sportsdataverse :: CHEAT SHEET

The SportsDataverse's Node.js / TypeScript client (npm `sportsdataverse`) — sister to sportsdataverse-py and the R packages (hoopR, wehoop, cfbfastR, fastRhockey, baseballr). A cross-league ESPN client (116 endpoints × 29 leagues) plus 532 native / provider live-API wrappers, with a tidy `{ parsed: true }` layer. Page 1 of 4: install, namespaces, parsers, return shapes.

v3.1.0

Node ≥ 20.18.1 · ESM · TypeScript
js.sportsdataverse.org

GET STARTED

```
npm install sportsdataverse
```

ESM-only (`"type": "module"`) — there is no CommonJS entry point. Ships `dist/` with full `.d.ts` typings.

```
import sdv from "sportsdataverse";       // ESM
// from CommonJS: dynamic import only
const { default: sdv } =
  await import("sportsdataverse");
```

QUICK START

```
const board = await sdv.nba
  .espnNbaScoreboard({});
```

```
const game = await sdv.nba
  .espnNbaSummary({ event_id: 401584793 });
```

```
const roster = await sdv.nfl
  .espnNflTeamRoster({ team_id: 12 });
// legacy method, still works:
const pbp = await sdv.nba
  .getPlayByPlay(401584793);
```

Every wrapper is `async` and takes one params object; snake_case and camelCase keys are interchangeable: `{ team_id: 12 }` ≡ `{ teamId: 12 }`.

RUNTIME NOTES

- HTTP via `axios`; every wrapper returns a `Promise` — use `await` or `.then()`.
- Node ≥ 20.18.1. In the browser, import only `sportsdataverse/parsers` (no node HTTP deps); the docs playground proxies live calls.
- Tests are no-network; codegen-driven (`npm run codegen` regenerates wrappers + docs from endpoint YAML).

DUAL-CASE NAMES

Generated wrappers register under BOTH `espn_nfl_team_roster` (snake — py/R parity) and `espnNflTeamRoster` (idiomatic JS). Same function.

NAMESPACE LAYOUT

Namespace	Contents
`sdv.nba .wnba .mbb .wbb .cfb .nfl .mlb .nhl`	legacy `get*` methods (page 2) + generated `espn_*` wrappers
`sdv.mch .wch .college_baseball .college_softball .ufl .xfl .cfl`	ESPN-only leagues
`sdv.soccer` + `epl laliga bundesliga seriea ligue1 mls ligamx ucl uel nwsl wwc wc`	league-param catch-all + 12 curated aliases
`sdv.cricket`	league-param ESPN cricket
`sdv.nhl .mlb .nfl`	+ native APIs: `nhlApiWeb* nhlEdge* nhlStatsRest* nhlRecords* mlb* mlbStatcast* nflApi*`
`sdv.odds .recruiting .cbs .fox .yahoo .hockeytech .torvik`	7 standalone provider namespaces (page 4)
`sdv.ncaa` · `sdv.tennis`	legacy-only services (page 2)

29 ESPN leagues + providers = 38 namespaces on the default export.

MULTI-LEAGUE SPORTS

```
await sdv.soccer.espnSoccerScoreboard(
  { league: "eng.1" });   // Premier League
await sdv.soccer.espnSoccerScoreboard(
  { league: "esp.1" });   // La Liga
```

`soccer` / `cricket` take an extra `league` slug; the curated aliases (`sdv.epl`, …) pin it for you.

TIDY ROWS: { PARSED: TRUE }

Every wrapper returns the **raw** JSON payload by default. Pass `{ parsed: true }` to run the registered parser → an array of flat, snake_cased row objects (the JS analog of py's `return_parsed=True`). Empty / malformed → `[]`, never throws.

```
const raw  = await sdv.nba
  .espnNbaScoreboard({});
const rows = await sdv.nba
  .espnNbaScoreboard({ parsed: true });
```

Summary dispatcher (21 sections)

```
await sdv.nba.espnNbaSummary({
  event_id: 401584793, parsed: true,
  section: "boxscore_player" });
```

Sections: `boxscore_player boxscore_team plays winprobability leaders game_info officials header season_series against_the_spread standings broadcasts format pickcenter odds article injuries news drives drive_plays scoring_plays`. Omit `section` → all 21 as a dict.

BROWSER-SAFE PARSERS

```
import { parseEndpoint }
  from "sportsdataverse/parsers";
```

The `/parsers` subpath export has no node-only HTTP deps — it powers the in-browser docs playground. Also exported: `normalize`, `snakeCase`, `PARSERS`, `parserFor`.

RETURN SHAPES

Call	Returns
default	raw provider JSON (ESPN subset for legacy `get*`)
`{ parsed: true }`	`rows[]` — flat snake_cased objects
`summary` + `section`	one sub-frame; omit → dict of 21
empty / error body	`[]` when parsed — chain without null checks

PIPE WITH TIDY.JS

```
import { tidy } from "sportsdataverse";
tidy.tidy(rows, tidy.groupBy("team",
  tidy.summarize({ n: tidy.n() })));
```

`@tidyjs/tidy` is re-exported — grammar-of-data verbs over parsed rows.

