rectangle will stop the renderer from recursively measuring the bounds of each children and instead use this single boundArea. This is great for optimisation! If for example you have a 1000 spinning particles and you know they all sit within a specific bounds, then setting it will mean the renderer will not need to measure the 1000 children to find the bounds. Instead it will just use the bounds you set.

~~**cacheAsBitmap** boolean~~**Deprecated : Since PixiJS v8**inherited

Legacy property for backwards compatibility with PixiJS v7 and below. Use `cacheAsTexture` instead.

children C[] readonly inherited

The array of children of this container.

cullable boolean inherited

Should this object be rendered if the bounds of this object are out of frame?

Culling has no effect on whether updateTransform is called.

Default Value:

- false

cullableChildren boolean inherited

Determines if the children to the container can be culled Setting this to false allows PixiJS to bypass a recursive culling function Which can help to optimize very complex scenes

Default Value:

- true

cullArea Rectangle inherited

If set, this shape is used for culling instead of the bounds of this object. It can improve the culling performance of objects with many children. The culling area is defined in local space.

destroyed boolean inherited

If the object has been destroyed via destroy(). If true, it should not be used.

Default Value:

- false

effects Array<Effect> inherited

TODO

- Needs docs.

groupTransform Matrix readonly inherited

The group transform is a transform relative to the render group it belongs too. If this container is render group then this will be an identity matrix. other wise it will be the same as the relativeGroupTransform. Use this value when actually rendering things to the screen

hitArea IHitArea inherited

Interaction shape. Children will be hit first, then this shape will be checked. Setting this will cause this shape to be checked in hit tests rather than the container's bounds.

Default Value:

- undefined

Example

```
import { Rectangle, Sprite } from 'pixi.js';

const sprite = new Sprite(texture);
sprite.interactive = true;
sprite.hitArea = new Rectangle(0, 0, 100, 100);
```

interactiveChildren boolean inherited

Determines if the children to the container can be clicked/touched Setting this to false allows PixiJS to bypass a recursive `hitTest` function

Default Value:

- true

isCachedAsTexture boolean readonly inherited

Whether this container is currently cached as a texture.

isRenderable boolean inherited

Whether or not the object should be rendered.

isRenderGroup boolean inherited

Returns true if this container is a render group. This means that it will be rendered as a separate pass, with its own set of instructions

label string inherited

The instance label of the object.

Default Value:

- undefined

localTransform Matrix readonly inherited

Current transform of the object based on local factors: position, scale, other stuff.

~~name~~ stringDeprecated : since 8.0.0inherited

The instance name of the object.

See:

- label

onclick inherited

Property-based event handler for the `click` event.

Default Value:

- null

Example

```
this.onclick = (event) => {
 //some function here that happens on click
}
```

onglobalmousemove inherited

Property-based event handler for the `globalmousemove` event.

Default Value:

- null

Example

```
this.onglobalmousemove = (event) => {
 //some function here that happens on globalmousemove
}
```

onglobalpointermove inherited

Property-based event handler for the `globalpointermove` event.

Default Value:

- null

Example

```
this.onglobalpointermove = (event) => {
 //some function here that happens on globalpointermove
}
```

onglobaltouchmove inherited

Property-based event handler for the `globaltouchmove` event.

Default Value:

- null

Example

```
this.onglobaltouchmove = (event) => {
 //some function here that happens on globaltouchmove
}
```

onmousedown inherited

Property-based event handler for the `mousedown` event.

Default Value:

- null

Example

```
this.onmousedown = (event) => {
 //some function here that happens on mousedown
}
```

onmouseenter inherited

Property-based event handler for the `mouseenter` event.

Default Value:

- null

Example

```
this.onmouseenter = (event) => {
 //some function here that happens on mouseenter
}
```

onmouseleave inherited

Property-based event handler for the `mouseleave` event.

Default Value:

- null

Example

```
this.onmouseleave = (event) => {
 //some function here that happens on mouseleave
}
```

onmousemove inherited

Property-based event handler for the `mousemove` event.

Default Value:

- null

Example

```
this.onmousemove = (event) => {
 //some function here that happens on mousemove
}
```

onmouseout inherited

Property-based event handler for the `mouseout` event.

Default Value:

- null

Example

```
this.onmouseout = (event) => {
 //some function here that happens on mouseout
}
```

onmouseover inherited

Property-based event handler for the `mouseover` event.

Default Value:

- null

Example

```
this.onmouseover = (event) => {
 //some function here that happens on mouseover
}
```

onmouseup inherited

Property-based event handler for the `mouseup` event.

Default Value:

- null

Example

```
this.onmouseup = (event) => {
 //some function here that happens on mouseup
}
```

onmouseupoutside inherited

Property-based event handler for the `mouseupoutside` event.

Default Value:

- null

Example

```
this.onmouseupoutside = (event) => {
 //some function here that happens on mouseupoutside
}
```

onpointercancel inherited

Property-based event handler for the `pointercancel` event.

Default Value:

- null

Example

```
this.onpointercancel = (event) => {
 //some function here that happens on pointercancel
}
```

onpointerdown inherited

Property-based event handler for the `pointerdown` event.

Default Value:

- null

Example

```
this.onpointerdown = (event) => {
 //some function here that happens on pointerdown
}
```

onpointerenter inherited

Property-based event handler for the `pointerenter` event.

Default Value:

- null

Example

```
this.onpointerenter = (event) => {
 //some function here that happens on pointerenter
}
```

onpointerleave inherited

Property-based event handler for the `pointerleave` event.

Default Value:

- null

Example

```
this.onpointerleave = (event) => {
 //some function here that happens on pointerleave
}
```

onpointermove inherited

Property-based event handler for the `pointermove` event.

Default Value:

- null

Example

```
this.onpointermove = (event) => {
 //some function here that happens on pointermove
}
```

onpointerout inherited

Property-based event handler for the `pointerout` event.

Default Value:

- null

Example

```
this.onpointerout = (event) => {
 //some function here that happens on pointerout
}
```

onpointerover inherited

Property-based event handler for the `pointerover` event.

Default Value:

- null

Example

```
this.onpointerover = (event) => {
 //some function here that happens on pointerover
}
```

onpointertap inherited

Property-based event handler for the `pointertap` event.

Default Value:

- null

Example

```
this.onpointertap = (event) => {
 //some function here that happens on pointertap
}
```

onpointerup inherited

Property-based event handler for the `pointerup` event.

Default Value:

- null

Example

```
this.onpointerup = (event) => {
 //some function here that happens on pointerup
}
```

onpointerupoutside inherited

Property-based event handler for the `pointerupoutside` event.

Default Value:

- null

Example

```
this.onpointerupoutside = (event) => {
 //some function here that happens on pointerupoutside
}
```

onrightclick inherited

Property-based event handler for the `rightclick` event.

