Questão 01.

function EquacaoSegundoGrau($floatCoefA, $floatCoefB, $floatCoefC) {

$floatDelta = pow($floatCoefB, 2) - (4 \* $floatCoefA \* $floatCoefC);

if ($floatDelta < 0){

return ('Não possui raiz real, pois delta é menor que 0');

}

else if ($floatDelta == 0){

$floatRaiz = (- $floatCoefB) / (2 \* $floatCoefA);

return ('A raiz da equação é '.$floatRaiz);

}

else{

$floatRaizPos = ((- $floatCoefB) + sqrt($floatDelta)) / (2 \* $floatCoefA);

$floatRaizNeg = ((- $floatCoefB) - sqrt($floatDelta)) / (2 \* $floatCoefA);

return ('As raizes da equação são '.$floatRaizPos.' e '.$floatRaizNeg);

}

}

Questão 02

 Em Javascript a comparação de valores não é limitada pelo tipo da variável, ou seja, é possível comparar uma string com inteiro por exemplo.  
Então utilizando "==" você ignora o tipo da variável na comparação.  
Já utilizando "===" você não ignora o tipo da variável, retornando "false" em caso de tipos diferentes.  
A função abaixo pode ser utilizada para ver a diferença.  
function Questao02 (param0, param1) {  
  var Resposta = "";  
  if(param0 == param1) {  
  Resposta += "Para a comparação com '==' os valores são iguais.";  
  }  
  else {  
    Resposta += "Para a comparação com '==' os valores são diferentes.";  
   }  
   if(param0 === param1) {  
     Resposta += " Para a comparação com '===' os valores são iguais.";  
   }  
   else {  
     Resposta += " Para a comparação com '===' os valores são diferentes.";  
   }  
   return(Resposta);  
}  
Exemplo:   
Questao02(5, "5") retornará "Para a comparação com '==' os valores são iguais. Para a comparação com '===' os valores são diferentes."

Questão 03

class Veiculo {

private $stringPlaca;

private $stringCor;

private $stringModelo;

private $intAno;

private $intTipo;

private $dateTimeEntrada;

function \_\_construct($StringPlaca, $StringCor, $StringModelo, $IntAno){

$this->stringPlaca = $StringPlaca;

$this->stringCor = $StringCor;

$this->stringModelo = $StringModelo;

$this->intAno = $IntAno;

}

// Funções para bsucar valores

function getPlaca(){

return ($this->stringPlaca);

}

function getCor(){

return ($this->stringCor);

}

function getModelo(){

return ($this->stringModelo);

}

function getAno(){

return ($this->intAno);

}

function getTipo(){

return ($this->intTipo);

}

function getEntrada(){

return ($this->dateTimeEntrada);

}

// Funções para setar valores

function setPlaca($StringPlaca){

$this->stringPlaca = $StringPlaca;

}

function setCor($StringCor){

$this->stringCor = $StringCor;

}

function setModelo($StringModelo){

$this->stringModelo = $StringModelo;

}

function setAno($IntAno){

$this->intAno = $IntAno;

}

function setTipo($IntTipo){

$this->intTipo = $IntTipo;

}

function setEntrada($DateTimeEntrada){

$this->dateTimeEntrada = $DateTimeEntrada;

}

}

class Carro extends Veiculo {

private $boolIsPickup;

function \_\_construct($StringPlaca, $StringCor, $StringModelo, $IntAno){

parent::\_\_construct($StringPlaca, $StringCor, $StringModelo, $IntAno);

$this->setTipo(0);

$this->boolIsPickup = false;

}

// Funções para bsucar valores

function getIsPickup(){

return ($this->boolIsPickup);

}

// Funções para setar valores

function setIsPickup($BoolIsPickup){

$this->boolIsPickup = $BoolIsPickup;

}

}

class Moto extends Veiculo {

private $boolIsTriciculo;

function \_\_construct($StringPlaca, $StringCor, $StringModelo, $IntAno){

parent::\_\_construct($StringPlaca, $StringCor, $StringModelo, $IntAno);

$this->setTipo(1);

$this->boolIsTriciculo = false;

}

// Funções para bsucar valores

function getIsTriciculo(){

return ($this->boolIsTriciculo);

}

// Funções para setar valores

function setIsTriciculo($BoolIsTriciculo){

$this->boolIsTriciculo = $BoolIsTriciculo;

}

}

class Estacionamento {

private $stringNome;

private $floatPrecoCarro;

private $floatPrecoMoto;

private $intVagasCarros;

private $intVagasMotos;

private $arrayCarros;

private $arrayMotos;

function \_\_construct($StringNome, $FloatPrecoCarro, $FloatPrecoMoto, $IntVagasCarros, $IntVagasMotos) {