ADVANCED EXPORTS

`LEAGUES WRAPPERS FLAT_WRAPPERS makeLeagueModule makeFlatModule resolveFlat FLAT_HOSTS normalize PARSERS parserFor` + NFL auth: `nflTokenGen nflHeadersGen nflClearTokenCache jwtExp NFL_API_HOST`. Types: `LeagueConfig WrapperDef WrapperFn ParserFn`.

ECOSYSTEM

Python mirror: sportsdataverse-py — same `espn_<league>_<short>` names. R siblings: hoopR wehoop cfbfastR fastRhockey baseballr. Docs + live playground: **js.sportsdataverse.org**.

8 LEAGUE SERVICES · COMMON SHAPE

GAME ENDPOINTS

On `sdv.cfb .mbb .mlb .nba .nfl .nhl .wbb .wnba` — all take an ESPN game `id`:

Function	Returns / notes
`getPlayByPlay(id)`	teams, plays, competitions, season, boxScore subset
`getBoxScore(id)`	gamepackage boxscore (+ `id`)
`getSummary(id)`	full ESPN summary subset
`getPicks(id)`	pickcenter; cfb mbb mlb nba nfl nhl only

```
const pbp = await sdv.wnba
  .getPlayByPlay(401244185);
const box = await sdv.cfb
  .getBoxScore(401628334);
```

Source: `cdn.espn.com/core/<sport>/…` game-package feeds. `getPlayByPlay` returns `{teams, id, plays, competitions, season, boxScore, seasonSeries, standings}`.

TEAMS

Function	Args
`getTeamList()`	colleges: `{group}` (cfb 80, mbb/wbb 50)
`getTeamInfo(id)`	nfl: `{id}` object
`getTeamPlayers(id)`	roster; nfl: `{id}`

```
await sdv.mbb.getTeamList({ group: 50 });
await sdv.nba.getTeamInfo(13);
await sdv.nfl.getTeamPlayers({ id: 12 });
```

Default groups: 80 = FBS (cfb) · 50 = NCAA D1 (mbb / wbb).

SCHEDULES & STANDINGS

Function	Args
`getSchedule`	`{year, month, day}` + colleges: `groups, seasontype`
`getScoreboard`	same + `limit` (300–1000)
`getStandings`	`{year}` (defaults to now)
`getConferences`	`{year}`; cfb mbb wbb only
`getRankings`	`{year, week}`; cfb only
`getWeeklySchedule`	`{week, year, seasonType}`; nfl only

```
await sdv.cfb.getSchedule(
  { year: 2024, month: 9, day: 7,
    groups: 80, seasontype: 2 });
await sdv.nfl.getWeeklySchedule(
  { week: 1, year: 2024, seasonType: 2 });
await sdv.mbb.getStandings({ year: 2024 });
await sdv.wbb.getConferences({ year: 2024 });
await sdv.cfb.getRankings(
  { year: 2024, week: 5 });
```

AVAILABILITY MATRIX

Namespace	Method count
`sdv.cfb`	15 (+ rankings, recruiting)
`sdv.mbb`	14 (+ recruiting)
`sdv.nfl`	11 (+ weekly schedule)
`sdv.mlb sdv.nba sdv.nhl`	10 each
`sdv.wbb`	10 (conferences, no picks)
`sdv.wnba`	9 (no picks / conferences)

RECRUITING SCRAPERS · CFB + MBB

247SPORTS (LEGACY SCRAPE)

Function	Args
`getPlayerRankings`	`{year, page, group, position, state}` (+ cfb `rankingsType`)
`getSchoolRankings`	`(year, page = 1)`
`getSchoolCommits`	`(school, year)`

```
await sdv.cfb.getPlayerRankings(
  { year: 2025, position: "QB" });
await sdv.mbb.getSchoolCommits(
  "Duke", 2025);
```

HTML scrape via cheerio. Superseded by the `sdv.recruiting.*` 247Sports RDB API family (page 4) — prefer it for new code.

TENNIS

```
await sdv.tennis.getScoreboard(
  { league: "atp", year: 2024,
    month: 7, day: 14 });
```

`league`: `"atp"` or `"wta"` (ESPN site v2 scoreboard).

MODERN REPLACEMENT

```
await sdv.recruiting.recruitingRankings(
  { year: 2025, headers: { ... } });
```

The 247Sports RDB API family (page 4) — 25 endpoints vs the 3 legacy scrapers; needs a caller-supplied JWT.

NCAA.COM · SDV.NCAA

GAMES (DATA.NCAA.COM)

Function	Purpose
`getInfo(game)`	gameInfo.json
`getBoxScore(game)`	boxscore.json
`getPlayByPlay(game)`	pbp.json
`getScoreboard({...})`	`{sport, division, year, month, day}`
`getRedirectUrl(url)`	resolve an ncaa.com game URL → casablanca id

```
const sb = await sdv.ncaa.getScoreboard({
  sport: "basketball-men", division: "d1",
  year: 2024, month: 3, day: 21 });
const url = sb.games[0].game.url;
const gid = await
sdv.ncaa.getRedirectUrl(url);
const pbp = await sdv.ncaa.getPlayByPlay(gid);
```

STATS.NCAA.ORG RANKINGS

Function	Args
`getSports()`	—
`getSeasons(sport)`	sport code
`getDivisions(sport, season)`	—
`getSportDivisionData`	`(sport, season, division, type, gameHigh)`
`getPlayerData`	`(sport, season, division, rankingPeriod, gameHigh, category)`
`getTeamData`	same args as player

Covers every NCAA sport code (wvb, mih, wla, wsb, …) — see the npm keywords for the full list.

