Curso Completo de JavaScript + Canvas HTML5

Desenho de Mapa Profissional — Versão didática e com exemplos

Autor: Itamar Rocha Chaves Junior

Este documento é um esboço ampliado e reestruturado do curso — objetivo: guiar você, passo a passo, na construção de um desenhador de mapa astral profissional usando HTML5 Canvas e JavaScript moderno (ES6+). Inclui explicações conceituais, trechos de código práticos, exemplos e exercícios para fixação.

Sumário

Público-alvo: desenvolvedores com conhecimento básico de JavaScript que queiram aprender programação gráfica aplicada a mapas astrais.

Pré-requisitos

- HTML e CSS básicos
- JavaScript (variáveis, funções, arrays, objetos, loops)
- Noções mínimas de trigonometria são úteis, mas explicamos o necessário

Estrutura proposta do projeto

```
index.html        # página que contém o <canvas>
chart.js          # coordenação do desenho (fluxo principal)
math.js           # utilitários trigonométricos e conversões
zodiac.js         # desenho dos signos e legendas
houses.js         # manejo de cúspides e desenho das casas
bodies.js         # modelagem e desenho de planetas/pontos
aspects.js        # cálculo e desenho de aspectos
data.js           # exemplo de dados (posições, cúspides)
```

MÓDULO 1 — FUNDAMENTOS DO CANVAS HTML5

1.1 O Canvas como plano cartesiano

O `<canvas>` é uma superfície bitmap. Tudo é desenhado via API 2D.

- Origem (0,0) no canto superior esquerdo
- Eixo X cresce para a direita; eixo Y cresce para baixo

Exemplo mínimo:

```
<canvas id="mapa" width="800" height="800"></canvas>
<script src="chart.js"></script>
```

```
const canvas = document.getElementById('mapa');
const ctx = canvas.getContext('2d');
const cx = canvas.width / 2;
const cy = canvas.height / 2;
const R = Math.min(cx, cy) - 20; // raio útil
```

1.2 Desenhando formas básicas

As primitivas principais: `arc`, `moveTo`, `lineTo`, `fillText`.

Exemplo: desenhando o círculo base do mapa

```
ctx.lineWidth = 2;
ctx.strokeStyle = '#222';
ctx.beginPath();
ctx.arc(cx, cy, R, 0, Math.PI * 2);
ctx.stroke();
```

MÓDULO 2 — MATEMÁTICA DO MAPA ASTRAL

2.1 Graus e radianos

Canvas usa radianos; astrologia usa graus. Converta sempre.

```
function degToRad(deg) {
  return deg * Math.PI / 180;
}

function radToDeg(rad) {
  return rad * 180 / Math.PI;
}
```

2.2 Referencial astrológico

No mapa astral convencional, 0° (Aries 0°) aparece no ponto esquerdo-horizontal (ascendente) quando desenhamos com Ascendente à esquerda. Para converter graus astrológicos em ângulo de canvas (radianos), aplicamos um deslocamento de -90° (ou +270°):

```
function astroAngle(deg) {
  return degToRad(deg - 90);
}
```

2.3 Conversão polar → cartesiana

Função central para posicionamento de elementos ao longo do círculo:

```
function polarToXY(cx, cy, r, deg) {
  const a = astroAngle(deg);
  return {
    x: cx + r * Math.cos(a),
    y: cy + r * Math.sin(a)
  };
}
```

Exemplo de uso:

```
const p = polarToXY(cx, cy, R - 40, 120.5); // posição para 120.5°
ctx.fillRect(p.x - 3, p.y - 3, 6, 6);
```

MÓDULO 3 — ZODÍACO

3.1 Conceitos fundamentais

- 12 signos, cada um com 30°
- Signo i começa em i*30° (0..11)

```
const SIGNS =
['Áries','Touro','Gêmeos','Câncer','Leão','Virgem','Libra','Escorpião','Sagitário'
,'Capricórnio','Aquário','Peixes'];
```

