你好，我是winter。

从今天开始，我们就从前端知识学习的部分，过渡到了实践部分。这节课我来谈谈性能。

性能是个特别有意思的话题，在我之前的工作中，从入门的初级工程师到高级别的技术专家，大家都很喜欢谈性能，我以前参与晋升评审，每年总能听到很多关于性能的晋升述职。

那么，今天我就来谈谈我眼中的性能。

性能总论

> while循环快还是for循环快？
>
> |0 是不是比 Math.floor 性能好？

网上随处可以见到一类对性能的讨论。一些新人也非常热衷此类讨论。但是实际上，它们除了让你写代码的时候纠结之外，毫无意义。

为什么这样讲呢？我想讲一个小故事。

从前有个工程师，特别注重代码细节，有一天他发现系统中的一段代码写的性能很差，因此，他用汇编重写了整段代码，执行效率足足提升了三倍。但是最后，大家发现，用户反馈性能丝毫没有提高，因为他优化的那个进程名字叫“System Idle”。

所以你看，性能优化不能只着眼于局部的代码。这里，我要提出一个我的观点：**一切没有profiling的性能都是要流氓**。凡是真正有价值的性能优化，必定是从端到端的业务场景建立体系来考虑的。

在我的认识中，性能体系的建立可以分成以下几部分：

- 现状评估和建立指标；
- 技术方案；
- 执行；
- 结果评估和监控。

下面，我就来为你一一讲解。

现状评估和建立指标

要想做好性能优化，正确地评估现状和建立指标是最关键的一步，它又往往是会被轻视的一步。

作为一个工程师，指标又要考虑两个因素。一方面，对用户来说，什么样的性能指标能更好地评估它的体验？另一方面，对公司来说，什么样的指标会影响业务价值呢？

在我公布答案之前，我希望你能思考一下，你所负责的业务，是否有前端性能指标？它是否能够满足我上面提到的两个要求？

在我之前的工作中，整个用了长达一年的时间来探索，才找到了合适的指标，并且回答好了两个问题。

性能问题可以分成很多方面，最重要的几个点是：

- 页面加载性能；
- 动画与操作性能；
- 内存、电量消耗。

注意，这里我们仅仅是对“性能”两个字的分析和解读，在对大量的用户数据分析后，我们发现，其实这三部分中，“页面加载性能”跟用户的流失率有非常强的关联性，而用户流失率，正是公司业务非常看重的指标。

因此，在开始阶段，我们决定把性能优化的重点放在页面加载性能上。

那么，用什么指标来衡量页面加载性能呢？最容易想到的方案是“用户平均加载时间”，事实上，我们在相当长的一段时间，也都是在使用用户平均加载时间作为性能指标。

但是，很快我们发现，这个指标有严重的问题：

- 当加载时间低于一定数字，用户体感差别不大了，我们经过一定的研究，认为这个数字大约是1秒；
- 少数超长时间加载的用户（如2G），会极大影响整个指标，即指标不能反映大多数用户的体验。

于是，基于以上分析，我们设计了一个新的指标——秒开率，即一秒之内打开的用户占用户总量的百分比。这个指标后来逐渐推广到整个公司，甚至影响到了一些业内的其它企业，现在，谈秒开率已经是个非常自然的事情了，但是当初的设计确实走了不少弯路。

技术方案

有了指标，我们就有了优化的目标，接下来，就到了技术出场的环节了。

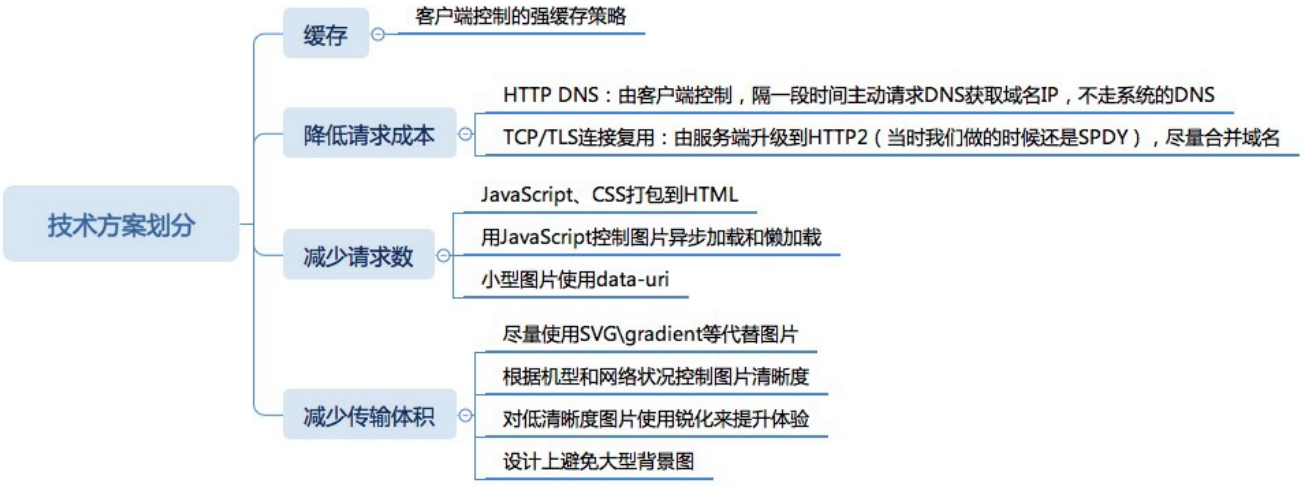
我们这里还是以加载过程为例，来讲解一下。

首先我们要简单分析一下，从输入URL后按下回车，到底发生了什么。

我们在浏览器的原理课程中，已经讲解了浏览器大致的工作过程，但是，我们必须理解几件事：

- 从域名到IP地址，需要用DNS协议查询；
- HTTP协议是用TCP传输的，所以会有TCP建立连接过程；
- 如果使用HTTPS，还有有HTTPS交换证书；
- 每个网页还有图片等请求。

从这个分析和实际试验的结果看，网页的加载时间，不但跟体积有关系，还跟请求数有很大关系，因此，我们最终设计的技术方案大约可以这样划分：



这里仅仅列出了性能优化的一部分技术方案，是我认为比较重要的部分，可以看到，这里涉及的并不仅仅是前端技术，有服务端、客户端、设计师团队，所以要想做好性能优化，绝对不能把自己限制在局部的视角，必须是整个业务一起考虑，才能有良好的收效。

