This is a huge document—heavily edited to be reduced in size—that specifies a significant portion of the data and functions for the game Accidental Woman. This information isn't mandatory or necessary to know in any way, but could be very useful if you are considering making your own game mods, or wish to submit content. The unedited document clocked in at 140 pages, and 27,000 words (it's now 79 and a little over 10,000).

The first section contains something of a master list of functions used by the game, contained in the "setup" and "aw" objects. It's set up in TypeScript syntax, showing the function parameters and return. After that is information on custom types and details of different data structures. After that is the class structure for the NPC and part of the PC characters. Finally there is the class structure for several game objects, not all-inclusive but fairly complete for items that are modable or will be in the near future.

Navigation: Good luck.

(use ctrl+f – yellow items typically have a corresponding item, and you can jump to it via the text search function in your browser/pdf reader.)

=== Game Functions ===

```
interface Engine {
  play: {(passage:string,something?:boolean):void};
}

interface Dialog {
  close: {():void};
  open: any;
  addClickHandler:any;
  setup:any;
  wiki:any;
  isOpen: { (): boolean};
}

interface setup {
  drag: {
    create: {(home:dragArea, ...ara:dragArea[]):void};
    homeListener: {():void};
    formatHome: { (items: string[]): string | Error};
```

```
    jewelry: { (): void};
    jewListener: { (): void};
    formatJewinv: { (items: string[]): string};
    formatJew: { (item: string): string};
  };
  operations: {
    tryGlobal: boolean;
    nicknames: boolean;
    fmRange: [number, number];
  };
  dice: {
    processDice: {(a:number|string,b?:number):number};
    processString: {(string:string):number|number[]};
    roll:{(a:number|string,b?:number):number};
  };
  fullscreen: {(element:any):void};
  messageMacro: {
    default: string;
  };
  playTime: setupPlaytime;
  pBar: {
    start: { (ident: cssID, cur: number, max: number): void};
    add: { (ident: cssID, amt: number): void};
    set: { (ident: cssID, amt: number): void};
    finish: { (ident: cssID, target: cssID, content: string): void};
  };
  selectStore: {(storyVar:string):[string,string]|false};
  storeCode: {(storyVar:string,code:string):boolean};
  evalCode: { (code: string, silent: boolean, $element: any, stream: any): boolean};
  acc: object;
  clothes: setupClothes;
  outfits: setupOutfits;
  clothesGen: setupClothesGen;
  consumables: setupConsumables;
  hair: setupHair;
  sInv: setupSInv;
  shop: setupShop;
  alert: {(msg:string):void};
  homeItems: setupHomeItems;
```

```
  isActive: { (npcid: npcid, restore: boolean): string};
  lactBreastCalc: { (): void};
  status: setupStatus;
  time: setupTime;
  refresh: {():void};
  notify: { (msg: twee, time?: number | false, classes?: string | false | string[]): void};
  jewel: setupJewel;
  statusLoad: { (): void};
  statusSave: { (): void};
  makeup: setupMakeup;
  generateStoreClothes: { (): void};
  shopInv: any;
  physicalATR: { (): void};
  totalATR: { (): void};
  bank: setupBank;
  bath: setupBath;
  condition: setupCondition;
  initCond: { (): void};
  fert: setupFert;
  food: setupFood;
  home: setupHome;
  job: setupJob;
  version: string;
  ver: number;
  map: setupMap;
  scene: setupScene;
  school: setupSchool;
  SCXfunc: { (): void};
  SCfunc: { (skillType: string, DCnum?: number, diceSize?: string | number, bonusVal?: number):
void};
  skillup: { (skillType: string): string};
  skillGain: { (type: string, dc: number): boolean};
  skillModCalc: { (skill: number): number};
  sleep: setupSleep;
  getReadySettings: { (): string};
  calcEnergyRate: { (): number};
  timeDisp: {};
  now: {};
  ui: setupUI;
```

```
unit: { (input: number, type: unit, num?: boolean): string | number};
unitHeight: { (input: number, type: unit): string};
climate: setupClimate;
weather: setupWeather;
playerHistoryComparison: { (): boolean};
week: setupWeek;
library: setupLibrary;
sex: setupSex;
dialog: { (title: string, content?: twee | 0): void};
loadOnce: number;
sexActs: any;
sexPos: any;
perf: any;
flipsat: number;
loadScript: { (sourceSrc: string): void};
resourceLoad: { (): void};
resourceLoadFinished: { (): void};
menuItemRunSaves: {};
appendStr: { (target: cssID, content: string): void};
bulkStoreNPC: { (npcidAr: npcid[]): void};
asyncBulkStoreNPC: {(NPCs: npcid[]): void};
randomDist: { (splooge: number[]): number};
backgroundThemeHandler: { (): void};
textThemeHandler: { (): void};
browserLimit: boolean;
colorThemeHandler: { (): void};
FontsizeHandler: { (): void};
RapeHandler: { (): void};
ExtremeHandler: { (): void};
ViolentHandler: { (): void};
refreshTheme: { (): void};
NPCStoreList: string[];
storeState: { (): void};
unpackVars: { (): void};
calculateNPCATR: { (main: any, body: any): [number, number, number]};
numWord: { (input: string | number): string | number};
monthName: { (mon: number): string};
numberLetAbrv: { (num: number): string};
updatePlayerHistory: { (): void};
```

```
  fillerText: { (chars: "rand" | number): string};
  sort: { (arr: object[], j: string): any[]};
  nameRandomizer: { (sex: number, race: race): string};
  nickRandomizer: { (sex: sexGenderNumber, name: string, body: object): string};
  surnameRandomizer: { (race: race): string};
  varanal: { (arg: string): any};
  calcBreastShape: {
    ({ size, silicone, weight, band, shape
    }: { size: number, silicone: number, weight: number, band: number, shape: string }):
string};
  calculateBreastStats: { (vol: number, shoulder: number, weight: number): any[]};
  deepThoughts: { (num: number): string};
  swimmer: boolean;
  swim: string;
  eMsg: { (reason: string | { name: string, message: string }): string};
  eliteStats: { (): string};
  log: { (output: string): void};
  phoneBGprint: { (): twee};
  phoneBGchange: { (back: boolean): string};
  phoneMenuPrint: { (): string};
  phoneMenuChange: { (back: boolean): string};
  colorretrieve: { (): true};
  scs: { (): void};
  scc: { (): void};
  insultGenerator: { (): string};
  cleanHome: { (unit: number): void};
  dirtyHome: { (): void};
  prop: { (prop: string): string};
  tooltipper: { (): void};
  pubeShape: { (style: string): string};
  forbiddenList: { (): void};
  addAppointment: { (day: number, week: number, { name, type, alert, start, end, place, loc,
code, msg, disc}: AppointmentInfo): void};
  daysList: { (days: weekdays): string};
  restoreNPC: { (npcid: npcid): boolean};
  storeNPC: { (npcid: npcid): boolean};
  textingMacroFunction: { (): void};
  initialize: setupInitialize;
  defineFixedNPCs: { (): void};
  ai: setupAI;
```

```
  breastCalc: { (): void};
  cleanStatus: { (): string};
  calcMilkCap: { (size: number, lact: number): number};
  backward: setupBackward;
  bot: {
    init: {};
    reply: {};
  };
  omni: setupOmni;
  imagedata: any;
  imageloaded: boolean;
  imageVersion: number;
  totalloadedimages: number;
  fload: number;
  loadImages: { (textdata: string, fname: string): void};
  fwait: number;
  imgwait: { (): void};
  imgbarinit: { (numFiles: number): void};
  imageAWRlist: { (name: string): void};
  imgbaradd: { (img?: string | boolean, totimg?: number): void};
  newImageLoader: { (): void};
  nimgbaradd: { (imgs: number): void};
  porn: setupPorn;
  AW: setupAW;
  sexCondomTempType:string;
  expectedImageLength:number;
  npcgen:setupNpcgen;
  npc: setupNPC;
  npcSetting: setupNPCsettings;
  startsPassage: string;
}

interface aw {
  code: awCode;
  Base64: awBase64;
  conLoad: awConLoad;
  drake: any
  awrMissing: number;
  ref: { (string: string): any};
```

```
  refName: {};
  acc: object;
  L: { (): void};
  S: { (): void};
  con: {
    enabled: { (): boolean};
    error: { (name: string, error: { name: string, message: string }):void};
    warn: { (msg: string): void};
    info: { (msg: string): void};
    obj: { (object: object, msg?: string | false): void};
    html: { (obj: Element, msg?: string | false): void};
    trace: { (msg?: string | false): void};
    table: { (obj: object, msg?: string | false): void}
  };
  get: { (key?: string): any};
  arrayCunt: { (arr: any[], fin: any, index?: number, del?: boolean): number};
  clothes: object;
  slot: {
    panties: any;
    bra: any;
    leg: any;
    top: any;
    bottom: any;
    coat: any;
    bag: any;
    accA: any;
    accB: any;
    accC: any;
    accD: any;
    shoes: any;
  };
  hair: {
    [propName: string]: Hairstyle;
  };
  hairDefine: {():void};
  invItems: awInvItems;
  homeItemsSwitch: { (orig: string, repl: string): void};
  homeItems: object;
  homeItemsDefine: {():void};
```

```
  capital: { (str: string): string};
  jewel: object;
  jewDefine: {():void};
  makeup: {
    eye: object;
    lip: object;
    gen: object;
  };
  makeupEyeDef: { (): void};
  makeupLipDef: { (): void};
  makeupGenDef: { (): void};
  job: object;
  school: object;
  dreams: {
    standard: {():string};
    none: { (): string};
    unsatisfied: { (): string};
    needy: { (): string};
    preg: { (): string};
    latePreg: { (): string}
  };
  sleep: awSleep;
  sexAct: object;
  sexActN: object;
  sexPos: object;
  base: {
    shop: {
      [propName: string]: number;
    };
    storeMod: {
      [propName: string]: number;
    };
    clothingAtrMod: {
      [propName: string]: [number,number];
    };
  }
  tempSPcunt: any;
  killit: { (target: cssID): boolean};
  append: { (target: cssID, content: twee): void};
```

```
  replace: { (target: cssID, content: twee): void};
  cash: { (amt: number, reason?: string): void};
  chad: any;
  ai: awAI;
  bot: any;
  botscript: string;
  n: { (id:npcid|number, activate?:boolean): any};
  parse: { (npc: string | number, cmd: string, ...args: any[]): string};
  imagedata: any;
  awrList: any;
  npc: {
    [propName: string]: NPC;
  }
}

interface State {
  temporary: any;
  active: {
    variables: {
      AW: object
    }
  };
  PC: {};
  makeup: object;
}

interface setupHair {
  prop: { (prop: string): any};
