
Earcut

The fastest and smallest JavaScript polygon triangulation library. 3KB gzipped.



The algorithm The library implements a modified ear slicing algorithm, optimized by z-order curve hashing and extended to handle holes, twisted polygons, degeneracies and self-intersections in a way that doesn't *guarantee* correctness of triangulation, but attempts to always produce acceptable results for practical data.

It's based on ideas from FIST: Fast Industrial-Strength Triangulation of Polygons by Martin Held and Triangulation by Ear Clipping by David Eberly.

Why another triangulation library? The aim of this project is to create a JS triangulation library that is **fast enough for real-time triangulation in the browser**, sacrificing triangulation quality for raw speed and simplicity, while being robust enough to handle most practical datasets without crashing or producing garbage. Some benchmarks using Node 0.12:

(ops/sec)	pts	earcut	libtess	poly2tri	pnlntri	polyk
OSM building	15	795,935	50,640	61,501	122,966	175,570
dude shape	94	35,658	10,339	8,784	11,172	13,557
holed dude shape	104	28,319	8,883	7,494	2,130	n/a
complex OSM water	2523	543	77.54	failure	failure	n/a
huge OSM water	5667	95	29.30	failure	failure	n/a

The original use case it was created for is Mapbox GL, WebGL-based interactive maps.

If you want to get correct triangulation even on very bad data with lots of self-intersections and earcut is not precise enough, take a look at libtess.js.

Usage

```
1 var triangles = earcut([10,0, 0,50, 60,60, 70,10]); // returns [1,0,3,3,2,1]
```

Signature: `earcut(vertices[, holes, dimensions = 2])`.

- `vertices` is a flat array of vertex coordinates like `[x0,y0, x1,y1, x2,y2, ...]`.

-
- `holes` is an array of hole *indices* if any (e.g. `[5, 8]` for a 12-vertex input would mean one hole with vertices 5–7 and another with 8–11).
 - `dimensions` is the number of coordinates per vertex in the input array (2 by default). Only two are used for triangulation (`x` and `y`), and the rest are ignored.

Each group of three vertex indices in the resulting array forms a triangle.

```
1 // triangulating a polygon with a hole
2 earcut([0,0, 100,0, 100,100, 0,100, 20,20, 80,20, 80,80, 20,80], [4]);
3 // [3,0,4, 5,4,0, 3,4,7, 5,0,1, 2,3,7, 6,5,1, 2,7,6, 6,1,2]
4
5 // triangulating a polygon with 3d coords
6 earcut([10,0,1, 0,50,2, 60,60,3, 70,10,4], null, 3);
7 // [1,0,3, 3,2,1]
```

If you pass a single vertex as a hole, Earcut treats it as a Steiner point.

Note that Earcut is a **2D** triangulation algorithm, and handles 3D data as if it was projected onto the XY plane (with Z component ignored).

If your input is a multi-dimensional array (e.g. GeoJSON Polygon), you can convert it to the format expected by Earcut with `earcut.flatten`:

```
1 var data = earcut.flatten(geojson.geometry.coordinates);
2 var triangles = earcut(data.vertices, data.holes, data.dimensions);
```

After getting a triangulation, you can verify its correctness with `earcut.deviation`:

```
1 var deviation = earcut.deviation(vertices, holes, dimensions, triangles
  );
```

Returns the relative difference between the total area of triangles and the area of the input polygon. 0 means the triangulation is fully correct.