执行

技术方案设计好了，它是不会自己变成线上页面的，所以，有了技术方案，我们只完成了一半的工作，接下来我们还需要一个执行过程。

执行也不简单，如果说方案主要靠技术，那么执行就是靠工程实施了。

根据公司的实际情况，工程实施可能有不同的程度，我把工程水平从低到高分成三个阶段：

- 纯管理；
- 制度化；
- 自动化。

纯行政管理，是由经理用纯粹的管理手段来执行方案，比如说，作为前端团队的Leader，我可以组织会议，要求整个团队使用我们前面谈的技术方案。

但是纯行政管理有一些问题，一方面，需要的行政资源不一定有，比如我没法强制让后端团队配合我，另一方面，纯粹的管理方式，团队本身的体验并不好，也不利于团队成长，最重要的是，纯粹管理方式容易造成执行不到位。这样的执行方式多数出现在非技术岗位。

制度化执行方式是用规则代替人的命令，指定责任人，通过培训、checklist、定期review等具体措施来保证实施。制度化执行可以极大地减轻管理工作量，一般现代互联网公司都会采用类似的方式。但是制度化执行方式还有很大成分是依靠人的主动性的，对程序员来说，还有更好的方式：自动化。

自动化的方式是在一些重要的操作路径上设置规则，针对我们的性能优化，有两个点适合做这件事：一个是把开发好的页面发布上线，另一个是开发好的页面URL投放到首页等处的链接。

在我之前的工作中，我们跟测试团队配合，开发了一套页面性能打分系统，它会自动扫面页面上的可优化点，并且跟发布平台和投放平台合作，把它加入日常机制中。现在多数公司都会采用制度化和自动化结合的执行方案。

结果评估和监控

执行完了之后，**就要向老板汇报争取升职加薪了**，还要有一定的结果总结，才是一个完整的工程实施，而且，凡是工程实施，肯定要有一定长效机制，不能优化完了退化，这些都要求有线上监控机制。

要想做线上监控，分两个部分：

- 数据采集；
- 数据展现。

数据采集部分，同样需要发布平台或者开发工具来配合，对性能数据来说，Performance API非常好用，它是浏览器记录的性能数据，一般来说，我们用统一的代码把它上传到服务器端就够用了。

数据的展现部分就比较自由了，可以用不同的数据可视化方案来展现性能数据，没有一定之规。一般的数据监控平台，会提供报警机制，对性能来说，报警需求不是特别强烈，但是也可以设置一些条件，针对秒开率特别低的网页报警。

有了监控，再配合一定制度，就可以保障整个团队产出的性能了，要注意，性能不是一个静态的事情，指标需要不断优化，技术方案还需要不断随着技术发展迭代，制度、自动化工具也需要不断改进，最终的监控平台产品也不能不做新需求，所以性能应该成为一个团队的日常工作的一部分，持续进行。

总结

今天我们学习了前端团队工程实施中的性能体系，首先我们介绍了总体思想：性能应该是基于业务和实际用户体验需求的一种工程实施，不是纯粹的技术游戏。

接下来我们分成四个步骤介绍了性能工程体系，首先介绍了现状评估和建立指标，建立指标应当从业务的角度考虑，接下来讲了技术方案设计，技术方案应当从整体角度，基于Profiling的结果分析来设计。

之后我们讲了实施，我们讲了工程实施的三个层次：纯管理、制度化、工程化，最后，我们讲了结果评估和线上监控，线上监控需要从数据采集和数据展现两个部分分别实现。

最后，留一个小问题，请你为自己的团队和业务设计一下性能的整体方案，欢迎来留言分享。

你好，我是winter。

从今天开始，我们就从前端知识学习的部分，过渡到了实践部分。这节课我来谈谈性能。

性能是个特别有意思的话题，在我之前的工作中，从入门的初级工程师到高级别的技术专家，大家都很喜欢谈性能，我以前参与晋升评审，每年总能听到很多关于性能的晋升述职。

那么，今天我就来谈谈我眼中的性能。

性能总论

> while循环快还是for循环快？
>
> |0 是不是比 Math.floor 性能好？

网上随处可以见到一类对性能的讨论。一些新人也非常热衷此类讨论。但是实际上，它们除了让你写代码的时候纠结之外，毫无意义。

为什么这样讲呢？我想讲一个小故事。

从前有个工程师，特别注重代码细节，有一天他发现系统中的一段代码写的性能很差，因此，他用汇编重写了整段代码，执行效率足足提升了三倍。但是最后，大家发现，用户反馈性能丝毫没有提高，因为他优化的那个进程名字叫“System Idle”。

所以你看，性能优化不能只着眼于局部的代码。这里，我要提出一个我的观点：**一切没有profiling的性能都是要流氓**。凡是真正有价值的性能优化，必定是从端到端的业务场景建立体系来考虑的。

在我的认识中，性能体系的建立可以分成以下几部分：

- 现状评估和建立指标；
- 技术方案；
- 执行；
- 结果评估和监控。

下面，我就来为你一一讲解。

现状评估和建立指标

要想做好性能优化，正确地评估现状和建立指标是最关键的一步，它又往往是会被轻视的一步。

作为一个工程师，指标又要考虑两个因素。一方面，对用户来说，什么样的性能指标能更好地评估它的体验？另一方面，对公司来说，什么样的指标会影响业务价值呢？

在我公布答案之前，我希望你能思考一下，你所负责的业务，是否有前端性能指标？它是否能够满足我上面提到的两个要求？

在我之前的工作中，整个用了长达一年的时间来探索，才找到了合适的指标，并且回答好了两个问题。

性能问题可以分成很多方面，最重要的几个点是：

- 页面加载性能；
- 动画与操作性能；
- 内存、电量消耗。

注意，这里我们仅仅是对“性能”两个字的分析和解读，在对大量的用户数据分析后，我们发现，其实这三部分中，“页面加载性能”跟用户的流失率有非常强的关联性，而用户流失率，正是公司业务非常看重的指标。

因此，在开始阶段，我们决定把性能优化的重点放在页面加载性能上。

那么，用什么指标来衡量页面加载性能呢？最容易想到的方案是“用户平均加载时间”，事实上，我们在相当长的一段时间，也都是在使用用户平均加载时间作为性能指标。

但是，很快我们发现，这个指标有严重的问题：

- 当加载时间低于一定数字，用户体感差别不大了，我们经过一定的研究，认为这个数字大约是1秒；
- 少数超长时间加载的用户（如2G），会极大影响整个指标，即指标不能反映大多数用户的体验。

于是，基于以上分析，我们设计了一个新的指标——秒开率，即一秒之内打开的用户占用户总量的百分比。这个指标后来逐渐推广到整个公司，甚至影响到了一些业内的其它企业，现在，谈秒开率已经是个非常自然的事情了，但是当初的设计确实走了不少弯路。