Default Value:

- null

Example

```
this.onrightclick = (event) => {
 //some function here that happens on rightclick
}
```

onrightdown inherited

Property-based event handler for the `rightdown` event.

Default Value:

- null

Example

```
this.onrightdown = (event) => {
 //some function here that happens on rightdown
}
```

onrightup inherited

Property-based event handler for the `rightup` event.

Default Value:

- null

Example

```
this.onrightup = (event) => {
 //some function here that happens on rightup
}
```

onrightupoutside inherited

Property-based event handler for the `rightupoutside` event.

Default Value:

- null

Example

```
this.onrightupoutside = (event) => {
 //some function here that happens on rightupoutside
}
```

ontap inherited

Property-based event handler for the `tap` event.

Default Value:

- null

Example

```
this.ontap = (event) => {
 //some function here that happens on tap
}
```

ontouchcancel inherited

Property-based event handler for the `touchcancel` event.

Default Value:

- null

Example

```
this.ontouchcancel = (event) => {
 //some function here that happens on touchcancel
}
```

ontouchend inherited

Property-based event handler for the `touchend` event.

Default Value:

- null

Example

```
this.ontouchend = (event) => {
 //some function here that happens on touchend
}
```

ontouchendoutside inherited

Property-based event handler for the `touchendoutside` event.

Default Value:

- null

Example

```
this.ontouchendoutside = (event) => {
 //some function here that happens on touchendoutside
}
```

ontouchmove inherited

Property-based event handler for the `touchmove` event.

Default Value:

- null

Example

```
this.ontouchmove = (event) => {
 //some function here that happens on touchmove
}
```

ontouchstart inherited

Property-based event handler for the `touchstart` event.

Default Value:

- null

Example

```
this.ontouchstart = (event) => {
 //some function here that happens on touchstart
}
```

onwheel inherited

Property-based event handler for the `wheel` event.

Default Value:

- null

Example

```
this.onwheel = (event) => {
 //some function here that happens on wheel
}
```

parent Container inherited

The display object container that contains this display object.

Default Value:

- undefined

parentRenderLayer IRenderLayer readonly inherited

The RenderLayer this container belongs to, if any. If it belongs to a RenderLayer, it will be rendered from the RenderLayer's position in the scene.

pivot ObservablePoint inherited

The center of rotation, scaling, and skewing for this display object in its local space. The `position` is the projection of `pivot` in the parent's local space.

By default, the pivot is the origin (0, 0).

Since:

- 4.0.0

position ObservablePoint inherited

The coordinate of the object relative to the local coordinates of the parent.

Since:

- 4.0.0

relativeGroupTransform Matrix readonly inherited

The relative group transform is a transform relative to the render group it belongs too. It will include all parent transforms and up to the render group (think of it as kind of like a stage - but the stage can be nested). If this container is is self a render group matrix will be relative to its parent render group

renderable inherited

Can this object be rendered, if false the object will not be drawn but the transform will still be updated.

rotation number inherited

The rotation of the object in radians. 'rotation' and 'angle' have the same effect on a display object; rotation is in radians, angle is in degrees.

scale ObservablePoint inherited

The scale factors of this object along the local coordinate axes.

The default scale is (1, 1).

Since:

- 4.0.0

skew ObservablePoint inherited

The skew factor for the object in radians.

Since:

- 4.0.0

sortableChildren boolean inherited

If set to true, the container will sort its children by `zIndex` value when the next render is called, or manually if `sortChildren()` is called.

This actually changes the order of elements in the array, so should be treated as a basic solution that is not performant compared to other solutions, such as PixiJS Layers

Also be aware of that this may not work nicely with the `addChildAt()` function, as the `zIndex` sorting may cause the child to automatically sorted to another position.

Default Value:

- false

sortDirty boolean inherited

Should children be sorted by zIndex at the next render call.

Will get automatically set to true if a new child is added, or if a child's zIndex changes.

Default Value:

- false

tabIndex number inherited

Default Value:

- 0

TODO

- Needs docs.

tint number inherited

The tint applied to the sprite. This is a hex value.

A value of 0xFFFFFF will remove any tint effect.

Default Value:

- 0xFFFFFF

uid number readonly inherited

unique id for this container

updateCacheTexture () ⇒ void inherited

Updates the cached texture of this container. This will flag the container's cached texture to be redrawn on the next render.

visible inherited

The visibility of the object. If false the object will not be drawn, and the transform will not be updated.

worldTransform readonly inherited

Current transform of the object based on world (parent) factors.

x number inherited

The position of the container on the x axis relative to the local coordinates of the parent. An alias to position.x

y number inherited

The position of the container on the y axis relative to the local coordinates of the parent. An alias to position.y

Inherited Methods

From class Container

_getGlobalBoundsRecursive (factorRenderLayers, bounds, currentLayer)void inherited

ts:31

Recursively calculates the global bounds for the container and its children. This method is used internally by getFastGlobalBounds to traverse the scene graph.

Name	Type	Description
`factorRenderLayers`	boolean	A flag indicating whether to consider render layers in the calculation.
`bounds`	Bounds	The bounds object to update with the calculated values.

`currentLayer`	IRenderLayer	The current render layer being processed.

addChild (...children)Container inherited

ts:625

Adds one or more children to the container.

Multiple items can be added like so: `myContainer.addChild(thingOne, thingTwo, thingThree)`

Name	Type	Description
`children`	Container	The Container(s) to add to the container

Returns:

Type	Description
Container	• The first child that was added.

addChildAt (child, index)Container inherited

ts:142

Adds a child to the container at a specified index. If the index is out of bounds an error will be thrown. If the child is already in this container, it will be moved to the specified index.

Name	Type	Description
`child`	Container	The child to add.
`index`	number	The absolute index where the child will be positioned at the end of the operation.

Returns:

Type	Description
Container	The child that was added.

addEventListener (type, listener, options)inherited

ts:681

Unlike `on` or `addListener` which are methods from EventEmitter, `addEventListener` seeks to be compatible with the DOM's `addEventListener` with support for options.

Name	Type	Attributes	Description
`type`	string		The type of event to listen to.
`listener`	EventListenerOrEventListenerObject		The listener callback or object.
`options`	AddListenerOptions	<optional>	Listener options, used for capture phase.

Example

```
// Tell the user whether they did a single, double, triple, or nth click.
button.addEventListener('click', {
    handleEvent(e): {
        let prefix;
```

```
        switch (e.detail) {
            case 1: prefix = 'single'; break;
            case 2: prefix = 'double'; break;
            case 3: prefix = 'triple'; break;
            default: prefix = e.detail + 'th'; break;
        }

        console.log('That was a ' + prefix + 'click');
    }
});

// But skip the first click!
button.parent.addEventListener('click', function blockClickOnce(e) {
    e.stopImmediatePropagation();
    button.parent.removeEventListener('click', blockClickOnce, true);
}, {
    capture: true,
});
```

cacheAsTexture (val)void inherited

ts:13

Caches this container as a texture. This allows the container to be rendered as a single texture, which can improve performance for complex static containers.