ESPN cross-league :: 116 endpoints × 29 leagues

Name parity with sportsdataverse-py's `espn_*` wrappers · keyless, no auth

PATTERN & SCOPES

One core, parameterized on ESPN `(sport, league)` slugs. Each league exposes `espn_<prefix>_<short>` / `espn<Prefix><Short>` for every short name in its scope set:

Scope	Leagues	Shorts
`universal`	all 29	110
`ncaa`	college leagues	+3
`football`	nfl · cfb	+2
`mlb`	mlb	+1

ncaa: `rankings season_recruits season_week_rankings` · football: `season_qbr season_qbr_week` · mlb: `athlete_hotzones`.

```
await sdv.cfb.espnCfbRankings({});
await sdv.wnba.espnWnbaStandings(
  { season: 2024 });
await sdv.nfl.espnNflScoreboard(
  { week: 1, season_type: 2 });
```

ESPN-ONLY LEAGUES

```
await sdv.mch.espnMchScoreboard({});
await sdv.college_softball
  .espnCollegeSoftballScoreboard({});
await sdv.ufl.espnUflStandings({});
```

No legacy service — the generated surface is the whole namespace.

LEAGUES (LEAGUES.YAML)

Sport	Prefixes
basketball	`nba wnba mbb wbb`
football	`nfl cfb ufl xfl cfl`
baseball	`mlb college_baseball college_softball`
hockey	`nhl mch wch`
soccer	`soccer` + 12 aliases
cricket	`cricket`

SITE V2 · 25 SHORTS

SCORES, TEAMS, NEWS

Group	Short names
League	`scoreboard summary calendar news injuries transactions conferences statistics_league draft standings rankings`
Teams	`teams_site team team_roster team_schedule team_record team_depthcharts team_injuries team_transactions team_history team_news team_leaders`
Athletes	`athlete_info athlete_bio athlete_news`

WEB V3 · 5 SHORTS

ATHLETE STATS

`athlete_overview athlete_stats athlete_gamelog athlete_splits leaders`

```
await sdv.nba.espnNbaAthleteGamelog(
  { athlete_id: 1966, season: 2024 });
await sdv.mbb.espnMbbLeaders({});
await sdv.wnba.espnWnbaAthleteSplits(
  { athlete_id: 4433403 });
```

SUMMARY SECTIONS

```
await sdv.cfb.espnCfbSummary({
  event_id: 401628334, parsed: true,
  section: "drive_plays" });
```

`drives drive_plays scoring_plays` populate for NFL / CFB; other sports get zero-row frames.

CORE V2 · 86 SHORTS (1/2)

SEASONS & LEAGUE

Group	Short names
League	`league_root season_pointer seasons season_info season_types season_type standings_core leaders_core league_notes talentpicks tournaments positions position awards award`
Season	`season_group season_groups season_group_teams season_group_children season_type_leaders season_type_corrections season_weeks season_week season_week_events season_teams season_team season_athletes season_coaches season_draft season_draft_round_picks season_futures season_freeagents season_powerindex season_powerindex_leaders season_awards season_recruits season_week_rankings season_qbr season_qbr_week`
Teams & venues	`teams_core team_core venues venue franchises franchise coach coach_record coach_season`

CORE V2 · 86 SHORTS (2/2)

EVENTS & ATHLETES

Group	Short names
Events	`events event event_competition event_competitors event_competitor event_competitor_roster event_competitor_linescores event_competitor_statistics event_competitor_record event_competitor_leaders event_odds event_probabilities event_plays event_play event_play_personnel event_situation event_status event_officials event_official_detail event_broadcasts event_predictor event_powerindex event_propbets event_leaders event_scoringplays`
Athletes	`athletes_index athlete_core athlete_career_stats athlete_statisticslog athlete_eventlog athlete_contracts athlete_awards athlete_seasons athlete_records athlete_injuries athlete_notes athlete_vs_athlete athlete_hotzones`

```
await sdv.nhl.espnNhlEventOdds(
  { event_id: 401559593 });
await sdv.nba.espnNbaAthleteVsAthlete(
  { athlete_id: 1966, opp_id: 110 });
```

Core v2 payloads are `$ref`-linked; the generic parsers (`parse_items`, single-entity) tidy list + resource shapes. ESPN Core v2 403s under aggressive parallel load — keep concurrency low.