技术方案

有了指标，我们就有了优化的目标，接下来，就到了技术出场的环节了。

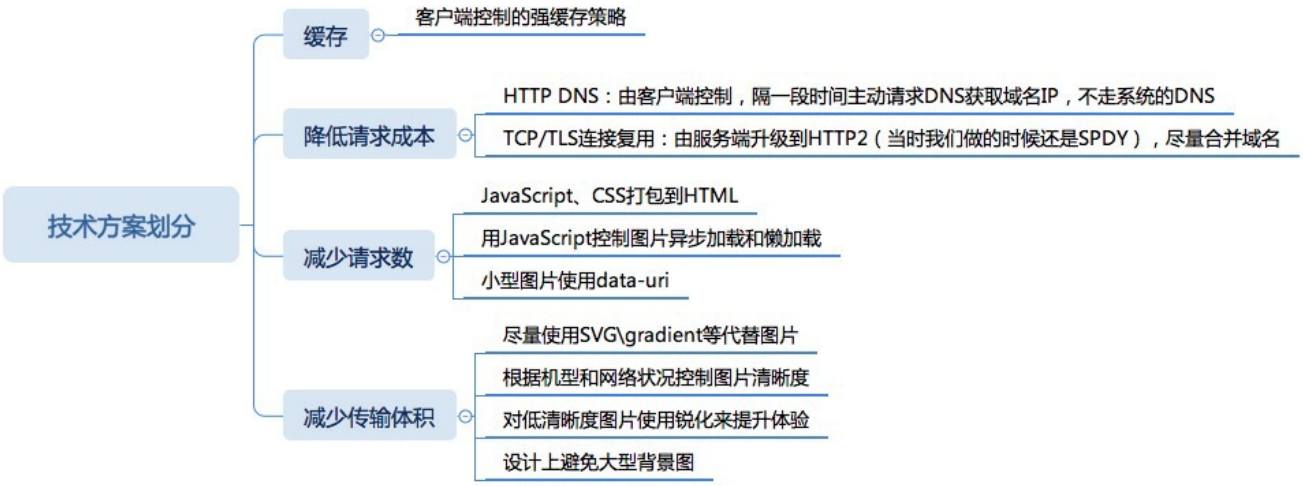
我们这里还是以加载过程为例，来讲解一下。

首先我们要简单分析一下，从输入URL后按下回车，到底发生了什么。

我们在浏览器的原理课程中，已经讲解了浏览器大致的工作过程，但是，我们必须理解几件事：

- 从域名到IP地址，需要用DNS协议查询；
- HTTP协议是用TCP传输的，所以会有TCP建立连接过程；
- 如果使用HTTPS，还有有HTTPS交换证书；
- 每个网页还有图片等请求。

从这个分析和实际试验的结果看，网页的加载时间，不但跟体积有关系，还跟请求数有很大关系，因此，我们最终设计的技术方案大约可以这样划分：



这里仅仅列出了性能优化的一部分技术方案，是我认为比较重要的部分，可以看到，这里涉及的并不仅仅是前端技术，有服务端、客户端、设计师团队，所以要想做好性能优化，绝对不能把自己限制在局部的视角，必须是整个业务一起考虑，才能有良好的收效。

执行

技术方案设计好了，它是不会自己变成线上页面的，所以，有了技术方案，我们只完成了一半的工作，接下来我们还需要一个执行过程。

执行也不简单，如果说方案主要靠技术，那么执行就是靠工程实施了。

根据公司的实际情况，工程实施可能有不同的程度，我把工程水平从低到高分成三个阶段：

- 纯管理；
- 制度化；
- 自动化。

纯行政管理，是由经理用纯粹的管理手段来执行方案，比如说，作为前端团队的Leader，我可以组织会议，要求整个团队使用我们前面谈的技术方案。

但是纯行政管理有一些问题，一方面，需要的行政资源不一定有，比如我没法强制让后端团队配合我，另一方面，纯粹的管理方式，团队本身的体验并不好，也不利于团队成长，最重要的是，纯粹管理方式容易造成执行不到位。这样的执行方式多数出现在非技术岗位。

制度化执行方式是用规则代替人的命令，指定责任人，通过培训、checklist、定期review等具体措施来保证实施。制度化执行可以极大地减轻管理工作量，一般现代互联网公司都会采用类似的方式。但是制度化执行方式还有很大成分是依靠人的主动性的，对程序员来说，还有更好的方式：自动化。

自动化的方式是在一些重要的操作路径上设置规则，针对我们的性能优化，有两个点适合做这件事：一个是把开发好的页面发布上线，另一个是开发好的页面URL投放到首页等处的链接。

在我之前的工作中，我们跟测试团队配合，开发了一套页面性能打分系统，它会自动扫面页面上的可优化点，并且跟发布平台和投放平台合作，把它加入日常机制中。现在多数公司都会采用制度化和自动化结合的执行方案。

结果评估和监控

执行完了之后，**就要向老板汇报争取升职加薪了**，还要有一定的结果总结，才是一个完整的工程实施，而且，凡是工程实施，肯定要有一定长效机制，不能优化完了退化，这些都要求有线上监控机制。

要想做线上监控，分两个部分：

- 数据采集；
- 数据展现。

数据采集部分，同样需要发布平台或者开发工具来配合，对性能数据来说，Performance API非常好用，它是浏览器记录的性能数据，一般来说，我们用统一的代码把它上传到服务器端就够用了。

数据的展现部分就比较自由了，可以用不同的数据可视化方案来展现性能数据，没有一定之规。一般的数据监控平台，会提供报警机制，对性能来说，报警需求不是特别强烈，但是也可以设置一些条件，针对秒开率特别低的网页报警。

有了监控，再配合一定制度，就可以保障整个团队产出的性能了，要注意，性能不是一个静态的事情，指标需要不断优化，技术方案还需要不断随着技术发展迭代，制度、自动化工具也需要不断改进，最终的监控平台产品也不能不做新需求，所以性能应该成为一个团队的日常工作的一部分，持续进行。

总结

今天我们学习了前端团队工程实施中的性能体系，首先我们介绍了总体思想：性能应该是基于业务和实际用户体验需求的一种工程实施，不是纯粹的技术游戏。

接下来我们分成四个步骤介绍了性能工程体系，首先介绍了现状评估和建立指标，建立指标应当从业务的角度考虑，接下来讲了技术方案设计，技术方案应当从整体角度，基于Profiling的结果分析来设计。

