Hack Cheat Sheet

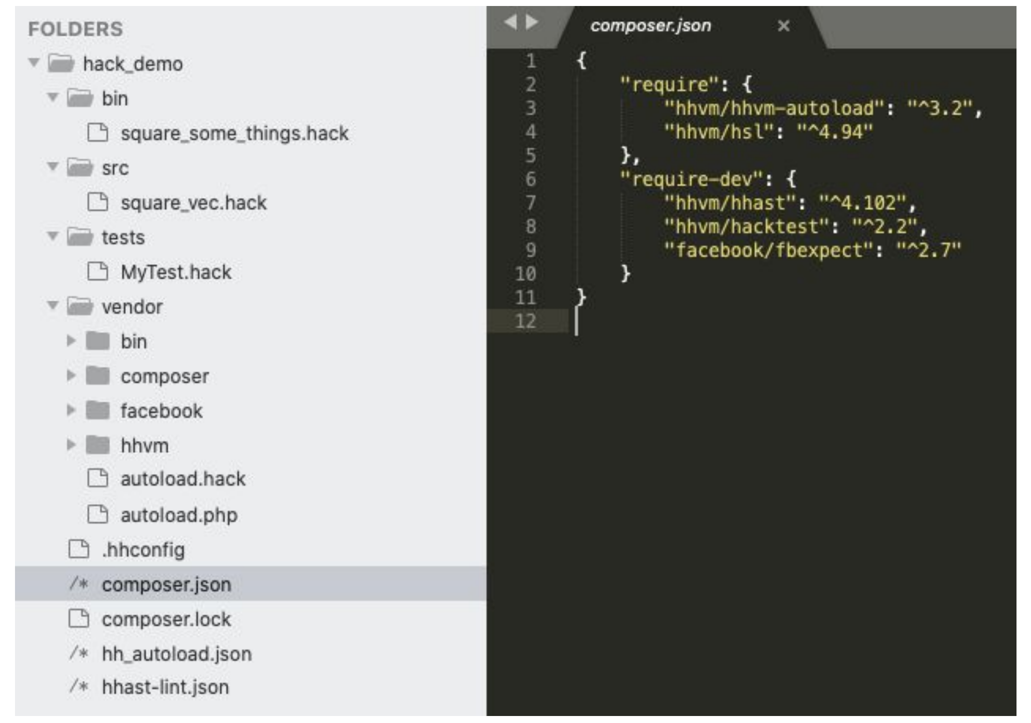
V2021.03.26

(Dr Yan Xu)

Tools

- ➔ hh_client (or hh in facebook)
  - static analysis
  - default running in some IDEs (e.g. VS-Code)
- ➔ hhvm
  - to execute Hack code, and can either be used for CLI (e.g. hhvm foo.hack) or as a server
- ➔ hackfmt
  - code formatter
- ➔ Composer
  - Dependency Management (e.g. npm)
- ➔ hhvm-autoload
  - to generate a map of what files define which classes, functions and so on for hhvm
- ➔ HackTest & expect
  - to create unit test classes
- ➔ hhvm/hsl
  - The Hack Standard Library, Str/Dict/Vec/C

Code Structure



```
FOLDERS
▾ hack_demo
  ▾ bin
      square_some_things.hack
  ▾ src
      square_vec.hack
  ▾ tests
      MyTest.hack
  ▾ vendor
    ▸ bin
    ▸ composer
    ▸ facebook
    ▸ hhvm
      autoload.hack
      autoload.php
    .hhconfig
    composer.json
    composer.lock
    hh_autoload.json
    hhast-lint.json
```

```
composer.json
{
    "require": {
        "hhvm/hhvm-autoload": "^3.2",
        "hhvm/hsl": "^4.94"
    },
    "require-dev": {
        "hhvm/hhast": "^4.102",
        "hhvm/hacktest": "^2.2",
        "facebook/fbexpect": "^2.7"
    }
}
```