Name	Type	Description
`val`	boolean | CacheAsTextureOptions	If true, enables caching with default options. If false, disables caching. Can also pass options object to configure caching behavior.

collectRenderables (instructionSet, renderer, currentLayer)void inherited

ts:15

Collects all renderables from the container and its children, adding them to the instruction set. This method decides whether to use a simple or advanced collection method based on the container's properties.

Name	Type	Description
`instructionSet`	InstructionSet	The set of instructions to which the renderables will be added.
`renderer`	Renderer	The renderer responsible for rendering the scene.
`currentLayer`	IRenderLayer	The current render layer being processed.

collectRenderablesSimple (instructionSet, renderer, currentLayer)void inherited

ts:25

Collects renderables using a simple method, suitable for containers marked as simple. This method iterates over the container's children and adds their renderables to the instruction set.

Name	Type	Description
`instructionSet`	InstructionSet	The set of instructions to which the renderables will be added.
`renderer`	Renderer	The renderer responsible for rendering the scene.
`currentLayer`	IRenderLayer	The current render layer being processed.

collectRenderablesWithEffects (instructionSet, renderer, currentLayer)void inherited

ts:35

Collects renderables using an advanced method, suitable for containers with complex processing needs. This method handles additional effects and transformations that may be applied to the renderables.

Name	Type	Description
`instructionSet`	InstructionSet	The set of instructions to which the renderables will be added.
`renderer`	Renderer	The renderer responsible for rendering the scene.
`currentLayer`	IRenderLayer	The current render layer being processed.

disableRenderGroup ()void inherited

ts:826

This will disable the render group for this container.

dispatchEvent (e)boolean inherited

ts:772

Dispatch the event on this Container using the event's EventBoundary.

The target of the event is set to `this` and the `defaultPrevented` flag is cleared before dispatch.

Name	Type	Description
`e`	Event	The event to dispatch.

Returns:

Type	Description
boolean	Whether the preventDefault() method was not invoked.

Example

```
// Reuse a click event!
button.dispatchEvent(clickEvent);
```

enableRenderGroup ()void inherited

ts:802

Calling this enables a render group for this container. This means it will be rendered as a separate set of instructions. The transform of the container will also be handled on the GPU rather than the CPU.

eventMode (value)inherited

ts:601

Enable interaction events for the Container. Touch, pointer and mouse. There are 5 types of interaction settings:

- `'none'`: Ignores all interaction events, even on its children.
- `'passive'`: **(default)** Does not emit events and ignores all hit testing on itself and non-interactive children. Interactive children will still emit events.
- `'auto'`: Does not emit events but is hit tested if parent is interactive. Same as `interactive = false` in v7
- `'static'`: Emit events and is hit tested. Same as `interaction = true` in v7
- `'dynamic'`: Emits events and is hit tested but will also receive mock interaction events fired from a ticker to allow for interaction when the mouse isn't moving

Name	Type	Description
`value`		

Since:

- 7.2.0

Example

```
import { Sprite } from 'pixi.js';

const sprite = new Sprite(texture);
sprite.eventMode = 'static';
sprite.on('tap', (event) => {
    // Handle event
});
```

filterArea ()Rectangle inherited

ts:238

The area the filter is applied to. This is used as more of an optimization rather than figuring out the dimensions of the displayObject each frame you can set this rectangle.

Also works as an interaction mask.

Returns:

Type	Description
Rectangle	

filters ()readonly inherited

ts:191

Sets the filters for the displayObject. IMPORTANT: This is a WebGL only feature and will be ignored by the canvas renderer. To remove filters simply set this property to `'null'`.

Returns:

Type	Description
readonly	

getBounds (skipUpdate, bounds)Bounds inherited

ts:111

Calculates and returns the (world) bounds of the display object as a Rectangle.

Name	Type	Attributes	Description
`skipUpdate`	boolean	<optional>	Setting to `true` will stop the transforms of the scene graph

			from being updated. This means the calculation returned MAY be out of date BUT will give you a nice performance boost.
`bounds`	Bounds	<optional>	Optional bounds to store the result of the bounds calculation.

Returns:

Type	Description
Bounds	• The minimum axis-aligned rectangle in world space that fits around this object.

getChildAt (index)U inherited

ts:91

Returns the child at the specified index

Name	Type	Description
`index`	number	The index to get the child at

Returns:

Type	Description
U	• The child at the given index, if any.

getChildByLabel (label, deep)Container inherited

ts:66

Returns the first child in the container with the specified label.

Recursive searches are done in a pre-order traversal.

Name	Type	Attributes	Default	Description
`label`	string | RegExp			Instance label.
`deep`	boolean	<optional>	false	Whether to search recursively

Returns:

Type	Description
Container	The child with the specified label.

~~getChildByName~~ (name, deep)ContainerDeprecated`` : since 8.0.0inherited

ts:53

Name	Type	Attributes	Default	Description
`name`	string			Instance name.

`deep`	boolean	<optional>	false	Whether to search recursively

See:

- getChildByLabel

Returns:

Type	Description
Container	The child with the specified name.

getChildIndex (child)number inherited

ts:124

Returns the index position of a child Container instance

Name	Type	Description
`child`	ContainerChild | IRenderLayer	The Container instance to identify

Returns:

Type	Description
number	• The index position of the child container to identify

getChildrenByLabel (label, deep, out)Array<Container> inherited

ts:103

Returns all children in the container with the specified label.

Name	Type	Attributes	Default	Description
`label`	string | RegExp			Instance label.
`deep`	boolean	<optional>	false	Whether to search recursively
`out`	Array<Container>	<optional>	[]	The array to store matching children in.

Returns:

Type	Description
Array<Container>	An array of children with the specified label.

getFastGlobalBounds (factorRenderLayers, bounds)Bounds inherited

ts:17

Computes an approximate global bounding box for the container and its children. This method is optimized for speed by using axis-aligned bounding boxes (AABBs), and uses the last render results from when it updated the transforms. This function does not update them. which may result in slightly larger bounds but never smaller than the actual bounds.
for accurate (but less performant) results use `container.getGlobalBounds`

Name	Type	Attributes	Description
`factorRenderLayers`	boolean	<optional>	A flag indicating whether to consider render layers in the calculation.
`bounds`	Bounds	<optional>	The output bounds object to store the result. If not

			provided, a new one is created.

Returns:

Type	Description
Bounds	The computed bounds.

getGlobalPosition (point, skipUpdate)Point inherited

ts:15

Returns the global position of the container.

Name	Type	Default	Description
`point`	Point		The optional point to write the global value to.
`skipUpdate`	boolean	false	Should we skip the update transform.