  stylist: { (hairstyle: string): string | Error};
  do: { (hairstyle: string): string | Error};
  undo: { (): void};
  shower: { (): void};
  print: { (limit?: number): string};
  known: { (): string};
  printButton: { (cunt: string, limit?: number): string};
  printWear: { (): string};
  doSet: { (set: string): any}
}
```

```
interface setupStatus {
  stress: { (amt: number, tgt?: number | string, restore?: boolean): void };
  anger: { (amt: number, tgt?: number | string, restore?: boolean): void };
  happy: { (amt: number, tgt?: number | string, restore?: boolean): void };
  tired: { (amt: number, tgt?: number | string, restore?: boolean): void };
  arousal: { (amt: number | string, tgt?: number | string, restore?: boolean): void };
  satisfact: { (amt: number, tgt?: number | string, restore?: boolean): void };
  getMilked: { (): void };
  milk: { (): number };
}

interface setupClothes {
  details: { (key: string): string};
  wardList: { (...slot: string[]): string};
  shopList: { (store: string, slot: clothingCategory): string};
  shopSalePrice: { (store: string, slot: clothingCategory, price: number): number};
  coordinate: { (top: number, bottom: number): 0 | 1 | 2 | 3};
  calculate: { (): void};
  wear: { (key: string, slot?: 0 | clothingSlot): "ERROR!" | "Success!"};
  referenceTryCount: number;
  referenceRebuild: { (): void};
  remove: { (slot: clothingSlot): "ERROR!" | "Success!"};
  delete: { (key: string, force: boolean): void};
  wearWords: { (key: string): string[] | string};
  wearKeyParse: { (inpt: string): string};
  sortButtons: { (arrayVarString: string, cuntID: string, twinePrinter: string): string};
  sort: { (array: Garment[], key: string, ascend: boolean): void};
  keyGen: { (): string};
  basePrice: { (type: clothingType, atr: number, formal: number, sexy: number, expose: number,
origin: string): number};
  initialize: {():void};
  defineObjects: { (): void};
  atrWord: { (val: number): string};
  dirty: { (val: number): string};
  health: { (val: number): string};
  exposureWord: { (val: number): string};
  sexyWord: { (val: number): string};
  formalWord: { (val: number): string};
  colorHex: { (code: number | string): string};
```

```
  gameLoad: { (string: string): void};
  colorWord: { (code: number): string};
  icon: setupClothesIcon;
  hemWord: { (num: number): string};
  isOutfitName: { (): string};
  PaperDollPrint: { (type: string): string};
  paperBody: { (): string};
  paperClothes: { (): string};
  paperAccessories: { (): string};
  paperShoes: { (): string};
  paperStatus: { (): string};
  PaperDollOptions: { (): string};
  clearStoreInv: { (): void};
  prologueGiver: { (type: string): void};
  quickPrint: { (slot: clothingCategory): string};
  defineCustomClothes: { (): void};
  outfitInitialize: { (): void};
  stained: boolean;
  kinky: boolean;
  exposed: setupClothesExposed;
  access: setupClothesAccess;
  dress: boolean;
  outfit: setupClothesOutfit;
  gameSave: { (): string};
  desc: setupClothesDesc;
}

interface setupClothesExposed {
  top: boolean;
  bottom: boolean
}

interface setupClothesAccess {
  pussy: boolean;
  ass: boolean;
  nip: boolean;
  tits: boolean;
  butt: boolean;
}
```

```
interface setupClothesDesc {

}

interface setupClothesOutfit {
  remove: { (): void};
  wear: { (name: string, rand?: boolean): void};
  empty: { (name: string):void};
  delete: { (name: string): void};
  save: { (name: string, group?: string, overwrite?: boolean): boolean};
  print: { (): string};
  check: { (): string};
  prologue: { (): boolean};
}

interface setupClothesIcon {
  panties: { (num: number, sub?: number): string};
  bra: { (num: number, sub?: number): string};
  "sports bra": { (num: number): string};
  stocking: { (num: number, sub?: number): string};
  top: { (num: number, sub?: number): string};
  bottoms: { (num: number, sub?: number): string};
  pants: { (num: number, sub?: number): string};
  shorts: { (num: number, sub?: number): string};
  skirt: { (num: number, sub?: number): string};
  dress: { (num: number, sub?: number): string};
  coat: { (num: number, sub?: number): string};
  swimTop: { (num: number, sub?: number): string};
  swimBottom: { (num: number, sub?: number): string};
  swimOnePiece: { (num: number, sub?: number): string};
  sportTop: { (num: number, sub?: number): string};
  athleticTop: { (num: number, sub?: number): string};
  sportBottom: { (num: number, sub?: number): string};
  athleticBottom: { (num: number, sub?: number): string};
}

interface setupClothesGen {
  panties: { (...args: [number, number, number, number, number, string]): string[]};
```

```
  bra: { (...args: [number, number, number, number, number, string]): string[]};
  stocking: { (...args: [number, number, number, number, number, string]): string[] };
  upperBody: { (...args: [number, number, number, number, number, string]): string[] };
  coat: { (...args: [number, number, number, number, number, string]): string[] };
  dress: { (...args: [number, number, number, number, number, string]): string[] };
  lowerBody: { (...args: [number, number, number, number, number, string]): string[] };
  swimBottom: { (...args: [number, number, number, number, number, string]): string[] };
  swimTop: { (...args: [number, number, number, number, number, string]): string[] };
  countSwimL: number;
  amtSwimL: number;
  countSwimU: number;
  amtSwimU:number;
  countLowerBody: number;
  amtLowerBody: number;
  countDress:number;
  amtDress:number;
  countOverWear:number;
  amtOverWear:number;
  countUpBody: number;
  countupperBody:number;
  amtUpBody:number;
  amtupperBody:number;
  countLeg:number;
  amtLeg:number;
  countBras:number;
  amtBras:number;
  countPanties:number;
  amtPanties:number;
}

interface setupOutfits {
  Casual: clothingOutfit;
  Work: clothingOutfit;
  Fancy: clothingOutfit;
  Home: clothingOutfit;
  Night: clothingOutfit;
}

interface setupConsumables {
```

```
  options: any;
  ref: any;
  getConsumable: { (id: string): any};
  hasConsumable: { (id: string, number: number): boolean};
  amtOfConsumable: { (id: string): number};
  consumableExists: { (id: string): boolean};
  getConsumableName: { (id: string): string};
  getConsumableCode: { (id: string): string};
  getConsumableDescr: { (id: string): any[]};
  getAllConsumables: { (): any};
  getCarriedConsumables: { (): string[]};
  findConsumableByIndex: { (index: number): string};
  findIndexOfConsumable: { (id: string): number};
  deleteConsumable: { (id: string):void};
  add: { (key: string, count?: number):void};
}

interface setupSInv {
  options: any;
  attachEvent: { (inv: any, loc: any, items: any, cont: any): void};
  inv: any;
}

interface awInvItems {
  image: object;
  info: object
}

interface setupShop {
  emptyCart: { (): void};
  viewCartShop: { (): void};
  viewCart: { (): void};
  cartTotal: { (): string};
  cartTotalNumber: { (): number};
  deleteCartItem: { (name: string): void};
  purchase: { (): "no afford" | "success"};
  process: { (): void};
  cartCheck: { (item: string, del?: boolean): number};
  skillPrice: { (amt: number): number};
```

```
  pushInv: { (item: string): void};
  launch: {
    clothes: { (store: string, category: clothingCategory): void};
  };
  sortButtons: { (arrayVarString: string, cuntID: string, twinePrinter: string): string};
  shopBanner: { (shop: string): string};
  storeImages: object;
  storeName: object;
  storeText: object;
}

interface setupJewel {
  prop: { (slot: jewelSlot, prop: string):any};
  slot: { (slot: jewelSlot): false | string | string[]};
  slotNames: jewelSlot[];
  comboNames: string[];
  worn: { (jewel: string): false | jewelSlot};
  exists: { (jewel: string): boolean};
  find: { (jewel: string): false | jewelSlot | "owned"};
  owned: { (jewel: string): boolean};
  removeAll: { (): string};
  putOn: { (item: string, slot: jewelSlot, give?: boolean): boolean | Error};
  takeOff: { (name: string): void};
  lose: { (name: string | jewelSlot):void};
  print: { ({ owned, slot, wearButt, removeButt, max, min, small} : {owned?:boolean,
slot?:false|jewelSlot, wearButt? : boolean, removeButt? : boolean, max? : false|number, min? :
false | number, small?:boolean}): string};
  sale: { ({ names, slot, wearButt, removeButt, max, min, small }: { names?: boolean|string[],
slot?: false | jewelSlot, wearButt?: boolean, removeButt?: boolean, max?: false | number, min?:
false | number, small?: boolean }): string};
  printWorn: { (): string};
  wearCount: { (): number};
  buttonGen: { (combo: jewelSlot, key: string): string};
  comboSlots: {
    wrist: string[];
    hand: string[];
    ring: string[];
    ear: string[];
    nip: string[]
  };
```

```
  slotWords: { (slot: jewelSlot): string};
  slotChecker: { (jew: string, slot: jewelSlot): boolean};
}

interface setupMakeup {
  calc: { (): void};
  wash: { (): string};
  smear: { (): void};
  shower: { (): void};
  applySet: { (set: string): string};
}

interface setupLibrary {
  callSexActN: { (book: string, chapter: string, page?: string | -1, chapArray?: 0 | any[]):
string};
  killSAN: { (): void};
  initSAN: { (): void};
  sexActN: any;
  callSexAct: { (book: string, chapter: string, page?: string | -1, chapArray?: 0 | any[]):
string};
  sexact: any;
  killSA: { (): void};
  initSA: { (): void};
  callSexPos: { (book: string, chapter: string, page?: string | -1, chapArray?: 0 | any[]):
string};
  callOrgasm: { (book: string, chapter: string, page?: string | -1, chapArray?: 0 | any[]):
string};
  initSP: { (): void};
  killSP: { (): void};
  killSAO: { (): void};
  orgasm: any;
  initSAO: { (): void};
  sexPos: any;
}

interface setupSex {
  defineAct: { (): void};
  defineActN: { (): void};
  sexActNList: string[];
  sexActList: string[];
```

```
  main: {
    (act: SexAct | SexPos, {type,ret,arg}: { type?: "act" | "pos", ret?: string, arg?:
string }): void};
  library: { (key: string, code: string, chapter?: string): string};
  npcActionSelect: { (ind: number): [string|number, string]};
  modifier: { (tgt, amt: number, sKink: kink[], sTrait: trait[], wKink: kink[], wTrait:
trait[]): number};