Native APIs & providers :: 532 flat wrappers · 15 families

All support `{ parsed: true }` · py siblings: **mlb_statcast_* nhl_web_* nfl_api_* hockeytech**

NATIVE · ON THE LEAGUE NAMESPACE

7 NATIVE FAMILIES

Wrappers	n	Host
`sdv.mlb.mlb*`	78	statsapi.mlb.com
`sdv.mlb.mlbStatcast*`	39+4	baseballsavant.mlb.com
`sdv.nhl.nhlApiWeb*`	27	api-web.nhle.com
`sdv.nhl.nhlEdge*`	35	…/v1/edge (tracking)
`sdv.nhl.nhlStatsRest*`	21	api.nhle.com/stats/rest
`sdv.nhl.nhlRecords*`	44	records.nhl.com
`sdv.nfl.nflApi*`	11	api.nfl.com

```
await sdv.mlb.mlbSchedule({ sport_id: 1,
  date: "2024-07-04", parsed: true });
await sdv.nhl.nhlApiWebPbp(
  { game_id: 2023030417, parsed: true });
await sdv.nfl.nflApiWeeklyGameDetails(
  { season: 2024, week: 1, parsed: true });
```

NFL.com auth is automatic: an anonymous `WEB_DESKTOP` bearer token is minted, cached and renewed (`nflTokenGen`) — no credential needed.

STATCAST EXTRAS (HAND-WRITTEN)

`mlb_statcast_search` (date-chunked, ≤25k rows/chunk) + `_minors`, `_wbc`, `mlb_statcast_player`; helpers `dateChunks translateFilters`.

```
await sdv.mlb.mlbStatcastSearch(
  { season: 2024, player_type: "batter" });
```

FAMILY HIGHLIGHTS

Family	Notable endpoints
mlb	`mlb_schedule mlb_pbp mlb_boxscore mlb_linescore mlb_people mlb_standings mlb_draft mlb_attendance mlb_awards`
statcast	`mlb_statcast_gamefeed` + 37 leaderboards + schedule (`…_bat_tracking …_arm_strength … _catch_probability`)
nhlweb	`nhl_api_web_pbp _boxscore _landing _club_stats _draft_picks _goalie_leaders`
nhledge	`nhl_edge_skater_detail _goalie_comparison _cat_skater_detail` …
nhlrest	`nhl_stats_rest_game _franchise _leaders_skaters _goalie_report`
records	`nhl_records_attendance _awards _coach _all_time_record_vs_franchise`
nflapi	`nfl_api_rosters _standings _injuries _draft_picks _combine_profiles _game_summaries _weeks`

Full per-family endpoint lists: the generated reference at js.sportsdataverse.org + the in-browser playground (run any endpoint live).

OPENAPI → NEW FAMILY

```
node tools/codegen/from-openapi.mjs \
  spec.yaml --api mystem
npm run codegen && npm run codegen:check
```

Providers are added mechanically from sdv-swagger OpenAPI specs; parsers + schemas are authored on top.

PROVIDERS · STANDALONE NAMESPACES

7 PROVIDER FAMILIES

Namespace	n	Auth
`sdv.odds` — The Odds API	10	`api_key` param
`sdv.recruiting` — 247Sports RDB	25	caller JWT via `headers`
`sdv.cbs` — CBS Sports	82	keyless
`sdv.fox` — Fox Sports	38	public apikey (defaulted)
`sdv.yahoo` — Yahoo (2 stems)	107	keyless + browser headers
`sdv.hockeytech` — HockeyTech	10	public per-league keys
`sdv.torvik` — BartTorvik	5	keyless, browser UA

```
await sdv.odds.oddsApiSports({
  api_key: process.env.ODDS_API_KEY,
  parsed: true });
await sdv.fox.foxScoreboard({ parsed: true });
await sdv.cbs.cbsGameScoringPlays(
  { game_id: "..." , parsed: true });
```

ODDS API ENDPOINTS

`sports sports_odds sports_scores sports_events sports_participants event_odds event_markets sports_odds_history sports_events_history event_odds_history` — R sibling: oddsapiR; usage counts against your Odds API quota.

HOCKEYTECH / LEAGUESTAT

One feed gateway, league-parameterized: `league` ∈ `pwhl ahl ohl whl qmjhl`. Endpoints: `seasons schedule teams team_roster player_stats game_shifts standings leaders pbp game_summary`.

```
await sdv.hockeytech.hockeytechSchedule(
  { league: "pwhl", season_id: 5 });
await sdv.hockeytech.hockeytechPbp(
  { league: "pwhl", game_id: 137 });
```

The runtime strips the JSONP envelope, injects each league's `client_code` / `key` / `site_id`, and handles the PWHL pbp key override. Env override: `SDV_<LEAGUE>_API_KEY`.

BARTTORVIK / T-RANK

`ratings team_factors game_stats player_stats game_schedule` — CSV + positional-array payloads, parsed to named rows (ported from hoopR).

```
await sdv.torvik.torvikRatings(
  { year: 2025, parsed: true });
```

LIMITS & GOTCHAS

- ESPN + most providers: keyless, **unofficial**, no documented rate limits — be polite; ESPN Core v2 403s under load.
- stats.nba.com is NOT wrapped here (TLS-blocked for JS clients) — use sportsdataverse-py's `nba_stats`.
- Tests are no-network (fixtures); typings ship in `dist/**/*.d.ts`.