Install NPM and Browserify:

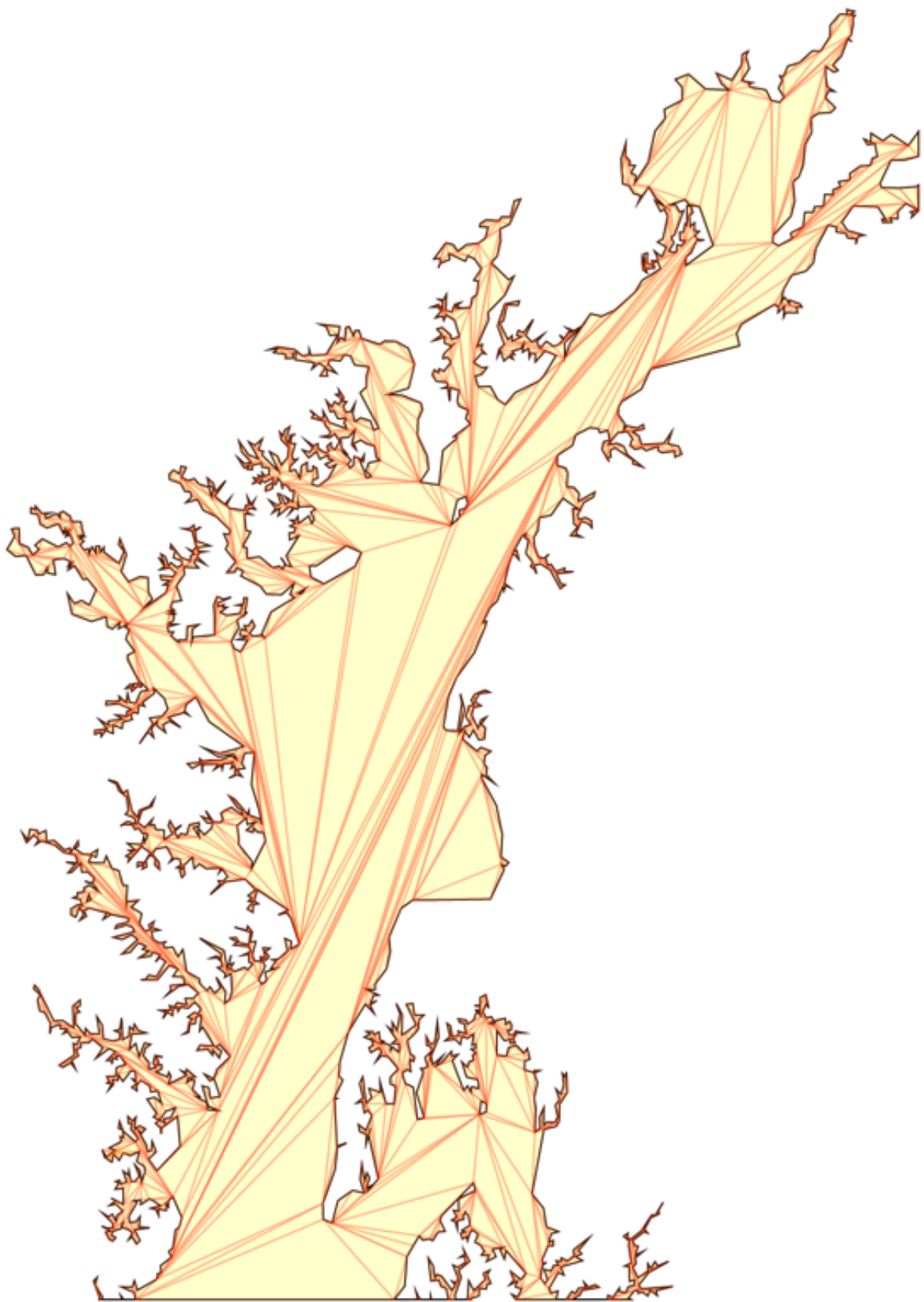
```
1 npm install earcut
```

Browser builds on CDN:

- development build
- minified production build

Running tests:

```
1 npm test
```



Ports to other languages

- mapbox/earcut.hpp (C++11)
- earcut4j/earcut4j (Java)
- the3deers/earcut-java (Java)
- Larpon/earcut (V)
- Cawfree/earcut-j (Java, outdated)

Changelog

2.2.4 (Jul 5, 2022)

- Improved performance by 10–15%.
- Fixed another rare race condition that could lead to an infinite loop.

2.2.3 (Jul 8, 2021)

- Fixed a rare race condition that could lead to an infinite loop.