Returns:

Type	Description
Point	• The updated point.

getLocalBounds ()Bounds inherited

ts:73

Retrieves the local bounds of the container as a Bounds object.

Returns:

Type	Description
Bounds	• The bounding area.

interactive (value)inherited

ts:585

Enable interaction events for the Container. Touch, pointer and mouse

Name	Type	Description
`value`	boolean	

isCachedAsTexture ()boolean inherited

ts:46

Is this container cached as a texture?

Returns:

Type	Description
boolean	

isInteractive ()boolean inherited

ts:631

Determines if the container is interactive or not

Since:

- 7.2.0

Returns:

Type	Description
boolean	Whether the container is interactive or not

Example

```
import { Sprite } from 'pixi.js';

const sprite = new Sprite(texture);
sprite.eventMode = 'static';
sprite.isInteractive(); // true

sprite.eventMode = 'dynamic';
sprite.isInteractive(); // true

sprite.eventMode = 'none';
sprite.isInteractive(); // false

sprite.eventMode = 'passive';
sprite.isInteractive(); // false

sprite.eventMode = 'auto';
sprite.isInteractive(); // false
```

mask ()unknown inherited

ts:110

Sets a mask for the displayObject. A mask is an object that limits the visibility of an object to the shape of the mask applied to it. In PixiJS a regular mask must be a Graphics or a {@link Sprite} object. This allows for much faster masking in canvas as it utilities shape clipping. Furthermore, a mask of an object must be in the subtree of its parent. Otherwise, `getLocalBounds` may calculate incorrect bounds, which makes the container's width and height wrong. To remove a mask, set this property to `null`.

For sprite mask both alpha and red channel are used. Black mask is the same as transparent mask.

Returns:

Type	Description
unknown	

Example

```
import { Graphics, Sprite } from 'pixi.js';

const graphics = new Graphics();
graphics.beginFill(0xFF3300);
graphics.drawRect(50, 250, 100, 100);
graphics.endFill();

const sprite = new Sprite(texture);
sprite.mask = graphics;
```

onRender ()(renderer: Renderer) ⇒ void inherited

ts:16

This callback is used when the container is rendered. This is where you should add your custom logic that is needed to be run every frame. In v7 many users used `updateTransform` for this, however the way v8 renders objects is different and "updateTransform" is no longer called every frame

Returns:

Type	Description
(renderer: Renderer) ⇒ void	

Example

```
const container = new Container();
container.onRender = () => {
   container.rotation += 0.01;
};
```

removeChild (...children)Container inherited

ts:704

Removes one or more children from the container.

Name	Type	Description
`children`	Container	The Container(s) to remove

Returns:

Type	Description
Container	The first child that was removed.

removeChildAt (index)U inherited

ts:78

Removes a child from the specified index position.

Name	Type	Description
`index`	number	The index to get the child from

Returns:

Type	Description
U	The child that was removed.

removeChildren (beginIndex, endIndex)ContainerChild[] inherited

ts:29

Removes all children from this container that are within the begin and end indexes.

Name	Type	Attributes	Default	Description
`beginIndex`	number		0	The beginning position.
`endIndex`	number	<optional>		The ending position. Default value is size of the container.

Returns:

Type	Description
ContainerChild[]	• List of removed children

removeEventListener (type, listener, options)inherited

ts:748

Unlike `off` or `removeListener` which are methods from EventEmitter, `removeEventListener` seeks to be compatible with the DOM's `removeEventListener` with support for options.

Name	Type	Attributes	Description
`type`	string		The type of event the listener is bound to.
`listener`	EventListenerOrEventListenerObject		The listener callback or object.
`options`	RemoveListenerOptions	<optional>	The original listener options. This is required to deregister a capture phase listener.

removeFromParent ()inherited

ts:240

Remove the Container from its parent Container. If the Container has no parent, do nothing.

reparentChild (...child)U[0] inherited

ts:249

Reparent the child to this container, keeping the same worldTransform.

Name	Type	Description
`child`	U	The child to reparent

Returns:

Type	Description
U[0]	The first child that was reparented.

reparentChildAt (child, index)U inherited

ts:267

Reparent the child to this container at the specified index, keeping the same worldTransform.

Name	Type	Description
`child`	U	The child to reparent
`index`	number	The index to reparent the child to

Returns:

Type	Description
U	

setChildIndex (child, index)void inherited

ts:107

Changes the position of an existing child in the container

Name	Type	Description
`child`	ContainerChild | IRenderLayer	The child Container instance for which you want to change the index number
`index`	number	The resulting index number for the child container

setFromMatrix (matrix)void inherited

ts:1149

Updates the local transform using the given matrix.

Name	Type	Description
`matrix`	Matrix	The matrix to use for updating the transform.

setMask (options)inherited

ts:132

Used to set mask and control mask options.

Name	Type	Description
`options`	Partial<MaskOptionsAndMask>	

Example

```
import { Graphics, Sprite } from 'pixi.js';

const graphics = new Graphics();
graphics.beginFill(0xFF3300);
graphics.drawRect(50, 250, 100, 100);
graphics.endFill();

const sprite = new Sprite(texture);
sprite.setMask({
    mask: graphics,
    inverse: true,
});
```

swapChildren (child, child2)void inherited

ts:212

Swaps the position of 2 Containers within this container.

Name	Type	Description
`child`	U	First container to swap
`child2`	U	Second container to swap

toGlobal (position, point, skipUpdate)P inherited

ts:37

Calculates the global position of the container.

Name	Type	Attributes	Default	Description
`position`	PointData			The world origin to calculate from.
`point`	P	<optional>		A Point object in which to store the value, optional (otherwise will create a new Point).
`skipUpdate`	boolean		false	Should we skip the update transform.

Returns:

Type	Description
P	• A point object representing the position of this object.

toLocal (position, from, point, skipUpdate)P inherited

ts:58

Calculates the local position of the container relative to another point.

Name	Type	Attributes	Description
`position`	PointData		The world origin to calculate from.
`from`	Container	<optional>	The Container to calculate the global position from.
`point`	P	<optional>	A Point object in which to store the

			value, optional (otherwise will create a new Point).
`skipUpdate`	boolean	<optional>	Should we skip the update transform

Returns:

Type	Description
P	• A point object representing the position of this object

updateCacheTexture ()void inherited

ts:70

Updates the cached texture. Will flag that this container's cached texture needs to be redrawn. This will happen on the next render.

updateLocalTransform ()void inherited

ts:1158

Updates the local transform.

updateTransform (opts)this inherited

ts:1113

Updates the transform properties of the container (accepts partial values).