Comments

```
// A single line comment.

# Also a single line comment.

/* A multi line comment.
 *
 */

/**
 * A doc comment starts with two asterisks.
 *
 * It summarises the purpose of a definition, such as a
 * function, class or method.
 */
function foo(): void {}
```

Naming

```
class IntBox {
  private int $x;

  public function __construct(int $x) {
    $this->x = $x; // Assigning to property.
  }

  public function getX(): int {
    return $this->x; // Accessing property.
  }
}

<<__EntryPoint>>
function main(): void {
  $ib = new IntBox(42);
  $x = $ib->getX(); // Calling instance method.
}
```

Script Inclusion

- ➔ The recommended way is to use an autoloader - however, need to include the autoloader itself.
  - require_once(__DIR__.'/../vendor/autoload.hack');
  - \Facebook\AutoloadMap\initialize();
- ➔ Then, you could access all functions in folder

Namespace

- ➔ A namespace is a container for a set of (typically related) classes, interfaces, traits, functions, and constants.
- ➔ In the absence of any namespace definition, the **default namespace**, which has no name, is used
- ➔ The names of the standard types that are introduced with Hack belong to namespace **HH**
- ➔ When the same namespace is defined in multiple scripts, and those scripts are combined into the same program, the namespace is considered the merger of them.
- ➔ Important scripts:
  - **namespace** NS1 {}
  - **use** NS1\{C, I, T};

Print/Echo

- ➔ Echo
  - echo can output an object with __toString()
  - Cannot output array
- ➔ Printf (formatted print)
  - printf("%d\n", $num)

Casting

```
(float)1; // 1.0
(int)3.14; // 3, rounds towards zero

(bool)0; // false
(string)new MyClass(); // calls __toString()
```

Type Assertions

- ➔ **is**
  - Checking Types
  - 'foo' is int;   // false
- ➔ **as / ?as**
  - enforcing Types

```
// Normally you'd want to make transport take a Vehicle
// directly, so you can check when you call the function.
function transport(mixed $m): void {
  // Exception if not a Vehicle.
  $v = $m as Vehicle;

  if ($v is Car) {
    $v->drive();
  } else {
    // Exception if $v is not a Boat.
    $v as Boat;
    $v->sail();
  }
}
```

- ➔ **Legacy** Type Predicates (e.g. is_int, is_bool)
  - use **is** instead
- ➔ **Legacy** instanceof
  - use **is** instead

Collections

- ➔ **Hack arrays (recommended - VALUE type)**
  - Huge functions in the C, Vec, Keyset and Dict namespaces

```
$v = vec[2, 1, 2];

$k = keyset[2, 1];

$d = dict['a' => 1, 'b' => 3];
```

- ➔ **Hack Collection (OBJECT types)**
  - Vector, Map, Set, Pair
- ➔ **PHP arrays (legacy)**
  - varray, darray
- ➔ **Vect** is dict with int as Key
- ➔ **C Namespace**
  - C\count, C\contains, C\contains_key
- ➔ **Vec**
  - Vec\concat, Vec\sort, Vec\Map, Vec\reverse
- ➔ **Dict**
  - Dict\merge
  - Vec\keys() -> keys
  - vec() -> values

```
// Converting from an Iterable.
vec(keyset[10, 11]); // vec[10, 11]
vec(Vector { 20, 21 }); // vec[20, 21]
vec(dict['key1' => 'value1']); // vec['value1']

// Type checks.
$items is vec<_>; // true
```

- ➔ **Keyset**
  - unset(), Keyset\union

```
// Converting from an Iterable.
dict(vec['a', 'b']); // dict[0 => 'a', 1 => 'b']
dict(Map {'a' => 5}); // dict['a' => 5]

// Type checks.
$items is dict<_, _>; // true
```