2.2.2 (Jan 21, 2020)

- Fixed yet another rare race condition when a hole shared an edge with an outer ring.

2.2.1 (Sep 19, 2019)

- Fixed another rare case with touching holes.

2.2.0 (Sep 18, 2019)

- Fixed a handful of rare race conditions.

2.1.5 (Feb 5, 2019)

- Fixed a race condition with coincident holes that could lead to bad triangulation.

2.1.4 (Dec 4, 2018)

- Fixed a race condition that could lead to a freeze on degenerate input.

2.1.3 (Jan 4, 2018)

- Improved performance for bigger inputs (5-12%).

2.1.2 (Oct 23, 2017)

- Fixed a few race conditions where bad input would cause an error.

2.1.1 (Mar 17, 2016)

- Fixed a rare race condition where the split routine would choose bad diagonals.
- Fixed a rare race condition in the “cure local intersections” routine.
- Fixed a rare race condition where a hole that shares a point with the outer ring would be handled incorrectly.
- Fixed a bug where a closing point wouldn’t be filtered as duplicate, sometimes breaking triangulation.

2.1.0 (Mar 11, 2016)

- Added `earcut.deviation` function for verifying correctness of triangulation.
- Added `earcut.flatten` function for converting GeoJSON-like input into a format Earcut expects.

2.0.9 (Mar 10, 2016)

- Fixed a rare race condition where a hole would be handled incorrectly.

2.0.8 (Jan 19, 2016)

- Fixed a rare race condition with a hole touching outer ring.

2.0.7 (Nov 18, 2015)

- Changed the algorithm to avoid filtering colinear/duplicate vertices unless it can’t triangulate the polygon otherwise. Improves performance on simpler shapes and fixes some 3D use cases.

2.0.6 (Oct 26, 2015)

- Improved robustness and reliability of the triangulation algorithm.
- Improved performance by up to 15%.
- Significantly improved source code clarity.

2.0.5 (Oct 12, 2015)

- Fixed a z-curve hashing bug that could lead to unexpected results in very rare cases involving shapes with lots of points.

2.0.4 (Oct 8, 2015)

- Fixed one more extremely rare race condition, lol!

2.0.3 (Oct 8, 2015)

- Fixed yet another rare race condition (multiple holes connected with colinear bridges).
- Fixed crash on empty input.

2.0.2 (Jul 8, 2015)

- Fixed one more rare race condition with a holed polygon.

2.0.1 (May 11, 2015)

- Added Steiner points support.

2.0.0 (Apr 30, 2015)

- **Breaking:** changed the API to accept a flat input array of vertices with hole indices and return triangle indices. It makes the indexed output much faster than it was before (up to 30%) and improves memory footprint.

1.4.2 (Mar 18, 2015)

- Fixed another rare edge case with a tiny hole in a huge polygon.

1.4.1 (Mar 17, 2015)

- Fixed a rare edge case that led to incomplete triangulation.

1.4.0 (Mar 9, 2015)

- Fixed indexed output to produce indices not multiplied by dimension and work with any number of dimensions.

1.3.0 (Feb 24, 2015)

- Added a second argument to `earcut` that switches output format to flat vertex and index arrays if set to `true`.

1.2.3 (Feb 10, 2015)

- Improved performance (especially on recent v8) by avoiding `Array push` with multiple arguments.

1.2.2 (Jan 27, 2015)

- Significantly improved performance for polygons with self-intersections (e.g. big OSM water polygons are now handled 2-3x faster)

1.2.1 (Jan 26, 2015)

- Significantly improved performance on polygons with high number of vertices by using z-order curve hashing for vertex lookup.
- Slightly improved overall performance with better point filtering.

1.1.0 (Jan 21, 2015)

- Improved performance on polygons with holes by switching from Held to Eberly hole elimination algorithm
- More robustness fixes and tests

1.0.1 — 1.0.6 (Jan 20, 2015)

- Various robustness improvements and fixes.

1.0.0 (Jan 18, 2015)

- Initial release.