  refresh: { (): void};
  effect: {
    ({pleasure,arousal,wetness,cum,satisfy,strong,weak,
    }: sexEffectObject, target?: number): boolean[]};
  posEffect: {
    ({pleasure,arousal,wetness,strong,weak}: sexEffectObject, char?: number): boolean |
number};
  orgasmPC: {
    (act: any, type?: "playerAction" | "npcAction" | "position" | "none"): void};
  orgasmNPC: { (ind: number, act: sexEffectObject | "none", type?: "playerAction" | "npcAction"
| "position" | "none"): void};
  cumRiskText: { (base: string, ind: number): void};
  cum: { (ind: number, type: "playerAction" | "npcAction" | "position"): string};
  cumCondIncrement: { (val: string, vol: number): string};
  changePosition: { (key: string): void};
  changePositionAct: { (key: string, argu?: any): void};
  playerPosition: { (): string};
  sexAction: { (key: string, argu?: any): void};
  changePositionNPC: { (key: string, ind?: number): void};
  sexActionNPC: { (key: string, ind?: number): boolean};
  dumbCount: number;
  actionButtonPrinter: { (): void};
  printText: { touch: string; self: string; kiss: string; speak: string; kink: string; move:
string; position: string; item: string; other: string; hover: string };
  badAction: { (num: number): twee};
  startSex: { (...args: string[]): void};
  close: { (): void};
  valid: { (partArr: any, tar: any): boolean};
  partRef: { (part: string): [string, string] | false};
  validNPC: { (partArr: any, tar: any): boolean};
  partDist: { (part: string): number};
  statusBars: { (): twee};
  occupyPrinter: { (i: number): string};
  wetIcon: { (wet: number): twee};
```

```
  characterButtons: { (): twee};
  click: { (num: "PC" | number): void};
  characterTargets: { (): twee};
  target: { (ind: number): void};
  equipCondom: { (type: string): void};
  definePos: { (): void}
  sexPosList: string[]
}

interface setupBank {
  loan: {
    apply: { (): void};
    pay: { (): void};
    minPayment: { (): void};
    interest: { (): void};
    accept: { (): void}
  };
  savings: {
    apply: { (): void};
    deposit: { (): void};
    debit: { (): void};
    interest: { (): void};
    termCheck: { (): void};
    accept: { (): void}
  };
  credit: {
    apply: { (): void};
    debit: { (): void};
    minPayment: { (): void};
    pay: { (): void};
    interest: { (): void};
    accept: { (): void}
  };
  atm: { (bank?: string): void};
  creditCheck: { (): void};
  faust: {
    name: string
  };
  indigo: {
```

```
    name: string
  }
}

interface setupBath {
  brushTeeth: { (): void };
  pubeLength: object;
  shave: { (leg?: boolean, arm?: boolean, biki?: boolean, pube?: boolean): void };
  shower: { ({ quick, vagWash, enema, relax, shaveArm, shaveLeg, shaveGroin, trimPubes }:
{ quick?: boolean, vagWash?: boolean, enema?: boolean, relax?: boolean, shaveArm?: boolean,
shaveLeg?: boolean, shaveGroin?: boolean, trimPubes?: boolean }): void };
}

interface setupCondition {
  add: { ({ loc, amt, tgt, wet, type}: { loc: conditionLocation, amt?: number, tgt?: string,
wet?: number, type?: string; }): void};
  fluid: { (loc?: "pussy" | "asshole"): string};
  print: { (input: any): string};
}

interface setupFert {
  test: { (): void };
  cycle: { (tgt?: 0 | npcid): void };
  riskyDay: { (day: number, length: number): number };
  thinkBC: { (tgt: 0 | npcid): boolean | Error};
  thinkPreg: { (tgt: 0 | npcid): boolean | Error };
  dayOfCycle: { (tgt: 0 | npcid): number | Error};
}

interface setupFood {
  eat: { (amt: number, type?: foodType): void};
  fastfood: { (place: string): string};
  fast: object;
}

interface setupHome {
  apartmentNameGen: { (): string};
  apartmentStreetGen: { (loc: number): string};
  apartmentScoreDisp: { (qual: 1 | 2 | 3 | 4 | 5): string | boolean};
```

```
  apartmentDesc: { (T: 1 | 2 | 3 | 4 | 5, L: 1 | 2 | 3 | 4 | 5, F: 1 | 2 | 3 | 4 | 5, U: 1 | 2 | 3 | 4 | 5): string}
}

interface setupHomeItems {
  sales: { ({ names, type, max, min, room }: { names: string | false, type: "string" | false, max: number | false, min: number | false, room: string | false }): string };
}

interface setupJob {
  goto: { (): void};
  startAt: { (): void};
  arrival: { (): void};
  startDay: { (): void};
  jobTasks: { (): void};
  focusEffect: { (focus: string, effort: number): [number, number, string]};
  taskOutcome: { (result: any, tCount: number): string};
  taskLabel: { (result: any): void};
  endJob: { (): void};
  promote: { (): void};
  fire: { (): void};
  warning: { (): void};
  workCalendar: { (): string};
  institute: { (): boolean};
  time: {
    until: { (): number | "none"};
    late: { (): boolean};
    today: { (): boolean};
    tomorrow: { (): boolean};
  };
}

interface setupMap {
  nav: { (main: locationMain, sub: string, tert?: string | false): void};
  time: { (start?: "def" | "current" | string[], dest?: "def" | "work" | "home" | string[]): number};
  downtownTime: { (s: string, st: string, d: string, dt: string): number};
  bullseyeTime: { (s: string, d: string): number};
  residentialTime: { (s: string, d: string): number};
  carTime: { (dist: number): number};
```

```
  worldDistance: { (s: string, d: string): number};
  distToExit: { (m: locationMain, s?: 0 | string, t?: 0 | string): number};
  lookup: { (loc: null | [locationMain, string, string]): locationInfo};
}

interface setupScene {
  isLocName: { (loc: string): boolean};
  locationLib: { (loc: string): string};
  set: { ({ loc, pc, npcs, group, refresh }: { loc?: string, pc?: string, npcs?: string |
string[], group?: string, refresh?: boolean }): void};
}

interface setupSchool {
  scheduler: { (): void};
  temp: string;
  hours: { (school: number[]): string};
  print: { (school: string): string};
  coursePrint: { (school: string, course: string): string};
  printButtons: { (school: string): string};
  define: { (): void };
}

interface setupSleep {
  bar: { (t: number): void};
  start: { (): void};
  dream: { (): string};
  sleepProc: { (): void};
  print: { (content): void};
  wakingUp: { (): void};
  startNap: { (): void};
  npcStart: { (): void};
  sleepTime: { (): void};
  getMissed: { (): number};
  loneliness: { (): void};
  sleepIn: { (): boolean};
  sleepTotal: { (): boolean};
  status: { (): void};
  groomItems: { (): void};
  goddessCheck: { (): void};
```

```
  nap: { (min: number): void};
  morningText: { (wu: string): string};
}

interface setupTime {
  daytime: { (): "N" | "D" | "SR" | "SS"};
  timeBackup: time;
  dateBackup: date;
  saver: { (): void};
  add: { (addMin: number, disable?: boolean): void};
  set: { (hour: number, min?: number, newDay?: boolean, stats?: boolean): void};
  chunk: { (min: number): void};
  dateChange: { (): void};
  schedCheck: { (): void};
  missedCheck: { (): void};
  missed: { (d: number, w: number): number};
  socialCount: { (): number};
  dayplans: { (date?: false | date): string};
  dayplansFull: { (date?: boolean | date): twee};
  upcoming: { (date?: boolean | date): twee};
  reminder: { (msgs: string[]): void};
  status: { (count: number): void};
  after: { (hr: number | time, min?: number, aft?: boolean): boolean};
  until: { (hr: time | number, min?: number, aft?: boolean): number};
  dayName: { (d?: "no" | number): string};
  monthName: { (m?: "no" | number): string};
  cycle: { (): void};
  dif: { (timeA: time, timeB: time): number};
  now: { (): [number, number, boolean]};
  dateDisplay: { (date?: false | date): twee};
  format: { (tim: time | number): twee};
  toSleepMessage: { (): false | string};
  appointmentAlert: { (appt: AppointmentInfo): string};
  addictNeedIncrease: { (): void};
  withdrawl: { (drug: string): void};
}

interface setupUI {
  mainPhone: { (): twee};
```

```
  sunSym: { (): twee};
  menuButtons: { (): twee};
  actionButts: { (): twee};
  warnIndicate: { (): twee};
  debugToolers: { (): twee};
  debugToolersFloat: { (): twee};
  statusInfo: { (): twee};
  progressBar: { (size: number, label?: string | boolean, color?: string, anim?: string |
false): twee};
  charList: { (): twee};
  segmentBar: {
    (val?: number, {width,height,radius,count,bgc,padding,color,empty}?: { width?: string,
height?: number, radius?: number, count?: number, bgc?: string, padding?: number, color?:
string, empty?: string }): twee};
  soundElement: boolean;
  rainyMood: { (): void};
}

interface setupClimate {
  maxTemp: number[];
  minTemp: number[];
  devTemp: number[];
  daysRainA: number[];
  daysRainB: number[];
  daysRainC: number[];
  daysRainD: number[];
  snowfall: number[];
  snowGround: number[]
}

interface setupWeather {
  seedGen: { (): void};
  daySummary: { (date?: 0 | date): wxObject};
  dist: { (num: number, dev: number): number};
  maxMin: { (wx: wxObject): [number, number, number]};
  curTemp: { (wx?: 0 | wxObject): number};
  tempPrint: { (): twee};
  dayWeather: { (input?: 0 | wxObject): { sky: string, con: string, mHour: number, leng:
number }};
  curWeather: { (): twee};
```

```
  cloudWord: { (cld: string): string};
  wxWord: { (code: string): string};
  seed: any[];
}

interface setupWeek {
  bar: { (t?: number): void};
  start: { (): string};
  main: { (): void};
  tutorial: { (): void};
  npcProc: { (): void};
  playerHistoryComparison: { (): void}
}

interface setupInitialize {
  one: { (): void };
  two: { (): void };
  three: { (): void };
  four: { (): void };
  five: { (): void };
  six: { (): void };
  seven: { (): void };
  sevenHalf: { (): void };
  eight: { (): void };
  nine: { (): void };
  ten: { (): void };
  eleven: { (): void };
  twelve: { (): void };
  thirteen: { (): void };
  fourteen: { (): void };
  final: { (): void };
}

interface setupAI {
  query: { (npc: any, note: string, ...tags: string[]): number};
  record: { (npc: NPC, result: any, inputs: object, tags: object)};
  load: { (): void};
  normalize: { (npc: { core: any }): any };
  normalTags: { (tags: string[]): number[]};
```

```
  export: { (num?: number): void };
  saveTraining: { (): void};
  runTrain: { (index: number): void};
  tagInfo: { (tag: string): string };
  tagger: { (npcid?: string): void};
  resultWord: { (reso: number): string};
}

interface setupBackward {
  //none yet...
