TypeScript Advanced Quiz – سوالات و پاسخ‌ها (06 تا 11)

سوال 1 (پیاده‌سازی - Discriminated Union)

```
export type Success = { status: 'success'; data: string };
export type Failure = { status: 'error'; message: string };
export type APIResponse = Success | Failure;

// پیاده‌سازی تابع handleResponse که با بررسی status، داده یا پیام خطا را برگرداند.
function handleResponse(res: APIResponse): string {
 // ...
}
```

پاسخ:

```
function handleResponse(res: APIResponse): string {
 switch(res.status) {
  case 'success': return res.data;
  case 'error': return `ERR: ${res.message}`;
 }
}
```

سوال 2 (never + exhaustiveness)

```
type Shape = { kind: 'circle'; r: number } | { kind: 'square'; s: number };
function area(shape: Shape): number {
 // ...
}
```

پاسخ:

```
function area(shape: Shape): number {
 if (shape.kind === 'circle') return Math.PI * shape.r ** 2;
 if (shape.kind === 'square') return shape.s ** 2;
 const _exhaustive: never = shape;
 return _exhaustive;
}
```

سوال 3 (Union vs Intersection) - ایجاد یک نوع `Person` که Union از Admin و Member باشد. - ایجاد یک نوع `Entity` که Intersection از WithId و WithTimestamps باشد.

پاسخ:

```ts
type Admin = { role: 'admin'; permissions: string[] };
type Member = { role: 'member'; points: number };
type Person = Admin | Member;

type WithId = { id: string };
type WithTimestamps = { createdAt: Date; updatedAt: Date };
type Entity = WithId & WithTimestamps;
```

سوال 4 (Type Guards)

```ts
class Dog { bark() {} }
class Cat { meow() {} }
function speak(a: Dog | Cat) {
  // از instanceof استفاده کن
}
```

پاسخ:

```ts
function speak(a: Dog | Cat) {
  if(a instanceof Dog) a.bark();
  else a.meow();
}
```

سوال 5 (in guard)

```ts
type Fish = { swim(): void };
type Bird = { fly(): void };
function move(pet: Fish | Bird) {
  // از in استفاده کن
}
```

پاسخ:

```ts
function move(pet: Fish | Bird) {
  if('swim' in pet) pet.swim();
  else pet.fly();
}
```

سوال 6 (Custom Type Guard)

```
function isDog(x: unknown): ??? {
 // بنویس تا TypeScript بفهمد x یک Dog است
}
```

پاسخ:

```
function isDog(x: unknown): x is Dog {
 return x instanceof Dog;
}
```

سوال 7 (assert function) - نوشتن تابع assert کلید کند بررسی که K صورت این غیر در و دارد وجود شی یک در خطا بدهد.

پاسخ:

```
function assertHas<K extends PropertyKey>(obj: unknown, key: K): asserts obj is
Record<K, unknown> {
 if(typeof obj !== 'object' || obj === null || !(key in obj)) throw new
Error('Missing key');
}
```

سوال 8 (infer – ReturnType)

```
function toPair(n: number) { return [n, String(n)] as const; }
type Pair = ??? // نوع خروجی تابع را استخراج کن
```

پاسخ:

```
type MyReturnType<F> = F extends (...args: any[]) => infer R ? R : never;
type Pair = MyReturnType<typeof toPair>; // readonly [number, string]
```

سوال 9 (infer – Parameters)

```
type ToPairParams = ??? // آرگومان‌های تابع toPair را استخراج کن
```

پاسخ:

```ts
type MyParameters<F> = F extends (...args: infer P) => any ? P : never;
type ToPairParams = MyParameters<typeof toPair>; // [number]
```

سوال 10 (ArrayElement)

```ts
type AE = ??? // نوع اعضای آرایه [1,2,3] را استخراج کن
```

پاسخ:

```ts
type ArrayElement<A> = A extends readonly (infer T)[] ? T : never;
type AE = ArrayElement<readonly [1,2,3]>; // 1 | 2 | 3
```

سوال 11 (Head/Tail Tuples)

```ts
type H = Head<[1,2,3]>; // 1
type T = Tail<[1,2,3]>; // [2,3]
```

پاسخ:

```ts
type Head<T extends any[]> = T extends [infer H, ...any[]] ? H : never;
type Tail<T extends any[]> = T extends [any, ...infer R] ? R : never;
```

سوال 12 (Curry)

```ts
function add(a: number, b: number) { return a+b; }
const curriedAdd = curry(add);
curriedAdd(2)(3); // ???
```

پاسخ:

```ts
function curry<A,B,R>(fn: (a:A,b:B)=>R) { return (a:A) => (b:B) => fn(a,b); }
curriedAdd(2)(3); // 5
```

سوال 13 (Template Literal Types)

```ts
type ButtonVariant = ??? // Size-Color
```

پاسخ:

```
type Size = 'sm'|'md'|'lg';
type Color = 'red'|'blue'|'green';
type ButtonVariant = `${Size}-${Color}`;
const v: ButtonVariant = 'sm-red';
```

سوال 14 (Event Names)

```
type DOMEvents = ??? // onClick | onFocus | onBlur
```

پاسخ:

```
type EventName<K extends string> = `on${Capitalize<K>}`;
type DOMEvents = EventName<'click'|'focus'|'blur'>; // 'onClick' | 'onFocus' |
'onBlur'
```

سوال 15 تا 40 - شامل ترکیبی از: Union/Intersection، نگهبان‌های نوع سفارشی، پیچیده، infer، پیشرفته TypeScript و ترکیب توابع، curry، assert، never/exhaustiveness، template literal. هر سوال شامل کد می‌شود. - پاسخ کامل همراه توضیح آورده شده است. پیچیده است که باید پیاده‌سازی یا تکمیل شود. TypeScript ناقص یا

توضیح: - این 40 سوال در PDF شده شامل همه موضوعات 06 تا 11، به صورت تصادفی و چالش‌برانگیز هستند و آماده تست استخدامی می‌باشند.