If contract MyContractA is derived from Contract MyContractB, then this would be the right syntax:

a) contract MyContractA is MyContractB { … }
b) contract MyContractA inherit (MyContractB) {…}
c) contract MyContractA extends MyContractB {…}
d) contract MyContractB derives MyContractA {…}

Inhertiance is useful, because a contract that is derived from another contract can make use of:

a) all public state variables and properties, public and internal functions and modifiers
b) all public and private state variables, public, internal and external functions, but not modifiers
c) all public state variables and properties, public functions and modifiers, but not internal, external or private ones.

Finish the sentence: The Library Web3.js is …

a) useful when developing distributed applications with HTML and JavaScript, because it already implements the abstraction of the JSON-RPC interface of Ethereum Nodes.
b) necessary when developing distributed applications with HTML and JavaScript, because the proprietary JSON-RPC interface of Ethereum Nodes is closed source
c) a great way to start with a boilerplate distributed application. Web3.js give you a lot of options to start either with React or Vue.js apps.

When solidity is compiled then also metadata is generated

a) The Metadata contains the ABI Array, which defines the Interface to interact with the Smart Contract. Metadata can also contain the address of the smart contract when it gets deployed.
b) Metadata contains the address, and the size of the smart contract. The ABI Array is generated externally upon deploying the smart contract
c) The ABI array and the Metadata are not generated when solidity is compiled to bytecode, its generated by a migration software which deploys the smart contract on the blockchain.

The difference between address.send() and address.transfer() is

a) .send returns a Boolean and .transfer throws an exception on error. Both just forward the gas-stipend of 2300 gas and are considered safe against re-entrancy.
b) .send throws an exception and .transfer returns a Boolean on error. Both just forward the gas-stipend of 2300 gas and considered safe against re-entrancy.
c) .send returns a Boolean and .transfer throws an exception on error. .send is considered dangerous, because it sends all gas along, while .transfer only sends the gas stipend of 2300 gas along.
d) .send and .transfer are both considered low-level functions which are dangerous, because they send all gas along. It’s better to use address.call.value()() to control the gas-amount.

All low-level functions on the address, so address.send(), address.call.value()(), address.callcode and address.delegatecall

a) Are interrupting execution on error, because they throw an exception
b) Continuing execution on error silently, which is the reason why they are so dangerous.
c) Returning Booleans to indicate an error during execution
d) .send() throws an exception, while the other functions are returning Booleans during execution to indicate an error.

When using assert to check invariants and it evaluates to false

a) All gas is consumed
b) All remaining gas is returned

When using require to check input parameters and it evaluates to false

a) All gas is consumed
b) All remaining gas is returned

To send ether to a contract without a function call:

a) A fallback function must be declared and it must be made payable. If there is no fallback function or the fallback function is not payable it will throw an exception.
b) Either a fallback function which is payable exists, or no fallback function at all exists.
c) You cannot send ether to a contract without explicitly calling a function. The fallback function can never receive ether.

Using selfdestruct(beneficiary) with the beneficiary being a contract without a payable fallback function:

a) Will throw an exception, because the fallback function is non-payable and thus cannot receive ether
b) It's impossible to secure a contract against receiving ether, because selfdestruct will always send ether to the address in the argument. This is a design decision of the Ethereum platform.
c) Selfdestruct doesn't send anything to a contract, it just re-assigns the owner of the contract to a new person. Sending ether must be done outside of selfdestruct.

If you need more fine-grained functionality than solidity offers out of the box

a) You can incorporate inline-assembly to get better controls
b) You have to import pre-compiled assembly files which are then hard-copied into the bytecode of the compiled solidity file
c) You can use Viper, the experimental assembly like language specifically to offer more flexibility

.Call vs. Delegatecall:

a) Address.call() is used for calling other contracts using the scope of the called contract in terms of storage variables. Address.delegatecall() is used for libraries, which uses the

storage variables of the contract who called. Libraries are a great way to re-use already existing code and delegatecall can make sure that no storage is used from the library, instead it looks like the code is directly copied into the calling contract.

b) Address.delegatecall() is used for calling other contracts using the scope of the called contract in terms of storage variables. Address. call() is used for libraries, which uses the storage variables of the contract who called. Libraries are a great way to re-use already existing code and call() can make sure that no storage is used from the library, instead it looks like the code is directly copied into the calling contract.