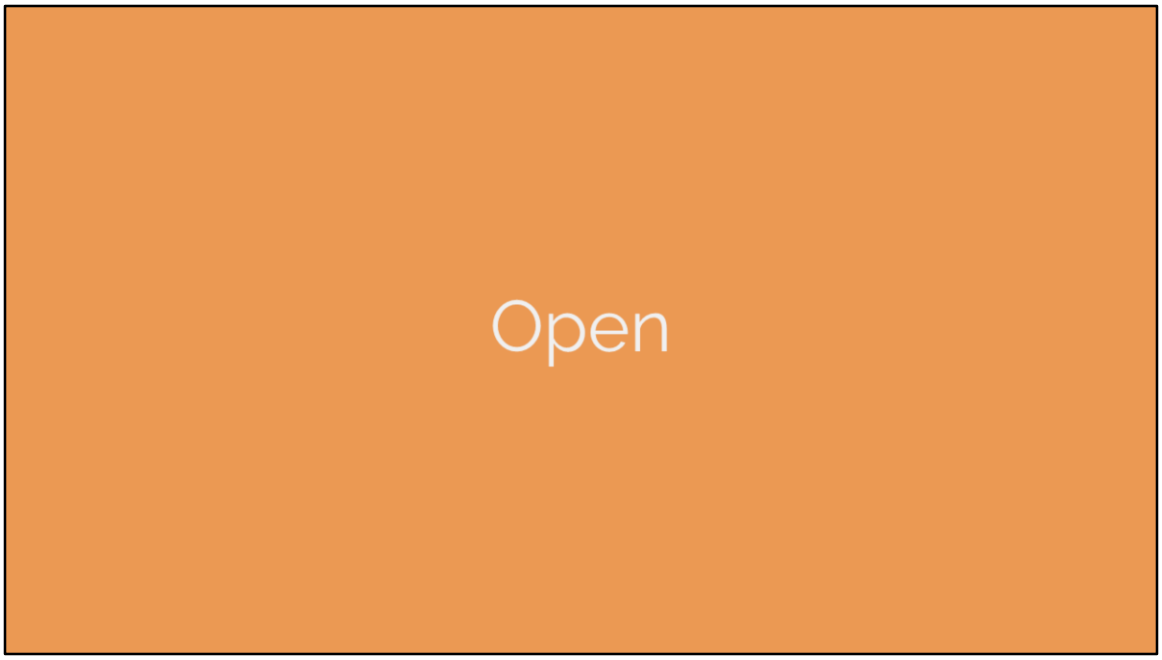
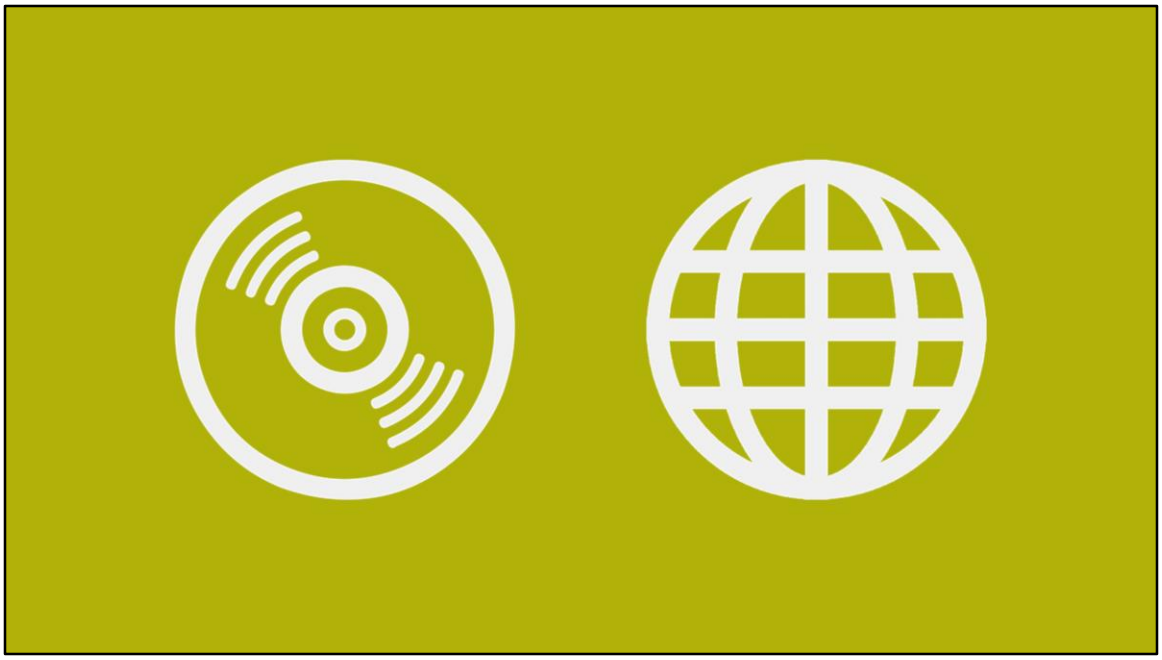
VOORSTELLEN

Goedemorgen, Mijn naam is Robin. **Twee jaar geleden** zat ik net als jullie in het publiek toen Jan Rauter met Jan Wielemans een presentatie over Angular kwam geven op deze hogeschool. Ik herriner mij nog goed dat ik daarna zo into Angular was en er direct mee aan de slag wou. Constant had ik zo iets in de trend van **SHOW ME THE CODE.** Toen had ik het vooral voor de code maar meer en meer komt de **kadering** erbij die zoveel meer waarde heeft dan de zoveelste hello world applicatie. Ik hoop vandaag **hetzelfde** bij jullie **los te weken** als ik toen ervaren had enkel nu met een nieuw topic, Meteor. Temper dus nog even het “SHOW ME THE CODE” gevoel tot na de middag. En geniet van de kadering. Mischien nog even kort rondvragen. Wie heeft er al eens van **Meteor gehoord**? (Een paar? Wel voor de rest. Meteor is Javascript.) En wie kent javascript nu niet.aag vertellen **waarom ik geloof dat Meteor** een tool is waarmee je snel business waarde kan scheppen maar eerst even dieper ingaan op de fundamentele **bouwstenen** die Meteor Mogelijk maken. Startende bij **javascript**.

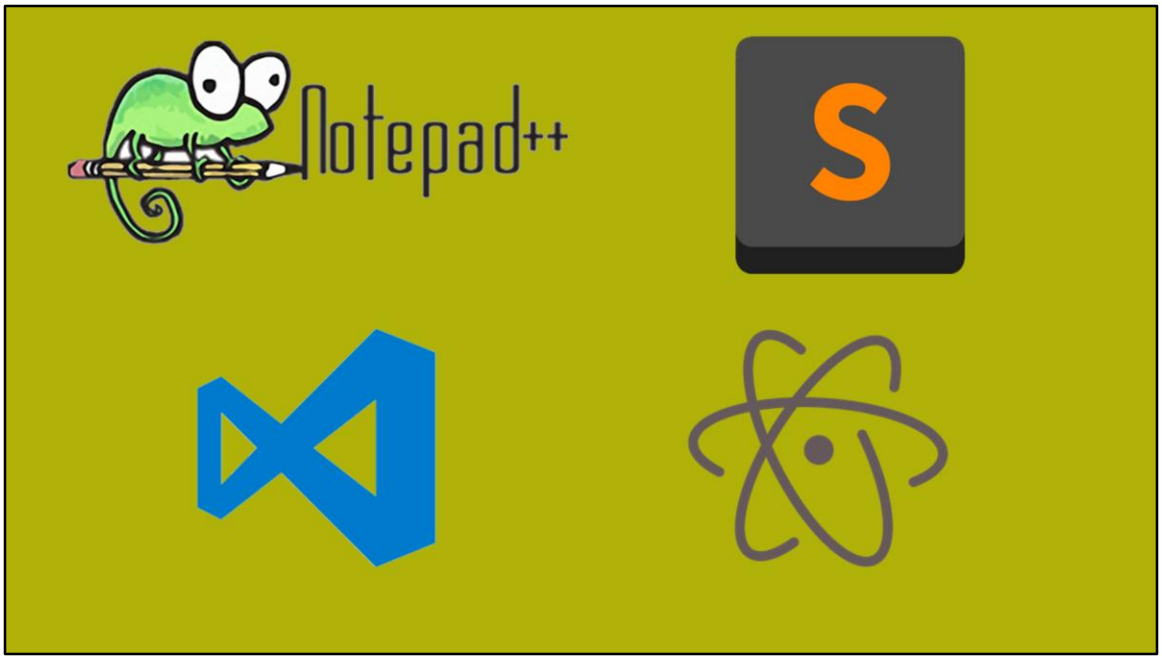
Waarom nu juist javascript en niet één of andere taal zoals C#, Java... Javascript heeft al wat **geschiedenis** achter de kiezen van in de begindagen waar javascript gebruikt werd als **animator van het web** tot het bouwen van **gigantische javascript applicaties** zoals een Facebook. Nu wat drijft onder andere het **success** van Javascript?



[Open] Javascript is heel open platform dat zeer **toegangkelijk** is voor iedere developer.



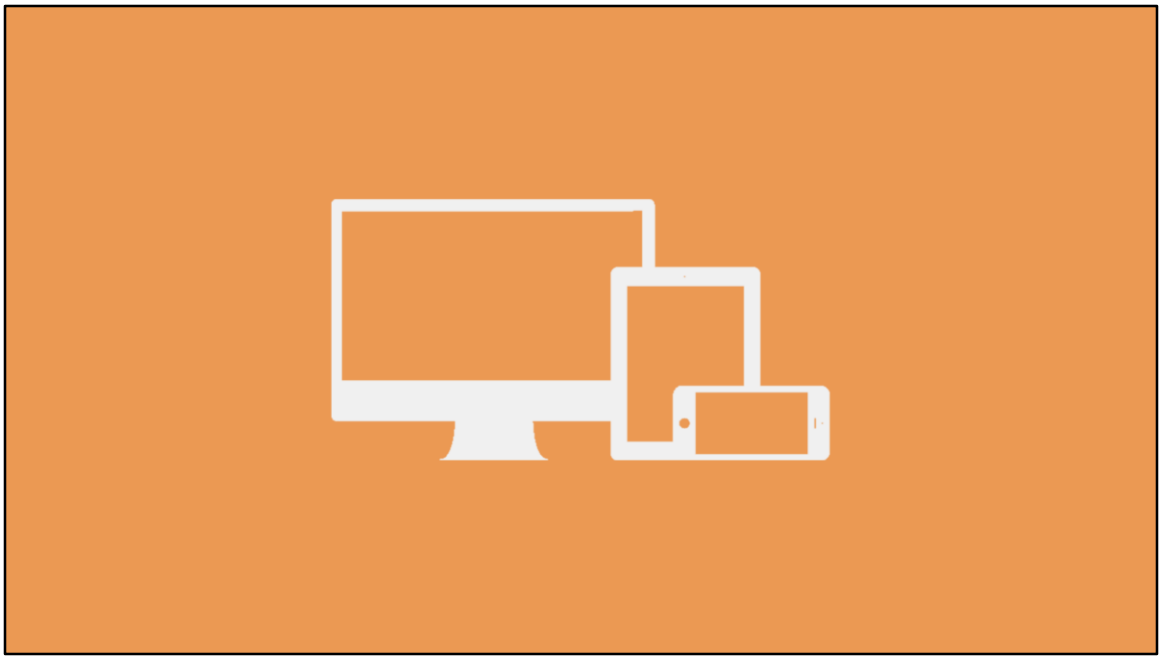
[Distributie] Het **internet** heeft Javascript als programeertaal een enorme boost gegeven. Vroeger werden **CD's in thick clients** gestoken die dan dat programma konden draaien. Nu surfen wij naar een **website** en brengt javascript ons diezelfde thick client ervaring op veel verschillende devices. CD's waren vroeger ook het medium waarop programmas werden verkocht. In een zekere zin zou je zelfs kunnen stellen dat Javascript een heel business model heeft onderuit gehaald. Door het web zijn **applicaties steeds meer connected** en kunnen deze zeer snel tot de gebruiker gebracht worden. Wat is dan juist het **voordeel** als je zegt van ah ja u software wordt op een andere manier verdeeld? Niemand hoeft te wachten tot de cd's klaar zijn. **Distributiekosten** dalen zeer hard. De nood voor een **dikke computer** met bepaalde hardware om je programma te draaien verdwijnt volledig. Je hebt enkel een **browser** nodig.



[Texteditor] Om javascript te schrijven heb je enkel een **text editor** en een **browser** nodig. Dit maakt het een eenvoudige instap om te programmeren. En door een gigantisch aantal mogelijkheden in **tooling** ishet mogelijk het grootste javascript project ooit tot een goed einde te brengen. Editors zoals visual studio code, atom, sublime, ... Maken het leven echt eenvoudig.

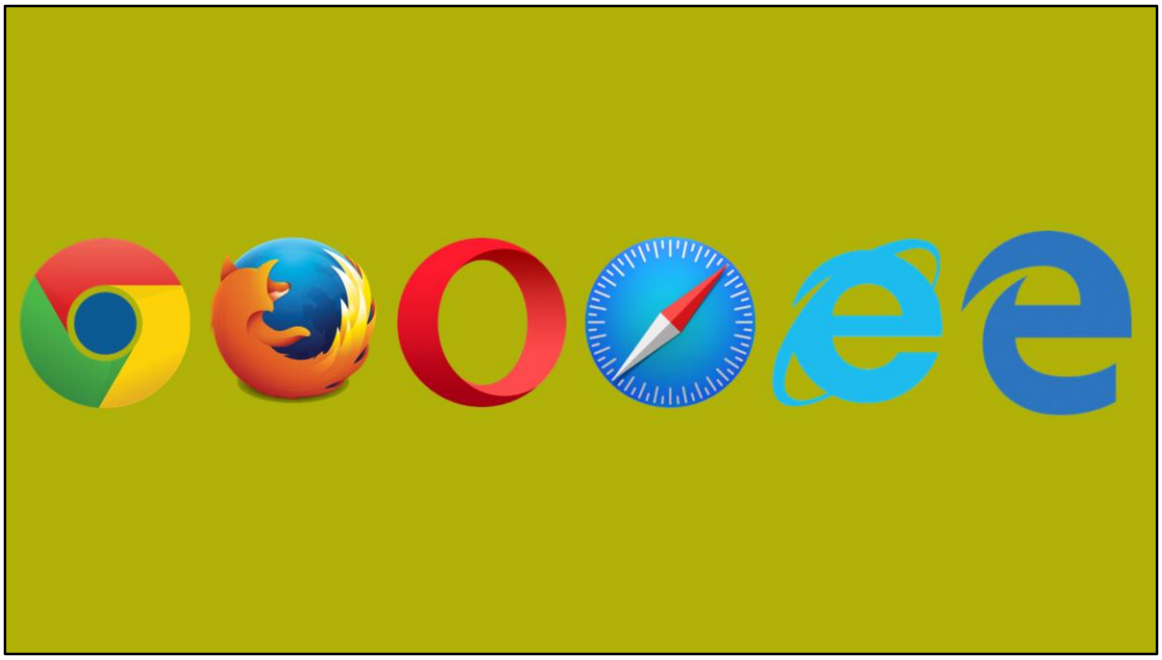
[Community] Zo open dat er al bijna **twee komma twee miljoen opensource** projecten op github zijn onstaan waarbij javascript de drijfveer is. Dat tegenover de bijna 2 miljoen voor **java** en 500 000 voor **C#**

[Community] een **dik miljoen** aantal vragen op stackoverflow waaruit blijkt dat javascript **populair** is en dat er een echte **communitie** is. In vergelijking hebben C# en Java rond een miljoen vragen.