}

interface setupOmni {
  run: {():void};
}

interface setupAW {
  compress: { (obj: object): string};
  decompress: { (str: string): string};
  localStore: { (id: string, data: string): boolean};
  localRestore: { (id: string, del?: boolean): string};
  compressNPC: { (npcid: string): void};
  decompressNPC: { (npcid: npcid): void};
  bulkCompressNPC: { (npcidAr: string[]): void};
  compressObj: { (objName: string): void};
  decompressObj: { (objName: string): void};
  compressAWT: { (): string | 69};
  decompressAWT: { (file: any): any};
  compressAWN: { (): string};
  decompressAWN: { (file: any): any};
  testStorage: { (): void};
  jsonNPCbody: { (name: string, npcid: npcid): void};
  jsonNPCmain: { (name: string, npcid: npcid): void};
  jsonNPCfert: { (name: string, npcid: npcid): void};
  jsonNPCbground: { (name: string, npcid: npcid): void};
  jsonNPC: { (name: string, npcid: npcid): void};
  clone: { (toClone: any): any};
}
```

```
interface awCode {
  compressToBase64: { (input: any): any };
  compress: { (uncompressed: any): any };
  decompressFromBase64: { (input: any): any };
  compressToUTF16: { (input: any): any };
  decompressFromUTF16: { (compressed: any): any };
  compressToUint8Array: { (uncompressed: any): any };
  decompressFromUint8Array: { (compressed: any): any };
  decompress: { (compressed: any): any };
  compressToEncodedURIComponent: { (input: any): any };
  decompressFromEncodedURIComponent: { (input: any): any };
  _compress: { (uncompressed: any, bitsPerChar: any, getCharFromInt: any): any };
  _decompress: { (length: any, resetValue: any, getNextValue: any): any };
}

interface awBase64 {
  compressToUTF16: { (input: any): any};
  decompressFromUTF16: { (input: any): any};
  _keyStr:string;
  decompress: { (input: any): any};
  compress: { (input: any): any};
}

interface setupPorn {
  imgid: { (index: number): string};
  imgclass: { (index: number): string};
  imgElement: { (id: string, source: string, classes?: 0 | string): twee};
  placeholder: string;
  gifrefresh: { (targets: string[], sources: string[]): void | Error};
  followon: { (targets: string[], sources: string[]): void | Error};
  preprint: { (imgar: string[]): string};
  maleNPC: { (npc?: any): any};
}

interface awConLoad {
  start: { (): void};
  add: { (): void};
  upload: { (file: File, fname: string): void};
  genItem: { (fname: string): void};
```

```
  loadMods: { (): void};
  copier: { (key: string): void};
  temp: any;
  output: any;
  data: {
    images: object;
    hairStyles: object;
    homeItems: object;
    jewelry: object;
    clothes: object;
    makeup: object;
  }
  images?: { (data: any): void};
  hairStyles?: { (data: any): void};
  homeItems?: { (data: any): void};
  jewelry?: { (data: any): void};
  clothes?: { (data: any): void};
  makeup?: { (data: any): void};
}

interface setupNpcgen {
  NPC: { (SemiNPCgenOptions):void};
  age: { (min: number, max: number): number};
  birthday: { (age: number, bday: 0 | date): date};
  setupMainVars: { (gender: number, npc: any): void};
  racistHitlerFuck: { (race: number, subrace: number, npc: any): void};
  maleBody: { (npc: any, tone: number, weight: number, should: number, hip: number, waist:
number): void};
  femaleBody: { (npc: any, tone: number, weight: number, should: number, hip: number, waist:
number): void };
  femaleFertility: { (npc: any, fert: number | "rand"): void};
  maleFertility: { (npc: any, fert: number | "rand"): void};
  futaFertility:any;
  sexCharMale: { (npc: any, tits: number, cock: number): void};
  sexCharFemale: { (npc: any, tits: number, pussy: number): void};
  sexCharFuta: { (gender: number, npc: any, tits: number, cock: number, pussy: number): void};
  maleFace: { (npc: any, beauty: number, face: number): void};
  femaleFace: { (npc: any, beauty: number, face: number): void};
  miscMale: { (npc: any): void};
  miscFemale: { (npc: any): void};
```

```
  mutateNPCmale: { (npc: any): void};
  mutateNPCfemale: { (npc: any): void};
  background: { (npc: any, gender: number, edu: number | "rand", wealth: number | "rand",
jobber: 0 | string): void};
  npcFlags: { (npc: any): void};
  relationship: { (npc: any): void};
  schedule: { (npc: any): void};
  maleStatus: { (npc: any): void};
  femaleStatus: { (npc: any): void};
  condition: { (npc: any): void};
  malePersonality: { (npc: any, homo: number): void};
  femalePersonality: { (npc: any, homo: number): void};
  traits: { (npc: any): void};
  malePrefs: { (npc: any): void};
  femalePrefs: { (npc: any): void};
  maleOutfits: { (npc: any): void};
  femaleOutfits: { (npc: any): void};
  donSomeClothing: { (npc: any): void};
  femalePortrait: { (npc: any, portName: 0 | string): void};
  malePortrait: { (npc: any, portName: 0 | string): void};
  setLists: { (npcid:string): void}
}

interface setupNPC {
  fixedIDs?: { [propName: string]: string,};
  ready: string[];
  stored: string[];
  single: string[];
  rShip: string[];
  married: string[];
  male: string[];
  female: string[];
  futa: string[];
  age13to17: string[];
  age18to21: string[];
  age22to27: string[];
  age28to33: string[];
  age34to39: string[];
  age40to49: string[];
```

```
  age50to59: string[];
  age60plus: string[];
  poor: string[];
  middle: string[];
  wealthy: string[];
  education: setupNPCeducation;
  friends: string[];
  bff: string[];
  boyFriend: string[];
  interested: string[];
  fling: string[];
  exes: string[];
  enemies: string[];
}

interface setupNPCeducation {
  dropout: string[];
  hschool: string[];
  assoc: string[];
  bach: string[];
  master: string[];
  doctor: string[];
}

interface setupNPCsettings {
  orgasmAdjust: number;
  futaRate: number;
  futaMale: boolean;
  gender: [boolean, number];
  arch: (boolean | number[])[];
  wealth: [boolean, number, number];
  age: [boolean, number[], number[]];
  homo: ((boolean | number)[] | boolean)[];
  race: (boolean | number[])[];
  height: (boolean | number[])[];
  tone: (boolean | number[])[];
  weight: (boolean | number[])[];
  fertility: [boolean, number, number];
  cock: (boolean | number)[];
```

```
  pussy: (boolean | number)[];
  circum: (boolean | number)[];
  beauty: (boolean | number[])[];
  orgasm: (boolean | number)[];
  pregTerm: [boolean, number];
  tits: (boolean | number)[];
  mutate: [boolean, number[], number[]];
  married: [boolean, number];
  open: (boolean | number)[];
  vert: (boolean | number)[];
  iq: [boolean, number, number];
  names: [boolean, string[] | [0], string[] | [0], string[] | [0]];
  persCore: (boolean | number[])[];
  pref: setupNPCsettingsPref;
}

interface setupNPCsettingsPref {
  enabled: boolean;
  Fweight: number[];
  Mweight: number[];
  Fheight: number[];
  Mheight: number[];
  Fmuscle: number[];
  Mmuscle: number[];
  Fother: number[];
  Mother: number[];
}

interface setupPlaytime {
  options: {
    tryGlobal: boolean;
    storyVar: string;
    pauseTag: string
  };
  getMS: any;
  getTimeArray: any;
  get: any;
  formatTime: any;
  output: any;
```

```
}

interface dragArea {
  ident: string;
  max: number | false;
  min: number | false;
}

interface awSleep {
  startTime: time;
  startDate: date;
  social: number;
  missed: number;
  earlyWakeMins: number;
  minsToWake: number;
  totalmins: number;
  extramins: number;
  sleepIn: boolean;
  timeUntilWork: number;
  slpMsg: string[];
  startLoc: [string, string, string];
}

interface awAI {
  record: any[];
  prime: any;
  procreate: any;
  morality: any;
  agreeable: any;
  conscient: any;
  loyalty: any;
  curiosity: any;
  neurotic: any;
  ego: any;
}
```

=== Prototype Stuff ===

```
interface Window {
```

```
  AWAI: any;
  jQuery: { (selector: any): any };
  $: { (selector: any): any };
  dice: any;
  Inventory: any;
  getPlayTime: { (arg: any): any };
  playTime: any;
  getConsumable: { (id: string): any[] };
  hasConsumable: { (id: string, num?: number): boolean };
  amtOfConsumable: { (id: string): number };
  consumableExists: {(id: string): boolean};
  getConsumableName: { (id: string): string};
  getConsumableCode: { (id: string): string };
  getConsumableDescr: { (id: string): any[] };
  getAllConsumables: { (): any[] };
  getCarriedConsumables: { (): any[] };
  findConsumableByIndex: { (index: number): string };
  findIndexOfConsumable: { (id: string): number};
  deleteConsumable: { (id: string): void};
}

interface Array<T> {
  delete: (item:any) => void;
  deleteAt: Function;
  flatten: Function;
  includes: { (params: any): boolean };
  includesAny: {(search:any[]): boolean};
}

interface JSON {
  reviveWrapper:any;
}

interface Number {
  dice: any;
  fairmath: any;
  fm: any;
  d: any;
}
```

```
interface Math {
  clamp: any;
  fairmath: any;
  fm: any;
}

/*
. 88888888888
.     888
.     888
.     888  888  888 88888b.   .d88b.  .d8888b
.     888  888  888 888 "88b d8P  Y8b 88K
.     888  888  888 888  888 88888888 "Y8888b.