String Lib

Str\capitalize	Str\capitalize_words	Str\chunk	Str\compare
Str\compare_ci	Str\contains	Str\contains_ci	Str\ends_with
Str\ends_with_ci	Str\format	Str\format_number	Str\is_empty
Str\join	Str\length	Str\lowercase	Str\pad_left
Str\pad_right	Str\repeat	Str\replace	Str\replace_ci
Str\replace_every	Str\replace_every_ci	Str\replace_every_nonrecursive	Str\replace_every_nonrecursive_ci
Str\reverse	Str\search	Str\search_ci	Str\search_last
Str\slice	Str\splice	Str\split	Str\starts_with
Str\starts_with_ci	Str\strip_prefix	Str\strip_suffix	Str\to_int
Str\trim	Str\trim_left	Str\trim_right	Str\uppercase

C Lib

C\any	C\contains	C\contains_key	C\count
C\every	C\find	C\find_key	C\findx
C\first	C\first_async	C\first_key	C\first_keyx
C\firstx	C\firstx_async	C\is_empty	C\is_sorted
C\is_sorted_by	C\last	C\last_key	C\last_keyx
C\lastx	C\nfirst	C\onlyx	C\pop_back
C\pop_backx	C\pop_front	C\pop_frontx	C\reduce

Vec Lib

Vec\cast_clear_legacy_array_mark	Vec\chunk	Vec\concat	Vec\diff
Vec\diff_by	Vec\drop	Vec\fill	Vec\filter
Vec\filter_async	Vec\filter_nulls	Vec\filter_with_key	Vec\flatten
Vec\from_async	Vec\intersect	Vec\keys	Vec\map
Vec\map_async	Vec\map_with_key	Vec\partition	Vec\range
Vec\reverse	Vec\sample	Vec\shuffle	Vec\slice
Vec\sort	Vec\sort_by	Vec\take	Vec\unique
Vec\unique_by	Vec\zip		

Dict Lib

C\reduce_with_key	Dict\associate	Dict\cast_clear_legacy_array_mark	Dict\chunk
Dict\count_values	Dict\diff_by_key	Dict\drop	Dict\equal
Dict\fill_keys	Dict\filter	Dict\filter_async	Dict\filter_keys
Dict\filter_nulls	Dict\filter_with_key	Dict\filter_with_key_async	Dict\flatten
Dict\flip	Dict\from_async	Dict\from_entries	Dict\from_keys
Dict\from_keys_async	Dict\from_values	Dict\group_by	Dict\map
Dict\map_async	Dict\map_keys	Dict\map_with_key	Dict\map_with_key_async
Dict\merge	Dict\partition	Dict\partition_with_key	Dict\pull
Dict\pull_with_key	Dict\reverse	Dict\select_keys	Dict\shuffle
Dict\sort	Dict\sort_by	Dict\sort_by_key	Dict\take
Dict\unique	Dict\unique_by	Keyset\chunk	Keyset\diff

Static Type Check

- ➔ Function must define type (parameters & return values)
- ➔ Class properties must have type and initialized
- ➔ Nullable (?int, ?string)
- ➔ Type Conversion (implicit & explicit)
- ➔ Type Refinement (e.g. xxx **is** num)
- ➔ Type Inferencing (local variables)
- ➔ Hack typechecker Error
- ➔ Silencing Error Comments (HH_FIXME)

```
/* HH_FIXME[4110] Your explanation here. */
takes_int("foo");
```