Name	Type	Description
`opts`	object	The options for updating the transform.
`opts.x`	number	The x position of the container.
`opts.y`	number	The y position of the container.
`opts.scaleX`	number	The scale factor on the x-axis.
`opts.scaleY`	number	The scale factor on the y-axis.
`opts.rotation`	number	The rotation of the container, in radians.
`opts.skewX`	number	The skew factor on the x-axis.
`opts.skewY`	number	The skew factor on the y-axis.
`opts.pivotX`	number	The x coordinate of the pivot point.
`opts.pivotY`	number	The y coordinate of the pivot point.

Returns:

Type	Description
this	

zIndex (value)inherited

ts:42

The zIndex of the container.

Setting this value, will automatically set the parent to be sortable. Children will be automatically sorted by zIndex value; a higher value will mean it will be moved towards the end of the array, and thus rendered on top of other display objects within the same container.

Name	Type	Description
`value`	number	

See:

- sortableChildren

Inherited Events

From class Container

click inherited

ts:120

Fired when a pointer device button (usually a mouse left-button) is pressed and released on the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

A `click` event fires after the `pointerdown` and `pointerup` events, in that order. If the mouse is moved over another Container after the `pointerdown` event, the `click` event is fired on the most specific common ancestor of the two target Containers.

The `detail` property of the event is the number of clicks that occurred within a 200ms window of each other upto the current click. For example, it will be `2` for a double click.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

clickcapture inherited

ts:136

Capture phase equivalent of `click`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

globalmousemove inherited

ts:206

Fired when a pointer device (usually a mouse) is moved globally over the scene. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

globalpointermove inherited

ts:391

Fired when a pointer device is moved globally over the scene. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

globaltouchmove inherited

ts:571

Fired when a touch point is moved globally over the scene. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mousedown inherited

ts:52

Fired when a mouse button (usually a mouse left-button) is pressed on the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	The mousedown event.

mousedowncapture inherited

ts:61

Capture phase equivalent of `mousedown`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	The capture phase mousedown.

mouseenter inherited

ts:249

Fired when the mouse pointer is moved over a Container and its descendant's hit testing boundaries.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mouseentercapture inherited

ts:257

Capture phase equivalent of `mouseenter`

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mouseleave inherited

ts:285

Fired when the mouse pointer exits a Container and its descendants.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	

mouseleavecapture inherited

ts:293

Capture phase equivalent of `mouseleave`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mousemove inherited

ts:215

Fired when a pointer device (usually a mouse) is moved while over the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mousemovecapture inherited

ts:224

Capture phase equivalent of `mousemove`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mouseout inherited

ts:265

Fired when a pointer device (usually a mouse) is moved off the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

This may be fired on a Container that was removed from the scene graph immediately after a `mouseover` event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mouseoutcapture inherited

ts:277

Capture phase equivalent of `mouseout`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mouseover inherited

ts:232

Fired when a pointer device (usually a mouse) is moved onto the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mouseovercapture inherited

ts:241

Capture phase equivalent of `mouseover`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mouseup inherited

ts:86

Fired when a pointer device button (usually a mouse left-button) is released over the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mouseupcapture inherited

ts:95

Capture phase equivalent of `mouseup`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mouseupoutside inherited

ts:164

Fired when a pointer device button (usually a mouse left-button) is released outside the container that initially registered a mousedown. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

This event is specific to the Federated Events API. It does not have a capture phase, unlike most of the other events. It only bubbles to the most specific ancestor of the targets of the corresponding `pointerdown` and `pointerup` events, i.e. the target of the `click` event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

mouseupoutsidecapture inherited

ts:179

Capture phase equivalent of `mouseupoutside`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointercancel inherited

ts:335

Fired when the operating system cancels a pointer event. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointercancelcapture inherited

ts:344

Capture phase equivalent of `pointercancel`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointerdown inherited

ts:301

Fired when a pointer device button is pressed on the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointerdowncapture inherited

ts:310

Capture phase equivalent of `pointerdown`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointerenter inherited

ts:434

Fired when the pointer is moved over a Container and its descendant's hit testing boundaries.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointerentercapture inherited

ts:442

Capture phase equivalent of `pointerenter`

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointerleave inherited

ts:467

Fired when the pointer leaves the hit testing boundaries of a Container and its descendants.

This event notifies only the target and does not bubble.
These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	The `pointerleave` event.

pointerleavecapture inherited

ts:477

Capture phase equivalent of `pointerleave`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointermove inherited

ts:400

Fired when a pointer device is moved while over the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description

`event`	FederatedPointerEvent	Event

pointermovecapture inherited

ts:409

Capture phase equivalent of `pointermove`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointerout inherited

ts:450

Fired when a pointer device is moved off the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointeroutcapture inherited

ts:459

Capture phase equivalent of `pointerout`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointerover inherited

ts:417

Fired when a pointer device is moved onto the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointerovercapture inherited

ts:426

Capture phase equivalent of `pointerover`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointertap inherited

ts:352

Fired when a pointer device button is pressed and released on the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointertapcapture inherited

ts:361

Capture phase equivalent of `pointertap`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointerup inherited

ts:318

Fired when a pointer device button is released over the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointerupcapture inherited

ts:327

Capture phase equivalent of `pointerup`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointerupoutside inherited

ts:369

Fired when a pointer device button is released outside the container that initially registered a pointerdown. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

This event is specific to the Federated Events API. It does not have a capture phase, unlike most of the other events. It only bubbles to the most specific ancestor of the targets of the corresponding `pointerdown` and `pointerup` events, i.e. the target of the `click` event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

pointerupoutsidecapture inherited

ts:383

Capture phase equivalent of `pointerupoutside`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

rightclick inherited

ts:144

Fired when a pointer device secondary button (usually a mouse right-button) is pressed and released on the container.

Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

This event follows the semantics of `click`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

rightclickcapture inherited

ts:156

Capture phase equivalent of `rightclick`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

rightdown inherited

ts:69

Fired when a pointer device secondary button (usually a mouse right-button) is pressed on the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

rightdowncapture inherited

ts:78

Capture phase equivalent of `rightdown`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	The rightdowncapture event.

rightup inherited

ts:103

Fired when a pointer device secondary button (usually a mouse right-button) is released over the container.

Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

rightupcapture inherited

ts:112

Capture phase equivalent of `rightup`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

rightupoutside inherited

ts:187

Fired when a pointer device secondary button (usually a mouse right-button) is released outside the container that initially registered a rightdown. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

rightupoutsidecapture inherited

ts:198

Capture phase equivalent of `rightupoutside`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

tap inherited

ts:536

Fired when a touch point is placed and removed from the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

tapcapture inherited

ts:545

Capture phase equivalent of `tap`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

touchcancel inherited

ts:519

Fired when the operating system cancels a touch. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

touchcancelcapture inherited

ts:528

Capture phase equivalent of `touchcancel`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

touchend inherited

ts:502

Fired when a touch point is removed from the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

touchendcapture inherited

ts:511

Capture phase equivalent of `touchend`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

touchendoutside inherited

ts:553

Fired when a touch point is removed outside of the container that initially registered a touchstart. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

touchendoutsidecapture inherited

ts:563

Capture phase equivalent of `touchendoutside`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

touchmove inherited

ts:580

Fired when a touch point is moved along the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

touchmovecapture inherited

ts:589

Capture phase equivalent of `touchmove`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

touchstart inherited

ts:485

Fired when a touch point is placed on the container. Container's `eventMode` property must be set to `static` or 'dynamic' to fire event.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

touchstartcapture inherited

ts:494

Capture phase equivalent of `touchstart`.