.     888  Y88b 888 888 d88P Y8b.          X88
.     888   "Y88888 88888P"   "Y8888   88888P'
.               888 888
.          Y8b d88P 888
.           "Y88P"  888
.
.  DEFINITION OF SPECIFIC DATA TYPES - HOORAY
*/

type twee = string;

type cssID = string;

type race =
"white"
| "Gaelic"
| "Nordic"
| "black"
| "native American"
| "middle eastern"
| "hispanic"
| "southern European"
| "southeast Asian"
| "Asian"
| "south Asian";
```

```
type skill =
"exhibition"
| "prostitute"
| "sex"
| "oral"
| "seduction"
| "comm"
| "org"
| "probSolving"
| "finance"
| "art"
| "athletic"
| "dancing"
| "clean"
| "shop"
| "cook"
| "fight"
| "willPower";

type weekdays = [
  any,
  boolean,
  boolean,
  boolean,
  boolean,
  boolean,
  boolean,
  boolean
];

type time = [number, number, boolean] | [number, number];

type date = [number, number, number] | [number, number, number, number];

type posBodyPart = "mouth" | "handL" | "handR" | "titL" | "titR" | "pussy" | "clit" | "asshole"
| "cock" | "balls";

type sexGenderNumber = 0 | 1 | 2 | 3 | 4;
```

```
type clothingType = "bra" | "pants" | "dress" | "panties" | "skirt" | "coat" | "sports bra" |
"shorts" | "stocking" | "top" | "jacket" | "swimTop" | "swimBottom" | "swimOnePiece" |
"sportBottom" | "sportTop" | "onepiece";


type clothingSlot = "panties" | "bra" | "leg" | "top" | "bottom" | "coat" | "bag" | "accA" |
"accB" | "accC" | "accD" | "shoes";


type clothingCategory = "panties" | "bra" | "leg" | "top" | "bottom" | "coat" | "bag" | "acc" |
"niteU" | "niteL" | "dress" |"swimU" | "swimL";


type jewelSlot =
  "neck"|
  "wristR"|
  "wristL"|
  "handR"|
  "handL"|
  "ringR"|
  "ringL"|
  "nose"|
  "lip"|
  "tongue"|
  "brow"|
  "earR"|
  "earL"|
  "upEar"|
  "belly"|
  "nipR"|
  "nipL"|
  "clit"|
  "labia"|
  "wrist"|
  "hand"|
  "ring"|
  "ear"|
  "nip"|
  "none";


type clothingOutfit = {
  panties: 0|string,
```

```
  bra: 0 | string,
  leg: 0 | string,
  top: 0 | string,
  bottom: 0 | string,
  accA: 0 | string,
  accB: 0 | string,
  accC: 0 | string,
  accD: 0 | string,
  coat: 0 | string,
  shoes: 0 | string,
  bag: 0 | string,
  rand: string,
};

type npcid = string;

type conditionLocation = "hair"|
"face"|
"chest"|
"back"|
"hands"|
"stomach"|
"butt"|
"groin"|
"genitals"|
"thighs"|
"legs"|
"feet"|
"vagFluid"|
"anusFluid";

type foodType = "junk" | "dessert" | "normal" | "health" | "diet";

type locationMain = "world" | "residential" | "bullseye" | "downtown" | "home" | "start" |
"BFhome";

type locationInfo = {
  image: string,
  name: string,
```

```
  passage: string,
  loc: string,
  desc: string,
};

type unit = "in"|"ft"|"yd"|"mi"|"gl"|"lbs"|"cm"|"m"|"km"|"l"|"kg";

type wxObject = {
  seed: number;
  max: number;
  min: number;
  dev: number;
  precip: number[];
  snowfall: number;
  snowGround: number;
  tSeed: number;
}

type sexEffectObject = {
  pleasure: { // pleasure (prog to orgasm) given by action
    pcAmt?: number, //actual amount
    pcMax?: number, //max allowed percent  - won't increase pleasure above that point.
    npcAmt?: number,
    npcMax?: number,
    amt?: number,
    max?: number,
  },
  arousal: { //arousal gain from action (base)
    pcAmt?: number,
    pcMax?: number,
    npcAmt?: number,
    npcMax?: number,
    amt?: number,
    max?: number,
  },
  wetness: { //amount of wetness to increase
    pcAmt?: number,
    pcMax?: number,
    npcAmt?: number,
```

```
    npcMax?: number,
    amt?: number,
    max?: number,
  },
  cum: false|object, //can override cum destination from position. Needs Object if override!
  satisfy: [number, number], //modifier to satisfaction gain from orgasm this way.
(value/10)*amt [pc,npc]
  strong: { //list of traits/kinks that this action is strong for
    pcKink?: kink[],
    pcTrait?: trait[],
    npcKink?: kink[],
    npcTrait?: trait[],
    kink?: kink[],
    trait?: trait[]
  },
  weak: { //list of traits/kinks that decrease effect/weak for.
    pcKink?: kink[],
    pcTrait?: trait[],
    npcKink?: kink[],
    npcTrait?: trait[],
    kink?: kink[],
    trait?: trait[]
  },
};


type kink =
"risky"|"pregnancy"|"sizequeen"|"cumSlut"|"sub"|"exhibition"|"masochist"|"buttSlut"|"public"|"s
lut"|"superSlut"|"hyperSlut"|"oral"|"anal"|"force"|"rape"|"liberate"|"easy"|"nips"|"dom"|"water
"|"bond"|"hard"|"fap"|"shame"|"none";


type trait =
"caring"|"bitch"|"maternal"|"romantic"|"deceptive"|"devious"|"persuasive"|"perceptive"|"forgetf
ul"|"forgiving"|"lowEsteem"|"picky"|"crude"|"friendly"|"approachable"|"relaxed"|"flirty"|"mater
ialist"|"none"|
  "-caring" | "-bitch" | "-maternal" | "-romantic" | "-deceptive" | "-devious" | "-persuasive"
| "-perceptive" | "-forgetful" | "-forgiving" | "-lowEsteem" | "-picky" | "-crude" | "-
friendly" | "-approachable" | "-relaxed" | "-flirty" | "-materialist";


type AppointmentInfo = {
  name: string,
  type: number,
  alert: boolean,
```

```
  start: [number, number],
  end?: false | [number,number],
  place: string | false,
  loc?: false | string[],
  code?: 0 | Function,
  msg?: false | string,
  disc?: false | string
};

type orificeTightness = -1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13;

type zeroFive = 0 | 1 | 2 | 3 | 4 | 5;

type pubeStyle = "bushy"|"trimmed"|"neatly-
trimmed"|"bikinitrim"|"bikiniline"|"stubble"|"garden"|"heart"|"neat patch"|"mohawk"|"landing-
strip"|"brazilian"|"triangular"|"martini"|"stamp"|"shaved"|"hairless";

type ConditionItem = {
  [propName:string]:{amt:number,wet?:number}
};
```

=== Data Structure ===

```
interface Makeup {
  atr: number;
  sexy: number;
  clown: boolean;
  type: string;
  desc: string;
  look: string;
```

```
}

interface HairSets {
  normal: string;
  fancy: string;
  casual: string;
  'custom 1'?: string;
  'custom 2'?: string;
  'custom 3'?: string;
}

interface VagFluid {
  vulva: any[];
  vest: any[];
  mid: any[];
  deep: any[];
  cervix: any[];
  womb: any[];
  ovary: any[];
}

interface SexRecord {
  vanilla: number;
  oralPC: number;
  oralNPC: number;
  anal: number;
  public: number;
  swallowed: number;
  creampie: number;
  accidentCP: number;
  forced: number;
  unprotected: number;
  interupted: number;
  nocumNPC: number;
  nocumPC: number;
  mob: number;
  bondage: number;
  sadoMaso: number;
  watersport: number;
```

```
  domsub: number;
  roleplay: number;
  fetish: number;
  exhibit: number;
  rapist: number;
  raped: number;
  saboPCbc: number;
  caughtSabo: number;
  PCsaboBC: number;
  PCsaboCaught: number;
  sexlocs: string[];
  tags: string[];
}

interface NPCflags {
  other: string[];
  events: string[];
  knows: string[];
  rumor: string[];
  exes: string[];
  kids: number;
  kidsPC: number;
  cheatonPC: number;
  cheatedon: number;
  cheatWithPC: number;
  knowPCcheated: number;
  PCknowCheated: number;
  toys: boolean;
  toysPublic: boolean;
  knowPCpreg: boolean;
  isFather: boolean;
  thinkFather: boolean;
  suspicion: number;
  PCsuspicion: number;
  thinkPCfaithful: boolean;
  thinkNPCfaithful: boolean;
}

interface AddictionStatus {
```

```
  sex: number;
  alc: number;
  heat: number;
  satyr: number;
  focus: number;
  cum: number;
  zone: number;
  cream: number;
  sexNeed: number;
  alcNeed: number;
  heatNeed: number;
  satyrNeed: number;
  focusNeed: number;
  cumNeed: number;
  zoneNeed: number;
  creamNeed: number;
  max: number;
}

interface Nutrition {
  [propName: string]: number;
}

interface Energy {
  amt: number;
  rate: number;
  regen: boolean;
}

interface WombData {
  preg: boolean;
  know: boolean;
  weeks: number;
  days: number;
  birth: number;
  count: number;
}

interface Ego {
```

```
  str: number;