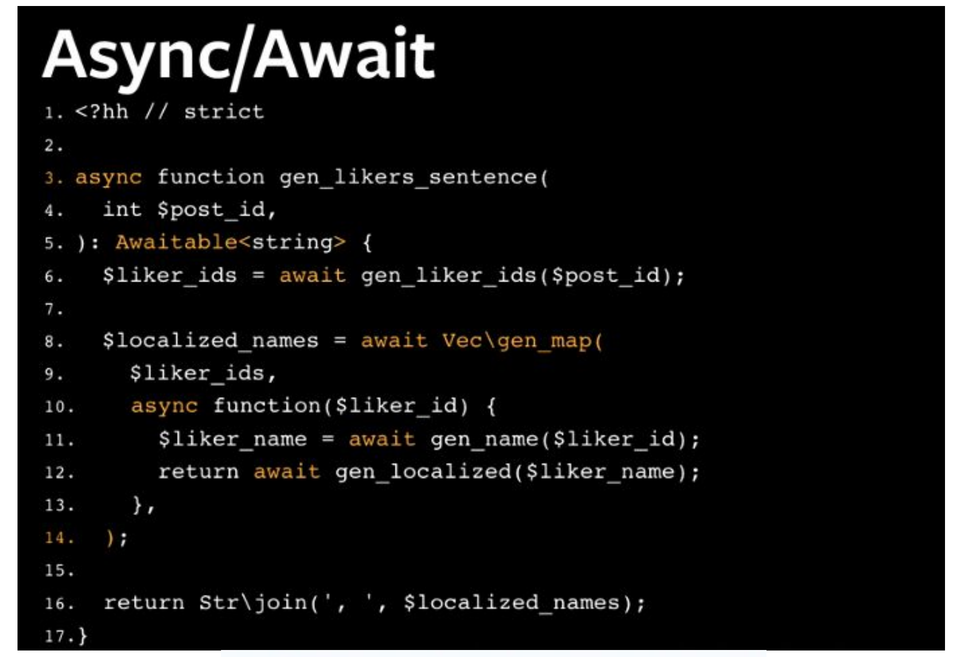
- ➔ Hack typechecker Error Codes
  - **1000 - 1999** are used for parsing errors
  - **2000 - 3999** are used for naming errors
  - **4000 - 4999** are used for typing errors
- ➔ Suppressing errors in one place can lead to runtime errors in other places.

Common Attributes

- ➔ <<__EntryPoint>>
- ➔ <<__Deprecated("mmessage", N)>>
- ➔ <<__LateInit>>
- ➔ <<__Memoize>>
  - memoization is per request
  - do not memoize funs with side impacts
- ➔ <<__Override>>

Asynchronous Operations

- ➔ Cooperative Multi-tasking Example



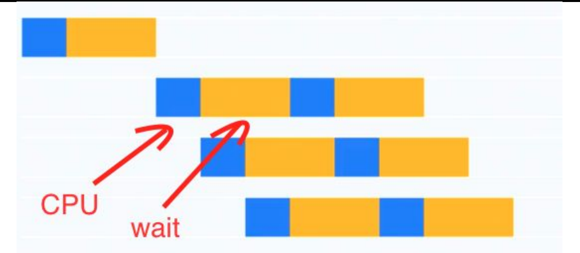
Async/Await

```
<?hh // strict

async function gen_likers_sentence(
  int $post_id,
): Awaitable<string> {
  $liker_ids = await gen_liker_ids($post_id);

  $localized_names = await Vec\gen_map(
    $liker_ids,
    async function($liker_id) {
      $liker_name = await gen_name($liker_id);
      return await gen_localized($liker_name);
    },
  );

  return Str\join(', ', $localized_names);
}
```



- ➔ Awaitables Func & Wait
  - ***Awaitables***: possibly asynchronous operation that may or may not have completed

```
async function foo(): Awaitable<int> { ... }
```

  - ***Wait***: block and let other tasks execute and only be used in an async function
    - HH\Lib\Vec\from_async
    - HH\Lib\Dict\from_async
    - \HH\Asio\join
    - ignores any successful awaitable results if one of the results was an exception.
- ➔ Do Not Use Async in Loops
- ➔ Don't Forget to Await an Awaitable
- ➔ Use Async Extensions
  - **MySQL** for database access and queries.
  - **cURL** for web page data and transfer.
  - **McRouter** for memcached-based operations.
  - **Streams** for stream-based resource operations.