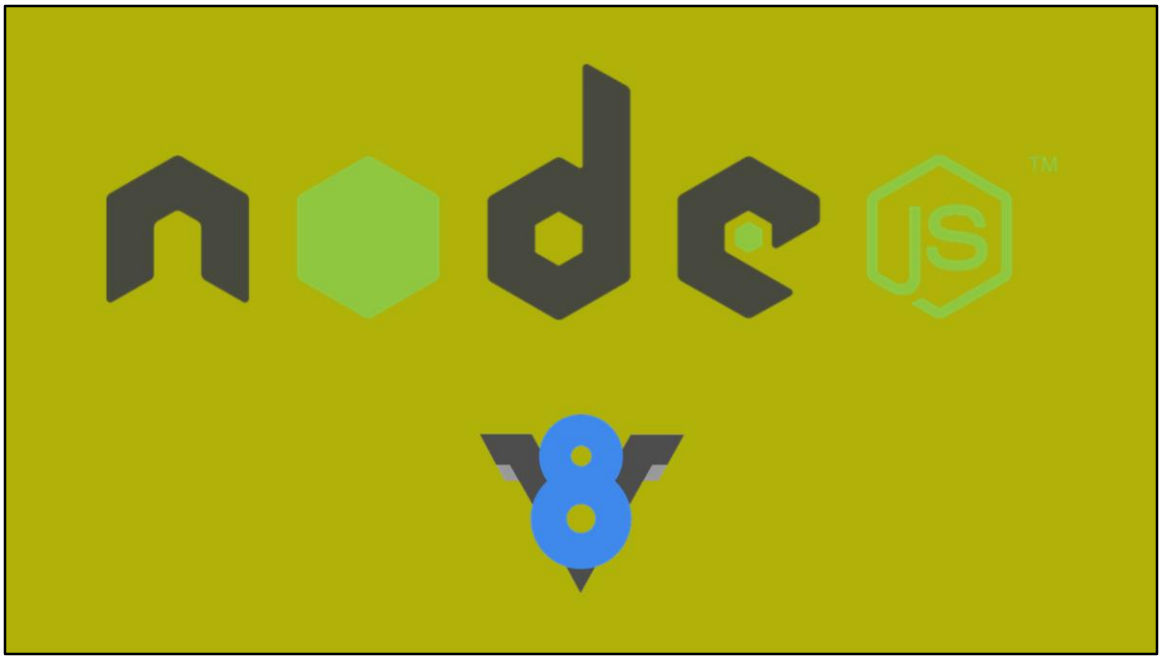


[Samenvatting L2] Nu we weten hoe eenvoudig het **opzetten** en **distribueren** van een Javascript applicatie kan zijn en op de hoogte zijn dat veel mensen dat al weten en we daar zeker **steun** zullen vinden gaan we nog een stap verder.

[Multiplatform] Javascript draait **overal**. Van desktop tablet, mobiele telefoon server, auto... OVERAL Kan javascript draaien en net **daarom** is het zo'n groot platform met extreem groote **potentie** waardoor er 1. Een grote **communitie** onstaat
2. De **tech giganten** zoals microsoft, google, ... Zich erachter scharen.

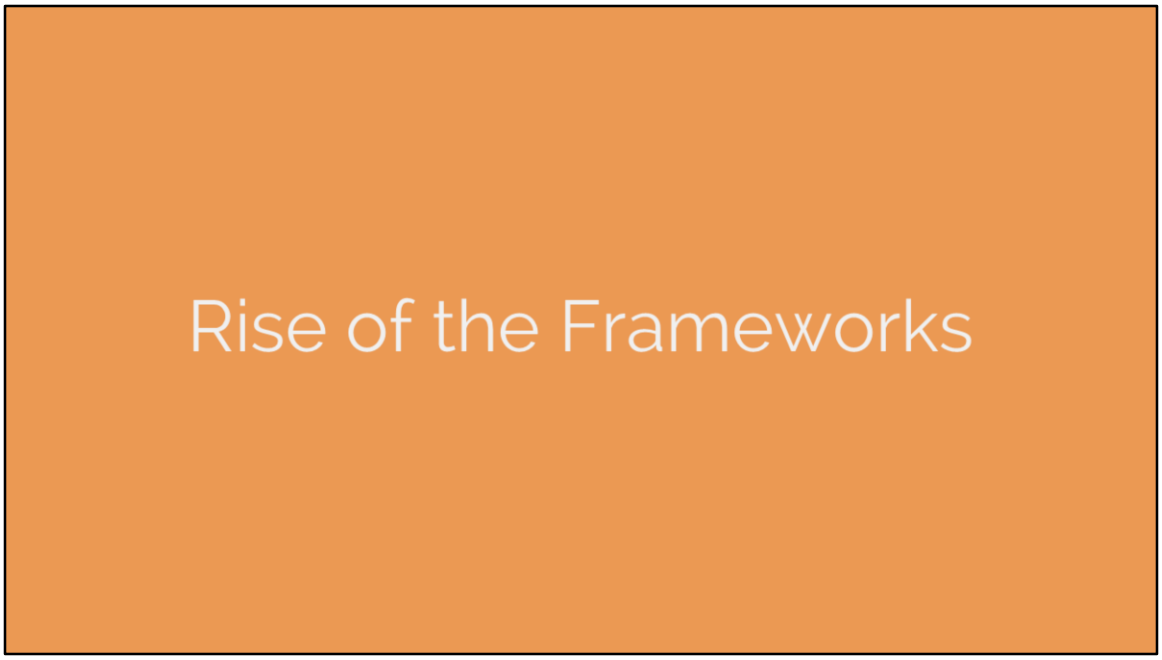


[Browser] In al deze browsers en nog veel meer is de mogelijk aanwezig om **rijke applicaties** te bouwen met javascript als bouwsteen. Ook hier weer je moet **geen cd** tot u gebruiker brengen om de ervaring van jou applicatie te delen. Internettoegang is de vereiste. Deze **browsers** zijn er voor **telefoons, tablets, desktops, autos**,... Kort de **kans** is heel groot tot zeker dat je gebruiker je **app** zal kunnen **gebruiken**.



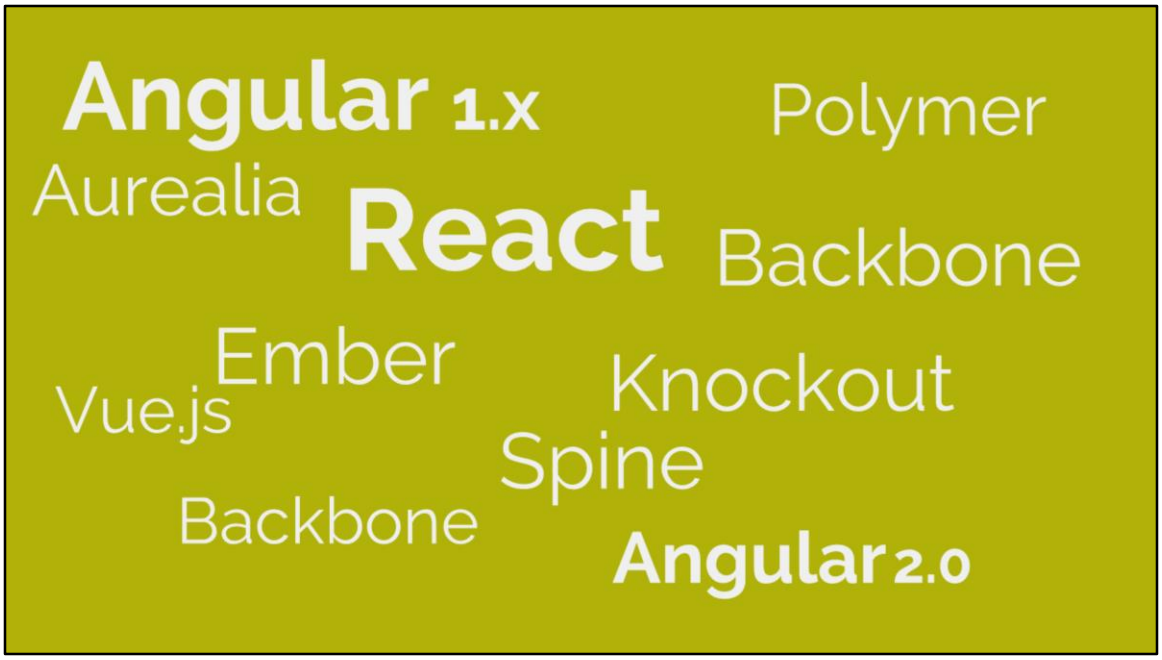
[Server] Als we zeggen ja oke javascript draait **overal** wat dan met **servers**, waar onze gedeelde logica zal komen en beslissingen genomen worden? Op servers kunnen door node en onderliggend google zijn V8 engine schaalbare javascript applicaties gemaakt worden met het oog op performantie en snelle oplevering van waarde.

[Wrappers] Oke maar als we als client **enkel** in de **browser internet?** kunnen draaien wat dan met bestanden opslaan, een foto nemen,... Er bestaan voor javascript vele **wrappers** die je toelaten native applicaties te maken waarbij javascript als basis genomen wordt en al het beste van het web html css geintegreerd wordt om native features aan te kunnen spreken.



[Samenvatting L2] Als we samen nemen dat we **overal** onze code kunnen draaien en er een gigantische **communitie** is moeten we toch ergens iets van **herbruikbaarheid** kunnen zien verschijnen.

[Framework] Die herbruikbaarheid uit zich nu vooral in de vorm van **frameworks**. Deze frameworks zijn vaak geschreven door **developers** die meewerkten aan een groot project en zo daar **herbruikbare dingen** uitgeplukt hebben om zo een framework te maken waar de wereld van kan profiteren.



[Development] Er zijn enorm veel frameworks enkel en alleen al voor **frontend** design decissions. Wat al deze frameworks gelijk hebben is dat ze een **doel willen oplossen**. Zo snel mogelijk zo veel mogelijk waarde kunnen developpen. Veel van deze frameworks zijn gebacked door de **grote bedrijven** google bij angular, facebook met react, knockout gestart door ene microsoft medewerker, ...

[Structuur] Een veel voorkomende vereiste is het tonen van **data** die van ergens komt op het **scherm**. Met dit in gedachten zijn verscheidene **frameworks** ontstaan. Zo is bijvoorbeeld react heel hard gefocust op de view en gaat react niet gaan instaan voor het ophalen van data terwijl angular dat wel meer gaat doen en binding gaat voorzien. Basicly vergemakkelijkt een framework de flow van data naar view.

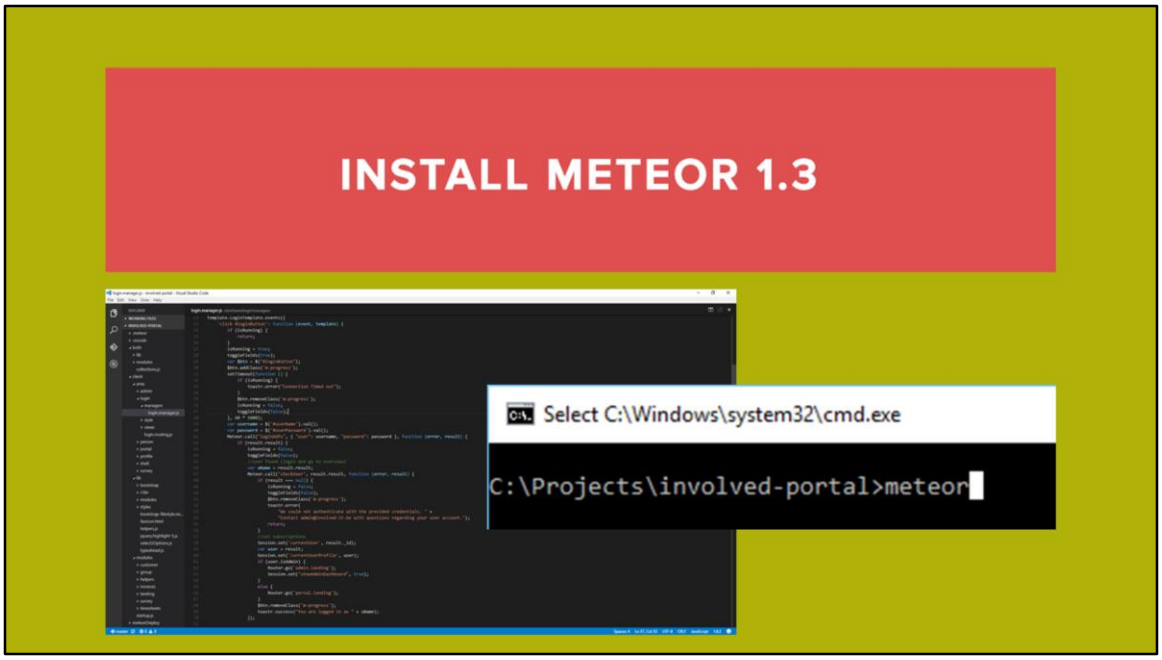
[Productie] Door een **framework** te gebruiken kan je grotere applicaties veel **eenvoudiger** en met **minder code** gaan schrijven. Wat zorgt voor **onderhoudbaardere** code, **snellere** oplevering van business value en loop je **minder risico** op eigen bugs waardoor je je risico ook verkleint

[Samenvatting L2] Oke we weten dat een framework dingen **abstraheerd** die voor velen **herhaling** vormen en zo je toelaten mits je het platform kent zeer snel te beginnen developpen en waarde te scheppen.

[Samenvatting L1] Om javascript in het **grote plaatje** te zetten weten we nu dat het een **open platform** is waarbij de **communitie** veel invloed heeft, werkelijk overal draait en dat componenten die bepaalde problemen oplossing in een framework gegoten worden.

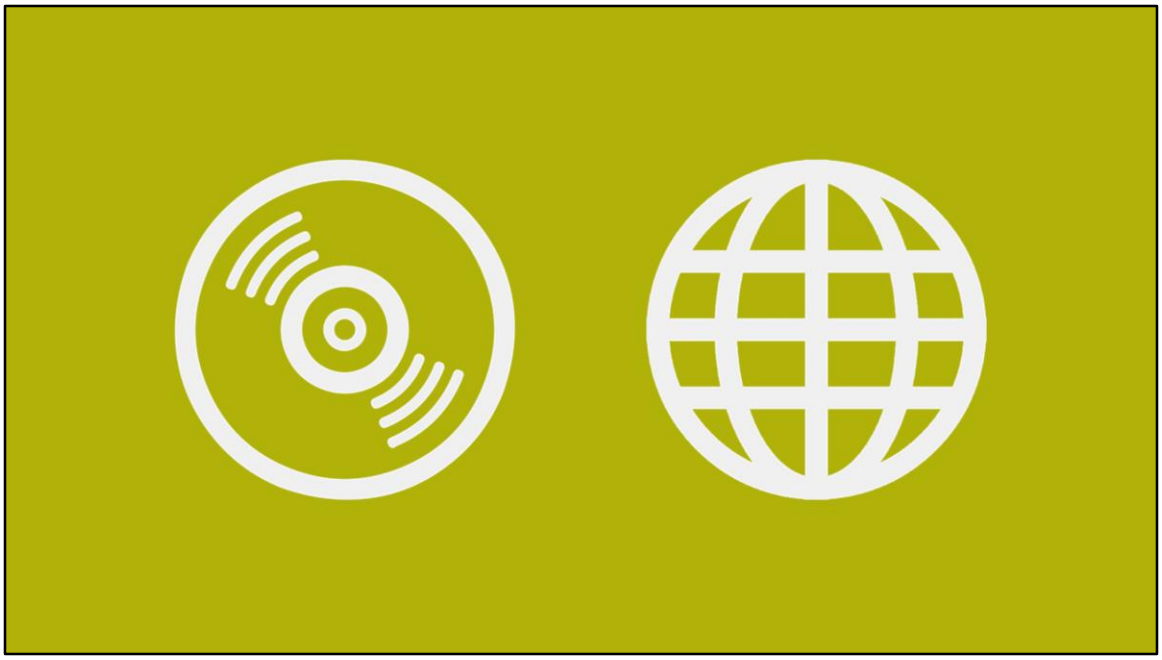
Why Meteor
Als we dan een framework kiezen waarom zouden we dan juist METEOR hiervoor kiezen?

[Setup] Het is nog steeds super **eenvoudig om te beginnen** alleen heb je nu buiten een browser en een texteditor nog extra nodig de meteor installer nodig. Door Meteor te gebruiken bouw je snel robuuste fullstack javascript applicaties die focussen op jouw code.



[Setup] Als je naar de Meteor website surft ga je deze grote knop zien staan op de homepage. Door hierop te klikken download je de meteor installer. Deze installer doet alles voor jou. Deze installeerd al zijn dependencies zoals bv mongo en node. Waardoor je met deze ene simpele installer direct toegang krijgt tot de pracht van Meteor. Vervolgens schrijf je je code en start je je project met één simpel commando “Meteor”.