$this->stringNome = $StringNome;

$this->floatPrecoCarro = $FloatPrecoCarro;

$this->floatPrecoMoto = $FloatPrecoMoto;

$this->intVagasCarros = $IntVagasCarros;

$this->intVagasMotos = $IntVagasMotos;

}

// Funções para bsucar valores

function getNome(){

return ($this->stringNome);

}

function getPrecoCarro(){

return ($this->floatPrecoCarro);

}

function getPrecoMoto(){

return ($this->floatPrecoMoto);

}

function getQtdCarros(){

return ($this->intVagasCarros);

}

function getQtdMotos(){

return ($this->intVagasMotos);

}

// Funções para setar valores

function setNome($StringNome){

$this->stringNome = $StringNome;

}

function setPrecoCarro($FloatPrecoCarro){

$this->floatPrecoCarro = $FloatPrecoCarro;

}

function setPrecoMoto($FloatPrecoMoto){

$this->floatPrecoMoto = $FloatPrecoMoto;

}

function setQtdCarros($IntVagasCarros){

$this->intVagasCarros = $IntVagasCarros;

}

function setQtdMotos($IntVagasMotos){

$this->intVagasMotos = $IntVagasMotos;

}

// Funções Utilitarias

function CalcularValorFinal($ObjVeiculo, $DateTimeFinal){

$dateTimeInicial = $ObjVeiculo->getEntrada();

$diferenca = $dateTimeInicial->diff($DateTimeFinal);

$minutos = (($diferenca->h + ($diferenca->days \* 24)) \* 60) + $diferenca->m;

$total = ($minutos \* $this->floatPrecoCarro) / 60;

return (abs($total));

}

function EntrarVeiculo($ObjVeiculo, $DateTimeMomento){

$alocado = false;

if ($DateTimeMomento !== null){

$ObjVeiculo->setEntrada($DateTimeMomento);

}

else{

$ObjVeiculo->setEntrada(now());

}

if ($ObjVeiculo->getTipo() == 0){ // 0 = Carros

for ($i = 0; $i < count($this->arrayCarros); $i++){

if ($this->arrayCarros[$i] === null){

$this->arrayCarros[$i] = $ObjVeiculo;

$alocado = true;

break;

}

}

if((!$alocado) && (count($this->arrayCarros) < $this->intVagasCarros)){

$this->arrayCarros[] = $ObjVeiculo;

$alocado = true;

}

}

else if ($ObjVeiculo->getTipo() == 1){ // 1 = Motos

for ($i = 0; $i < count($this->arrayMotos); $i++){

if ($this->arrayMotos[$i] === null){

$this->arrayMotos[$i] = $ObjVeiculo;

$alocado = true;

break;

}

}

if((!$alocado) && (count($this->arrayMotos) < $this->intVagasMotos)){

$this->arrayMotos[] = $ObjVeiculo;

$alocado = true;

}

}

return ($alocado);

}

function SairVeiculo($IdVeiculo, $TipoVeiculo, $DateTimeMomento){

$floatValorFinal = 0;

$dateTimeMomento = now();

if ($DateTimeMomento !== null){

$dateTimeMomento = $DateTimeMomento;

}

if ($TipoVeiculo == 0){ // 0 = Carros

if (($IdVeiculo > 0) && ($IdVeiculo < count($this->arrayCarros))){

$floatValorFinal = CalcularValorFinal(($this->arrayCarros[$IdVeiculo]), $dateTimeMomento);

$this->arrayCarros[$IdVeiculo] = null;

}

}

else if ($TipoVeiculo == 1){ // 1 = Motos

if (($IdVeiculo > 0) && ($IdVeiculo < count($this->arrayMotos))){

$floatValorFinal = CalcularValorFinal($this->arrayMotos[$IdVeiculo], $dateTimeMomento);

$this->arrayMotos[$IdVeiculo] = null;

}

}

return ($floatValorFinal);

}

}

}

Questão 04

Retorno da função é 42;

Questão 05

O Node.Js historicamente era conhecido por não utilizar a tecnologia de threads, mas recentemente a partir da versão 10.5.0 foi adicionado o suporte a threads.  
O objetivo aparentemente é melhorar o desempenho nas aplicações que envolvem "machine learning", "inteligencia artificial" e "data science", pois exigem mais da CPU.  
Para os outros casos ainda é recomendado a utilização do "event loop" e do "async I/O".

Questão 06 e 07

Código fonte e o demonstrativo já estão disponíveis.