之后我们讲了实施，我们讲了工程实施的三个层次：纯管理、制度化、工程化，最后，我们讲了结果评估和线上监控，线上监控需要从数据采集和数据展现两个部分分别实现。

最后，留一个小问题，请你为自己的团队和业务设计一下性能的整体方案，欢迎来留言分享。

你好，我是winter。

从今天开始，我们就从前端知识学习的部分，过渡到了实践部分。这节课我来谈谈性能。

性能是个特别有意思的话题，在我之前的工作中，从入门的初级工程师到高级别的技术专家，大家都很喜欢谈性能，我以前参与晋升评审，每年总能听到很多关于性能的晋升述职。

那么，今天我就来谈谈我眼中的性能。

性能总论

> while循环快还是for循环快？
>
> |0 是不是比 Math.floor 性能好？

网上随处可以见到一类对性能的讨论。一些新人也非常热衷此类讨论。但是实际上，它们除了让你写代码的时候纠结之外，毫无意义。

为什么这样讲呢？我想讲一个小故事。

从前有个工程师，特别注重代码细节，有一天他发现系统中的一段代码写的性能很差，因此，他用汇编重写了整段代码，执行效率足足提升了三倍。但是最后，大家发现，用户反馈性能丝毫没有提高，因为他优化的那个进程名字叫“System Idle”。

所以你看，性能优化不能只着眼于局部的代码。这里，我要提出一个我的观点：**一切没有profiling的性能都是耍流氓**。凡是真正有价值的性能优化，必定是从端到端的业务场景建立体系来考虑的。

在我的认识中，性能体系的建立可以分成以下几部分：

- 现状评估和建立指标；
- 技术方案；
- 执行；
- 结果评估和监控。

下面，我就来为你一一讲解。

现状评估和建立指标

要想做好性能优化，正确地评估现状和建立指标是最关键的一步，它又往往是会被轻视的一步。

作为一个工程师，指标又要考虑两个因素。一方面，对用户来说，什么样的性能指标能更好地评估它的体验？另一方面，对公司来说，什么样的指标会影响业务价值呢？

在我公布答案之前，我希望你能思考一下，你所负责的业务，是否有前端性能指标？它是否能够满足我上面提到的两个要求？

在我之前的工作中，整个用了长达一年的时间来探索，才找到了合适的指标，并且回答好了两个问题。

性能问题可以分成很多方面，最重要的几个点是：

- 页面加载性能；
- 动画与操作性能；
- 内存、电量消耗。

注意，这里我们仅仅是对“性能”两个字的分析和解读，在对大量的用户数据分析后，我们发现，其实这三部分中，“页面加载性能”跟用户的流失率有非常强的关联性，而用户流失率，正是公司业务非常看重的指标。

因此，在开始阶段，我们决定把性能优化的重点放在页面加载性能上。

那么，用什么指标来衡量页面加载性能呢？最容易想到的方案是“用户平均加载时间”，事实上，我们在相当长的一段时间，也都是在使用用户平均加载时间作为性能指标。

但是，很快我们发现，这个指标有严重的问题：

- 当加载时间低于一定数字，用户体感差别不大了，我们经过一定的研究，认为这个数字大约是1秒；
- 少数超长时间加载的用户（如2G），会极大影响整个指标，即指标不能反映大多数用户的体验。

于是，基于以上分析，我们设计了一个新的指标——秒开率，即一秒之内打开的用户占用户总量的百分比。这个指标后来逐渐推广到整个公司，甚至影响到了一些业内的其它企业，现在，谈秒开率已经是个非常自然的事情了，但是当初的设计确实走了不少弯路。

技术方案

有了指标，我们就有了优化的目标，接下来，就到了技术出场的环节了。

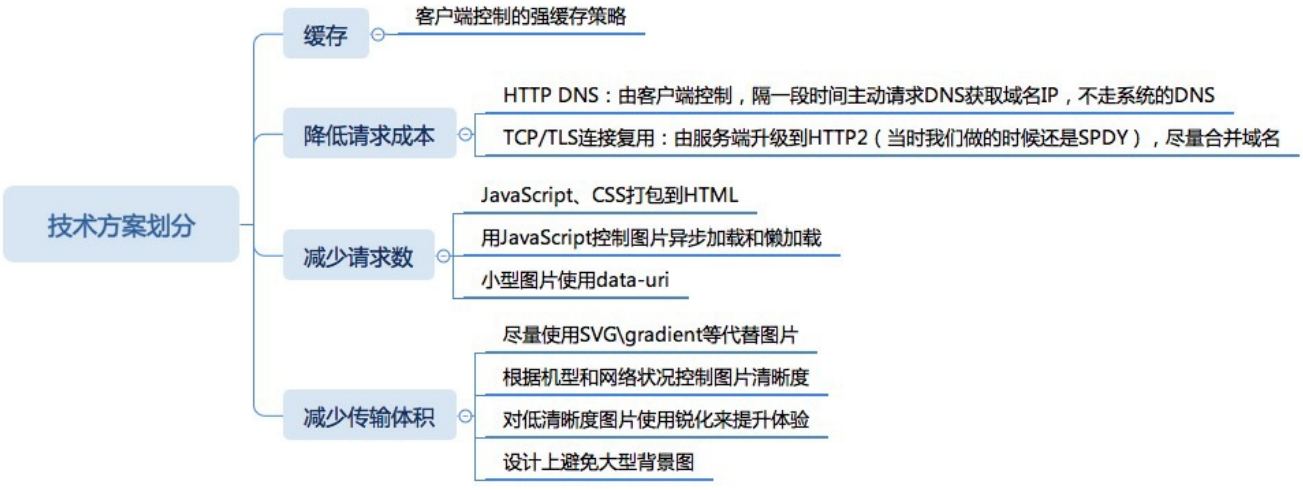
我们这里还是以加载过程为例，来讲解一下。

首先我们要简单分析一下，从输入URL后按下回车，到底发生了什么。

我们在浏览器的原理课程中，已经讲解了浏览器大致的工作过程，但是，我们必须理解几件事：

- 从域名到IP地址，需要用DNS协议查询；
- HTTP协议是用TCP传输的，所以会有TCP建立连接过程；
- 如果使用HTTPS，还有有HTTPS交换证书；
- 每个网页还有图片等请求。

从这个分析和实际试验的结果看，网页的加载时间，不但跟体积有关系，还跟请求数有很大关系，因此，我们最终设计的技术方案大约可以这样划分：



这里仅仅列出了性能优化的一部分技术方案，是我认为比较重要的部分，可以看到，这里涉及的并不仅仅是前端技术，有服务端、客户端、设计师团队，所以要想做好性能优化，绝对不能把自己限制在局部的视角，必须是整个业务一起考虑，才能有良好的收效。