[Focus op de essentie] Je hoeft geen **eindeloos werk** te doen om code op bv de backend aan te roepen. In andere frameworks stel je zelf je **http calls** samen. Bij meteor roep je rechtstreeks de backend code aan in jouw code. Meteor lost wel op hoe die data er geraakt. Waarom is dat nu een **voordeel**? Ook weer **time to market.** Je schrijft **jouw code en enkel jouw code**. VALUE!



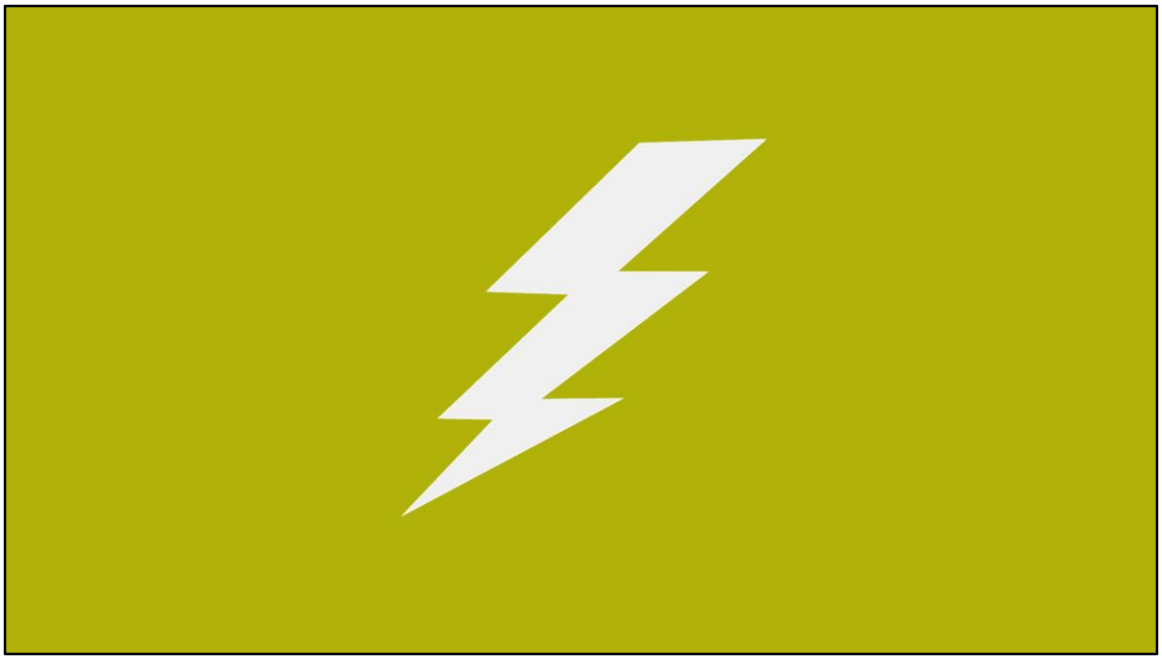
[Eenvoudige deploy] Je code is geschreven met **1 commando deploy** je je applicatie naar een webserver en sta je live. Geen gedoe met servers configureren, meteor doet dit voor jou. Ook hier weer nu iets developpen binnen 5 minuten live staan. Van **time to market** gesproken.

Fullstack reactivity

[Samenvatting L2] Dat we met Meteor **snel productierijpe** code kunnen voortbrengen op een korte tijd is alvast duidelijk.

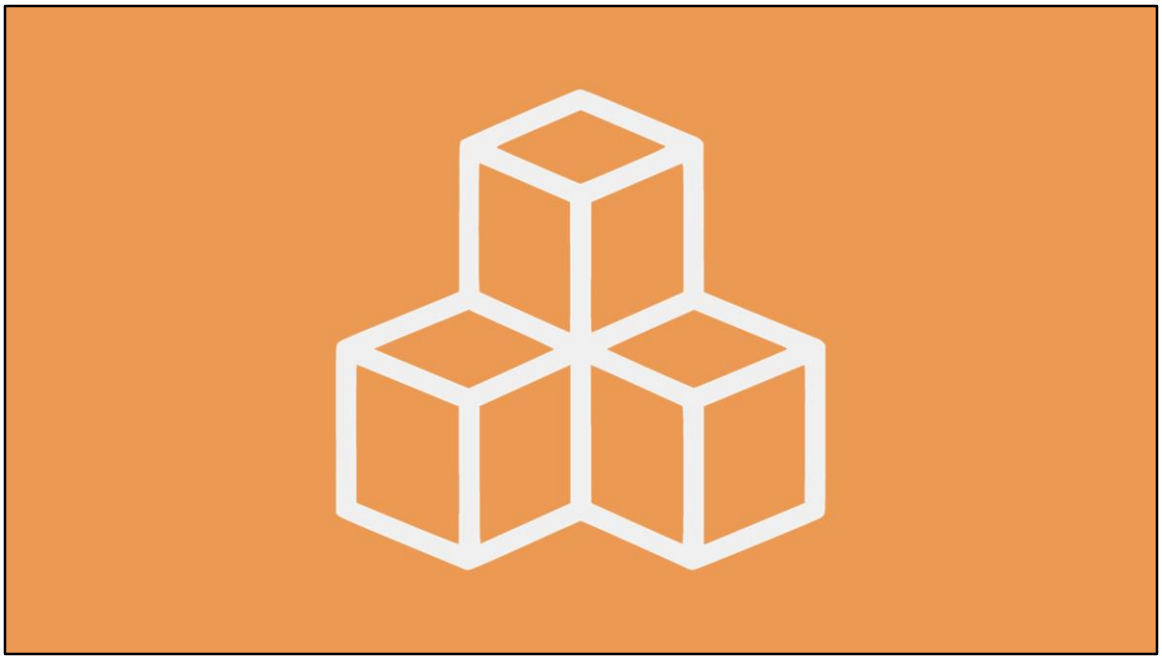
[Fullstack reactivity] Beeld u in wat je zou moeten doen om op client nummer **1** een **boodschap** te tonen wannner client nummer **2** iets **wijzigt** aan **data** die op het scherm van client **1** staat. Elke client laten pollen? Uw server de database laten pollen? Veel dingen zijn mogelijk. Meteor heeft dit probleem **omarmd** met een term die zij Fullstack reactivity noemen.

[Actions affect every subscriber] Als ik hier op een knop duw die data veranderd zal iedereen dat die data ook op zijn scherm ziet, die data ook zien veranderen. Elke actie die gebeurd heeft een gevolg op diegene die zich daarop inschrijven.



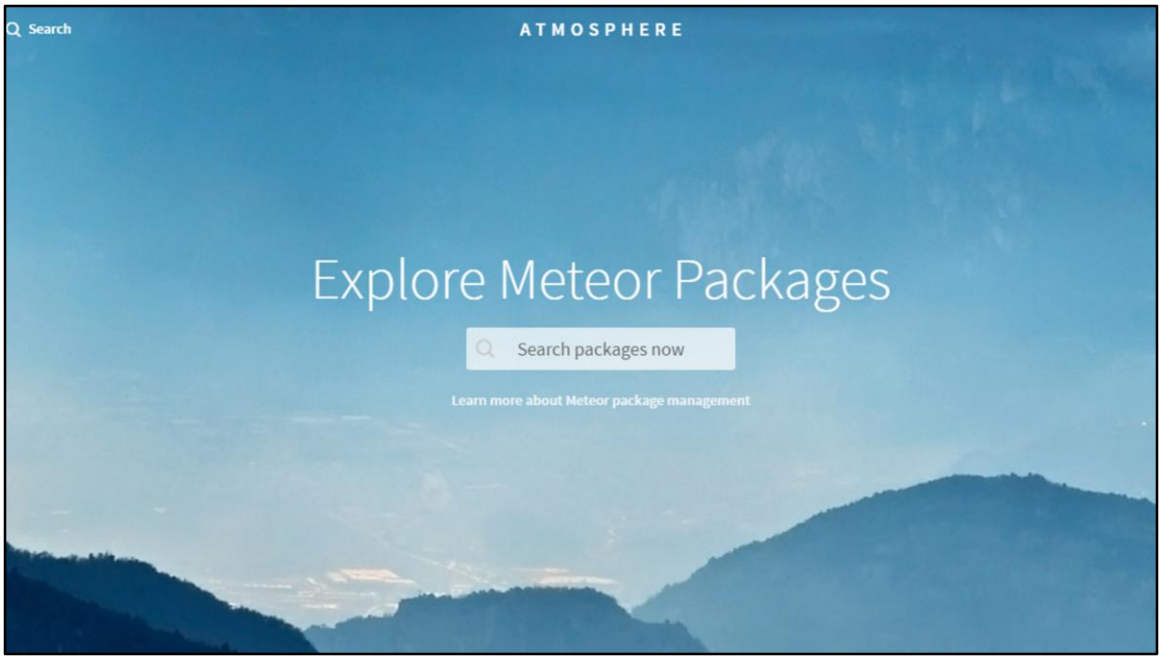
[Blazingly fast] Deze actie reactie gebeurd razend snel wat zorgt voor zo goed als real time communicatie tussen frontend, backend en andere connected clients. Dit komt niet enkel de gebruikservaring ten goede maar ook de **productiviteit** van de uiteindelijke gebruikers.

[Out of the box] Het beste aan deze oplossing is dat je er als developer niet voor hoeft te doen (scheelt ook weer in time to market). Fullstack reactivity werkt out of the box en is een van de pilaren waarop meteor gebouwd is.

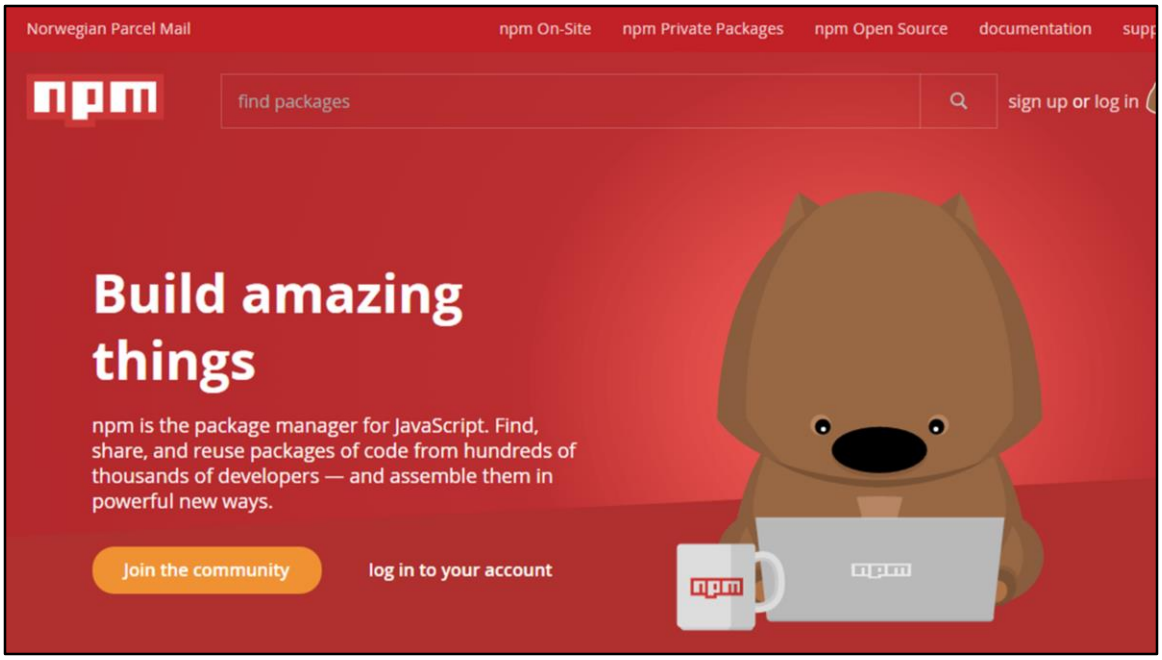


[Samenvatting L2] Wat is er zo **goed** aan deze fullstack reactivity**?** Vandaag is veel **geconnecteerd** en morgen alles. Meteor bied **performantie** out of the box door deze snelle actie reactie pilaar.

[Packages] Een **framework** met enkel en alleen zijn **basisfunctionaliteit** stelt weinig voor vaak willen we meer **dingen herhalen** maar die niet perce door iederen die het framework gebruikt nuttig zijn. Een antwoord hierop is een **library** of een package. Een package is een klein **mini frameworkje** dat in het beste geval **1 probleem probeert op te lossen**. Een voorbeeld van een package is bijvoorbeeld bij meteor een package die **meteor-accounts** heet. Deze package gaat ervoor zorgen dat je eenvoudig met 1 lijntje code ergens op je html pagina een login scherm kan zetten. Dit zijn meestal dingen die vaak herbruikt worden maar niet iedereen nodig heeft.

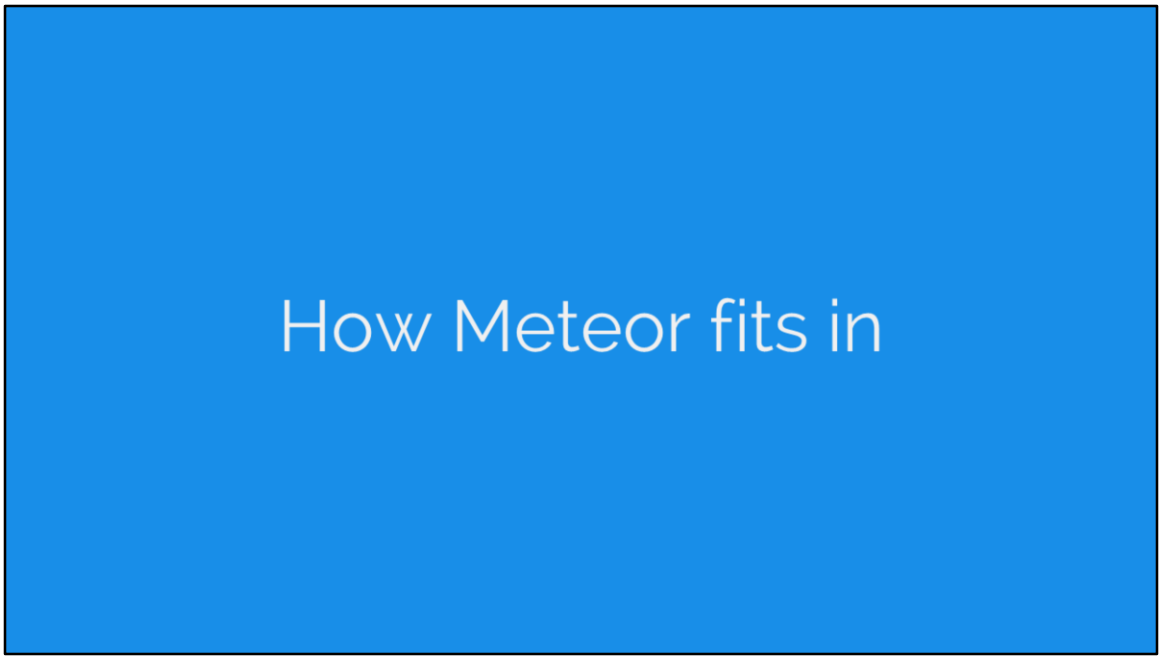


[Atmosphere] Zo bestaat er een **dedicated packages** systeem voor meteor geneaamd atmosphere js. Dit systeem laat toe om meteor packages te installeren om bijvoorbeeld account toe te voegen en dergelijke



[NMP] Via een atmospere js package kan je node packages ook gebruiken dit opend de deuren voor eindeloos veel packages