  selfinterest: number;
  selfworth: number;
  confidence: number;
  fragility: number;
  selfimage: number;
  mach: number;
  risk: number;
}

interface Neurotic {
  str: number;
  impulsive: number;
  unstable: number;
  addiction: number;
  anxiety: number;
  sensitive: number;
  anger: number;
  sadness: number;
}

interface Curiosity {
  str: number;
  complex: number;
  learning: number;
  abstract: number;
  curiosity: number;
  novelty: number;
}

interface Loyalty {
  str: number;
  betrayal: number;
  cheating: number;
  effort: number;
  permanence: number;
  family: number;
}
```

```
interface Conscient {
  str: number;
  thoughtful: number;
  responsible: number;
  attention: number;
  trustworthy: number;
  structure: number;
}

interface Agreeable {
  str: number;
  interest: number;
  empathy: number;
  caring: number;
  trust: number;
  altruism: number;
}

interface Morality {
  str: number;
  life: number;
  liberty: number;
  property: number;
  honesty: number;
  integrity: number;
}

interface Procreate {
  str: number;
  secure: number;
  preg: number;
  kids: number;
  evolve: number;
  pleasure: number;
}

interface NPCpreference {
  Fweight: [number, number, number, number, number, number];
  Mweight: [number, number, number, number, number, number];
```

```
  Fheight: [number, number, number, number, number];
  Mheight: [number, number, number, number, number];
  Fmuscle: [number, number, number, number, number, number];
  Mmuscle: [number, number, number, number, number, number];
  Fother: [number, number, number, number, number, number, number, number, number, number];
  Mother: [number, number, number, number, number, number, number, number, number];
  active: number;
  romance: number;
  novel: number;
  excite: number;
  night: number;
  expensive: number;
  fancy: number;
  popular: number;
  position: string[];
  sexact: string[];
  kinks: string[];
}

interface Mutations {
  Smooth?: boolean;
  LilithCurse?: boolean;
  NoRefract?: boolean;
  MegaNuts?: boolean;
  KillerSperm?: boolean;
  BitchBreaker?: boolean;
  MegaLong?: boolean;
  Iron?: boolean;
  Virile?: boolean;
  AcidPre?: boolean;
  Girth?: boolean;
  Contort?: boolean;
  Cumpire?: boolean;
  PowerEjac?: boolean;
  Multgasm?: boolean;
  Immune?: boolean;
  Milk?: boolean;
  Acid?: boolean;
  BC?: boolean;
```

```
  Multiple?: boolean;
  Gestate?: boolean;
  Cycle?: boolean;
  TwinWomb?: boolean;
  Pheromone?: boolean;
  Period?: boolean;
  Mouth?: boolean;
  PseudoPreg?: boolean;
  Elastic?: boolean;
  LitePhero?: boolean;
}

interface KinkList {
  risky: boolean;
  pregnancy: boolean;
  sizequeen: boolean;
  cumSlut: boolean;
  sub: boolean;
  exhibition: boolean;
  masochist: boolean;
  buttSlut: boolean;
  public?: boolean;
  publix?: boolean;
  slut: boolean;
  superSlut: boolean;
  hyperSlut: boolean;
  oral: boolean;
  anal: boolean;
  force: boolean;
  rape: boolean;
  liberate: boolean;
  easy: boolean;
  nips: boolean;
  dom: boolean;
  water: boolean;
  bond: boolean;
  hard: boolean;
  fap: boolean;
  shame: boolean;
```

```
}

interface ClothingPC {
  worn: ClothingWorn;
  keys: ClothingKeys;
  stats: ClothingStats;
  coordinate: ClothingCoordinate;
  spots: AccessorySpots;
}

interface AccessorySpots {
  mouth: boolean;
  nipL: boolean;
  nipR: boolean;
  clit: boolean;
  pussy: boolean;
  ass: boolean;
  face: boolean;
  head: boolean;
  handL: boolean;
  handR: boolean;
}

interface ClothingCoordinate {
  outfit: boolean;
  shoes: boolean;
  coat: boolean;
}

interface ClothingStats {
  atr: number;
  sexy: number;
  formal: number;
  exposureTop: number;
  exposureBot: number;
}

interface ClothingWorn {
  panties: string | 0;
```

```
  bra: string | 0;
  leg: string | 0;
  top: string | 0;
  bottom: string | 0;
  bag: string | 0;
  coat: string | 0;
  accA: string | 0;
  accB: string | 0;
  accC: string | 0;
  accD: string | 0;
  shoes: string | 0;
}

interface ClothingKeys {
  panties: string | 0;
  bra: string | 0;
  leg: string | 0;
  top: string | 0;
  bottom: string | 0;
  bag: string | 0;
  coat: string | 0;
  accA: string | 0;
  accB: string | 0;
  accC: string | 0;
  accD: string | 0;
  shoes: string | 0;
}

interface PCjewelry {
  neck: string;
  wristR: string;
  wristL: string;
  handR: string;
  handL: string;
  ringR: string;
  ringL: string;
  nose: string;
  lip: string;
  tongue: string;
```

```
  brow: string;
  earR: string;
  earL: string;
  upEar: string;
  belly: string;
  nipR: string;
  nipL: string;
  clit: string;
  labia: string;
  pierced: PiercingSlots;
  owned: string[];
}

interface PiercingSlots {
  neck: boolean;
  wristR: boolean;
  wristL: boolean;
  handR: boolean;
  ringR: boolean;
  ringL: boolean;
  handL: boolean;
  nose: boolean;
  lip: boolean;
  tongue: boolean;
  brow: boolean;
  earR: boolean;
  earL: boolean;
  upEar: boolean;
  belly: boolean;
  nipR: boolean;
  nipL: boolean;
  clit: boolean;
  labia: boolean;
}

interface PCskills {
  exhibition: number;
  prostitute: number;
  sex: number;
```

```
  oral: number;
  seduction: number;
  comm: number;
  org: number;
  probSolving: number;
  finance: number;
  art: number;
  athletic: number;
  dancing: number;
  clean: number;
  shop: number;
  cook: number;
  martial: number;
}

interface Persona {
  type: string;
  bCon: number;
  mentionFertile: boolean;
  fidelity: number;
  sweet: boolean;
  sexy: boolean;
  slutty: boolean;
  lookingFor: string;
}

interface PCtraits {
  vert: string;
  intro: boolean;
  extro: boolean;
  open: string;
  op: boolean;
  cl: boolean;
  will: 0 | 1 | 2 | 3 | 4 | 5;
  libido: 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8;
  caring: -1 | 0 | 1;
  bitch: -1 | 0 | 1;
  maternal: -1 | 0 | 1;
  romantic: -1 | 0 | 1;
```

```
  deceptive: -1 | 0 | 1;
  devious: -1 | 0 | 1;
  persuasive: -1 | 0 | 1;
  perceptive: -1 | 0 | 1;
  forgetful: -1 | 0 | 1;
  forgiving: -1 | 0 | 1;
  lowEsteem: -1 | 0 | 1;
  picky: -1 | 0 | 1;
  crude: -1 | 0 | 1;
  friendly: -1 | 0 | 1;
  approachable: -1 | 0 | 1;
  relaxed: -1 | 0 | 1;
  flirty: -1 | 0 | 1;
  materialist: -1 | 0 | 1;
}

interface SemiNPCgenOptions {
  npcid: string;
  gender?: sexGenderNumber,
  age?: [number, number],
  name?: 0 | string,
  surname?: 0 | string,
  nickname?: 0 | string,
  portName?: 0 | string,
  bday?: 0 | date,
  homo?: 0 | 1 | 2,
  race?: number,
  subrace?: number,
  tone?: number,
  weight?: number,
  fert?: "rand" | number,
  tits?: number,
  cock?: number,
  pussy?: number,
  beauty?: number,
  edu?: "rand" | number,
  wealth?: "rand" | number,
  jobber?: 0 | string,
  should?: number,
```

```
  hip?: number,
  waist?: number,
  face?: number,
}

interface NPCoutfit {
  atr: number,
  sexy: number,
  formal: number,
  expTop: number,
  expBot: number,
  panties: false | string,
  bra: false | string,
  leg: false | string,
  top: false | string,
  bottom: false | string,
  accA: false | string,
  accB: false | string,
  accC: false | string,
  coat: false | string,
  shoes: false | string,
  bag: false | string,
}

interface NPCslots {
  panties: false | string,
  bra: false | string,
  leg: false | string,
  top: false | string,
  bottom: false | string,
  accA: false | string,
  accB: false | string,
  accC: false | string,
  coat: false | string,
  shoes: false | string,
  bag: false | string,
}

//  888b    888 8888888b.   .d8888b.
```

```
//  8888b   888 888   Y88b d88P  Y88b
//  88888b  888 888    888 888    888
//  888Y88b 888 888   d88P 888        .d8888b
//  888 Y88b888 8888888P"  888        88K
//  888  Y88888 888        888    888 "Y8888b.