执行

技术方案设计好了，它是不会自己变成线上页面的，所以，有了技术方案，我们只完成了一半的工作，接下来我们还需要一个执行过程。

执行也不简单，如果说方案主要靠技术，那么执行就是靠工程实施了。

根据公司的实际情况，工程实施可能有不同的程度，我把工程水平从低到高分成三个阶段：

- 纯管理；
- 制度化；
- 自动化。

纯行政管理，是由经理用纯粹的管理手段来执行方案，比如说，作为前端团队的Leader，我可以组织会议，要求整个团队使用我们前面谈的技术方案。

但是纯行政管理有一些问题，一方面，需要的行政资源不一定有，比如我没法强制让后端团队配合我，另一方面，纯粹的管理方式，团队本身的体验并不好，也不利于团队成长，最重要的是，纯粹管理方式容易造成执行不到位。这样的执行方式多数出现在非技术岗位。

制度化执行方式是用规则代替人的命令，指定责任人，通过培训、checklist、定期review等具体措施来保证实施。制度化执行可以极大地减轻管理工作量，一般现代互联网公司都会采用类似的方式。但是制度化执行方式还有很大成分是依靠人的主动性的，对程序员来说，还有更好的方式：自动化。

自动化的方式是在一些重要的操作路径上设置规则，针对我们的性能优化，有两个点适合做这件事：一个是把开发好的页面发布上线，另一个是开发好的页面URL投放到首页等处的链接。

在我之前的工作中，我们跟测试团队配合，开发了一套页面性能打分系统，它会自动扫面页面上的可优化点，并且跟发布平台和投放平台合作，把它加入日常机制中。现在多数公司都会采用制度化和自动化结合的执行方案。

结果评估和监控

执行完了之后，**就要向老板汇报争取升职加薪了**，还要有一定的结果总结，才是一个完整的工程实施，而且，凡是工程实施，肯定要有一定长效机制，不能优化完了退化，这些都要求有线上监控机制。

要想做线上监控，分两个部分：

- 数据采集；
- 数据展现。

数据采集部分，同样需要发布平台或者开发工具来配合，对性能数据来说，Performance API非常好用，它是浏览器记录的性能数据，一般来说，我们用统一的代码把它上传到服务器端就够用了。

数据的展现部分就比较自由了，可以用不同的数据可视化方案来展现性能数据，没有一定之规。一般的数据监控平台，会提供报警机制，对性能来说，报警需求不是特别强烈，但是也可以设置一些条件，针对秒开率特别低的网页报警。

有了监控，再配合一定制度，就可以保障整个团队产出的性能了，要注意，性能不是一个静态的事情，指标需要不断优化，技术方案还需要不断随着技术发展迭代，制度、自动化工具也需要不断改进，最终的监控平台产品也不能不做新需求，所以性能应该成为一个团队的日常工作的一部分，持续进行。

总结

今天我们学习了前端团队工程实施中的性能体系，首先我们介绍了总体思想：性能应该是基于业务和实际用户体验需求的一种工程实施，不是纯粹的技术游戏。

接下来我们分成四个步骤介绍了性能工程体系，首先介绍了现状评估和建立指标，建立指标应当从业务的角度考虑，接下来讲了技术方案设计，技术方案应当从整体角度，基于Profiling的结果分析来设计。

之后我们讲了实施，我们讲了工程实施的三个层次：纯管理、制度化、工程化，最后，我们讲了结果评估和线上监控，线上监控需要从数据采集和数据展现两个部分分别实现。

最后，留一个小问题，请你为自己的团队和业务设计一下性能的整体方案，欢迎来留言分享。

你好，我是winter。

从今天开始，我们就从前端知识学习的部分，过渡到了实践部分。这节课我来谈谈性能。

性能是个特别有意思的话题，在我之前的工作中，从入门的初级工程师到高级别的技术专家，大家都很喜欢谈性能，我以前参与晋升评审，每年总能听到很多关于性能的晋升述职。

那么，今天我就来谈谈我眼中的性能。

性能总论

> while循环快还是for循环快？
>
> |0 是不是比 Math.floor 性能好？

网上随处可以见到一类对性能的讨论。一些新人也非常热衷此类讨论。但是实际上，它们除了让你写代码的时候纠结之外，毫无意义。

为什么这样讲呢？我想讲一个小故事。

从前有个工程师，特别注重代码细节，有一天他发现系统中的一段代码写的性能很差，因此，他用汇编重写了整段代码，执行效率足足提升了三倍。但是最后，大家发现，用户反馈性能丝毫没有提高，因为他优化的那个进程名字叫“System Idle”。

所以你看，性能优化不能只着眼于局部的代码。这里，我要提出一个我的观点：**一切没有profiling的性能都是耍流氓**。凡是真正有价值的性能优化，必定是从端到端的业务场景建立体系来考虑的。

在我的认识中，性能体系的建立可以分成以下几部分：

- 现状评估和建立指标；
- 技术方案；
- 执行；
- 结果评估和监控。

下面，我就来为你一一讲解。

现状评估和建立指标

要想做好性能优化，正确地评估现状和建立指标是最关键的一步，它又往往是会被轻视的一步。

作为一个工程师，指标又要考虑两个因素。一方面，对用户来说，什么样的性能指标能更好地评估它的体验？另一方面，对公司来说，什么样的指标会影响业务价值呢？

在我公布答案之前，我希望你能思考一下，你所负责的业务，是否有前端性能指标？它是否能够满足我上面提到的两个要求？

在我之前的工作中，整个用了长达一年的时间来探索，才找到了合适的指标，并且回答好了两个问题。

性能问题可以分成很多方面，最重要的几个点是：

- 页面加载性能；
- 动画与操作性能；
- 内存、电量消耗。

注意，这里我们仅仅是对“性能”两个字的分析和解读，在对大量的用户数据分析后，我们发现，其实这三部分中，“页面加载性能”跟用户的流失率有非常强的关联性，而用户流失率，正是公司业务非常看重的指标。

因此，在开始阶段，我们决定把性能优化的重点放在页面加载性能上。

那么，用什么指标来衡量页面加载性能呢？最容易想到的方案是“用户平均加载时间”，事实上，我们在相当长的一段时间，也都是在使用用户平均加载时间作为性能指标。

但是，很快我们发现，这个指标有严重的问题：

- 当加载时间低于一定数字，用户体感差别不大了，我们经过一定的研究，认为这个数字大约是1秒；
- 少数超长时间加载的用户（如2G），会极大影响整个指标，即指标不能反映大多数用户的体验。

于是，基于以上分析，我们设计了一个新的指标——秒开率，即一秒之内打开的用户占用户总量的百分比。这个指标后来逐渐推广到整个公司，甚至影响到了一些业内的其它企业，现在，谈秒开率已经是个非常自然的事情了，但是当初的设计确实走了不少弯路。

