

【JS】API

Canvas

- 準備
- ▶ ☆ 1. HTML 内に canvas 要素を配置
 - ▶ ☆ 2. (CSS で canvas 要素に背景色を設定)
 - ▶ ☆ 3. JS に draw() 関数を設ける ※draw という名前じゃなくてもいいけれど
 - ▶ ☆ 4. 高解像度ディスプレイに対応させる

- 詳解
- ▶ ※ ピクセル単位でも px を書く必要はない。

キャンバスの情報

- ▶ キャンバスの幅、高さを取得
- ▶ キャンバスの幅、高さを変更
- ▶ 表示する際の大きさを変更

図形

- ▶ 四角形（塗りつぶしのみ）
- ▶ 四角形（枠線のみ）
- ▶ 塗りつぶしの色の変更
- ▶ 枠線の色を変更
- ▶ 枠線の太さ
- ▶ 枠線の角の状態を変える
- ▶ 線形グラデーション
- ▶ 円形グラデーション
- ▶ 影をつける
- ▶ 線を描く
- ▶ 線を閉じる
- ▶ 線を閉じて中を塗りつぶす
- ▶ 点線にする

【JS】API

Canvas

- 準備
- ▶ ☆ 1. HTML 内に canvas 要素を配置
 - ▶ ☆ 2. (CSS で canvas 要素に背景色を設定)
 - ▶ ☆ 3. JS に draw() 関数を設ける ※draw という名前じゃなくてもいいけれど
 - ▶ ☆ 4. 高解像度ディスプレイに対応させる

- 詳解
- ▶ ※ ピクセル単位でも px を書く必要はない。

キャンバスの情報

- ▶ キャンバスの幅、高さを取得 canvas.width、canvas.height
- ▶ キャンバスの幅、高さを変更 canvas.width = 数値;、.height = 数値;
- ▶ 表示する際の大きさを変更 canvas.style.width = '数値px';、.height = '数値px';

図形

- ▶ 四角形（塗りつぶしのみ） ctx.fillRect(x, y, width, height);
- ▶ 四角形（枠線のみ） ctx.strokeRect(x, y, width, height);
- ▶ 塗りつぶしの色の変更 ctx.fillStyle = '色';
- ▶ 枠線の色を変更 ctx.strokeStyle = '色';
- ▶ 枠線の太さ ctx.lineWidth = width;
- ▶ 枠線の角の状態を変える ctx.lineJoin = '状態'; ※丸くしたり斜めにしたり
- ▶ 線形グラデーション const g = ctx.createLinearGradient(x0, y0, x1, y1);
- ▶ 円形グラデーション const g = ctx.createRadialGradient(x0, y0, r0, x1, y1, r1);
- ▶ 影をつける ctx.shadowOffsetX, Y, .shadowBlur, .shadowColor
- ▶ 線を描く ctx.beginPath();, .moveTo(x, y);, .lineTo(x, y);, .stroke();
- ▶ 線を閉じる closePath();
- ▶ 線を閉じて中を塗りつぶす ctx.fill();
- ▶ 点線にする ctx.setLineDash([点線長さ, 間隙, . . .]);

▶ 線の端をかかではなく円く	
▶ 円弧	
▶ 扇形	
▶ 楕円	
▶ 四角形（パスをえがくだけ）	

テキスト

▶ テキスト	
▶ フォントを変える	
▶ テキスト全体の配置方法を変更	
▶ テキスト全体の最大幅を設定	
▶ テキストの外枠だけ	

画像

▶ ※ 必ずロードしてから	
▶ 画像	
▶ 画像をテクスチャとして使う	
▶ 画像のトリミング	

座標空間の変形

▶ 座標空間を拡大縮小	
▶ 座標空間を平行移動	
▶ 座標空間を回転移動	

描画設定の保存・復元

▶ 描画設定を保存	
▶ 描画設定を復元	

アニメーション

▶ ☆ アニメーション

■用例

▶ ☆ ローディングアイコン
▶ ☆ アナログ時計

▶ 線の端をかかではなく円く	ctx.lineCap = 'round';
▶ 円弧	ctx.arc(x, y, r, startRad, endRad); ※Rad=ラジアン
▶ 扇形	ctx.moveTo(円中心座標);, ctx.arc(諸引数);
▶ 楕円	ctx.ellipse(x, y, rx, ry, rotation, startRad, endRad);
▶ 四角形（パスをえがくだけ）	ctx.rect(x, y, width, height);