LINKS

npm: **npmjs.com/package/sportsdataverse** · GitHub: **github.com/sportsdataverse/sportsdataverse-js** · docs + playground: **js.sportsdataverse.org** · Python: **py.sportsdataverse.org**.

sportsdataverse :: CHEAT SHEET

The SportsDataverse's Node.js / TypeScript client (npm `sportsdataverse`) — sister to **sportsdataverse-py** and the R packages (hoopR, wehoop, cfbfastR, fastRhockey, baseballr). A cross-league ESPN client (116 endpoints × 29 leagues) plus 532 native / provider live-API wrappers, with a tidy `{ parsed: true }` layer. Page 1 of 4: install, namespaces, parsers, return shapes.

v3.1.0

Node ≥ 20.18.1 · ESM · TypeScript
js.sportsdataverse.org

GET STARTED

```
npm install sportsdataverse
```

ESM-only (`"type": "module"`) — there is no CommonJS entry point. Ships `dist/` with full `.d.ts` typings.

```
import sdv from "sportsdataverse";        // ESM
// from CommonJS: dynamic import only
const { default: sdv } =
  await import("sportsdataverse");
```

QUICK START

```
const board = await sdv.nba
  .espnNbaScoreboard({});
```

```
const game = await sdv.nba
  .espnNbaSummary({ event_id: 401584793 });
```

```
const roster = await sdv.nfl
  .espnNflTeamRoster({ team_id: 12 });
// legacy method, still works:
const pbp = await sdv.nba
  .getPlayByPlay(401584793);
```

Every wrapper is `async` and takes one params object; snake_case and camelCase keys are interchangeable: `{ team_id: 12 }` ≡ `{ teamId: 12 }`.

RUNTIME NOTES

- HTTP via `axios`; every wrapper returns a `Promise` — use `await` or `.then()`.
- Node ≥ 20.18.1. In the browser, import only `sportsdataverse/parsers` (no node HTTP deps); the docs playground proxies live calls.
- Tests are no-network; codegen-driven (`npm run codegen` regenerates wrappers + docs from endpoint YAML).

DUAL-CASE NAMES

Generated wrappers register under BOTH `espn_nfl_team_roster` (snake — py/R parity) and `espnNflTeamRoster` (idiomatic JS). Same function.

NAMESPACE LAYOUT

Namespace	Contents
`sdv.nba .wnba .mbb .wbb .cfb .nfl .mlb .nhl`	legacy `get*` methods (page 2) + generated `espn_*` wrappers
`sdv.mch .wch .college_baseball .college_softball .ufl .xfl .cfl`	ESPN-only leagues
`sdv.soccer` + `epl laliga bundesliga seriea ligue1 mls ligamx ucl uel nwsl wwc wc`	league-param catch-all + 12 curated aliases
`sdv.cricket`	league-param ESPN cricket
`sdv.nhl .mlb .nfl`	+ native APIs: `nhlApiWeb*` `nhlEdge*` `nhlStatsRest*` `nhlRecords*` `mlb*` `mlbStatcast*` `nflApi*`
`sdv.odds .recruiting .cbs .fox .yahoo .hockeytech .torvik`	7 standalone provider namespaces (page 4)
`sdv.ncaa` · `sdv.tennis`	legacy-only services (page 2)

29 ESPN leagues + providers = 38 namespaces on the default export.

MULTI-LEAGUE SPORTS

```
await sdv.soccer.espnSoccerScoreboard(
  { league: "eng.1" });   // Premier League
await sdv.soccer.espnSoccerScoreboard(
  { league: "esp.1" });   // La Liga
```

`soccer` / `cricket` take an extra `league` slug; the curated aliases (`sdv.epl`, …) pin it for you.

TIDY ROWS: { PARSED: TRUE }

Every wrapper returns the **raw** JSON payload by default. Pass `{ parsed: true }` to run the registered parser → an array of flat, snake_cased row objects (the JS analog of py's `return_parsed=True`). Empty / malformed → `[]`, never throws.

```
const raw  = await sdv.nba
  .espnNbaScoreboard({});
const rows = await sdv.nba
  .espnNbaScoreboard({ parsed: true });
```

Summary dispatcher (21 sections)

```
await sdv.nba.espnNbaSummary({
  event_id: 401584793, parsed: true,
  section: "boxscore_player" });
```

Sections: `boxscore_player boxscore_team plays winprobability leaders game_info officials header season_series against_the_spread standings broadcasts format pickcenter odds article injuries news drives drive_plays scoring_plays`. Omit `section` → all 21 as a dict.

BROWSER-SAFE PARSERS

```
import { parseEndpoint }
  from "sportsdataverse/parsers";
```

The `/parsers` subpath export has no node-only HTTP deps — it powers the in-browser docs playground. Also exported: `normalize`, `snakeCase`, `PARSERS`, `parserFor`.

RETURN SHAPES

Call	Returns
default	raw provider JSON (ESPN subset for legacy `get*`)
`{ parsed: true }`	`rows[]` — flat snake_cased objects
`summary` + `section`	one sub-frame; omit → dict of 21
empty / error body	`[]` when parsed — chain without null checks

PIPE WITH TIDY.JS

```
import { tidy } from "sportsdataverse";
tidy.tidy(rows, tidy.groupBy("team",
  tidy.summarize({ n: tidy.n() })));
```

`@tidyjs/tidy` is re-exported — grammar-of-data verbs over parsed rows.