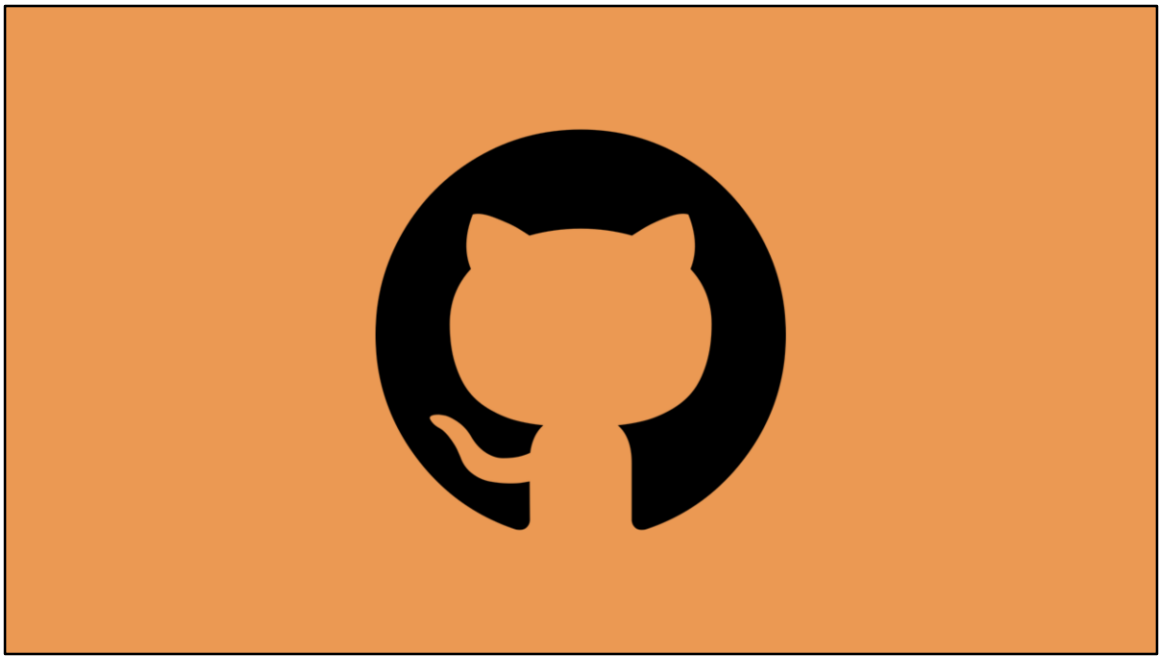
[Javascript] Niets houd je ook tegen je **eigen packages** te schrijven en deze te delen. Zo kan je bijvoorbeeld je eigen coole loading indicator in een package gieten en zo zelf herbruiken en andere mensen gelukkig maken. Allemaal in Javascript



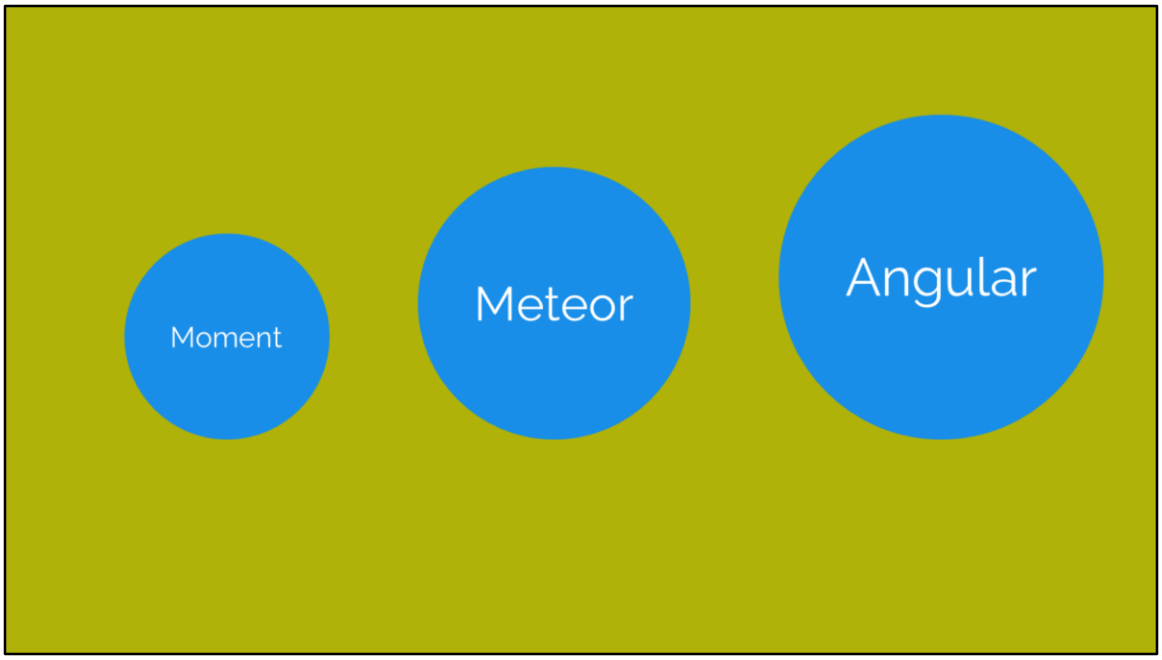
[Samenvatting L2] Packages laten je dus toe **kleindere herbruikbare componenten** te delen en gebruiken om zo functionaliteit af te schermen.

[Samenvatting L1] **Waarom** is Meteor nu het platform om voor te ontwikkelen en te gebruiken? (echt vragen) (antw: Snelle dev, Reactief, Uitbreidbaar)

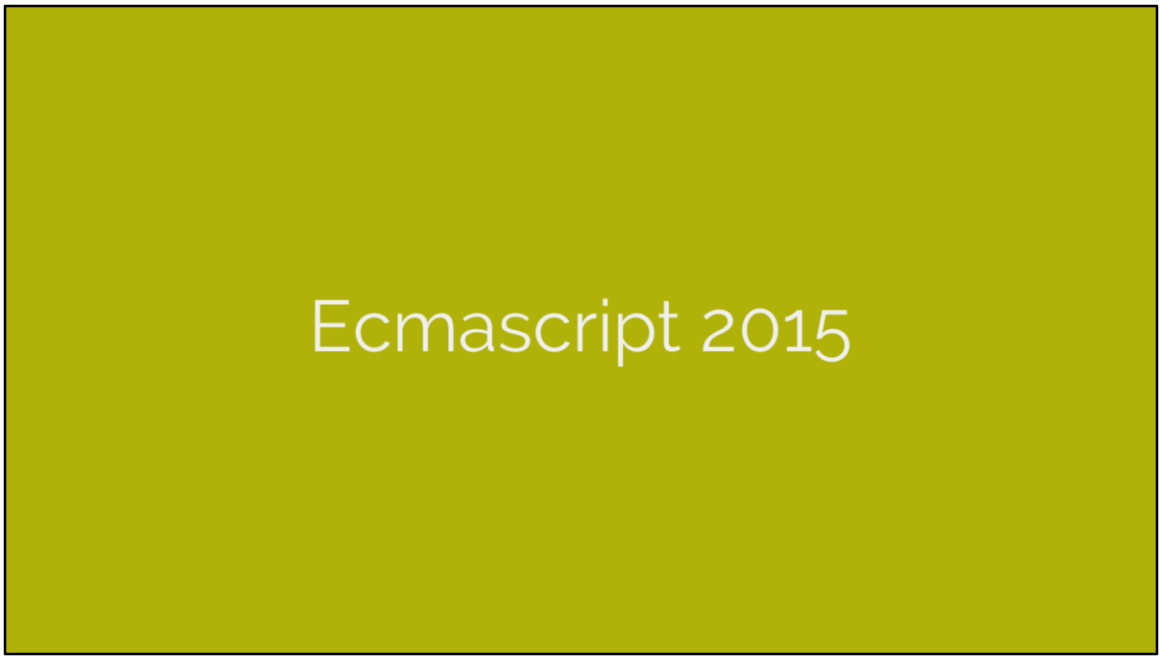
[How Meteor fits in] Nu onze **keuze** geprikt is op **Meteor** om zijn snelheid in waarde scheppen, out of the box functionaliteit en uitbreidbaarheid kunnen we eens kijken hoe meteor juist **past** binnen de javascript **mindset**.



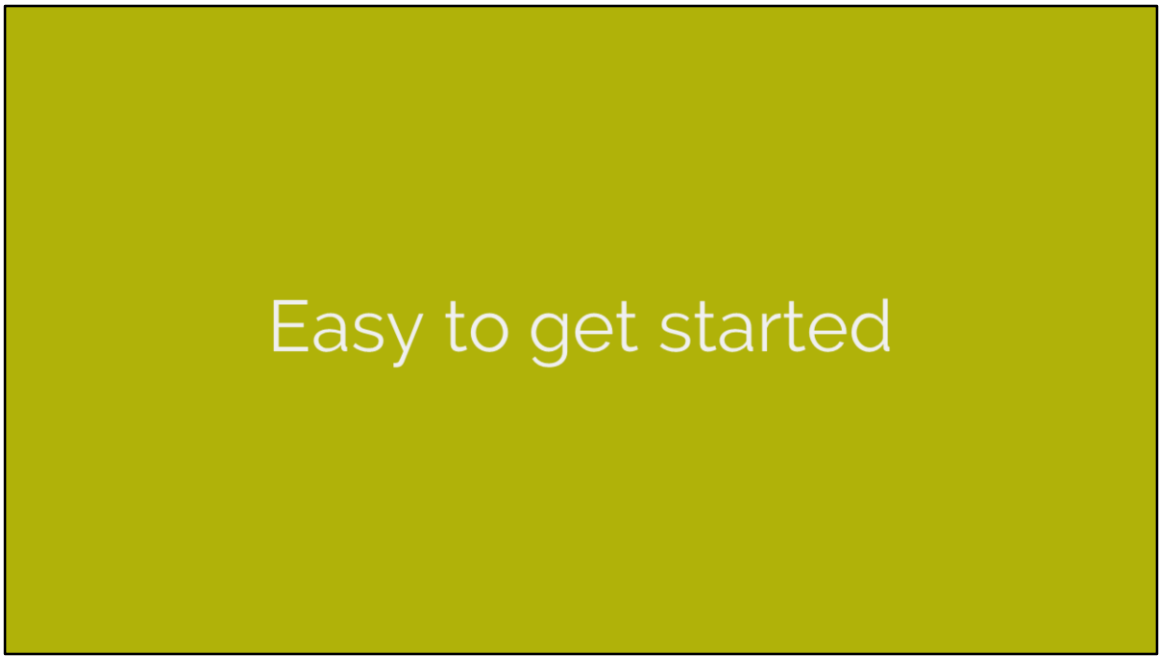
[Open source] Meteor is een **opensource** project gehost op Github. Waar iedereen de broncode kan bekijken en verbeteringen kan voorstellen.



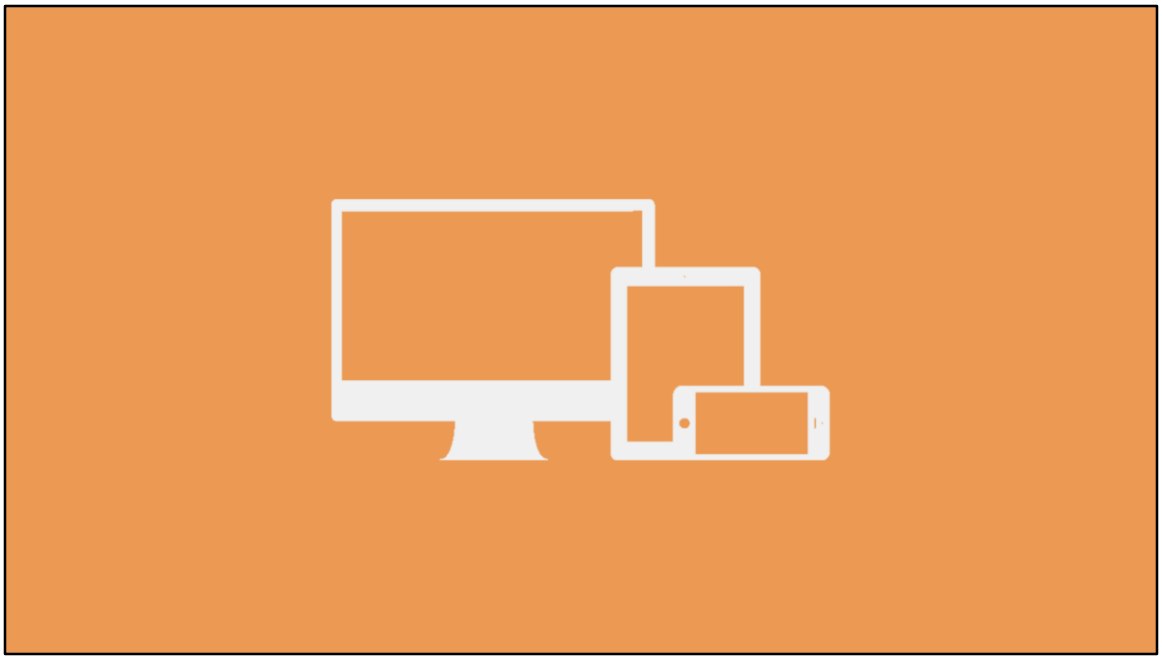
[Community] Met een **dikke 33 duizend stars** op github zit meteor op de **8e plek** tussen de andere grootte javascript frameworks.



[Ecmascript] Meteor laat toe je code te schrijven in de laatste versie van de javascript taal. Deze standaard wordt door veel grote bedrijven ontwikkeld (zoals microsoft, google, yahoo, ...) Op deze manier bouw je je applicaties altijd met een open specificatie.

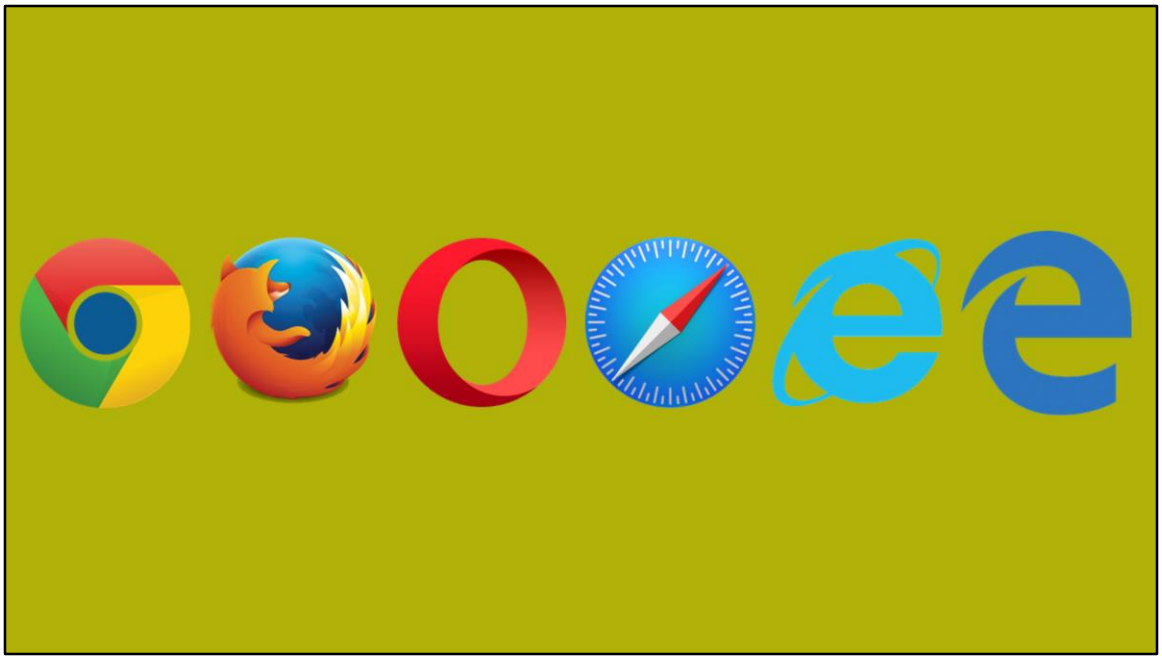


[Instap] De **instap** om met meteor te beginnen is nog steeds laag. Een **texteditor** en de **installer** volstaan om snel waarde te beginnen creeren.