テキスト

▶ テキスト	ctx.fillText('文字列', x, y);
▶ フォントを変える	ctx.font = 'CSSのフォント指定';
▶ テキスト全体の配置方法を変更	ctx.textAline = '適値';, ctx.textBaseline = '適値';
▶ テキスト全体の最大幅を設定	ctx.fillText("文字列", x, y, 最大幅);
▶ テキストの外枠だけ	ctx.strokeText('文字列', x, y);

画像

▶ ※ 必ずロードしてから	
▶ 画像	ctx.drawImage(img要素, x, y); ※大きさ変更も
▶ 画像をテクスチャとして使う	const pattern = ctx.createPattern(img要素, 'repeat');
▶ 画像のトリミング	ctx.drawImage(img要素, sx, sy, sw, sh, dx, dy, dw, dh);

座標空間の変形

▶ 座標空間を拡大縮小	ctx.scale(x方向の割合, y方向の割合);
▶ 座標空間を平行移動	ctx.translate(x, y);
▶ 座標空間を回転移動	ctx.rotate(rotation);

描画設定の保存・復元

▶ 描画設定を保存	ctx.save();
▶ 描画設定を復元	ctx.restore();

アニメーション

▶ ☆ アニメーション

■用例

▶ ☆ ローディングアイコン
▶ ☆ アナログ時計

- ▶ ☆ 迷路
- ▶ ☆ 15パズル
- ▶ ☆ ピンポンゲーム

Intersection Observer

■用語

- ▶ ☆ target, root, Intersection Ratio

■主な用途

- ページがスクロールした際の画像やその他のコンテンツの遅延読み込み。
- 「無限スクロール」をするウェブサイトを実装し、スクロールに従って次々とコンテンツを読み込んで、ユーザーがページの切り替えをせずに済むようにすること。
- 広告費用を計算するための広告が表示されたかどうかのレポート。
- ユーザーが見るかどうかによって、タスクを実行するかどうか、アニメーションを処理するかを決定すること

■手順

- ▶ ☆ 1. IntersectionObserverインスタンスを作成する
- ▶ ☆ 2. そのインスタンスのobserveメソッドの第1引数に監視対象を渡す
- ▶ ☆ 3. optionsの中身を設定する
- ▶ ☆ 4. callback関数を設定する

■完成版

- ▶ ☆ 監視対象が1つの場合
- ▶ ☆ 監視対象が複数の場合

- ▶ ☆ 迷路
- ▶ ☆ 15パズル
- ▶ ☆ ピンポンゲーム

Intersection Observer

■用語

- ▶ ☆ target, root, Intersection Ratio

■主な用途

- ページがスクロールした際の画像やその他のコンテンツの遅延読み込み。
- 「無限スクロール」をするウェブサイトを実装し、スクロールに従って次々とコンテンツを読み込んで、ユーザーがページの切り替えをせずに済むようにすること。
- 広告費用を計算するための広告が表示されたかどうかのレポート。
- ユーザーが見るかどうかによって、タスクを実行するかどうか、アニメーションを処理するかを決定すること

■手順

- ▶ ☆ 1. IntersectionObserverインスタンスを作成する
- ▶ ☆ 2. そのインスタンスのobserveメソッドの第1引数に監視対象を渡す
- ▶ ☆ 3. optionsの中身を設定する
- ▶ ☆ 4. callback関数を設定する

■完成版

- ▶ ☆ 監視対象が1つの場合
- ▶ ☆ 監視対象が複数の場合