ADVANCED EXPORTS

`LEAGUES WRAPPERS FLAT_WRAPPERS makeLeagueModule makeFlatModule resolveFlat FLAT_HOSTS normalize PARSERS parserFor` + NFL auth: `nflTokenGen nflHeadersGen nflClearTokenCache jwtExp NFL_API_HOST`. Types: `LeagueConfig WrapperDef WrapperFn ParserFn`.

ECOSYSTEM

Python mirror: **sportsdataverse-py** — same `espn_<league>_<short>` names. R siblings: **hoopR wehoop cfbfastR fastRhockey baseballr**. Docs + live playground: **js.sportsdataverse.org**.

Legacy services :: sdv.<league>.get*

Pre-3.0 hand-written methods — preserved for back-compat; new code should prefer the generated `espn_*` wrappers (page 3)

8 LEAGUE SERVICES · COMMON SHAPE

GAME ENDPOINTS

On `sdv.cfb .mbb .mlb .nba .nfl .nhl .wbb .wnba` — all take an ESPN game `id`:

Function	Returns / notes
`getPlayByPlay(id)`	teams, plays, competitions, season, boxScore subset
`getBoxScore(id)`	gamepackage boxscore (+ `id`)
`getSummary(id)`	full ESPN summary subset
`getPicks(id)`	pickcenter; cfb mbb mlb nba nfl nhl only

```
const pbp = await sdv.wnba
  .getPlayByPlay(401244185);
const box = await sdv.cfb
  .getBoxScore(401628334);
```

Source: `cdn.espn.com/core/<sport>/…` game-package feeds. `getPlayByPlay` returns `{teams, id, plays, competitions, season, boxScore, seasonSeries, standings}`.

TEAMS

Function	Args
`getTeamList()`	colleges: `{group}` (cfb 80, mbb/wbb 50)
`getTeamInfo(id)`	nfl: `{id}` object
`getTeamPlayers(id)`	roster; nfl: `{id}`

```
await sdv.mbb.getTeamList({ group: 50 });
await sdv.nba.getTeamInfo(13);
await sdv.nfl.getTeamPlayers({ id: 12 });
```

Default groups: 80 = FBS (cfb) · 50 = NCAA D1 (mbb / wbb).

SCHEDULES & STANDINGS

Function	Args
`getSchedule`	`{year, month, day}` + colleges: `groups, seasontype`
`getScoreboard`	same + `limit` (300–1000)
`getStandings`	`{year}` (defaults to now)
`getConferences`	`{year}`; cfb mbb wbb only
`getRankings`	`{year, week}`; cfb only
`getWeeklySchedule`	`{week, year, seasonType}`; nfl only

```
await sdv.cfb.getSchedule(
  { year: 2024, month: 9, day: 7,
    groups: 80, seasontype: 2 });
await sdv.nfl.getWeeklySchedule(
  { week: 1, year: 2024, seasonType: 2 });
await sdv.mbb.getStandings({ year: 2024 });
await sdv.wbb.getConferences({ year: 2024 });
await sdv.cfb.getRankings(
  { year: 2024, week: 5 });
```

AVAILABILITY MATRIX

Namespace	Method count
`sdv.cfb`	15 (+ rankings, recruiting)
`sdv.mbb`	14 (+ recruiting)
`sdv.nfl`	11 (+ weekly schedule)
`sdv.mlb sdv.nba sdv.nhl`	10 each
`sdv.wbb`	10 (conferences, no picks)
`sdv.wnba`	9 (no picks / conferences)

RECRUITING SCRAPERS · CFB + MBB

247SPORTS (LEGACY SCRAPE)

Function	Args
`getPlayerRankings`	`{year, page, group, position, state}` (+ cfb `rankingsType`)
`getSchoolRankings`	`(year, page = 1)`
`getSchoolCommits`	`(school, year)`

```
await sdv.cfb.getPlayerRankings(
  { year: 2025, position: "QB" });
await sdv.mbb.getSchoolCommits(
  "Duke", 2025);
```

HTML scrape via cheerio. Superseded by the `sdv.recruiting.*` 247Sports RDB API family (page 4) — prefer it for new code.

TENNIS

```
await sdv.tennis.getScoreboard(
  { league: "atp", year: 2024,
    month: 7, day: 14 });
```

`league`: `"atp"` or `"wta"` (ESPN site v2 scoreboard).

MODERN REPLACEMENT

```
await sdv.recruiting.recruitingRankings(
  { year: 2025, headers: { ... } });
```

The 247Sports RDB API family (page 4) — 25 endpoints vs the 3 legacy scrapers; needs a caller-supplied JWT.

NCAA.COM · SDV.NCAA

GAMES (DATA.NCAA.COM)

Function	Purpose
`getInfo(game)`	gameInfo.json
`getBoxScore(game)`	boxscore.json
`getPlayByPlay(game)`	pbp.json
`getScoreboard({...})`	`{sport, division, year, month, day}`
`getRedirectUrl(url)`	resolve an ncaa.com game URL → casablanca id

```
const sb = await sdv.ncaa.getScoreboard({
  sport: "basketball-men", division: "d1",
  year: 2024, month: 3, day: 21 });
const url = sb.games[0].game.url;
const gid = await sdv.ncaa.getRedirectUrl(url);
const pbp = await sdv.ncaa.getPlayByPlay(gid);
```

STATS.NCAA.ORG RANKINGS

Function	Args
`getSports()`	—
`getSeasons(sport)`	sport code
`getDivisions(sport, season)`	—
`getSportDivisionData`	`(sport, season, division, type, gameHigh)`
`getPlayerData`	`(sport, season, division, rankingPeriod, gameHigh, category)`
`getTeamData`	same args as player

Covers every NCAA sport code (wvb, mih, wla, wsb, …) — see the npm keywords for the full list.