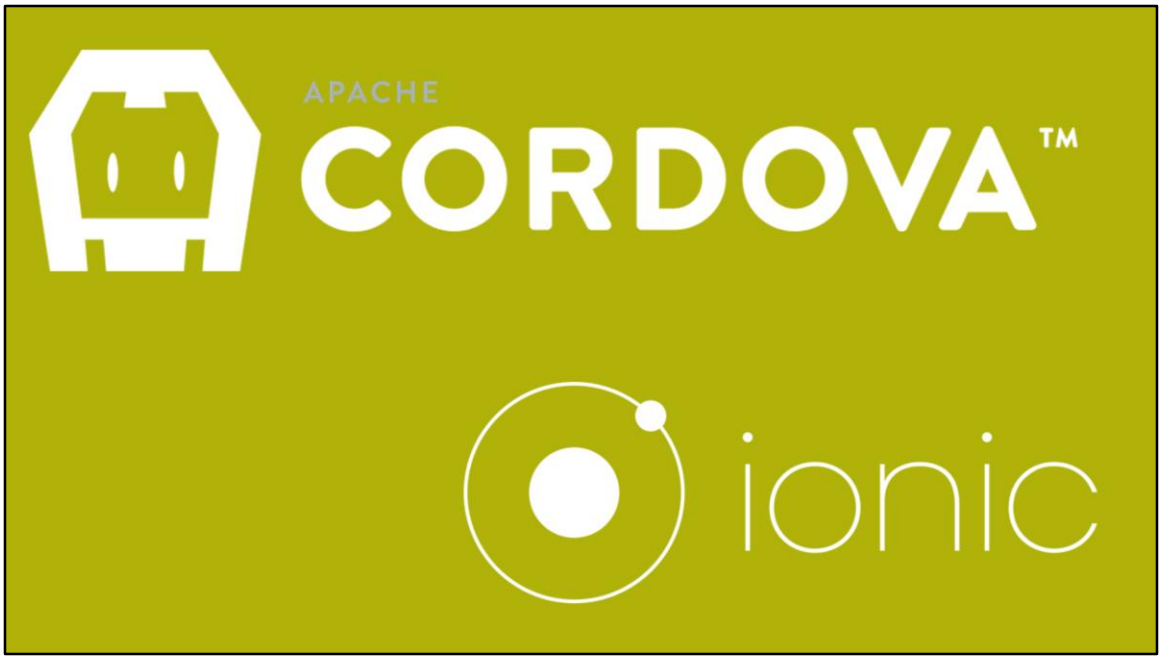


[Samenvatting L2] Meteor is **open** en **actief**. Een **communitie** staat klaar en je bent vertrokken op 1 2 3. Past perfect in de **Javascript mindset**.

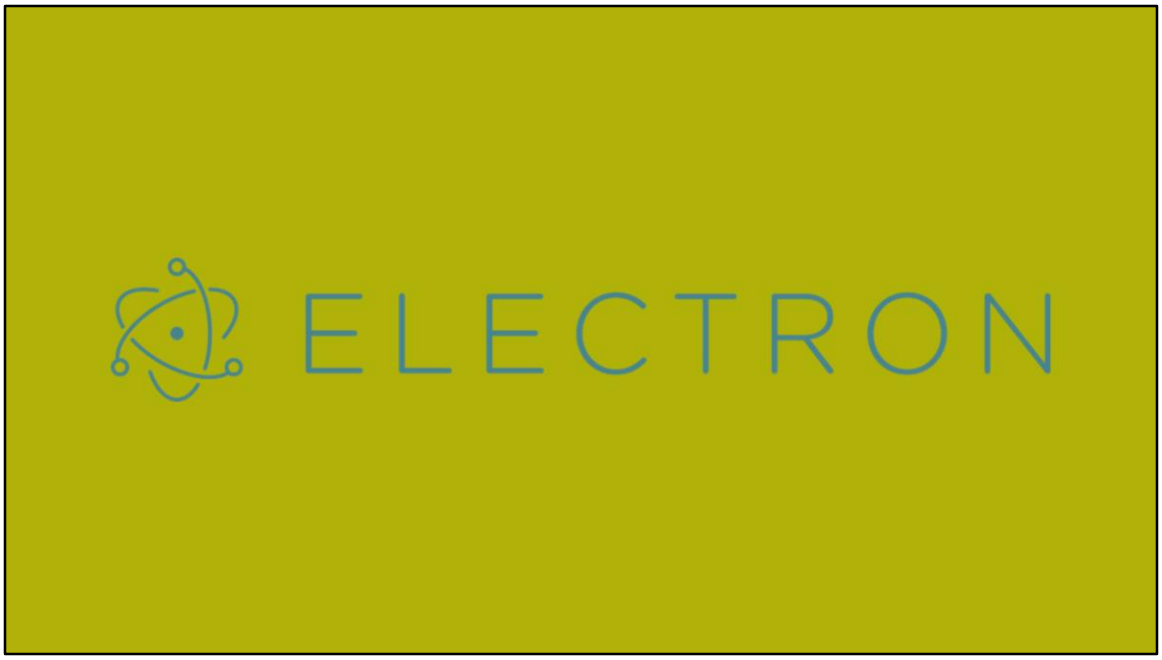
[Build targets] We hadden het er daarnet al even over maar Meteor draait **overal**. Om niet terug te moeten grijpen naar **het CD'tje. INTERNET**!!



[Browser] Met meteor kan je browsers targetten door een **webapplicatie** hosten. Zoals daarstraks al gezegt. Dit draait op mobiele toestellen, desktops, ... overal!



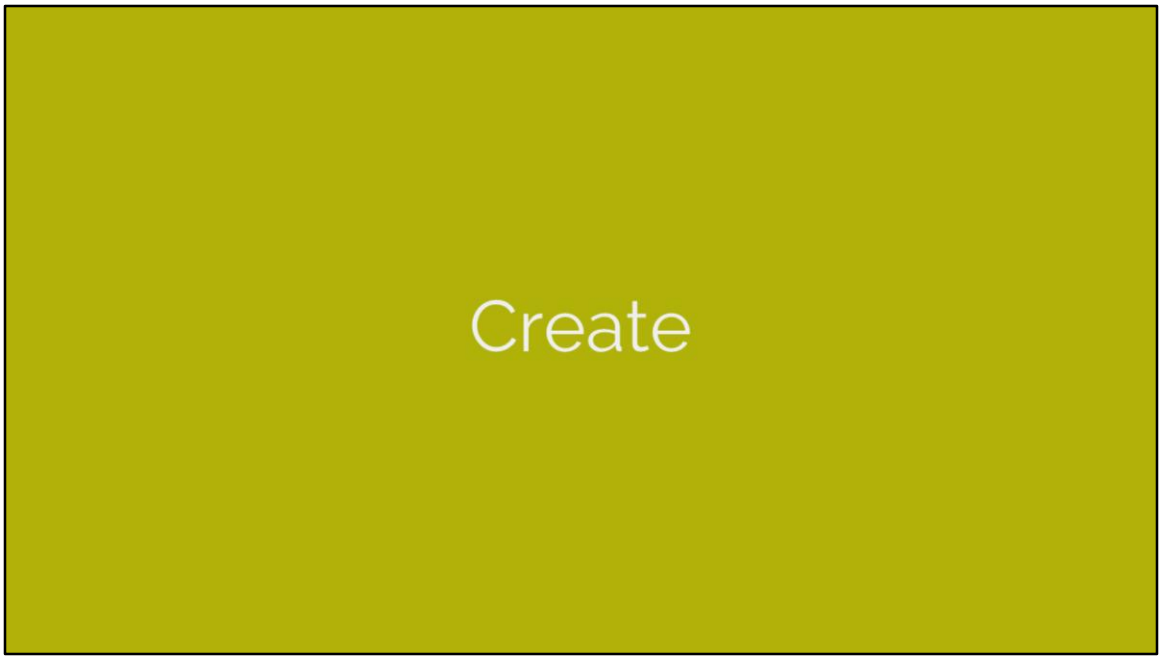
[Mobile wrappers] Daarnet heb ik ook wrappers aangehaald. Meteor heeft firstclass support om **wrappers** zoals een cordova of ionic te gebruiken. Zo kan je ook weer eenvoudig aan de **native features** van je apparaat. Deze functionaliteit komt ook voort uit de packages die we daarnet gezien hebben.



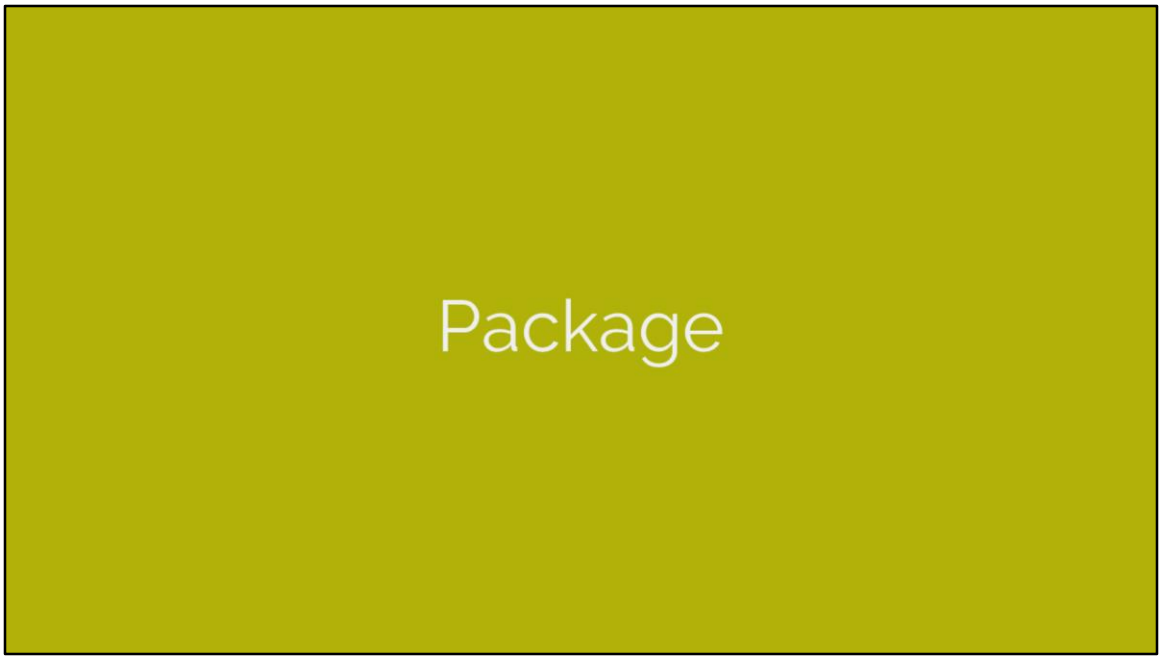
[Desktop wrappers] Nu al die mobiele apparaten is leuk maar wat met de good **old desktop/laptop**. Wel, Meteor ondersteund de **electron** wrapper. De Electron desktop wrapper laat ook weer toe via een package een **native windows mac of linux** applicatie te maken van je applicatie en zo native desktop features te kunnen gebruiken zoals notificaties, schijf IO, ...

[Samenvatting L2] **Waar** draait meteor? **Overal**!

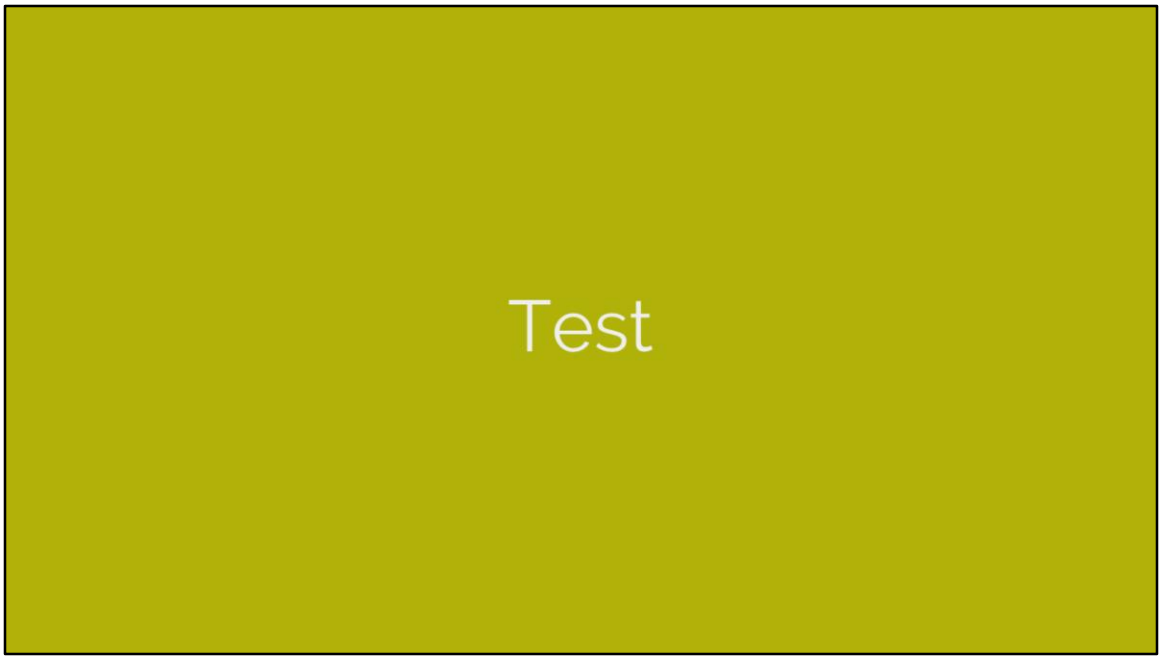
[Ease of development] Bij meteor zijn vertrokken uit het gedacht alles moet **zo simpel mogelijk** zijn. Veel herhalende stappen zijn zo simpel mogelijk gemaakt.



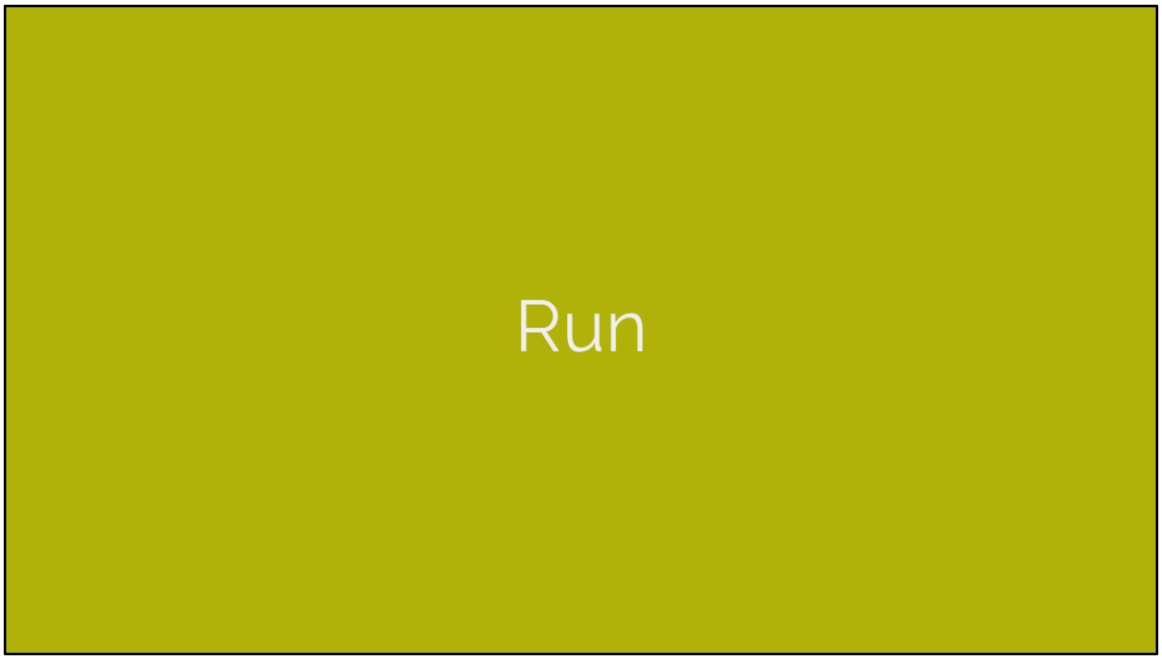
[Build pipeline] Je begint met het schrijven van functionaliteit, het maken van je applicatie. Je werkt met je favorite editor in **mapjes** en **files** die je maakt zoals je zelf wil en developt je code.



