Prototypes

Explain the role of the prototype property via an example! (5 points)

**Answer:** JavaScript functions defined using the ES5 syntax have prototypes. (1 point)

Remark: ES6 arrow functions don’t have prototypes. Methods defined using the concise method syntax don’t have prototypes.

These prototypes become important once a function is used for instantiation. In JavaScript terminology, these functions are *constructor functions*. (1 point)

A prototype may contain functions that are available in every instance of created by the constructor functions. (1 point)

Example:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | function Wallet() {      this.amount = 0;  }    Wallet.prototype.deposit = function( amount ) {      this.amount += amount;  }  Wallet.prototype.withdraw = function( amount ) {      if ( this.amount >= amount ) {          this.amount -= amount;      } else {          throw 'Insufficient funds.';      }  }    let myWallet = new Wallet();  myWallet.deposit( 100 );  myWallet.amount  > 100 |

rototypal inheritance

Extend your example from question 3 to demonstrate prototypal inheritance! (5 points)

**Answer:**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | function BoundedWallet( maxAmount ) {      Wallet.call( this );               // 1 point      this.maxAmount = maxAmount;        // 1 point  }    BoundedWallet.prototype = Object.create( Wallet.prototype ); // 1 point  BoundedWallet.prototype.constructor = BoundedWallet;         // 1 point    BoundedWallet.prototype.deposit = function( amount ) {      if ( this.amount + amount > this.maxAmount ) {          throw 'Insufficient wallet capacity';      }      Wallet.prototype.deposit.call( this, amount );  // this.amount += amount;  } // 1 point |

One construct worth mentioning is Wallet.prototype.deposit.call( this, amount );. Here instead of writing this.amount += amount;, it is semantically more correct to reuse the deposit functionality of the base class. In ES5, this is the way to go.

Although the rest of the code is not self-explanatory, you can find an explanation for a very similar example in [Chapter 4: Classes of ES6 in Practice](https://leanpub.com/es6-in-practice). You can access this chapter even in the free sample.

### ES6 classes

Use the ES6 class syntax to rewrite the code you wrote in questions 3 and 4. (7 points)

**Answer:**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | class Wallet {                             // 1 point      constructor() { this.amount = 0; }     // 1 point      deposit( amount ) { this.amount += amount; }      withdraw( amount ) {          if ( this.amount >= amount ) {              this.amount -= amount;          } else {              throw 'Insufficient funds.';          }      }                                      // 1 point  }    class BoundedWallet extends Wallet {       // 1 point      constructor( maxAmount ) {          super();                           // 1 point          this.maxAmount = maxAmount;      }      deposit( amount ) {          if ( this.amount + amount > this.maxAmount ) {              throw 'Insufficient wallet capacity';          }          super.deposit( amount );      }                                      // 1 point  }    let myWallet = new Wallet();  myWallet.deposit( 100 );  myWallet.amount                             // 1 point  > 100 |

You need to demonstrate the following six items for a complete solution:

* class keyword, class name, and braces to define a constructor function
* a properly working constructor using the concise method syntax
* at least one method using the concise method syntax
* proper usage of the extends keyword
* proper usage of super in the constructor of the child class
* at least one redefined method. It is optional to access the shadowed base class method using super. In this example, super.deposit( amount ); accessed the deposit method of the Wallet class
* some code demonstrating instantiation, which should be unchanged compared to the ES5 prototypal inheritance syntax