ESPN cross-league :: 116 endpoints × 29 leagues

Name parity with **sportsdataverse-py**'s `espn_*` wrappers · keyless, no auth

PATTERN & SCOPES

One core, parameterized on ESPN `(sport, league)` slugs. Each league exposes `espn_<prefix>_<short>` / `espn<Prefix><Short>` for every short name in its scope set:

Scope	Leagues	Shorts
`universal`	all 29	110
`ncaa`	college leagues	+3
`football`	nfl · cfb	+2
`mlb`	mlb	+1

ncaa: `rankings season_recruits season_week_rankings` · football: `season_qbr season_qbr_week` · mlb: `athlete_hotzones`.

```
await sdv.cfb.espnCfbRankings({});
await sdv.wnba.espnWnbaStandings(
  { season: 2024 });
await sdv.nfl.espnNflScoreboard(
  { week: 1, season_type: 2 });
```

ESPN-ONLY LEAGUES

```
await sdv.mch.espnMchScoreboard({});
await sdv.college_softball
  .espnCollegeSoftballScoreboard({});
await sdv.ufl.espnUflStandings({});
```

No legacy service — the generated surface is the whole namespace.

LEAGUES (LEAGUES.YAML)

Sport	Prefixes
basketball	`nba wnba mbb wbb`
football	`nfl cfb ufl xfl cfl`
baseball	`mlb college_baseball college_softball`
hockey	`nhl mch wch`
soccer	`soccer` + 12 aliases
cricket	`cricket`

SITE V2 · 25 SHORTS

SCORES, TEAMS, NEWS

Group	Short names
League	`scoreboard summary calendar news injuries transactions conferences statistics_league draft standings rankings`
Teams	`teams_site team team_roster team_schedule team_record team_depthcharts team_injuries team_transactions team_history team_news team_leaders`
Athletes	`athlete_info athlete_bio athlete_news`

WEB V3 · 5 SHORTS

ATHLETE STATS

`athlete_overview athlete_stats athlete_gamelog athlete_splits leaders`

```
await sdv.nba.espnNbaAthleteGamelog(
  { athlete_id: 1966, season: 2024 });
await sdv.mbb.espnMbbLeaders({});
await sdv.wnba.espnWnbaAthleteSplits(
  { athlete_id: 4433403 });
```

SUMMARY SECTIONS

```
await sdv.cfb.espnCfbSummary({
  event_id: 401628334, parsed: true,
  section: "drive_plays" });
```

`drives drive_plays scoring_plays` populate for NFL / CFB; other sports get zero-row frames.

CORE V2 · 86 SHORTS (1/2)

SEASONS & LEAGUE

Group	Short names
League	`league_root season_pointer seasons season_info season_types season_type standings_core leaders_core league_notes talentpicks tournaments positions position awards award`
Season	`season_group season_groups season_group_teams season_group_children season_type_leaders season_type_corrections season_weeks season_week season_week_events season_teams season_team season_athletes season_coaches season_draft season_draft_round_picks season_futures season_freeagents season_powerindex season_powerindex_leaders season_awards season_recruits season_week_rankings season_qbr season_qbr_week`
Teams & venues	`teams_core team_core venues venue franchises franchise coach coach_record coach_season`

CORE V2 · 86 SHORTS (2/2)

EVENTS & ATHLETES

Group	Short names
Events	`events event event_competition event_competitors event_competitor event_competitor_roster event_competitor_linescores event_competitor_statistics event_competitor_record event_competitor_leaders event_odds event_probabilities event_plays event_play event_play_personnel event_situation event_status event_officials event_official_detail event_broadcasts event_predictor event_powerindex event_propbets event_leaders event_scoringplays`
Athletes	`athletes_index athlete_core athlete_career_stats athlete_statisticslog athlete_eventlog athlete_contracts athlete_awards athlete_seasons athlete_records athlete_injuries athlete_notes athlete_vs_athlete athlete_hotzones`

```
await sdv.nhl.espnNhlEventOdds(
  { event_id: 401559593 });
await sdv.nba.espnNbaAthleteVsAthlete(
  { athlete_id: 1966, opp_id: 110 });
```

Core v2 payloads are `$ref`-linked; the generic parsers (`parse_items`, single-entity) tidy list + resource shapes. ESPN Core v2 403s under aggressive parallel load — keep concurrency low.