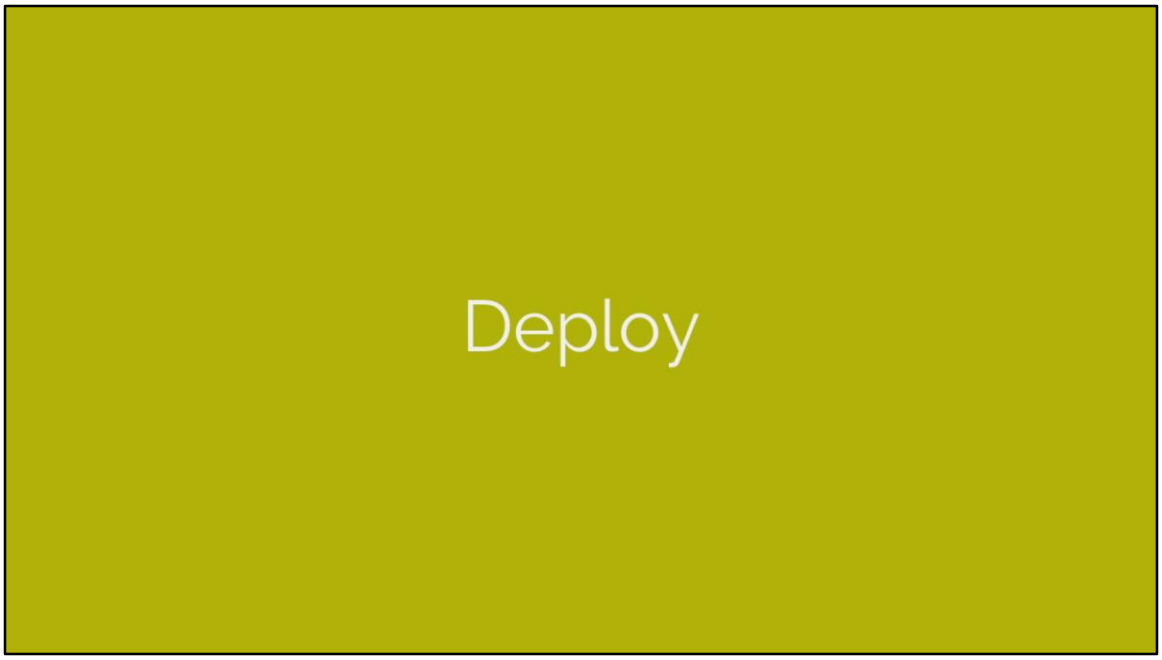
[Build pipeline] Oke, je code is klaar? Wat doe je dan? **Inpakken en shippen?** Wel we zullen al beginnen met **inpakken**. Van al die **losse files** kan meteor dan een **package** maken die je kan **runnen** om tegen te debuggen (een beetje zoals een **exe** bij een C# applicatie)



[Build pipeline] Wanneer je je package hebt kan je **testen schrijven** en deze draaien tegenover je output. Zo kan je eenvoudig unit en end to end testen met een atmospherejs package genaamd velocity.



[Build pipeline] Een package die je gemaakt hebt met de package functie kan eenvoudig opgestart worden op een platform. Het **packagen van je code, testen draaien op je code en het starten** van je applicatie gebeurd met 1 **commando METEOR**!. Nu dit is allemaal lokaal.

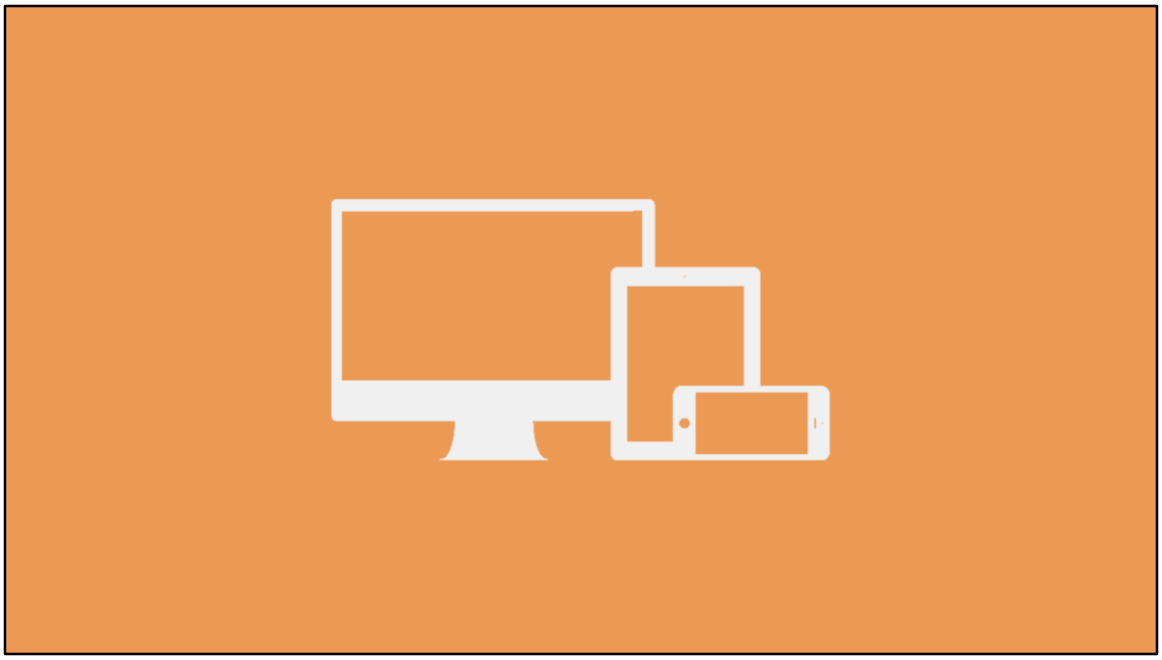


[Build pipeline] Nadat we lokaal alles getest hebben gaan we verder. We gaan live. **Deployen** naar een server, je applicatie online zetten, Ook hier weer zo **eenvoudig** mogelijk gemaakt. We hebben **één commando** nodig en een **lege vm** nodig om een deploy te doen. Meteor up is de package die daar voor zorgt. Momenteel houd meteor wel nog het liefst van Linux om op te draaien. Maar gezien Microsft zijn laatste moves verwacht ik hier ook onderstuening komen in de nabije toekomst Verder is er recentelijk ook **Galaxy** een infrastructuur voor meteor applicaties uitgekomen waar ook naar gedeployed kan worden. Dit is echter wel duur maar er gaat een gratis tier komen. Dus in het oog houden.

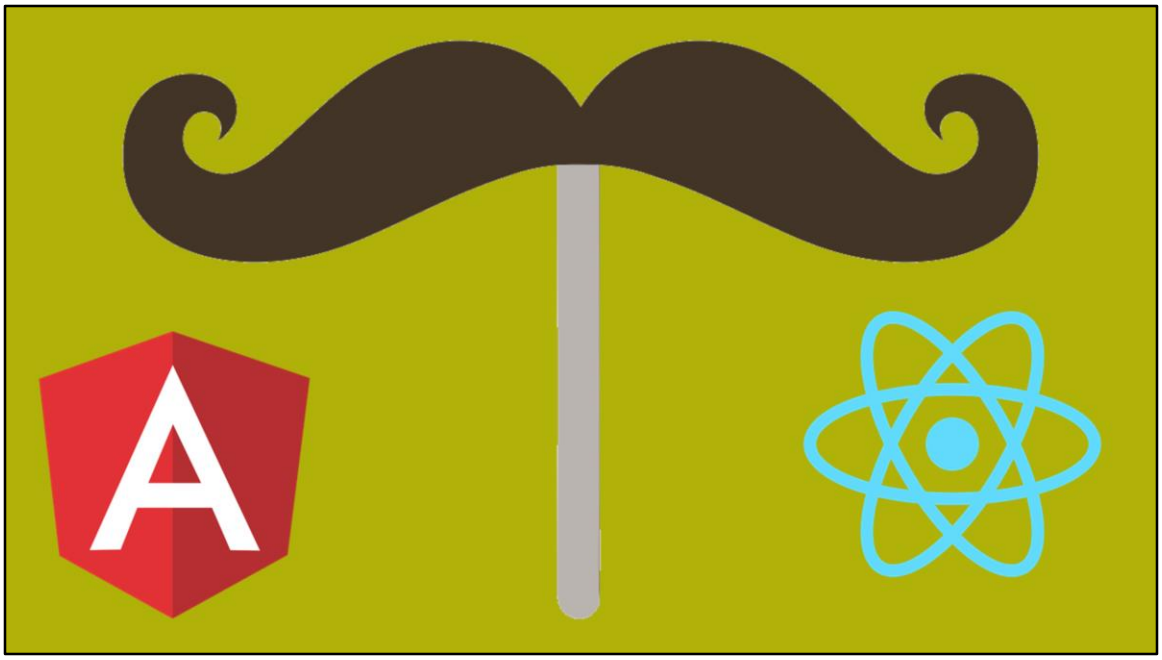
[Samenvatting L2] **Eenvoudig** van **Idee** naar live **applicatie**, dat is **meteor**, dat is zeker **Javascript**. **Eenvoud** heerst.

[Samenvatting L1] Meteor heeft een **gigantische communitie** uitgebouwd, wordt al **actief gebruikt** en draait overal met **native functionaliteit**, Te goed om waar te zijn? Nee het is zo! Javascript!

[How Meteor empowers you] Hoe gaat meteor jou nu helpen om te doen wat je wil doen **het beste te maken** dat je kan maken



[Client] [Frontend famework choice] Omdat **Meteor** echt een **platform** is gaat het je nog toelaten om je **frontend tooling** te kiezen. Angular/React of hun eigen spacebars



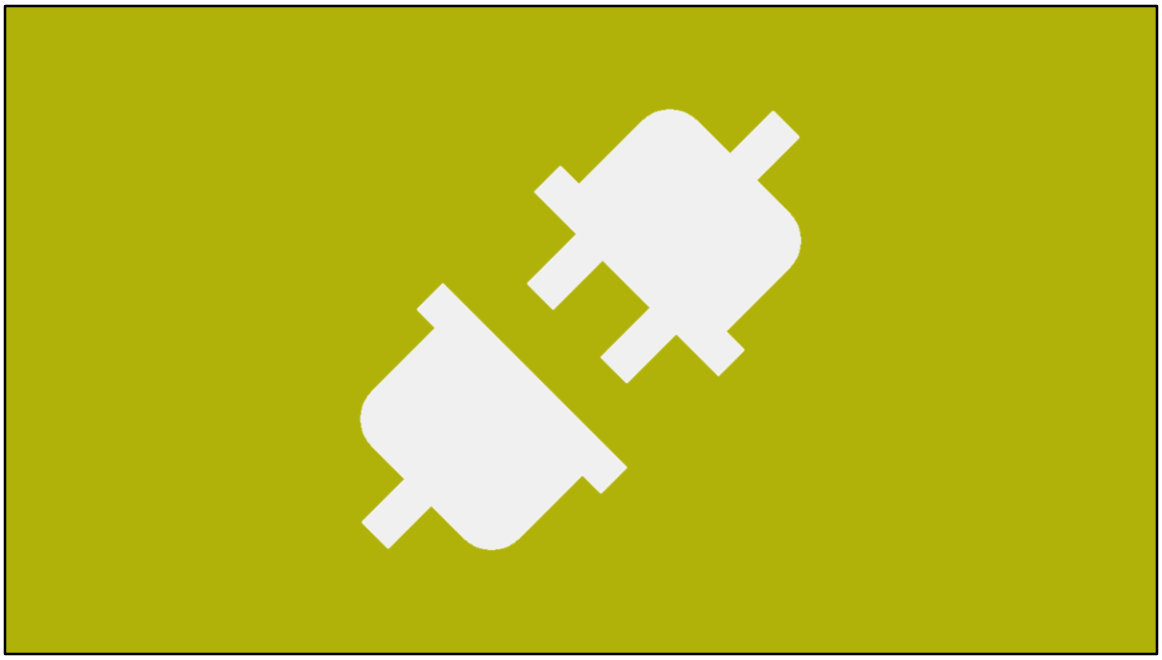
[Frontenkeuze] **Meteor** is oorspronkelijk begonnen met zijn eigen **view engine**. **Spacebars**, gebaseerd op handlebars. (De snor). **Later** zijn **Angular** en **React** erbij gekomen als mogelijke frontend keuzen. Zoals het er nu naar uit ziet zal in de **toekomst React** als standaard gebruikt gaan worden. React leent zich perfect voor de manier waarop Meteor met **data** omgaat waarover sewes meer.

[Samenvatting L2] Niet nodig

[Transport] **Optimistisch transport** wat ervoor zorgt dat alles snel in de **juiste state** verkeerd **zonder tijd te verspillen**. Meteor gaat er van uit dat alles wel in orde komt.

[Sync data] De meteor **client** houd de **data lokaal** bij waarop die gesubscribed is. (Das dat reactive van daarnet). De client gaat **geen eigen calls naar backend** doen voor persistentie van data gewoon lokaal doen en **meteor sync** deze optimistisch en zorgt er wel voor dat het in orde komt.

[Latency comenpensation] Een voordeel / gevolg hiervan is dat je **lokaal** al kan **veronderstellen** dat het **in orde** is. En meteor doet dat dan ook voor jou. Voor dat er echt iets op de draad gezet word wordt lokaal al aangenomen dat het in orde is. Wanneer het niet in orde is, word de actie terug gedraait en is er niets aan de hand. Je kan dat dan uiteraard best aan je gebruiker laten weten. Op deze manier hoeft de gebruiker niet te wachten op de '**het is oke' van de server**.

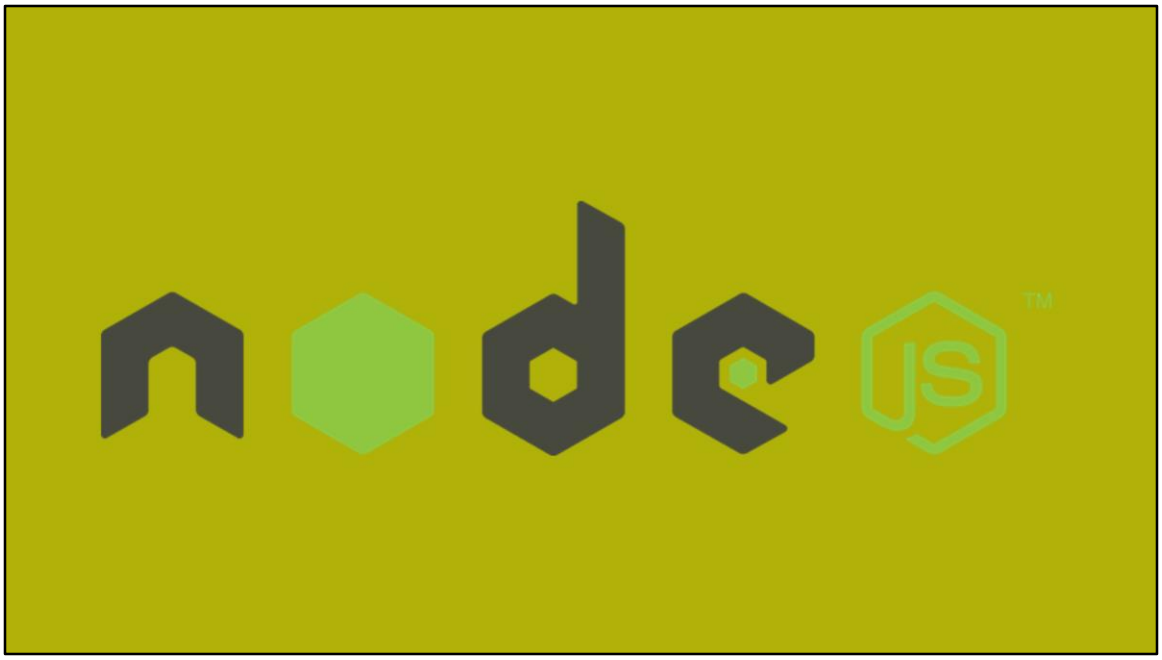


[Sockets] Mocht er dan toch iets **serieus mis** gaan zoals bv **geen internet** dan vangt meteor dit op door te bv te **retryen** en kan ook via package in offline mode en later synces, zoveel mogelijkheden

[Samenvatting L2] Van wat kunnen we zeker zijn bij **Meteor**? Onze data gaat niet verloren. En meteor **lost veel problemen** vanzelf op.