技术方案

有了指标，我们就有了优化的目标，接下来，就到了技术出场的环节了。

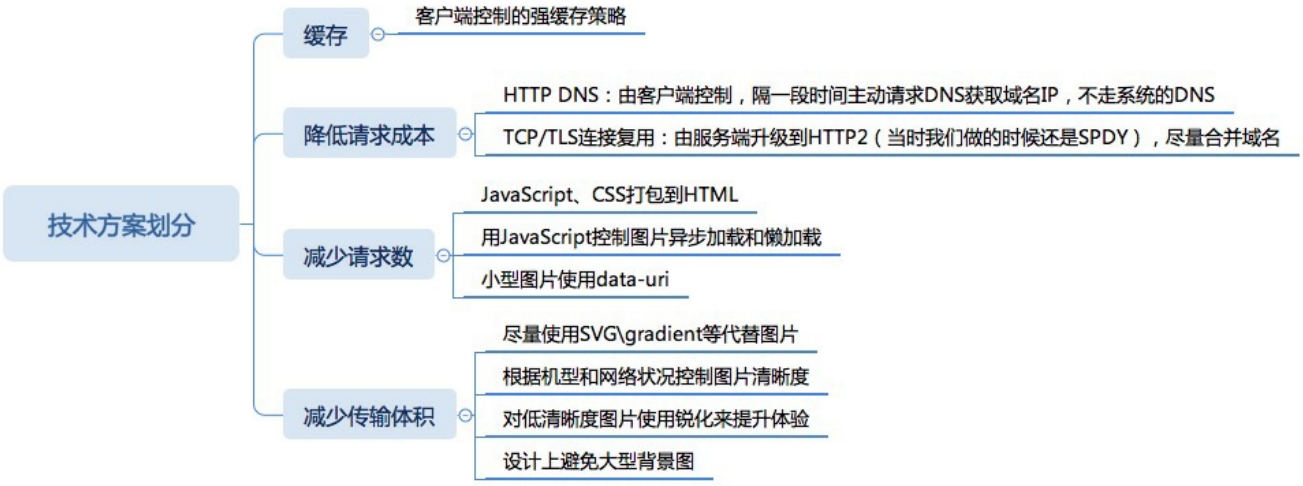
我们这里还是以加载过程为例，来讲解一下。

首先我们要简单分析一下，从输入URL后按下回车，到底发生了什么。

我们在浏览器的原理课程中，已经讲解了浏览器大致的工作过程，但是，我们必须理解几件事：

- 从域名到IP地址，需要用DNS协议查询；
- HTTP协议是用TCP传输的，所以会有TCP建立连接过程；
- 如果使用HTTPS，还有有HTTPS交换证书；
- 每个网页还有图片等请求。

从这个分析和实际试验的结果看，网页的加载时间，不但跟体积有关系，还跟请求数有很大关系，因此，我们最终设计的技术方案大约可以这样划分：



这里仅仅列出了性能优化的一部分技术方案，是我认为比较重要的部分，可以看到，这里涉及的并不仅仅是前端技术，有服务端、客户端、设计师团队，所以要想做好性能优化，绝对不能把自己限制在局部的视角，必须是整个业务一起考虑，才能有良好的收效。

执行

技术方案设计好了，它是不会自己变成线上页面的，所以，有了技术方案，我们只完成了一半的工作，接下来我们还需要一个执行过程。

执行也不简单，如果说方案主要靠技术，那么执行就是靠工程实施了。

根据公司的实际情况，工程实施可能有不同的程度，我把工程水平从低到高分成三个阶段：

- 纯管理；
- 制度化；
- 自动化。

纯行政管理，是由经理用纯粹的管理手段来执行方案，比如说，作为前端团队的Leader，我可以组织会议，要求整个团队使用我们前面谈的技术方案。

但是纯行政管理有一些问题，一方面，需要的行政资源不一定有，比如我没法强制让后端团队配合我，另一方面，纯粹的管理方式，团队本身的体验并不好，也不利于团队成长，最重要的是，纯粹管理方式容易造成执行不到位。这样的执行方式多数出现在非技术岗位。

制度化执行方式是用规则代替人的命令，指定责任人，通过培训、checklist、定期review等具体措施来保证实施。制度化执行可以极大地减轻管理工作量，一般现代互联网公司都会采用类似的方式。但是制度化执行方式还有很大成分是依靠人的主动性的，对程序员来说，还有更好的方式：自动化。

自动化的方式是在一些重要的操作路径上设置规则，针对我们的性能优化，有两个点适合做这件事：一个是把开发好的页面发布上线，另一个是开发好的页面URL投放到首页等处的链接。

在我之前的工作中，我们跟测试团队配合，开发了一套页面性能打分系统，它会自动扫面页面上的可优化点，并且跟发布平台和投放平台合作，把它加入日常机制中。现在多数公司都会采用制度化和自动化结合的执行方案。

结果评估和监控

执行完了之后，**就要向老板汇报争取升职加薪了**，还要有一定的结果总结，才是一个完整的工程实施，而且，凡是工程实施，肯定要有一定长效机制，不能优化完了退化，这些都要求有线上监控机制。

要想做线上监控，分两个部分：

- 数据采集；
- 数据展现。

数据采集部分，同样需要发布平台或者开发工具来配合，对性能数据来说，Performance API非常好用，它是浏览器记录的性能数据，一般来说，我们用统一的代码把它上传到服务器端就够用了。

数据的展现部分就比较自由了，可以用不同的数据可视化方案来展现性能数据，没有一定之规。一般的数据监控平台，会提供报警机制，对性能来说，报警需求不是特别强烈，但是也可以设置一些条件，针对秒开率特别低的网页报警。

有了监控，再配合一定制度，就可以保障整个团队产出的性能了，要注意，性能不是一个静态的事情，指标需要不断优化，技术方案还需要不断随着技术发展迭代，制度、自动化工具也需要不断改进，最终的监控平台产品也不能不做新需求，所以性能应该成为一个团队的日常工作的一部分，

持续进行。

总结

今天我们学习了前端团队工程实施中的性能体系，首先我们介绍了总体思想：性能应该是基于业务和实际用户体验需求的一种工程实施，不是纯粹的技术游戏。

接下来我们分成四个步骤介绍了性能工程体系，首先介绍了现状评估和建立指标，建立指标应当从业务的角度考虑，接下来讲了技术方案设计，技术方案应当从整体角度，基于Profiling的结果分析来设计。

