JAVASCRIPTCORE 实际应用

JSPATCH 灵活性

- 举个最简单的例子，RN与Weex，对于微信支付这种很native的模块，或许都是采用预先在native代码里写好了实现，最后让JS去调用（无论是开发者自己扩展的还是系统扩展的），这个东西就叫jsbridge，但这个bridge很普通，你写了微信支付的jsbridge，那你的js就有了微信支付的能力，你写了支付宝支付的jsbridge，那你的js就有了支付宝的能力，你写了iOS平台的IAP的jsbridge，那你的js就有了IAP的能力，如果你没写咋办？你的JS就办不到，要知道，扩写这些jsbridge，都是需要发版的，都是不能动态更新的。换句话说，如果一开始使用RN的app只开发了支付宝的支付功能，写好了支付宝的jsbridge，于是发版上线了，等某一天突然想接入苹果的applepay，那么只有一个办法，那就是重新发版，重新写好了applepay的jsbridge，再次提交苹果审核

- 但是对于JSPatch来说，JSPatch也是一套基于JSCore的bridge，但是牛逼就牛逼在JSPatch不是这种常规的normal bridge，而是runtime bridge，runtime的特点就是实现没有被任何oc代码预先写好的功能，同时还能修改已经被OC代码预先写好的功能！

JSPATCH 灵活性

- 举个最简单的例子，RN与Weex，对于微信支付这种很native的模块，或许都是采用预先在native代码里写好了实现，最后让JS去调用（无论是开发者自己扩展的还是系统扩展的），这个东西就叫jsbridge，但这个bridge很普通，你写了微信支付的jsbridge，那你的js就有了微信支付的能力，你写了支付宝支付的jsbridge，那你的js就有了支付宝的能力，你写了iOS平台的IAP的jsbridge，那你的js就有了IAP的能力，如果你没写咋办？你的JS就办不到，要知道，扩写这些jsbridge，都是需要发版的，都是不能动态更新的。换句话说，如果一开始使用RN的app只开发了支付宝的支付功能，写好了支付宝的jsbridge，于是发版上线了，等某一天突然想接入苹果的applepay，那么只有一个办法，那就是重新发版，重新写好了applepay的jsbridge，再次提交苹果审核
- 但是对于JSPatch来说，JSPatch也是一套基于JSCore的bridge，但是牛逼就牛逼在JSPatch不是这种常规的normal bridge，而是runtime bridge，runtime的特点就是实现没有被任何oc代码预先写好的功能，同时还能修改已经被OC代码预先写好的功能！

APATCH