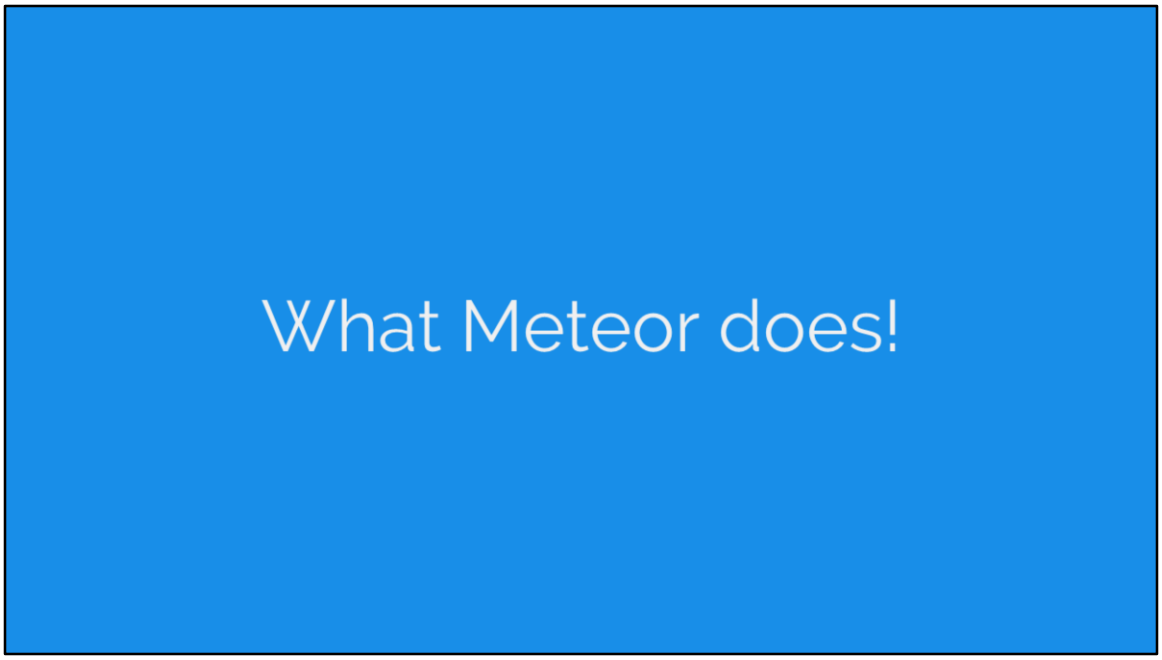
[Server] Op de **server** draaien verschillende technologien

[Meteor] Te beginnen met de **highlevel meteor runtime**. Deze zorgt ervoor dat **jouw code** draait op de server. Onderliggend gebruikt meteor node.



[Node] Node heeft zich op de server al **bewezen**. Alle niet meteor dingen zoals bijvoorbeeld het **schrijven naar disk**,... Worden door meteor aangeroepen in node.

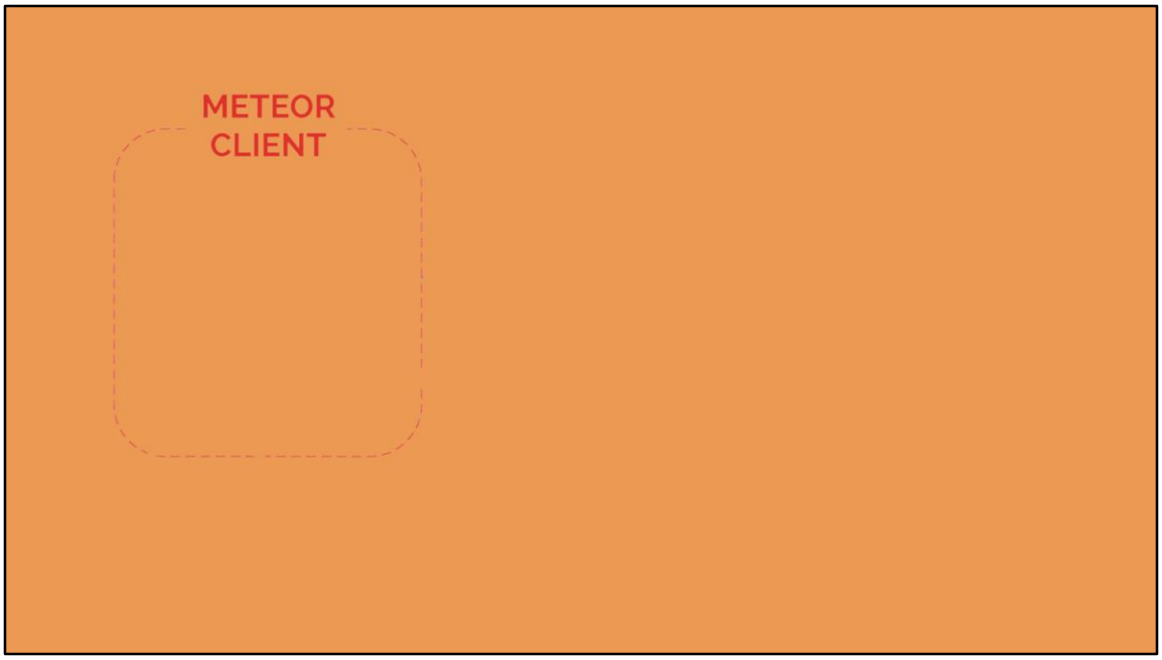
[Mongo] Mongodb de document database wordt gebruikt voor **persistentie**. Al onze data moet ergens gestockeerd worden. De **meteor runtime** gebruikt zijn **driver** om met **mongodb** te spreken en zo data te kunnen opslaan en observeren met die reactiviteit op het oog.



[Samenvatting L2] Op de server draaijen 3 **opensource** technologien meteor, node en mongo.

[Samenvatting L1] Door alle **technologie** die we gezien hebben kunnne we zeker zijn dat meteor **alles ter uwe beschikking** stelt om de **snelste beste** en **goedkoopste** applicatie te maken!

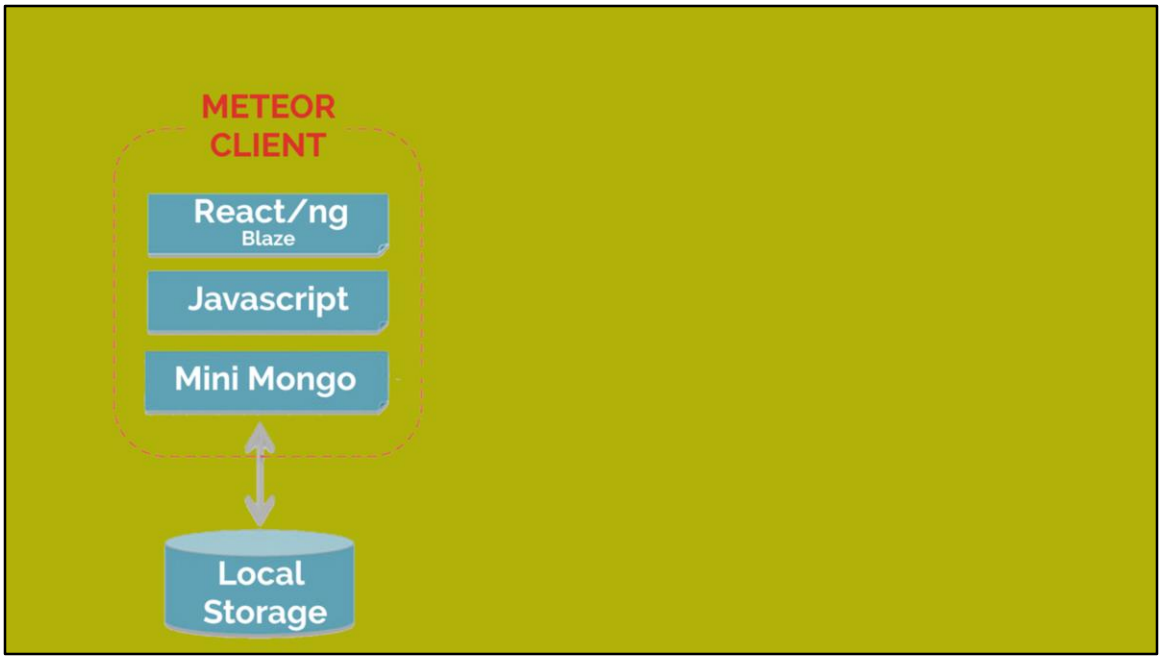
[What Meteor does] Nu we alle bouwstenen kennen gaan we even goed **diep inzoemen** wat meteor juist doet om deze samen te kleven.



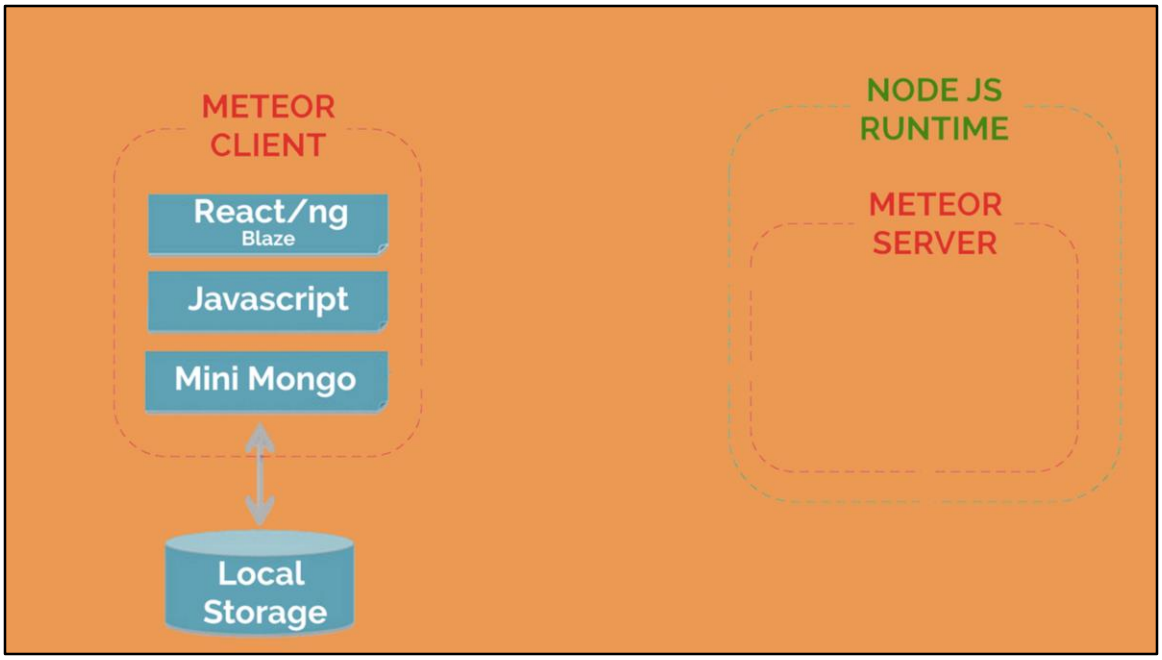
[Client] Om te beginnen spreken we over een **meteor client**. Dit is de applicatie die bv in de **browser** of in een **mobiele** wrapper zal draaien.

[Frontend] Binnen deze client kan je zoals we eerder gezien hebben je eigen **frontend framework** kiezen. Dat de **data** zal presenteren die ergens vandaan komt.

[Backend] Dit frontendframework is gaat **jou javascript** aanroepen welke **interacties** zal doen met de **data**. En **acties** uitvoeren

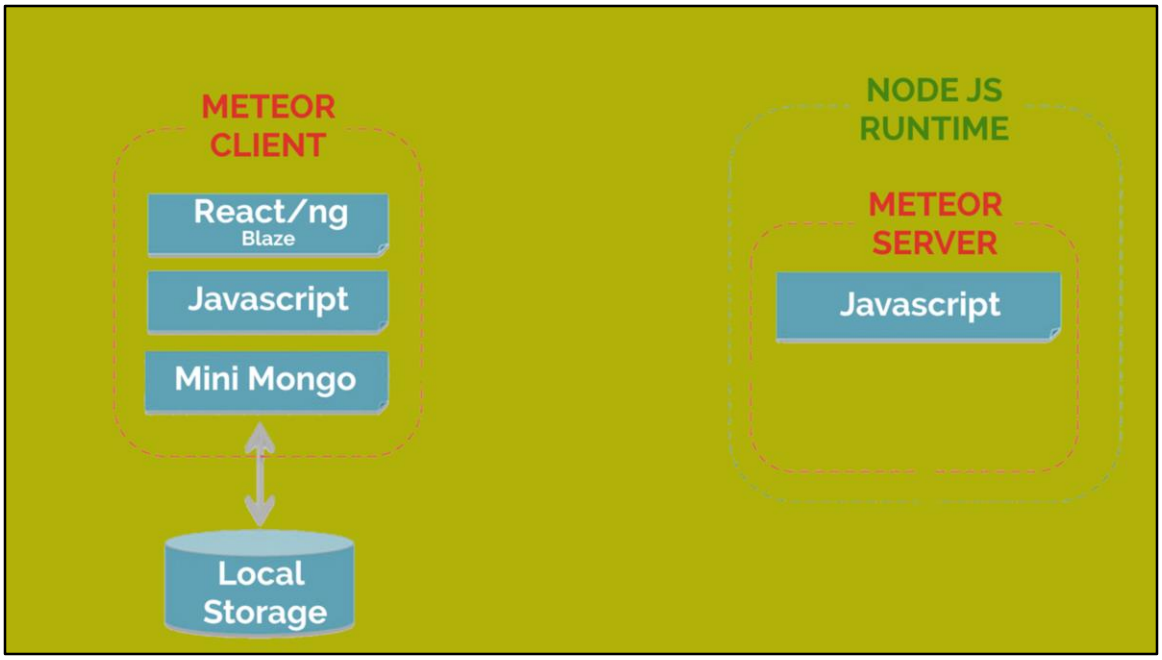


[Storage] Op de client draait een stukje code dat **lokaal alle data** heeft, die je applicatie nodig heeft van de server om te kunnen functioneren. Binnen meteor noemen we deze data de data waarop we **subscriben**. Binnenin de **client** gaat deze data **gebruikt worden** en aangepast worden. Deze kan in local storage opgeslagen worden bij bijvoorbeeld offline gebruik.

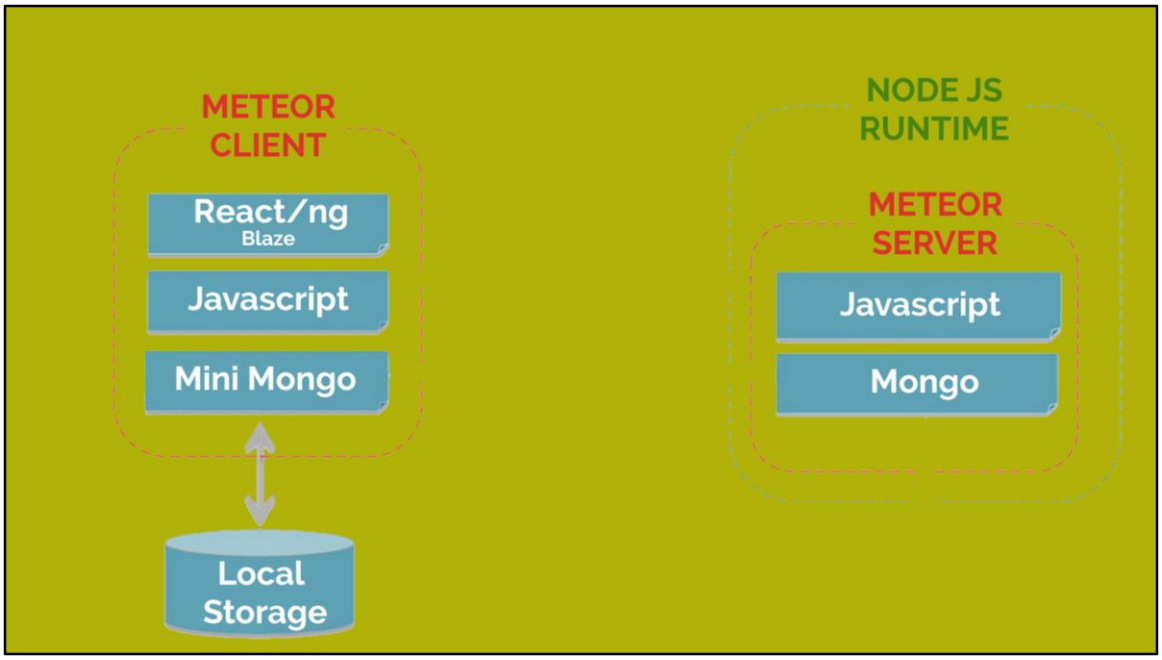


[Samenvatting L2] Op de client speelt al vanalles af. **Views** worden **gerenderd**, **state** word **gemanaged**, **data** wordt bijgehouden.