之后我们讲了实施，我们讲了工程实施的三个层次：纯管理、制度化、工程化，最后，我们讲了结果评估和线上监控，线上监控需要从数据采集和数据展现两个部分分别实现。

最后，留一个小问题，请你为自己的团队和业务设计一下性能的整体方案，欢迎来留言分享。

你好，我是winter。

从今天开始，我们就从前端知识学习的部分，过渡到了实践部分。这节课我来谈谈性能。

性能是个特别有意思的话题，在我之前的工作中，从入门的初级工程师到高级别的技术专家，大家都很喜欢谈性能，我以前参与晋升评审，每年总能听到很多关于性能的晋升述职。

那么，今天我就来谈谈我眼中的性能。

性能总论

> while循环快还是for循环快？
>
> |0 是不是比 Math.floor 性能好？

网上随处可以见到一类对性能的讨论。一些新人也非常热衷此类讨论。但是实际上，它们除了让你写代码的时候纠结之外，毫无意义。

为什么这样讲呢？我想讲一个小故事。

从前有个工程师，特别注重代码细节，有一天他发现系统中的一段代码写的性能很差，因此，他用汇编重写了整段代码，执行效率足足提升了三倍。但是最后，大家发现，用户反馈性能丝毫没有提高，因为他优化的那个进程名字叫“System Idle”。

所以你看，性能优化不能只着眼于局部的代码。这里，我要提出一个我的观点：**一切没有profiling的性能都是耍流氓**。凡是真正有价值的性能优化，必定是从端到端的业务场景建立体系来考虑的。

在我的认识中，性能体系的建立可以分成以下几部分：

- 现状评估和建立指标；
- 技术方案；
- 执行；
- 结果评估和监控。

下面，我就来为你一一讲解。

现状评估和建立指标

要想做好性能优化，正确地评估现状和建立指标是最关键的一步，它又往往是会被轻视的一步。

作为一个工程师，指标又要考虑两个因素。一方面，对用户来说，什么样的性能指标能更好地评估它的体验？另一方面，对公司来说，什么样的指标会影响业务价值呢？

在我公布答案之前，我希望你能思考一下，你所负责的业务，是否有前端性能指标？它是否能够满足我上面提到的两个要求？

在我之前的工作中，整个用了长达一年的时间来探索，才找到了合适的指标，并且回答好了两个问题。

性能问题可以分成很多方面，最重要的几个点是：

- 页面加载性能；
- 动画与操作性能；
- 内存、电量消耗。

注意，这里我们仅仅是对“性能”两个字的分析和解读，在对大量的用户数据分析后，我们发现，其实这三部分中，“页面加载性能”跟用户的流失率有非常强的关联性，而用户流失率，正是公司业务非常看重的指标。

因此，在开始阶段，我们决定把性能优化的重点放在页面加载性能上。

那么，用什么指标来衡量页面加载性能呢？最容易想到的方案是“用户平均加载时间”，事实上，我们在相当长的一段时间，也都是在使用用户平均加载时间作为性能指标。

但是，很快我们发现，这个指标有严重的问题：

- 当加载时间低于一定数字，用户体感差别不大了，我们经过一定的研究，认为这个数字大约是1秒；
- 少数超长时间加载的用户（如2G），会极大影响整个指标，即指标不能反映大多数用户的体验。

于是，基于以上分析，我们设计了一个新的指标——秒开率，即一秒之内打开的用户占用户总量的百分比。这个指标后来逐渐推广到整个公司，甚至影响到了一些业内的其它企业，现在，谈秒开率已经是个非常自然的事情了，但是当初的设计确实走了不少弯路。

技术方案

有了指标，我们就有了优化的目标，接下来，就到了技术出场的环节了。

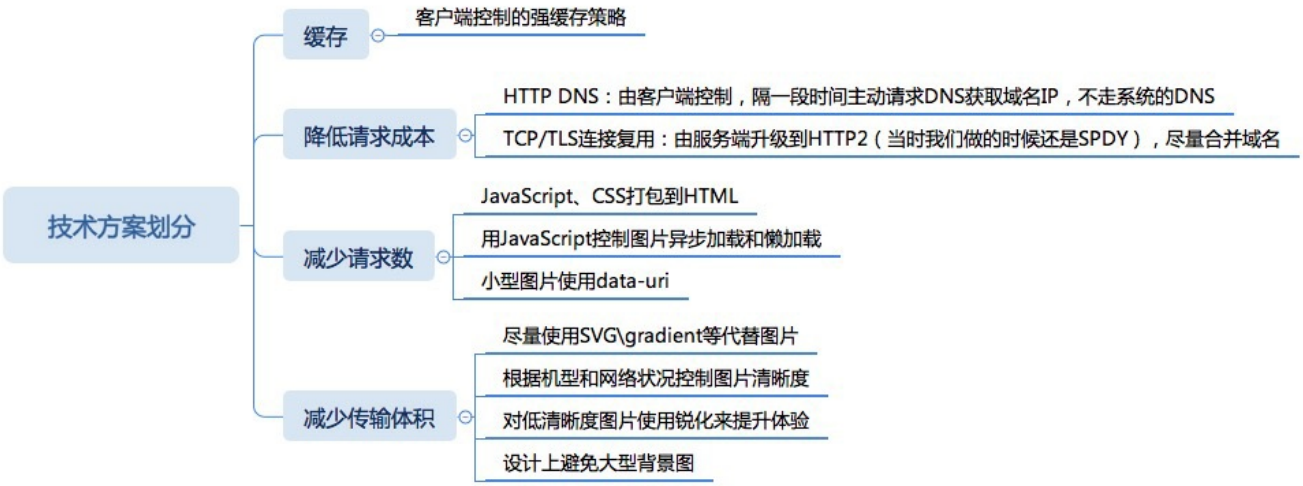
我们这里还是以加载过程为例，来讲解一下。

首先我们要简单分析一下，从输入URL后按下回车，到底发生了什么。

我们在浏览器的原理课程中，已经讲解了浏览器大致的工作过程，但是，我们必须理解几件事：

- 从域名到IP地址，需要用DNS协议查询；
- HTTP协议是用TCP传输的，所以会有TCP建立连接过程；
- 如果使用HTTPS，还有有HTTPS交换证书；
- 每个网页还有图片等请求。

从这个分析和实际试验的结果看，网页的加载时间，不但跟体积有关系，还跟请求数有很大关系，因此，我们最终设计的技术方案大约可以这样划分：



这里仅仅列出了性能优化的一部分技术方案，是我认为比较重要的部分，可以看到，这里涉及的并不仅仅是前端技术，有服务端、客户端、设计师团队，所以要想做好性能优化，绝对不能把自己限制在局部的视角，必须是整个业务一起考虑，才能有良好的收效。