3.2 Desenhando os setores zodiacais

Usamos `ctx.arc` com ângulos iniciais e finais convertidos via `degToRad`.

```
SIGNS.forEach((s, i) => {
  const start = degToRad(i * 30 - 90);
  const end = degToRad((i + 1) * 30 - 90);
```

```
  ctx.beginPath();
  ctx.moveTo(cx, cy);
  ctx.arc(cx, cy, R, start, end);
  ctx.closePath();
  ctx.strokeStyle = '#ccc';
  ctx.stroke();

  // label do signo (centro do setor)
  const midDeg = i * 30 + 15;
  const labelPos = polarToXY(cx, cy, R - 24, midDeg);
  ctx.font = '14px sans-serif';
  ctx.fillStyle = '#000';
  ctx.textAlign = 'center';
  ctx.fillText(s, labelPos.x, labelPos.y + 4);
});
```

Exercício: altere `R` e veja como o texto se reposiciona. Tente mudar `textAlign` para experimentar alinhamentos.

MÓDULO 4 — CASAS ASTROLÓGICAS (CÚSPIDES)

4.1 Dados de cúspides

As cúspides são ângulos (em graus) determinando as divisões das casas. Não são uniformes: dependem de latitude, hora e sistema (Placidus, Koch, Equal, etc.). Em produção, obtenha dados de uma ephemeris.

Exemplo de array de cúspides (dados reais de exemplo):

```
const CUSPS = [82.70154304778679, 111.1341504020764, 141.3461512349346,
173.6225546549408, 205.73919610408942, 235.36167285257102, 262.7015430477868,
291.1341504020764, 321.3461512349346, 353.62255465494087, 25.73919610408942,
55.361672852571004];
```

4.2 Desenho das linhas das casas

Trace linhas do centro até o perímetro em cada cúspide:

```
CUSPS.forEach((deg, i) => {
  const p = polarToXY(cx, cy, R, deg);
  ctx.beginPath();
  ctx.moveTo(cx, cy);
  ctx.lineTo(p.x, p.y);
  ctx.strokeStyle = '#888';
  ctx.stroke();

  // rótulo da cúspide perto da borda
  const label = (i + 1).toString().padStart(2, '0');
  const labPos = polarToXY(cx, cy, R - 8, deg);
  ctx.fillStyle = '#222';
```

```
  ctx.fillText(label, labPos.x, labPos.y + 4);
});
```

Exercício: desenhe as casas preenchendo setores entre cúspides com transparência.

MÓDULO 5 — PLANETAS E PONTOS

5.1 Modelagem dos corpos

Estrutura sugerida para um corpo:

```
// exemplo simplificado
const BODIES = [
  { code: 'sol', name: 'Sol', lon: 97.66258, sign: 'Câncer' },
  { code: 'lua', name: 'Lua', lon: 307.45898, sign: 'Aquário' },
  // ... outros
];
```

5.2 Desenhando planetas no raio interior

Coloque os planetas num círculo interior (por ex. R - 60) e desenhe um marcador e o nome.

```
BODIES.forEach(b => {
  const p = polarToXY(cx, cy, R - 60, b.lon);
  ctx.beginPath();
  ctx.fillStyle = '#000';
  ctx.arc(p.x, p.y, 6, 0, Math.PI * 2);
  ctx.fill();

  ctx.font = '12px sans-serif';
  ctx.textAlign = 'left';
  ctx.fillText(b.name, p.x + 8, p.y + 4);
});
```

Dica: para evitar sobreposição de rótulos, calcule deslocamentos pequenos (y offsets) quando distâncias entre marcadores forem menores que um limiar.