//  888   Y8888 888        Y88b  d88P      X88
//  888    Y888 888         "Y8888P"   88888P'
//
//
```

```
class Tits {
  public shape: string;
  public nipple: string;
  public nipLength: 1|2|3|4|5|6|7|8;
  public nipGirth: 1 | 2 | 3 | 4 | 5;
  public areola: 1 | 2 | 3 | 4 | 5;
  public puffy: 1 | 2 | 3 | 4 | 5;
  public band: 24 | 26 | 28| 30 | 32 | 34 | 36 | 38 | 40 | 42 | 44 | 46;
  public base: {
    cupNum: number;
    cupRaw: number;
    size: number;
    cup: string;
    bra: string;
  };
  public silicone: number;
  public lact: {
    on: boolean;
    max: number;
    size: number;
    cupNum: number;
    cup: string;
    bra: string;
  };
  get size() {}
  get cupNum() {}
  get cup() {}
  get bra() {}
  get has() {}
```

```
  public calculate() {}
}

class Pussy {
  public virgin: boolean;
  public tight: orificeTightness;
  public stretch: zeroFive;
  public time: number;
  public wetness: number;
}

class Anus {
  public virgin: boolean;
  public tight: orificeTightness;
  public stretch: zeroFive;
  public time: number;
  public wetness: number;
}

class Penis {
  public length: number;
  public girth: number;
  public head: string;
  public vol: number;
  public circum: boolean;
  public hard: number;
  public smegma: boolean;
  public tags: string[];
  public calculate() {}
}

class Balls {
  public count: number;
  public size: number;
  public sac: number;
  public hang: number;
  public tags: string[];
}
```

```
class BirthCon {
  public diaphragm: {
    worn: boolean;
    type: string;
    effect: number;
    health: number;
    break: boolean;
    sabo: number;
  };
  public femaleCondom: {
    worn: boolean;
    type: string;
    effect: number;
    health: number;
    break: boolean;
    sabo: number;
  };
  public menstrualCup: {
    worn: boolean;
    type: string;
    effect: number;
    health: number;
    break: boolean;
    sabo: number;
  };
  public sponge: {
    worn: boolean;
    type: string;
    effect: number;
    health: number;
    break: boolean;
    sabo: number;
  };
  public condom: {
    worn: boolean;
    type: string;
    effect: number;
    health: number;
```

```
    break: boolean;
    sabo: number;
  };
  public headCap: {
    worn: boolean;
    type: string;
    effect: number;
    health: number;
    break: boolean;
    sabo: number;
  };
  public hormone: number;
  public hormoneType: string;
  public knowIneffective: boolean;
  public bcIneffective: boolean;
}

class Body {
  public tits: Tits;
  public pussy: Pussy;
  public asshole: Anus;
  public cock: Penis;
  public balls: Balls;
  public race: string;
  public skinColor: string;
  public tone: number;
  public weight: number;
  public shoulders: number;
  public hips: number;
  public waist: number;
  public height: number;
  public ass: number;
  public pelvis: number;
  public clit: number;
  public labia: number;
  public beauty: number;
  public face: string;
  public brow: string;
  public lips: number;
```

```
  public nose: string;
  public jaw: string;
  public eyeColor: string;
  public lactation: number;
  public lactCapacity: number;
  public orgasm: number;
  public energy: number;
  public topATR?: any;
  public botATR?: any;
  public ATR?: any;
  public tags?: string[];
  public ears: string;
}

class PCmain {
  public ageOriginal: number;
  public age: number;
  public ageID: number;
  public name: string;
  public surname: string;
  public background: string;
  public bd: number[];
  public female: boolean;
  public male: boolean;
}

class NPCmain {
  public id: npcid;
  public age: number;
  public bd: date;
  public female: boolean;
  public male: boolean;
  public genes: string;
  public seen: boolean;
  public interact: boolean;
  public relation: boolean;
  public suicide: boolean;
  public lifetime: number;
  public count: number;
```

```
  public tags: string[];
  public name: string;
  public surname: string;
  public portrait: string;
  public nickname: string;
}

class Jewelry {
  public neck: string;
  public wristR: string;
  public wristL: string;
  public handR: string;
  public handL: string;
  public ringR: string;
  public ringL: string;
  public nose: string;
  public lip: string;
  public tongue: string;
  public brow: string;
  public earR: string;
  public earL: string;
  public upEar: string;
  public belly: string;
  public nipR: string;
  public nipL: string;
  public clit: string;
  public labia: string;
  public pierced: PiercingSlots;
  public owned: string[];
}

class NPCgroom {
  public hairColor: string;
  public hairCurl: number;
  public pubeColor: string;
  public pubeLength: number;
  public hairdye: string;
  public hairLength: number;
  public hairStyle: string;
```

```
  public hairDefaultFancy: string;
  public hairDefaultCasual: string;
  public pubeStyle: pubeStyle;
  public bikini: string;
  public pubeShape: string;
  public leghair: string;
  public armpit: string;
  public makeup: Makeup;
  public teeth: string;
}

class PCgroom {
  public hairColor: string;
  public hairCurl: number;
  public pubeColor: string;
  public pubes: string;
  public hairdye: string;
  public ears: string;
  public bottomATR: number;
  public hairLength: number;
  public hairStyle: string;
  public hairFlag: number;
  public hairSets: HairSets;
  public pubeStyle: string;
  public pubeGrow: number;
  public pubeFreq: number;
  public pube: number;
  public bikiniFreq: number;
  public bikiniCount: number;
  public pubeCount: number;
  public pubeShape: string;
  public leghair: number;
  public leghairCount: number;
  public leghairFreq: number;
  public armpit: number;
  public armpitCount: number;
  public armpitFreq: number;
  public eyeMU: string;
  public lipMU: string;
```

```
  public genMU: string;
  public makeup: Makeup;
  public teeth: string;
  public toothHealth: number;
  public toothbrush: number;
  public lastToothbrush: number[];
}

class Fert {
  public fertility: number;
  public egg: number;
  public implant: number;
  public vagHostile: number;
  public period: number;
  public wombHealth: number;
  public multEgg: number;
  public barren: boolean;
  public IUD: boolean;
  public femaleFlag: string[];
  public cycle: number;
  public cycStart: date;
  public boost: number;
  public ovuMod: number;
  public fluid?: VagFluid;
  public iud: boolean;
  public pregTerm: number;
  public flagF: string[];
  public quality: number;
  public ejac: number;
  public resMax: number;
  public reserve: number;
  public refact: number;
  public quantity: number;
  public surv: number;
  public maleFlag: string[];
}

class Background {
  public hschool: boolean;
```

```
  public college: boolean;
  public associate: boolean;
  public bachelor: boolean;
  public master: boolean;
  public doctor: boolean;
  public inschool: boolean;
  public education: number;
  public homeParents: boolean;
  public wealth: number;
  public cash: number;
  public bank: number;
  public debt: number;
  public home: number;
  public job: string;
  public car: string[];
  public timeApple: number;
  public sister: number;
  public sisterYounger: boolean;
  public brother: number;
  public brotherYounger: boolean;
  public parentDivorced: boolean;
  public stepParent: string;
  public dadDead: boolean;
  public momDead: boolean;
  public married: boolean;
  public exSpouse: number;
  public rShip: boolean;
  public affair: boolean;
}

class Relation {
  public friend: boolean;
  public acquaint: boolean;
  public dating: boolean;
  public lovers: boolean;
  public exclusive: boolean;
  public engaged: boolean;
  public married: boolean;
  public likePC: number;
```

```
  public likeNPC: number;
  public lovePC: number;
  public loveNPC: number;
  public companion: number;
  public domsub: number;
  public mesh: number;
  public daysince: number;
  public space: number;
  public dates: number;
  public hangout: number;
  public met: number;
  public sleptover: number;
  public pcslept: number;
}

class Schedule {
  public workdays: boolean[];
  public workhours: number[];
  public workLoc: string;
  public outhours: number[];
  public locations: any[];
}

class Clothes {
  public worn: ClothingWorn;
  public keys: ClothingKeys;
  public stats: ClothingStats;
  public coordinate: ClothingCoordinate;
  public spots: AccessorySpots;
}

class NPCclothes {
  public outfits: {
    casual: NPCoutfit,
    fancy: NPCoutfit,
    work: NPCoutfit,
    athletic: NPCoutfit,
    swim: NPCoutfit,
  };
```

```
  public current: string;
  public worn: NPCslots;
}

class Womb {
  public preg: boolean;
  public know: boolean;
  public weeks: number;
  public days: number;
  public birth: number;
  public count: number;
}

class Addiction {
  public sex: number;
  public alc: number;
  public heat: number;
  public satyr: number;
  public focus: number;
  public cum: number;
  public zone: number;
  public cream: number;
  public sexNeed: number;
  public alcNeed: number;
  public heatNeed: number;
  public satyrNeed: number;
  public focusNeed: number;
  public cumNeed: number;
  public zoneNeed: number;
  public creamNeed: number;
  public jonesing: number;
  public withdrawl: boolean;
  get max(): string {}
  get maxValue(): number {}
}

class Status {
  public birthCon: BirthCon;
  public alcohol: number;
```

```
public drugs: number;
public wetness: number;
public fertText: string;
public risk: number;
public wombA: Womb;
public wombB: Womb;
public period: number;
public milk: number;
public milkStore: number;
public arousal: number;
public pleasure: number;
public need: number;
public satisfaction: number;
public atr?: any;
public stress: number;
public happy: number;
public anger: number;
public lonely: number;
public fatigue: number;
public sleep: boolean;
public health: number;
public healthOld: number;
public will: number;
public overAnger: boolean;
public overStress: boolean;
public overDepress: boolean;
public underSatisfy: number;
public addict: Addiction;
public energy: Energy;
public kids: number;
public morality: number;
public perversion: number;
public clean: number;
public exercise: number;
public bimbo: number;
public inPublic: boolean;
public injury: string[];
public disease: string[];
public mindbreak: boolean;
```

```
  public nutrition: Nutrition;
}

class Condition {
  public hair: ConditionItem;
  public face: ConditionItem;
  public chest: ConditionItem;
  public back: ConditionItem;
  public hands: ConditionItem;
  public stomach: ConditionItem;
  public butt: ConditionItem;
  public groin: ConditionItem;
  public genitals: ConditionItem;
  public thighs: ConditionItem;
  public legs: ConditionItem;
  public feet: ConditionItem;
  public vagFluid: ConditionItem;
  public anusFluid: ConditionItem;
}

class Mutation {
  public smooth: boolean;
  public lilithCurse: boolean;
  public noRefract: boolean;
  public megaNuts: boolean;
  public killerSperm: boolean;
  public bitchBreaker: boolean;
  public megaLong: boolean;
  public iron: boolean;
  public virile: boolean;
  public acidPre: boolean;
  public girth: boolean;
  public contort: boolean;
  public cumpire: boolean;
  public powerEjac: boolean;
  public multgasm: boolean;
  public immune: boolean;
  public milk: boolean;
  public acid: boolean;
```

```
  public bc: boolean;
  public multiple: boolean;
  public gestate: boolean;
  public cycle: boolean;
  public twinWomb: boolean;
  public pheromone: boolean;
  public period: boolean;
  public mouth: boolean;
  public pseudoPreg: boolean;
  public elastic: boolean;
  public litePhero: boolean;
}

class Persona {
  public type: string;
  public bCon: number;
  public mentionFertile: boolean;
  public fidelity: number;
  public sweet: boolean;
  public sexy: boolean;
  public slutty: boolean;
  public lookingFor: string;
}

class NPCprefs {
  public Fweight: [number, number, number, number, number, number];
  public Mweight: [number, number, number, number, number, number];
  public Fheight: [number, number, number, number, number];
  public Mheight: [number, number, number, number, number];
  public Fmuscle: [number, number, number, number, number, number];