These events are propagating from the EventSystem.

Name	Type	Description
`event`	FederatedPointerEvent	Event

wheel inherited

ts:597

Fired when a the user scrolls with the mouse cursor over a Container.

These events are propagating from the EventSystem.

Type:

- FederatedWheelEvent

wheelcapture inherited

ts:605

Capture phase equivalent of `wheel`.

These events are propagating from the EventSystem.

Type:

- FederatedWheelEvent

assets

A one stop shop for all Pixi resource management! Super modern and easy to use, with enough flexibility to customize and do what you need!

Use the singleton class Assets to easily load and manage all your assets.

```
import { Assets, Texture } from 'pixi.js';

const bunnyTexture = await Assets.load<Texture>('bunny.png');
const sprite = new Sprite(bunnyTexture);
```

Check out the sections below for more information on how to deal with assets.

Asset Loading

Do not be afraid to load things multiple times - under the hood, it will **NEVER** load anything more than once.

For example:

```
import { Assets } from 'pixi.js';

promise1 = Assets.load('bunny.png')
promise2 = Assets.load('bunny.png')

// promise1 === promise2
```

Here both promises will be the same. Once resolved... Forever resolved! It makes for really easy resource management!

Out of the box Pixi supports the following files:

- Textures (***avif***, ***webp***, ***png***, ***jpg***, ***gif***, ***svg***) via loadTextures, loadSvg
- Video Textures (***mp4***, ***m4v***, ***webm***, ***ogg***, ***ogv***, ***h264***, ***avi***, ***mov***) via loadVideoTextures
- Sprite sheets (***json***) via spritesheetAsset
- Bitmap fonts (***xml***, ***fnt***, ***txt***) via assets.loadBitmapFont
- Web fonts (***ttf***, ***woff***, ***woff2***) via loadWebFont
- JSON files (***json***) via loadJson
- Text Files (***txt***) via loadTxt

  More types can be added fairly easily by creating additional LoaderParsers.

Textures

- Textures are loaded as ImageBitmap on a worker thread where possible. Leading to much less janky load + parse times.
- By default, we will prefer to load AVIF and WebP image files if you specify them. But if the browser doesn't support AVIF or WebP we will fall back to png and jpg.
- Textures can also be accessed via `Texture.from()` (see Texture.from) and now use this asset manager under the hood!
- Don't worry if you set preferences for textures that don't exist (for example you prefer 2x resolutions images but only 1x is available for that texture, the Assets manager will pick that up as a fallback automatically)

Sprite sheets

- It's hard to know what resolution a sprite sheet is without loading it first, to address this there is a naming convention we have added that will let Pixi understand the image format and resolution of the spritesheet via its file name: `my-spritesheet{resolution}.{imageFormat}.json`

  For example:
  - `my-spritesheet@2x.webp.json` * // 2x resolution, WebP sprite sheet*
  - `my-spritesheet@0.5x.png.json` * // 0.5x resolution, png sprite sheet*
- This is optional! You can just load a sprite sheet as normal. This is only useful if you have a bunch of different res / formatted spritesheets.

Fonts

Web fonts will be loaded with all weights. It is possible to load only specific weights by doing the following:

```
import { Assets } from 'pixi.js';

// Load specific weights..
await Assets.load({
    data: {
        weights: ['normal'], // Only loads the weight
```

```
    },
    src: `outfit.woff2`,
});

// Load everything...
await Assets.load(`outfit.woff2`);
```

Background Loading

Background loading will load stuff for you passively behind the scenes. To minimize jank, it will only load one asset at a time. As soon as a developer calls `Assets.load(...)` the background loader is paused and requested assets are loaded as a priority. Don't worry if something is in there that's already loaded, it will just get skipped!

You still need to call `Assets.load(...)` to get an asset that has been loaded in the background. It's just that this promise will resolve instantly if the asset has already been loaded.

Manifest and Bundles

- Manifest is a descriptor that contains a list of all assets and their properties.
- Bundles are a way to group assets together.

```
import { Assets } from 'pixi.js';

// Manifest Example
const manifest = {
    bundles: [
        {
            name: 'load-screen',
            assets: [
                {
                    alias: 'background',
                    src: 'sunset.png',
                },
                {
                    alias: 'bar',
                    src: 'load-bar.{png,webp}',
                },
            ],
        },
        {
            name: 'game-screen',
            assets: [
                {
                    alias: 'character',
                    src: 'robot.png',
                },
                {
                    alias: 'enemy',
                    src: 'bad-guy.png',
                },
            ],
        },
    ]
};

await Assets.init({ manifest });

// Load a bundle...
loadScreenAssets = await Assets.loadBundle('load-screen');
// Load another bundle...
gameScreenAssets = await Assets.loadBundle('game-screen');
```

Classes

Assets

BackgroundLoader

Cache

Loader

Resolver

Spritesheet

Interface Definitions

AssetExtension

This developer convenience object allows developers to group together the various asset parsers into a single object.

Example

```
import { AssetExtension, extensions } from 'pixi.js';

// create the CacheParser
const cache = {
   test(asset: item): boolean {
      // Gets called by the cache when a dev caches an asset
   },
   getCacheableAssets(keys: string[], asset: item): Record<string, any> {
      // If the test passes, this function is called to get the cacheable assets
      // an example may be that a spritesheet object will return all the sub textures it has so they can
      // be cached.
   },
};

// create the ResolveURLParser
const resolver = {
   test(value: string): boolean {
      // the test to perform on the url to determine if it should be parsed
   },
   parse(value: string): ResolvedAsset {
      // the function that will convert the url into an object
   },
};

// create the LoaderParser
const loader = {
   name: 'itemLoader',
   extension: {
      type: ExtensionType.LoadParser,
   },
   async testParse(asset: any, options: ResolvedAsset) {
      // This function is used to test if the parse function should be run on the asset
   },
   async parse(asset: any, options: ResolvedAsset, loader: Loader) {
      // Gets called on the asset it testParse passes. Useful to convert a raw asset into something more
useful
   },
   unload(item: any) {
      // If an asset is parsed using this parser, the unload function will be called when the user
requests an asset
      // to be unloaded. This is useful for things like sounds or textures that can be unloaded from
memory
   },
};
```

```
// put it all together and create the AssetExtension
extensions.add({
    extension: ExtensionType.Asset,
    cache,
    resolver,
    loader,
}
```

AssetExtensionAdvanced

A more verbose version of the AssetExtension, allowing you to set the cached, loaded, parsed, and unloaded asset separately

Properties:

Name	Type	Description
`cache`	Partial<CACHE_ASSET<CacheParser>>	the asset cache parser
`detection`	Partial<FormatDetectionParser>	the asset format detection parser
`extension`	Asset	The type of extension
`loader`	LoaderParserAdvanced<META_DATA, ASSET, PARSED_ASSET, UNLOAD_ASSET>	the asset loader
`resolver`	Partial<ResolveURLParser>	the asset resolve parser

AssetInitOptions

Initialization options object for the Assets Class.

