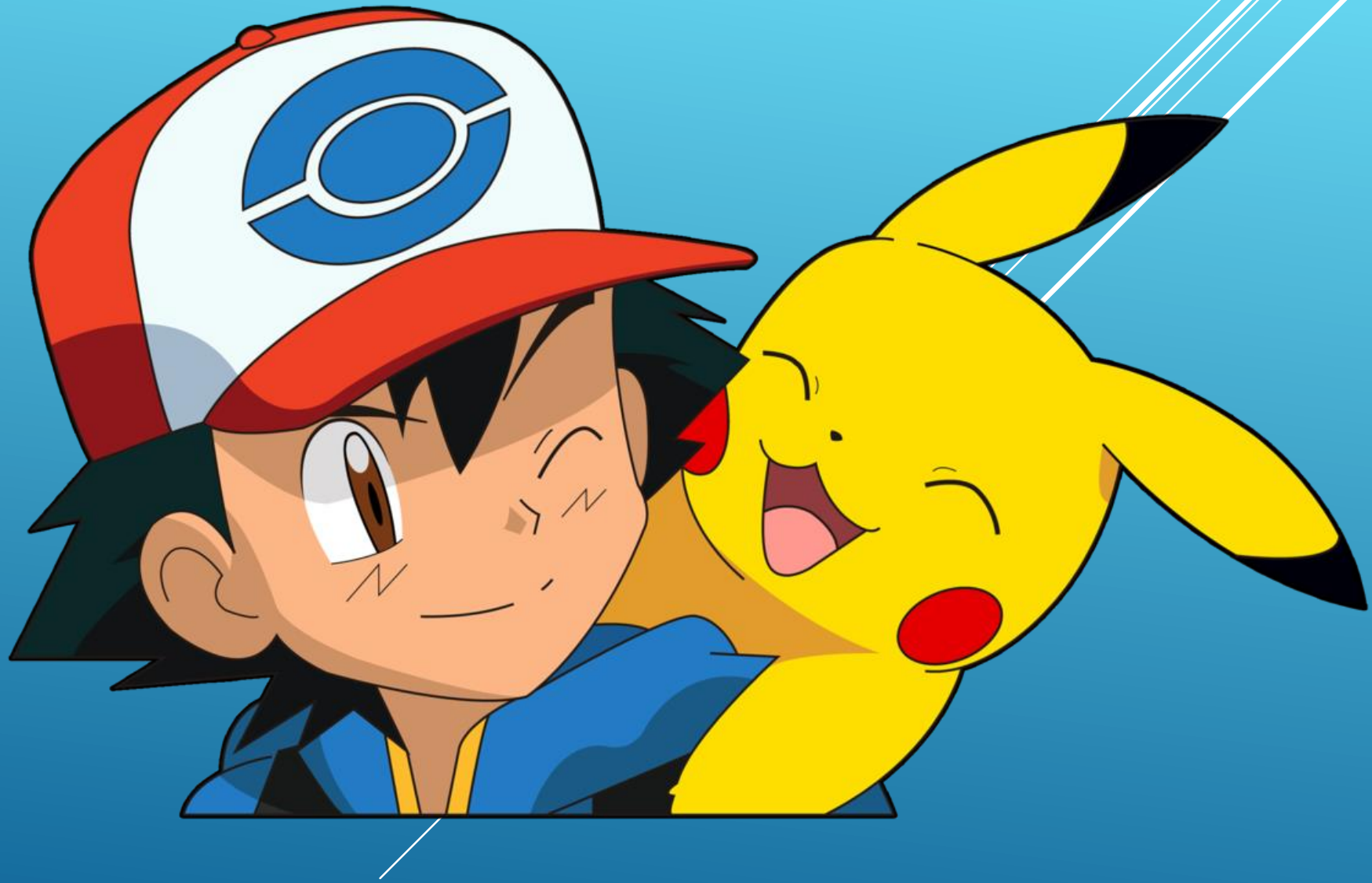
POKEBOT



- Listar escolhas possíveis:
- Listar todos os movimentos possíveis com o pokemon atual;
- Listar todos os pokemons vivos que não sejam o atual;
- Biblioteca de apoio: underscore (_)

```
function escolhasPossiveis(estado)  {
    var escolhas = [];
    _.each(estado.self.active.moves, function(move) {
                if (!move.disabled) {
                    escolhas.push({
                        "type": "move",
                        "id": move.id
                    });
                }
            });

    var trapped = (estado.self.active) ? (estado.self.active.trapped || estado.self.active.maybeTrapped) : false;
    var troca = estado.forceSwitch || !trapped;

    if (troca) {
        _.each(estado.self.reserve, function(pokemon, index) {
                if (pokemon.condition.indexOf("fnt") < 0 && !pokemon.active) {
                    escolhas.push({
                        "type": "switch",
                        "id": index
                    });
                }
            });
        }

        return escolhas;
};
```

- A partir das escolhas possíveis, escolher qual a melhor decisão:
- Para isso, definir prioridades:
- Movimentos sempre tem a preferência, visto que "perdemos" um turno caso seja efetuada a troca de pokemón
- Procurar o melhor movimento, dando preferencia a movimentos que enfraqueçam o adversário, movimentos que afetem o campo de batalha e movimentos que causem dano por segundo
- Após estes, movimentos que tem prioridade são os de cura (se o HP estiver abaixo de 50%) e movimentos com bonificação
- Se nenhum destes for listado, segue para a troca de pokemóns

- Prioridade maior para pokemóns que resistem ao tipo do pokemón adversário
- Procurar o melhor pokemón contra aquele adversário talvez não seja a estratégia mais eficaz, visto que se o oponente efetua a troca, invalida nossa escolha

- Após defini-las, escolhe a ação com maior prioridade
- Retorna para a função principal um vetor com o tipo (move ou switch), o id (“numero” do pokemon ou nome do movimento) e a prioridade

```
function tomarDecisao(estado, escolhas){
    var e = _.max(escolhas, function(opt) {
        var p = getP(estado, opt);
        opt.prior = p;
        return p;
    });

    return {
        type: e.type,
        id: e.id,
        prior: e.prior
    };
}
```

- Para checar a eficácia do movimento ou do Pokemon contra o adversário, usamos a função compare do typechart disponibilizado:

```
//retorna verdadeiro se o alvo é imune
function checkImune(source, target) {
    var myType = source.type || source;
    var targetType = target.getTypes && target.types || target.types || target;

    if (Array.isArray(targetType)) {
        for (var i = 0; i < targetType.length; i++) {
            if (!checkImune(myType, targetType[i])) return true;
        }
        return false;
    }
    var resMove = Typechart.compare(myType, targetType);
    if (resMove == 0) return true;
    return false;
};

//retorna 1 se superefetivo, -1 se não muito efetivo e 0 se for dano normal;
function checkEffect(source, target) {

        var myType = source.type || source;
        var totalPrior = 0;
        var targetType = target.getTypes && target.types || target.types || target;

        if (Array.isArray(targetType)) {
            for (var i = 0; i < targetType.length; i++) {
                totalPrior += checkEffect(myType, targetType[i]);
            }
            return totalPrior;
        }



        var resMove = Typechart.compare(myType, targetType);
        if (!resMove) return 0;
        if (resMove == 0.5) return -1;
        else if (resMove == 2) return 1;
        else return 0;
    };
```