  public Mmuscle: [number, number, number, number, number, number];
  public Fother: [number, number, number, number, number, number, number, number, number,
number];
  public Mother: [number, number, number, number, number, number, number, number, number];
  public active: number;
  public romance: number;
  public novel: number;
  public excite: number;
  public night: number;
```

```
  public expensive: number;
  public fancy: number;
  public popular: number;
}

class Kinks {
  public risky: boolean;
  public pregnancy: boolean;
  public sizequeen: boolean;
  public cumSlut: boolean;
  public sub: boolean;
  public exhibition: boolean;
  public masochist: boolean;
  public buttSlut: boolean;
  public public: boolean;
  public slut: boolean;
  public superSlut: boolean;
  public hyperSlut: boolean;
  public oral: boolean;
  public anal: boolean;
  public force: boolean;
  public rape: boolean;
  public liberate: boolean;
  public easy: boolean;
  public nips: boolean;
  public dom: boolean;
  public water: boolean;
  public bond: boolean;
  public hard: boolean;
  public fap: boolean;
  public shame: boolean;
}

class NPCkinks {
  public position: string[];
  public sexact: string[];
  public risky: boolean;
  public pregnancy: boolean;
  public sizequeen: boolean;
```

```
  public cumSlut: boolean;
  public sub: boolean;
  public exhibition: boolean;
  public masochist: boolean;
  public buttSlut: boolean;
  public public: boolean;
  public slut: boolean;
  public superSlut: boolean;
  public hyperSlut: boolean;
  public oral: boolean;
  public anal: boolean;
  public force: boolean;
  public rape: boolean;
  public liberate: boolean;
  public easy: boolean;
  public nips: boolean;
  public dom: boolean;
  public water: boolean;
  public bond: boolean;
  public hard: boolean;
  public fap: boolean;
  public shame: boolean;
}

class NPCtraits {
  public will: number;
  public libido: number;
  public open: number;
  public vert: number;
  public perv: number;
  public corrupt: number;
  public diq: number;
  public iq: number;
  public bimbo: number;
  public op: boolean;
  public cl: boolean;
  public intro: boolean;
  public extro: boolean;
  public sexuality: number;
```

```
  public straight: boolean;
  public bi: boolean;
  public homo: boolean;
  public bitch: number;
  public lowEsteem: number;
}
class Traits {
  public vert: string;
  public intro: boolean;
  public extro: boolean;
  public open: string;
  public op: boolean;
  public cl: boolean;
  public will: 0 | 1 | 2 | 3 | 4 | 5;
  public libido: 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8;
  public caring: -1 | 0 | 1;
  public bitch: -1 | 0 | 1;
  public maternal: -1 | 0 | 1;
  public romantic: -1 | 0 | 1;
  public deceptive: -1 | 0 | 1;
  public devious: -1 | 0 | 1;
  public persuasive: -1 | 0 | 1;
  public perceptive: -1 | 0 | 1;
  public forgetful: -1 | 0 | 1;
  public forgiving: -1 | 0 | 1;
  public lowEsteem: -1 | 0 | 1;
  public picky: -1 | 0 | 1;
  public crude: -1 | 0 | 1;
  public friendly: -1 | 0 | 1;
  public approachable: -1 | 0 | 1;
  public relaxed: -1 | 0 | 1;
  public flirty: -1 | 0 | 1;
  public materialist: -1 | 0 | 1;
}

class Core {
  public procreate: Procreate;
  public morality: Morality;
  public agreeable: Agreeable;
```

```
  public conscient: Conscient;
  public loyalty: Loyalty;
  public curiosity: Curiosity;
  public neurotic: Neurotic;
  public ego: Ego;
}

class Records {
  public makeout: string[];
  public sex: SexRecord;
  public flag: NPCflags;
}

class Skills {
  public exhibition: number;
  public prostitute: number;
  public sex: number;
  public oral: number;
  public seduction: number;
  public comm: number;
  public org: number;
  public probSolving: number;
  public finance: number;
  public art: number;
  public athletic: number;
  public dancing: number;
  public clean: number;
  public shop: number;
  public cook: number;
  public martial: number;
}

class NPC {
  public body: Body;
  public main: NPCmain;
  public groom: NPCgroom;
  public fert: Fert;
  public status: Status;
  public rship: Relation;
```

```
  public cond: Condition;
  public background: Background;
  public friends: string[];
  public mutate: Mutation;
  public pref: NPCprefs;
  public core: Core;
  public clothes: NPCclothes;
  public prefs: NPCprefs;
  public trait: NPCtraits;
  public kink: Kinks;
  public sched: Schedule;
  public record: Records;
}




/*
/*  ██████╗██╗      █████╗ ███████╗███████╗███████╗███████╗
/* ██╔════╝██║     ██╔══██╗██╔════╝██╔════╝██╔════╝██╔════╝
/* ██║     ██║     ███████║███████╗███████╗█████╗  ███████╗
/* ██║     ██║     ██╔══██║╚════██║╚════██║██╔══╝  ╚════██║
/* ╚██████╗███████╗██║  ██║███████║███████║███████╗███████║
/*  ╚═════╝╚══════╝╚═╝  ╚═╝╚══════╝╚══════╝╚══════╝╚══════╝
*/
class Timer {
  public name: string;
  public start: time;
  public end: boolean | time;
  public startDate: date;
  public endDate: boolean | date;
  public dur: boolean | number;
}


class Garment {
  public key: string;
  public a;
  public nick: string;
```

```
public type: clothingType;
public slot: clothingSlot;
public color: string;
public style: string;
public subStyle: string;
public tertiary: string;
public fabric: string;
public wetness: number;
public wear: string[];
public cond: object;
public price: number;
public flag: {
  text?: string,
  hem?: number,
};
public origin: string;
public swimwear: boolean;
public nightwear: boolean;
public athletic: boolean;
public kinky: boolean;
public access: {
  nip: boolean,
  pussy: boolean,
  ass: boolean,
  butt: boolean,
  tits: boolean,
};
public save: boolean;
public img: string|0;
public values: {
  atr: number,
  sexy: number,
  formal: number,
  exposure: number,
  style: number,
  subStyle: number,
  fabric: number,
  color: number,
  damage: number,
```

```
    dirty: number,
    price: number,
  };
  public print: () => string;
    this.print = function(size = "big", cls = "none"): string {}
  // color hex code
  get hex(): string {}
  // attraction word
  get atr(): string {}
  // exposure word
  get exposure(): string {}
  // formal word
  get formal() {}
  // image string no brackets
  get image(): string {}
  // sexiness word
  get sexy(): string {}
  // returns dirtiness words
  get dirty(): string {}
  // returns words for damage
  get health(): string {}
  // returns string describing the status (health, dirty, condition)
  get status(): string {}
  // describes condition
  get condition(): string {}
  // runs setup.clothes.details on this for infos
  get details(): string {}
}



class Hairstyle {
  public name: string;
  public key: string;
  public sDesc: string;
  public lDesc: string;
  public learn: number;
  public atr: number;
  public time: number;
  public diff: number;
```

```
  public autoPass: number;
  public image: string;
  public max: number;
  public min: number;
}

class HomeItem {
  public name: string;
  public key: string;
  public type: string;
  public image: string;
  public tags: string[];
  public mult: boolean;
  public desc: string;
  public quality: number;
  public cost: number;
  public fragile: boolean;
  public notRoom: string[];
  public button: string;
  public info: string;
  public action: () => void;
  public actionText: string;
  public effect: () => void;
  public effectText: string;
  public remove: () => void;
  public breaks: () => void;
    this.remove = function() {}
    this.breaks = function() {}
  get owned() {}
}

class Jewel {
  public key: string;
  public short: string;
  public long: string;
  public slot: jewelSlot;
  public atr: number;
  public cost: number;
  public type: string;
```

```
  public image: string;
  public name: string;
}

class EyeMakeup {
  public key: string;
  public name: string;
  public short: string;
  public long: string;
  public learn: number;
  public atr: number;
  public thick: number;
  public garish: number;
  public sexy: number;
  public time: number;
  public diff: number;
  public autoPass: number;
  public image: string|boolean;
  public putOn() {}
  public learnIt() {}
  public print() {}
  public button(cmd) {}
}

class LipMakeup {
  public key: string;
  public name: string;
  public short: string;
  public long: string;
  public learn: number;
  public atr: number;
  public thick: number;
  public garish: number;
  public sexy: number;
  public time: number;
  public diff: number;
  public autoPass: number;
  public image: string | boolean;
  public putOn() {}
```

```
  public learnIt() {}
  public print() {}
  public button(cmd) {}
}

class GenMakeup {
  public key: string;
  public name: string;
  public short: string;
  public long: string;
  public learn: number;
  public atr: number;
  public thick: number;
  public garish: number;
  public sexy: number;
  public time: number;
  public diff: number;
  public autoPass: number;
  public image: string | boolean;
  public putOn() {}
  public learnIt() {}
  public print() {}
  public button(cmd) {}
}

class SexAct {
  public name: string;
  public key: string;
  public label: string;
  public hover: string;
  public tab: string;
  public cat: string;
  public kink: string[];
  public skill: any;
  public parts: [string, string];
  public effect: sexEffectObject;
  public sex: sexGenderNumber;
  public tags: string[];
  public req: string[];
```

```
  public forbid: string[];
  public special: () => void;
  public button: string;
  public uniqueReq: () => void;
  get allowed() {}
}

class SexActN {
  public name: string;
  public key: string;
  public cat: string;
  public kink: string[];
  public parts: [string, string];
  public effect: sexEffectObject;
  public sex: sexGenderNumber;
  public tags: string[];
  public req: string[];
  public forbid: string[];
  public special: () => void;
  public uniqueReq: () => void;
  get allowed() {}
}



interface Ipos {
  head: [number, number, number, string];
  chest: [number, number, number, string];
  belly: [number, number, number, string];
  groin: [number, number, number, string];
  kneeR: false | [number, number, number, string];
  kneeL: false | [number, number, number, string];
  footR: false | [number, number, number, string];
  footL: false | [number, number, number, string];
  handR: false | [number, number, number, string];
  handL: false | [number, number, number, string];
  blocked: posBodyPart[];
  occupy: posBodyPart[];
  sex: sexGenderNumber; // required gender 0:none 1:male 2:female (requires parts, so 1 can be
futa)
```

```
  effect: sexEffectObject;
}
class SexPos {
  public name: string;
  public key: string;
  public desc: string;
  public movText: string;
  public img: string;
  public cat: string;
  public basic: string;
  public kink: string[];
  public min: number;
  public count: number;
  public sex: boolean;
  public anal: boolean;
  public speed: {max: number, min: number};
  public cum: [{any}];
  public mult: number;
  public req: string[];
  public forbid: string[];
  public control: [boolean, number];
  public tags: string[];
  public pos: Ipos[];
  public group: string;
  public special: () => void;
  public action: () => void;
  public label: string;
  public hover: string;
  public button: string;
  public npc: () => void;
  get allowed() {}
}

class Cum {
  public owner: npcid;
  public vol: number;
  public qual: number;
  public surv: number;
  public quant: number;
```

```
  public amt: number;
  public time: time;
  public date: date;
}

class School {
  public key: string;
  public name: string;
  public desc: string;
  public img: string;
  public days: boolean[];
  public hours: [number, number];
  public basePrice: number;
  public courses: any;
  public hooks: string[];
  public instructor: string[];
  public member: boolean;
  public loc: string[];
  public signUp(course: string) {}
  public reqCheck(course: string) {}
  public quit(course: string) {}
  public reallyQuit(course: string, fee: number) {}
  public enrolled(course: string): boolean {}
  public available(course: string): boolean {}
  public attend(course: string) {}
  public schedule(course: string) {}
}
```