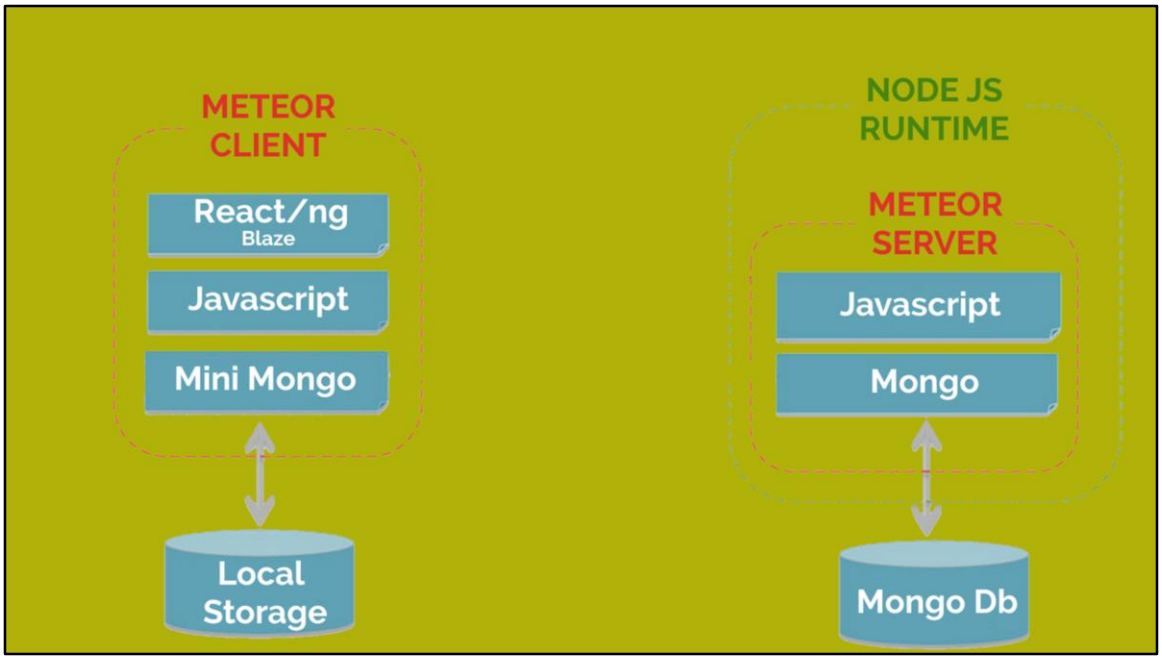
[Server] Als we dan naar de server kijken zien we dat de **meteor server** binnen de nodejs runtime draait. Deze meteor server **regelt alles** voor jou zodat jou meteor **applicatie** kan **draaien**



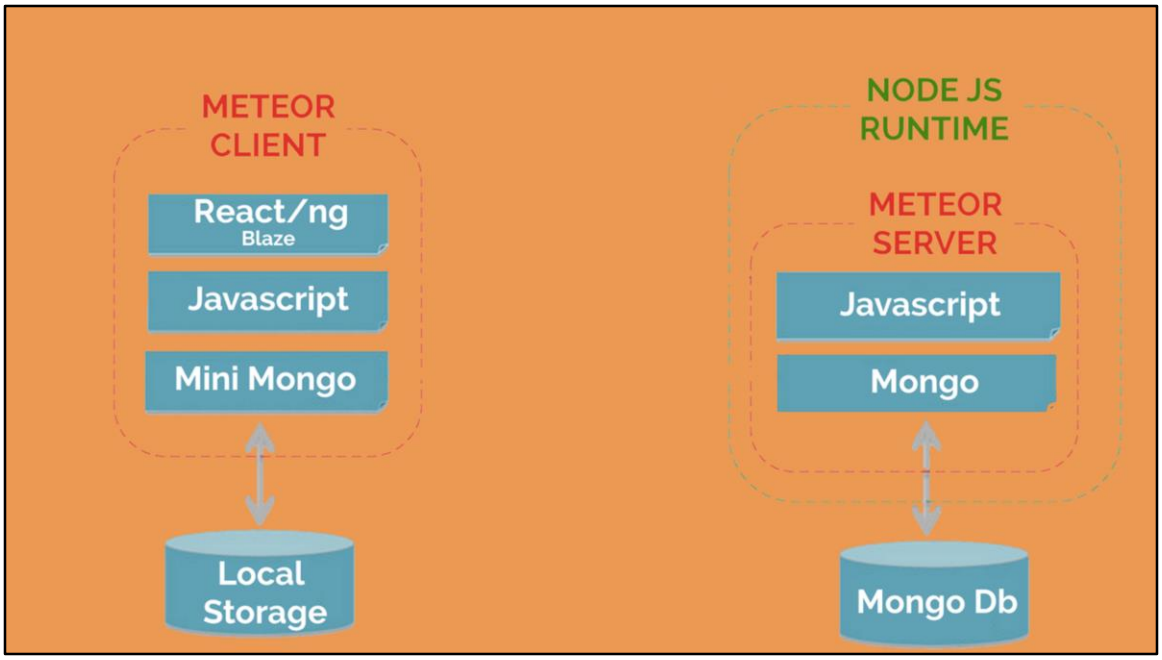
[Code] Wil je een **email** sturen? Of iets **opslaan** op de server? Dat doe je op de server in **javascript**. In javascript die door meteor uitgevoerd wordt schrijf je je **busniess logica**.



[DB] Als we iets willen ophalen uit onze **database** hebben we code nodig die dat doet. Daarvoor dient de **Mongo driver**. Deze draait binnen meteor en houd de **echte mongo database** in de gaten en doet daar **calls** naar.

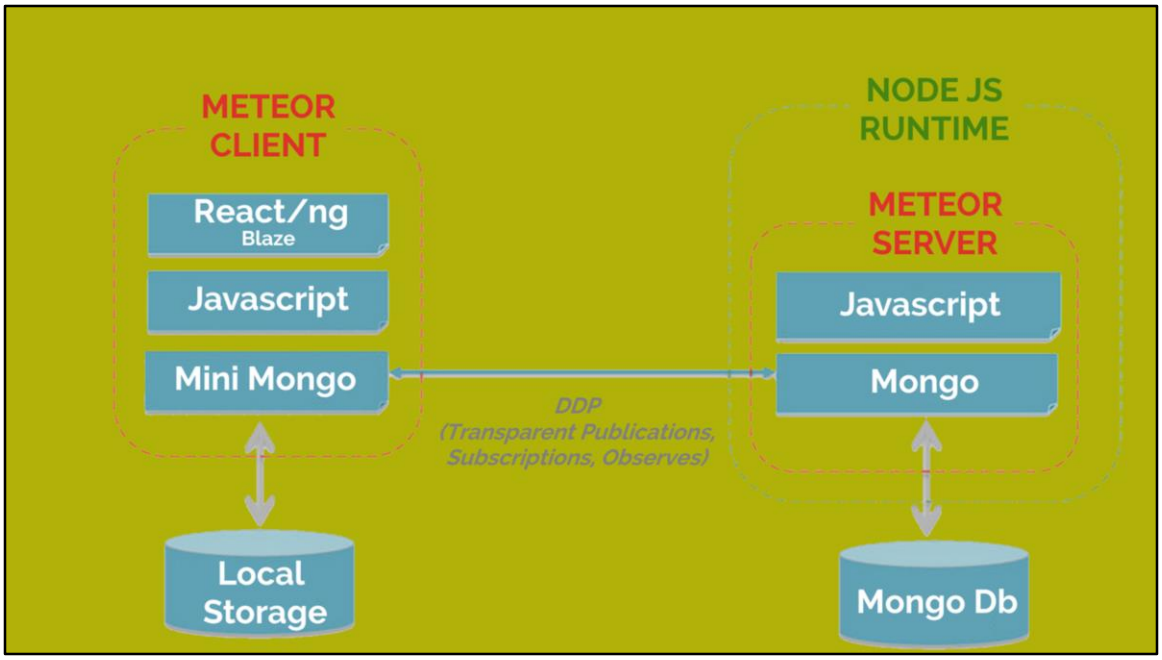


[DB] Tenslotte draait op de server of een andere server kort **ergens** draait een **Mongodb**. Hier wordt de data **echt in opgeslagen** en deze wordt nauwlettend in de **gaten** gehouden door de **mongo driver** om **wijzigingen** op te pikken en deze te kunnen **propageren**.

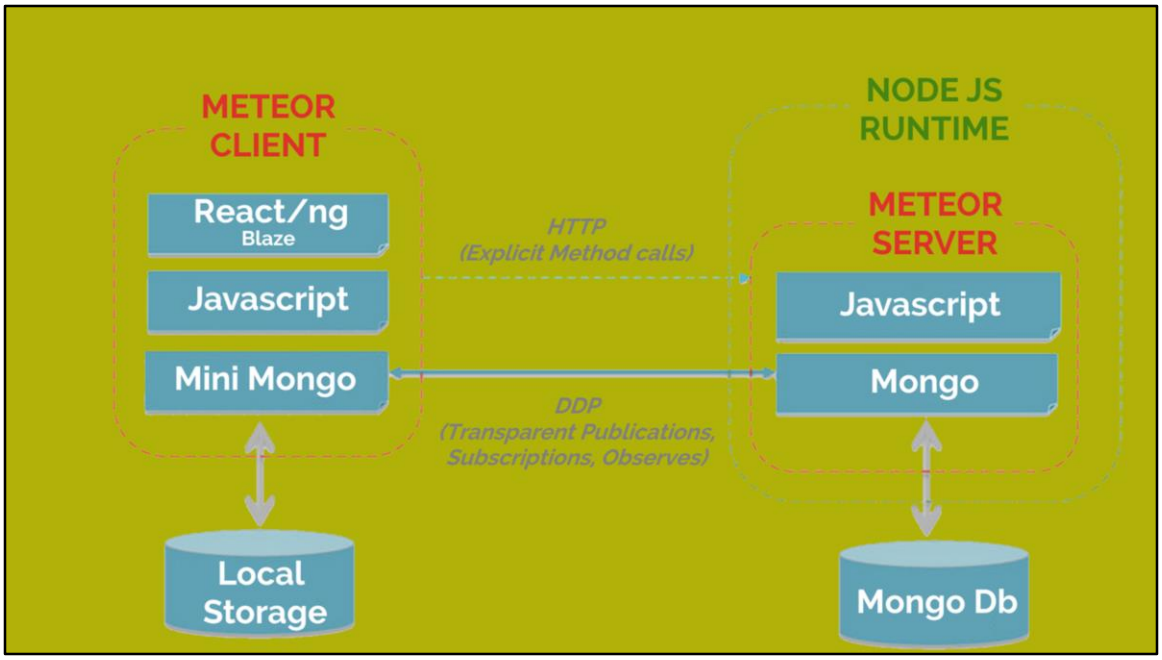


[Samenvatting L2] Op de server zit veel **functionaliteit**. **Business logica** wordt uitgevoerd**, Mongo driver** volgt de database op.

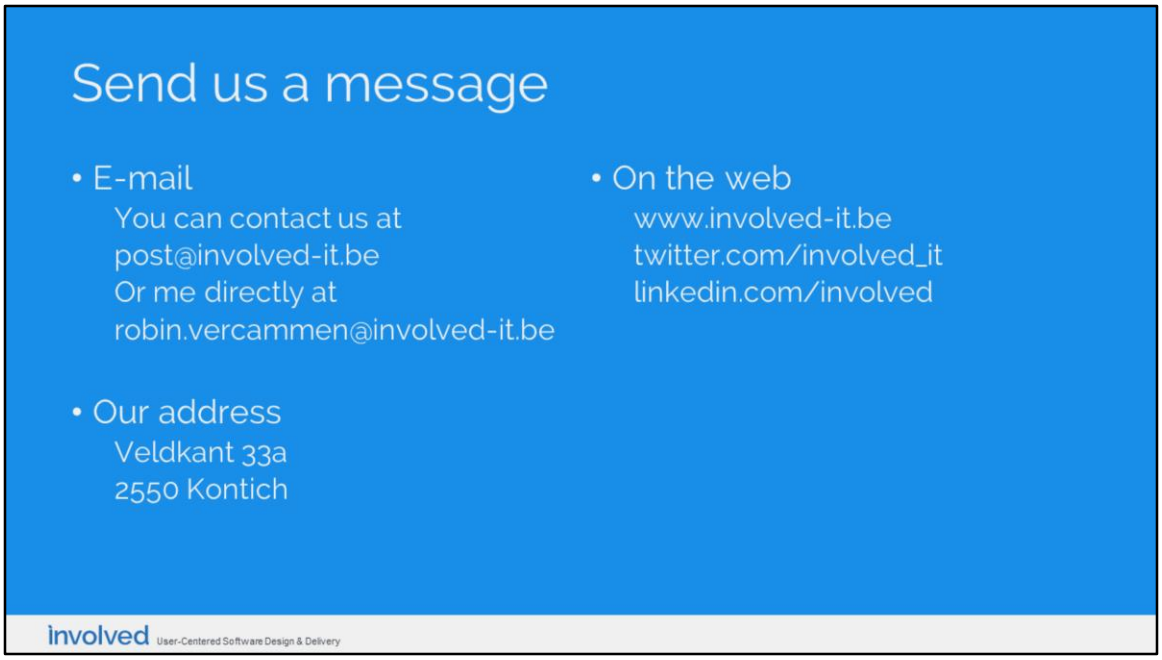
[**Transport**] Uiteraard moeten nu client en server kunnen **praten**. Hier blinkt **Meteor echt uit**. Hier gebeurd veel magie die ons het leven gemakkelijker maakt.



[DDP] Om de data op de **server gelijk** te houden met de data die de **client** nodig heeft, heeft meteor een **eigen protocol** ontwikkeld: DDP. DDP gebruikt ondrliggend **websockets** om snel packetejes over te brengen. Concreet gaat hier de data over die zegt van object met id 1 is veranderd op de server en mini mongo gaat dit oppikken en zijn locale opslag updaten om het model te matchen op de server. Omgekeerd werkt dat uiteraard ook. **Minimongo** dedecteerd dat wij een waarde veranderd hebben en gaat die publishen naar de mongo driver op de server. Deze gaat dit dan opslaan in de Mongo DB.



[Basic] Wanneer we **geen data** willen updaten maar gewoon **code** op de server willen **aanroepen** (bv een email sturen) dan kan je in je client code een **server method** aanroepen. Deze gaat achterliggend een http call doen naar de server maar in jouw code lijkt het alsof je gewoon een **normale functie aanroept**.



[Samenvatting L2] Waar blinkt meteor dus enorm in uit en versnelt het onze developtijd? **Transport**. Zo eenvoudig je server aanroepen is echt wonderbaarlijk.

[Samenvatting L1] Passen de **blokken** mooi in elkaar? Zeker! Is meteor **klaar voor de toekomst**? Voor meteor is de toekomst vandaag al! We weten dat onze frontend gekozen kan worden onze server gebruik maakt van prachtige technologien en het transport een pareltje is waar wij als developers lyrisch van worden.

[Samenvatting L0] Ik hoop duidelijk gemaakt te hebben dat ik enorme **fan van Javascript** ben. Ik daar zeker **niet alleen in ben** als we de getallen mogen geloven. **Meteor** al het **beste** van **Javascript bundeld** in een eenvoud te gebruiken platform. Meteor een uitstekend platform is om **snel** en dus **goedkoop realtime applicaties** te bouwen. En Meteor ieder van ons in staat stelt de toekomst van Javascript en het internet vandaag te bouwen.

[Einde] Op **10 mei** gaat er hier een nieuwe editie van **JSValley** door. Waar de topics allemaal **Javascript** zijn en nu met de nadruk op **functioneel programmeren**. Kijk zeker is naar onze prachtige **website** (involved-it.be) en volg ons **twitter** of **linkedin**. Vragen mag je altijd sturen naar **post@involved-it.be** of naar mij persoonlijk

robin.Vercammen@involved-it.be. Na de middag gaan we een **workshop** doorlopen en alle **goodies** van Meteor ervaren. En voor nu smakelijk en tot straks!