Properties:

Name	Type	Description
`basePath`	string	a base path for any assets loaded
`bundleIdentifier`	BundleIdentifierOptions	advanced - override how bundlesIds are generated
`defaultSearchParams`	string | Record<string, any>	a default URL parameter string to append to all assets loaded
`manifest`	string | AssetsManifest	a manifest to tell the asset loader upfront what all your assets are this can be the manifest object itself, or a URL to the manifest.
`preferences`	Partial<AssetsPreferences>	Optional loader preferences
`skipDetections`	boolean	If true, don't attempt to detect whether browser has preferred

		formats available. May result in increased performance as it skips detection step.
`texturePreference`	{ resolution?: number | number[], format?: ArrayOr<string> }	optional preferences for which textures preferences you have when resolving assets for example you might set the resolution to 0.5 if the user is on a rubbish old phone or you might set the resolution to 2 if the user is on a retina display

AssetsBundle

Structure of a bundle found in a Manifest file

Properties:

Name	Type	Description
`assets`	UnresolvedAsset[] | UnresolvedAsset<string, ArrayOr<string> | Record>	The assets in the bundle
`name`	string	The name of the bundle

AssetsManifest

The expected format of a manifest. This could be auto generated or hand made

Properties:

Name	Type	Description
`bundles`	AssetsBundle[]	array of bundles

AssetsPreferences

Extensible preferences that can be used, for instance, when configuring loaders.

Since:

- 7.2.0

BundleIdentifierOptions

Options for how the resolver deals with generating bundle ids

Properties:

Name	Type	Description
`connector`	string	The character that is used to connect the bundleId and the assetId when generating a bundle asset id key
`createBundleAssetId`	(bundleId: string, assetId: string) ⇒ string	A function that generates a bundle asset id key from a

		bundleId and an assetId
`extractAssetIdFromBundle`	(bundleId: string, assetBundleId: string) ⇒ string	A function that generates an assetId from a bundle asset id key. This is the reverse of generateBundleAssetId

CacheParser

For every asset that is cached, it will call the parsers test function the flow is as follows:

1. `cacheParser.test()` : Test the asset.
2. `cacheParser.getCacheableAssets()` : If the test passes call the getCacheableAssets function with the asset

Useful if you want to add more than just a raw asset to the cache (for example a spritesheet will want to make all its sub textures easily accessible in the cache)

Properties:

Name	Type	Description
`config`	Record<string, any>	A config to adjust the parser
`extension`	ExtensionMetadata	The extension type of this cache parser
`getCacheableAssets`	(keys: string[], asset: T) ⇒ Record<string, any>	If the test passes, this function is called to get the cacheable assets an example may be that a spritesheet object will return all the sub textures it has so they can be cached.
`test`	(asset: T) ⇒ boolean	Gets called by the cache when a dev caches an asset

FormatDetectionParser

Format detection is useful for detecting feature support on the current platform.

Properties:

Name	Type	Description
`add`	(formats: string[]) ⇒ Promise<string[]>	Add formats (file extensions) to the existing list of formats. Return an new array with added formats, do not mutate the formats argument.
`extension`	ExtensionMetadata	Should be ExtensionType.DetectionParser
`remove`	(formats: string[]) ⇒ Promise<string[]>	Remove formats (file extensions) from the list of supported formats. This is used when uninstalling this DetectionParser. Return an new array with filtered formats, do not mutate the formats argument.

`test`	() ⇒ Promise<boolean>	Browser/platform feature detection supported if return true

LoaderParser

The interface to define a loader parser *(all functions are optional)*.

When you create a `parser` object, the flow for every asset loaded is:

1. `parser.test()` - Each URL to load will be tested here, if the test is passed the assets are loaded using the load function below. Good place to test for things like file extensions!
2. `parser.load()` - This is the promise that loads the URL provided resolves with a loaded asset if returned by the parser.
3. `parser.testParse()` - This function is used to test if the parse function should be run on the asset If this returns true then parse is called with the asset
4. `parse.parse()` - Gets called on the asset it testParse passes. Useful to convert a raw asset into something more useful

Some loaders may only be used for parsing, some only for loading, and some for both!

LoadSVGConfig

Configuration for the loadSVG plugin.

Properties:

Name	Type	Default	Description
`crossOrigin`	HTMLImageElement["crossOrigin"]	`'anonymous'`	The crossOrigin value to use for loading the SVG as an image.
`parseAsGraphicsContext`	boolean	`false`	When set to `true`, loading and decoding images will happen with `new Image()`,

See:

- assets.loadSVG

LoadTextureConfig

Configuration for the loadTextures plugin.

Properties:

Name	Type	Default	Description
`crossOrigin`	HTMLImageElement["crossOrigin"]	`'anonymous'`	The crossOrigin value to use for images when `preferCreateImageBitmap` is `fals`
`preferCreateImageBitmap`	boolean	`true`	When set to `true`, loading and decoding images will happen with `createImageBitmap`, otherwise it will use `new Image()`.
`preferWorkers`	boolean	`true`	When set to `true`, loading and decoding images will happen with Worker thread, if available on the browser. This is much more performant as network requests and decodi can be expensive on the CPU. However, not environments support Workers, in some cas it can be helpful to disable by setting to `false`.

See:

- loadTextures

PreferOrder

A prefer order lets the resolver know which assets to prefer depending on the various parameters passed to it.

Properties:

Name	Type	Description
`priority`	string[]	the importance order of the params

PromiseAndParser

A promise and parser pair

Properties:

Name	Type	Description
`parser`	LoaderParser	the parser that is loading the asset
`promise`	Promise<any>	the promise that is loading the asset

ResolvedAsset

A fully resolved asset, with all the information needed to load it.

Properties:

Name	Type	Description
`alias`	string[]	Aliases associated with asset
`data`	T	Optional data
`format`	string	Format, usually the file extension
`loadParser`	LoadParserName	An override that will ensure that the asset is loaded with a specific parser
`src`	string	The URL or relative path to the asset

ResolveURLParser

Format for url parser, will test a string and if it pass will then parse it, turning it into an ResolvedAsset

Properties:

Name	Type	Description
`config`	Record<string, any>	A config to adjust the parser
`parse`	(value: string) ⇒ ResolvedAsset & { [key: string]: any }	the function that will convert the url into an object
`test`	(url: string) ⇒ boolean	the test to perform on the url to determine if it should be parsed

SpritesheetData

Atlas format.

