# General architecture

C:\Users\stijn.koopal\AppData\Local\Microsoft\Windows\INetCache\Content.Word\architecture-physical (1).png

## Json-RPC

One way of communicating to the Ethereum network, and thereby communicating to smart contracts, is through the JSON-RPC protocol (<http://www.jsonrpc.org/specification>) over HTTP.

This protocol is based on the lightweight JSON format and it’s used to remotely invoke procedures on the exposing node, hence the name Remote Procedure Call (RPC). The underlying JSON format is of little importance, however the JavaScript Ethereum implementation of JSON-RPC, named *web3*, is.

## Web3

This is the Ethereum compatible JavaScript API which implements the JSON-RPC spec. It comprises library functions to get Ethereum node specifications, network details and contract details. Furthermore, it provides means to invoke contract functions and utility functions.

The web3 library is packaged into the web application, and it is therefore served with every request to the website.

The Metamask plugin for Chrome enhances Chrome with means to communicate to the *web3* instance of a given website. See a more detailed description below.

## Metamask

Metamask (<https://metamask.io/>) is a Chrome plugin that injects Ethereum authentication into the browser. That is, for every website that uses web3, Metamask will inject details of the authenticated Ethereum user to the website.

The injection principle provided by Metamask is very simple. Once installed, it will provide a *web3* object into the *window* object of the browser. Something similar to this will be seen in the source code of the website:

if (typeof web3 !== 'undefined') {   
 window.web3 = new Web3(web3.currentProvider);  
 window.web3.eth.defaultAccount = web3.eth.accounts[0];  
}

## Solidity

Solidity is a contract-oriented, high-level language whose syntax is similar to that of JavaScript and it is designed to target the Ethereum Virtual Machine (EVM).

Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.

# DTAP

C:\Users\stijn.koopal\AppData\Local\Microsoft\Windows\INetCache\Content.Word\otap.png

Figure 1 DTAP for DApps

Any application that is worth mentioning has something called a DTAP street: Development-Test-Acceptance-Production. For DApps something similar is in place—see Figure 1.

Development and testing can be done on a Sandbox environment. This fully configurable environment can be started and stopped on request. A sandbox is provided by Ethereum Camp (<https://live.ether.camp/eth-studio>). Any exchange with the Sandbox is instant. That is, every action you do a contract is executed immediately, in contrast to BlockChain where actions are mined.

Multiple Ethereum test environments are available. On test environments it is possible to acquire as much Ether as you want, but the environments are not configurable. They look a lot like the Main network, but it does not operate on real Ether.

# Javascript design principles

## ES2016

Vanilla JavaScript currently supported by all major browsers currently is ECMAScript 5. In order to work with Promises, lambda’s are essential. For this reason, our JavaScript is transpiled from ECMAScript 2016.

## Promises

The *web3* API allows for two programming models: synchronously, and asynchronously. While the former one provides easier access to the Ethereum Network, it will freeze any application due to the asynchronous nature of the blockchain. Hence, the asynchronous programming model provides the best fit for dealing with the blockchain.

The asynchronous API provided by *web3* is based on callbacks and therefore provides a suboptimal solution. (Why you say? <http://callbackhell.com/>). We chose to add some syntactic sugar to the API in order to support Promises.

## Solidity Events

Any state change to a contract can be distributed to connected applications by means of Events. The contract can throw events that can be catched by the application. We use this principle in order to refresh frontends whenever the contract state is changed.