MÓDULO 6 — ASPECTOS ASTROLÓGICOS

6.1 Diferença angular (módulo circular)

Para comparar ângulos, use uma diferença mínima ao redor do círculo:

```
function angleDiff(a, b) {
  const d = Math.abs((a - b + 360) % 360);
  return d > 180 ? 360 - d : d;
}
```

6.2 Identificando aspectos com orbes

Defina aspectos e orbes (tolerâncias):

```
const ASPECTS = [
  { name: 'Conjunção', angle: 0, orbe: 8, color: '#444' },
  { name: 'Sextil', angle: 60, orbe: 6, color: '#2E86AB' },
  { name: 'Quadratura', angle: 90, orbe: 6, color: '#C0392B' },
  { name: 'Trígono', angle: 120, orbe: 6, color: '#27AE60' },
  { name: 'Oposição', angle: 180, orbe: 8, color: '#8E44AD' }
];

function getAspect(a, b) {
  const d = angleDiff(a, b);
  return ASPECTS.find(as => Math.abs(d - as.angle) <= as.orbe) || null;
}
```

6.3 Desenho das linhas de aspecto

Desenhe linhas entre os pontos dos planetas, colorindo conforme o aspecto.

```
for (let i = 0; i < BODIES.length; i++) {
  for (let j = i + 1; j < BODIES.length; j++) {
    const a = BODIES[i].lon, b = BODIES[j].lon;
    const asp = getAspect(a, b);
    if (!asp) continue;

    const p1 = polarToXY(cx, cy, R - 60, a);
    const p2 = polarToXY(cx, cy, R - 60, b);
    ctx.beginPath();
    ctx.moveTo(p1.x, p1.y);
    ctx.lineTo(p2.x, p2.y);
    ctx.strokeStyle = asp.color;
    ctx.lineWidth = 1.5;
    ctx.stroke();
  }
}
```

Exercício: desenhe um pequeno rótulo no meio da linha indicando o tipo de aspecto.

MÓDULO 7 — ORGANIZAÇÃO E CAMADAS

Ordem recomendada de desenho (do fundo para a frente):

1. Fundo e gradientes
2. Setores zodiacais
3. Círculo externo e marcações de graus

4. Linhas das casas (cúspides)
5. Aspectos (linhas entre planetas)
6. Planetas / pontos (marcadores)
7. Textos, legendas e interatividade (tooltips)

Manter essa ordem evita sobreposição indesejada e melhora clareza.

Boas práticas e performance

- Agrupe operações de desenho semelhantes (mesma cor/estilo) para reduzir trocas de estado do contexto.
- Use camadas: desenhe em canvas offscreen para partes estáticas (fundo, zodiac, casas) e redesenhe apenas as camadas dinâmicas (aspectos, planetas) ao interagir.
- Para grandes resoluções, aumente `canvas.width/height` proporcionalmente ao `devicePixelRatio` e ajuste `ctx.scale`.

Exportação

- Para salvar como PNG: `canvas.toDataURL('image/png')` e criar link de download.
- Para PDF: renderize o canvas em uma imagem e insira em um PDF via bibliotecas como `jsPDF`.

Exemplo simples de download PNG:

```
function downloadPNG(canvas, filename = 'mapa.png') {
  const url = canvas.toDataURL('image/png');
  const a = document.createElement('a');
  a.href = url;
  a.download = filename;
  a.click();
}
```

Exercícios práticos

1. Modifique `BODIES` para adicionar pontos fictícios (ex: Part of Fortune) e desenhe-os com símbolo diferente.
2. Adicione hover: ao passar o mouse sobre um planeta, destaque o aspecto e mostre uma tooltip com detalhes.
3. Implemente label collision: realoque rótulos próximos para evitar sobreposição.

Dados de exemplo (bloco usado no esboço original)

O arquivo original inclui um bloco de saída com posições reais. Você pode transformar esse texto em `data.js` parseando linhas e extraindo `lon` e `cusps`.

Próximos passos e recursos

- Integrar Swiss Ephemeris (WASM) para cálculos precisos
- Adicionar sistema de casas alternativos (Placidus, Koch, Equal)
- Gerar trânsitos e animações temporais

Referências

- Documentação Canvas 2D (MDN)
- Swiss Ephemeris
- Artigos sobre visualização de dados circulares (radar charts)

Se quiser, eu posso:

- Gerar uma implementação completa mínima (HTML + JS) copiável e testável
- Criar `data.js` com parsing do bloco de exemplo
- Implementar a lógica de tooltip e resolução de rótulos

Diga qual desses próximos passos prefere que eu implemente primeiro.