Native APIs & providers :: 532 flat wrappers · 15 families

All support `{ parsed: true }` · py siblings: **mlb_statcast_* nhl_web_* nfl_api_* hockeytech**

NATIVE · ON THE LEAGUE NAMESPACE

7 NATIVE FAMILIES

Wrappers	n	Host
`sdv.mlb.mlb*`	78	statsapi.mlb.com
`sdv.mlb.mlbStatcast*`	39+4	baseballsavant.mlb.com
`sdv.nhl.nhlApiWeb*`	27	api-web.nhle.com
`sdv.nhl.nhlEdge*`	35	…/v1/edge (tracking)
`sdv.nhl.nhlStatsRest*`	21	api.nhle.com/stats/rest
`sdv.nhl.nhlRecords*`	44	records.nhl.com
`sdv.nfl.nflApi*`	11	api.nfl.com

```
await sdv.mlb.mlbSchedule({ sport_id: 1,
  date: "2024-07-04", parsed: true });
await sdv.nhl.nhlApiWebPbp(
  { game_id: 2023030417, parsed: true });
await sdv.nfl.nflApiWeeklyGameDetails(
  { season: 2024, week: 1, parsed: true });
```

NFL.com auth is automatic: an anonymous `WEB_DESKTOP` bearer token is minted, cached and renewed (`nflTokenGen`) — no credential needed.

STATCAST EXTRAS (HAND-WRITTEN)

`mlb_statcast_search` (date-chunked, ≤25k rows/chunk) + `_minors`, `_wbc`, `mlb_statcast_player`; helpers `dateChunks translateFilters`.

```
await sdv.mlb.mlbStatcastSearch(
  { season: 2024, player_type: "batter" });
```

FAMILY HIGHLIGHTS

Family	Notable endpoints
mlb	`mlb_schedule mlb_pbp mlb_boxscore mlb_linescore mlb_people mlb_standings mlb_draft mlb_attendance mlb_awards`
statcast	`mlb_statcast_gamefeed` + 37 leaderboards + schedule (`…_bat_tracking …_arm_strength … _catch_probability`)
nhlweb	`nhl_api_web_pbp _boxscore _landing _club_stats _draft_picks _goalie_leaders`
nhledge	`nhl_edge_skater_detail _goalie_comparison _cat_skater_detail` …
nhlrest	`nhl_stats_rest_game _franchise _leaders_skaters _goalie_report`
records	`nhl_records_attendance _awards _coach _all_time_record_vs_franchise`
nflapi	`nfl_api_rosters _standings _injuries _draft_picks _combine_profiles _game_summaries _weeks`

Full per-family endpoint lists: the generated reference at js.sportsdataverse.org + the in-browser playground (run any endpoint live).

OPENAPI → NEW FAMILY

```
node tools/codegen/from-openapi.mjs \
  spec.yaml --api mystem
npm run codegen && npm run codegen:check
```

Providers are added mechanically from sdv-swagger OpenAPI specs; parsers + schemas are authored on top.

PROVIDERS · STANDALONE NAMESPACES

7 PROVIDER FAMILIES

Namespace	n	Auth
`sdv.odds` — The Odds API	10	`api_key` param
`sdv.recruiting` — 247Sports RDB	25	caller JWT via `headers`
`sdv.cbs` — CBS Sports	82	keyless
`sdv.fox` — Fox Sports	38	public apikey (defaulted)
`sdv.yahoo` — Yahoo (2 stems)	107	keyless + browser headers
`sdv.hockeytech` — HockeyTech	10	public per-league keys
`sdv.torvik` — BartTorvik	5	keyless, browser UA

```
await sdv.odds.oddsApiSports({
  api_key: process.env.ODDS_API_KEY,
  parsed: true });
await sdv.fox.foxScoreboard({ parsed: true });
await sdv.cbs.cbsGameScoringPlays(
  { game_id: "..." , parsed: true });
```

ODDS API ENDPOINTS

`sports sports_odds sports_scores sports_events sports_participants event_odds event_markets sports_odds_history sports_events_history event_odds_history` — R sibling: **oddsapiR**; usage counts against your Odds API quota.

HOCKEYTECH / LEAGUESTAT

One feed gateway, league-parameterized: `league` ∈ `pwhl ahl ohl whl qmjhl`. Endpoints: `seasons schedule teams team_roster player_stats game_shifts standings leaders pbp game_summary`.

```
await sdv.hockeytech.hockeytechSchedule(
  { league: "pwhl", season_id: 5 });
await sdv.hockeytech.hockeytechPbp(
  { league: "pwhl", game_id: 137 });
```

The runtime strips the JSONP envelope, injects each league's `client_code` / `key` / `site_id`, and handles the PWHL pbp key override. Env override: `SDV_<LEAGUE>_API_KEY`.

BARTTORVIK / T-RANK

`ratings team_factors game_stats player_stats game_schedule` — CSV + positional-array payloads, parsed to named rows (ported from **hoopR**).

```
await sdv.torvik.torvikRatings(
  { year: 2025, parsed: true });
```

LIMITS & GOTCHAS

- ESPN + most providers: keyless, **unofficial**, no documented rate limits — be polite; ESPN Core v2 403s under load.
- stats.nba.com is NOT wrapped here (TLS-blocked for JS clients) — use **sportsdataverse-py**'s `nba_stats`.
- Tests are no-network (fixtures); typings ship in `dist/**/*.d.ts`.

LINKS

npm: **npmjs.com/package/sportsdataverse** · GitHub: **github.com/sportsdataverse/sportsdataverse-js** · docs + playground: **js.sportsdataverse.org** · Python: **py.sportsdataverse.org**.