Properties:

Name	Type	Description
`animations`	Dict<string[]>	The animations of the atlas.

`frames`	Dict<SpritesheetFrameData>	The frames of the atlas.

`meta`	{
	app?: string,
	format?: string,
	frameTags?: {
	from: number,
	name: string,
	to: number,
	direction: string
	}[],
	image?: string,
	layers?: {
	blendMode: string,
	name: string,
	opacity: number
	}[],
	scale: number | string,
	size?: {
	h: number,
	w: number
	},
	slices?: {
	color: string,
	name: string,
	keys: {
	frame: number,
	bounds: {
	x: number,
	y: number,
	w: number,
	h: number
	}
	}[]
	}[],
	related_multi_packs?: string[],
	version?: string
	}
		The meta data of the atlas.

SpritesheetFrameData

Represents the JSON data for a spritesheet atlas.

Properties:

Name	Type	Description
`anchor`	PointData	The anchor point of the texture.
`borders`	TextureBorders	The 9-slice borders of the texture.

`frame`	{
	x: number,
	y: number,
	w: number,
	h: number
	}
		The frame rectangle of the texture.

`rotated`	boolean	Whether the texture is rotated.

`sourceSize`	{
	w: number,
	h: number
	}
		The source size of the texture.

`spriteSourceSize`	{
	h?: number,
	w?: number,
	x: number,
	y: number
	}
		The sprite source size.

`trimmed`	boolean	Whether the texture is trimmed.

Members

cacheTextureArray readonly

Returns an object of textures from an array of textures to be cached

detectAvif readonly

Detects if the browser supports the AVIF image format.

detectDefaults readonly

Adds some default image formats to the detection parser

detectMp4 readonly

Detects if the browser supports the MP4 video format.

detectOgv readonly

Detects if the browser supports the OGV video format.

detectWebm readonly

Detects if the browser supports the WebM video format.

detectWebp readonly

Detects if the browser supports the WebP image format.

loadJson readonly

A simple loader plugin for loading json data

loadSvg readonly

A simple loader plugin for loading json data

loadTextures readonly

A simple plugin to load our textures! This makes use of imageBitmaps where available. We load the `ImageBitmap` on a different thread using workers if possible. We can then use the `ImageBitmap` as a source for a Pixi texture

You can customize the behavior of this loader by setting the `config` property. Which can be found here

```
// Set the config
import { loadTextures } from 'pixi.js';

loadTextures.config = {
   // If true we will use a worker to load the ImageBitmap
   preferWorkers: true,
   // If false we will use new Image() instead of createImageBitmap,
   // we'll also disable the use of workers as it requires createImageBitmap
   preferCreateImageBitmap: true,
   crossOrigin: 'anonymous',
};
```

loadTxt readonly

A simple loader plugin for loading text data

loadVideoTextures readonly

A simple plugin to load video textures.

You can pass VideoSource options to the loader via the .data property of the asset descriptor when using Asset.load().

```
// Set the data
const texture = await Assets.load({
    src: './assets/city.mp4',
    data: {
        preload: true,
        autoPlay: true,
    },
});
```

loadWebFont readonly

A loader plugin for handling web fonts

Example

```
import { Assets } from 'pixi.js';

Assets.load({
  alias: 'font',
  src: 'fonts/titan-one.woff',
  data: {
    family: 'Titan One',
    weights: ['normal', 'bold'],
  }
})
```

resolveJsonUrl readonly

A parser that will resolve a json urls resolution for spritesheets e.g. `assets/spritesheet@1x.json`

resolveTextureUrl readonly

A parser that will resolve a texture url

spritesheetAsset AssetExtension readonly

Asset extension for loading spritesheets

Example

```
import { Assets } from 'pixi.js';

Assets.load({
    alias: 'spritesheet',
    src: 'path/to/spritesheet.json',
    data: {
        ignoreMultiPack: true,
        textureOptions: {
            scaleMode: "nearest"
        }
    }
})
```

Type Definitions

AssetSrc

ts:53

A valid asset src. This can be a string, or a ResolvedSrc, or an array of either.

LoadFontData

ts:25

Data for loading a font

Properties:

Name	Type	Description

`display`	string	A set of optional descriptors passed as an object. It can contain any of the descriptors available for @font-face:
`family`	string	Font family name
`featureSettings`	string	The featureSettings property of the FontFace interface retrieves or sets infrequently used font features that are not available from a font's variant properties.
`stretch`	string	The stretch property of the FontFace interface retrieves or sets how the font stretches.
`style`	string	The style property of the FontFace interface retrieves or sets the font's style.
`unicodeRange`	string	The unicodeRange property of the FontFace interface retrieves or sets the range of unicode code points encompassing the font.
`variant`	string	The variant property of the FontFace interface programmatically retrieves or sets font variant values.
`weights`	string[]	The weight property of the FontFace interface retrieves or sets the weight of the font.

LoadParserName

ts:3

Names of the parsers that are built into PixiJS. Can be any of the following defaults:

- `loadJson`
- `loadSVG`

- `loadTextures`
- `loadTxt`
- `loadVideo`
- `loadWebFont` or a custom parser name.

ProgressCallback

ts:33

Callback for when progress on asset loading is made. The function is passed a single parameter, `progress`, which represents the percentage (0.0 - 1.0) of the assets loaded.

Name	Type	Description
`progress`	number	The percentage (0.0 - 1.0) of the assets loaded.

Example

```
(progress) => console.log(progress * 100 + '%')
```

ResolvedSrc

ts:42

A fully resolved src, Glob patterns will not work here, and the src will be resolved to a single file.

Properties:

Name	Type	Description
`data`	any	Optional data
`format`	string	Format, usually the file extension
`loadParser`	string	An override that will ensure that the asset is loaded with a specific parser
`src`	string	The URL or relative path to the asset

UnresolvedAsset

ts:60

An asset that has not been resolved yet.

Properties:

Name	Type	Description
`alias`	ArrayOr<string>	Aliases associated with asset
`src`	AssetSrc	The URL or relative path to the asset

Methods

crossOrigin (element, url, crossorigin)void

ts:18

Set cross origin based detecting the url and the crossorigin

Name	Type	Attributes	Description
`element`	HTMLImageElement | HTMLVideoElement		Element to apply crossOrigin
`url`	string		URL to check
`crossorigin`	boolean | string	<optional>	Cross origin value to use

getFontFamilyName (url)string

ts:60

Return font face name from a file name Ex.: 'fonts/titan-one.woff' turns into 'Titan One'

Name	Type	Description
`url`	string	File url

Returns:

Type	Description
string	