执行

技术方案设计好了，它是不会自己变成线上页面的，所以，有了技术方案，我们只完成了一半的工作，接下来我们还需要一个执行过程。

执行也不简单，如果说方案主要靠技术，那么执行就是靠工程实施了。

根据公司的实际情况，工程实施可能有不同的程度，我把工程水平从低到高分成三个阶段：

- 纯管理；
- 制度化；
- 自动化。

纯行政管理，是由经理用纯粹的管理手段来执行方案，比如说，作为前端团队的Leader，我可以组织会议，要求整个团队使用我们前面谈的技术方案。

但是纯行政管理有一些问题，一方面，需要的行政资源不一定有，比如我没法强制让后端团队配合我，另一方面，纯粹的管理方式，团队本身的体验并不好，也不利于团队成长，最重要的是，纯粹管理方式容易造成执行不到位。这样的执行方式多数出现在非技术岗位。

制度化执行方式是用规则代替人的命令，指定责任人，通过培训、checklist、定期review等具体措施来保证实施。制度化执行可以极大地减轻管理工作量，一般现代互联网公司都会采用类似的方式。但是制度化执行方式还有很大成分是依靠人的主动性的，对程序员来说，还有更好的方式：自动化。

自动化的方式是在一些重要的操作路径上设置规则，针对我们的性能优化，有两个点适合做这件事：一个是把开发好的页面发布上线，另一个是开发好的页面URL投放到首页等处的链接。

在我之前的工作中，我们跟测试团队配合，开发了一套页面性能打分系统，它会自动扫面页面上的可优化点，并且跟发布平台和投放平台合作，把它加入日常机制中。现在多数公司都会采用制度化和自动化结合的执行方案。

结果评估和监控

执行完了之后，**就要向老板汇报争取升职加薪了**，还要有一定的结果总结，才是一个完整的工程实施，而且，凡是工程实施，肯定要有一定长效机制，不能优化完了退化，这些都要求有线上监控机制。

要想做线上监控，分两个部分：

- 数据采集；
- 数据展现。

数据采集部分，同样需要发布平台或者开发工具来配合，对性能数据来说，Performance API非常好用，它是浏览器记录的性能数据，一般来说，我们用统一的代码把它上传到服务器端就够用了。

数据的展现部分就比较自由了，可以用不同的数据可视化方案来展现性能数据，没有一定之规。一般的数据监控平台，会提供报警机制，对性能来说，报警需求不是特别强烈，但是也可以设置一些条件，针对秒开率特别低的网页报警。

有了监控，再配合一定制度，就可以保障整个团队产出的性能了，要注意，性能不是一个静态的事情，指标需要不断优化，技术方案还需要不断随着技术发展迭代，制度、自动化工具也需要不断改进，最终的监控平台产品也不能不做新需求，所以性能应该成为一个团队的日常工作的一部分，持续进行。

总结

今天我们学习了前端团队工程实施中的性能体系，首先我们介绍了总体思想：性能应该是基于业务和实际用户体验需求的一种工程实施，不是纯粹的技术游戏。

接下来我们分成四个步骤介绍了性能工程体系，首先介绍了现状评估和建立指标，建立指标应当从业务的角度考虑，接下来讲了技术方案设计，技术方案应当从整体角度，基于Profiling的结果分析来设计。

之后我们讲了实施，我们讲了工程实施的三个层次：纯管理、制度化、工程化，最后，我们讲了结果评估和线上监控，线上监控需要从数据采集和数据展现两个部分分别实现。

最后，留一个小问题，请你为自己的团队和业务设计一下性能的整体方案，欢迎来留言分享。

你好，我是winter。

从今天开始，我们就从前端知识学习的部分，过渡到了实践部分。这节课我来谈谈性能。

性能是个特别有意思的话题，在我之前的工作中，从入门的初级工程师到高级别的技术专家，大家都很喜欢谈性能，我以前参与晋升评审，每年总能听到很多关于性能的晋升述职。

那么，今天我就来谈谈我眼中的性能。

性能总论

> while循环快还是for循环快？
>
> |0 是不是比 Math.floor 性能好？

网上随处可以见到一类对性能的讨论。一些新人也非常热衷此类讨论。但是实际上，它们除了让你写代码的时候纠结之外，毫无意义。

为什么这样讲呢？我想讲一个小故事。

从前有个工程师，特别注重代码细节，有一天他发现系统中的一段代码写的性能很差，因此，他用汇编重写了整段代码，执行效率足足提升了三倍。但是最后，大家发现，用户反馈性能丝毫没有提高，因为他优化的那个进程名字叫“System Idle”。

所以你看，性能优化不能只着眼于局部的代码。这里，我要提出一个我的观点：**一切没有profiling的性能都是耍流氓**。凡是真正有价值的性能优化，必定是从端到端的业务场景建立体系来考虑的。

在我的认识中，性能体系的建立可以分成以下几部分：

- 现状评估和建立指标；
- 技术方案；
- 执行；
- 结果评估和监控。

下面，我就来为你一一讲解。

现状评估和建立指标

要想做好性能优化，正确地评估现状和建立指标是最关键的一步，它又往往是会被轻视的一步。

作为一个工程师，指标又要考虑两个因素。一方面，对用户来说，什么样的性能指标能更好地评估它的体验？另一方面，对公司来说，什么样的指标会影响业务价值呢？

在我公布答案之前，我希望你能思考一下，你所负责的业务，是否有前端性能指标？它是否能够满足我上面提到的两个要求？

在我之前的工作中，整个用了长达一年的时间来探索，才找到了合适的指标，并且回答好了两个问题。

性能问题可以分成很多方面，最重要的几个点是：

- 页面加载性能；
- 动画与操作性能；
- 内存、电量消耗。

注意，这里我们仅仅是对“性能”两个字的分析和解读，在对大量的用户数据分析后，我们发现，其实这三部分中，“页面加载性能”跟用户的流失率有非常强的关联性，而用户流失率，正是公司业务非常看重的指标。

因此，在开始阶段，我们决定把性能优化的重点放在页面加载性能上。

那么，用什么指标来衡量页面加载性能呢？最容易想到的方案是“用户平均加载时间”，事实上，我们在相当长的一段时间，也都是在使用用户平均加载时间作为性能指标。

但是，很快我们发现，这个指标有严重的问题：

- 当加载时间低于一定数字，用户体感差别不大了，我们经过一定的研究，认为这个数字大约是1秒；
- 少数超长时间加载的用户（如2G），会极大影响整个指标，即指标不能反映大多数用户的体验。

于是，基于以上分析，我们设计了一个新的指标——秒开率，即一秒之内打开的用户占用户总量的百分比。这个指标后来逐渐推广到整个公司，甚至影响到了一些业内的其它企业，现在，谈秒开率已经是个非常自然的事情了，但是当初的设计确实走了不少弯路。

技术方案

有了指标，我们就有了优化的目标，接下来，就到了技术出场的环节了。

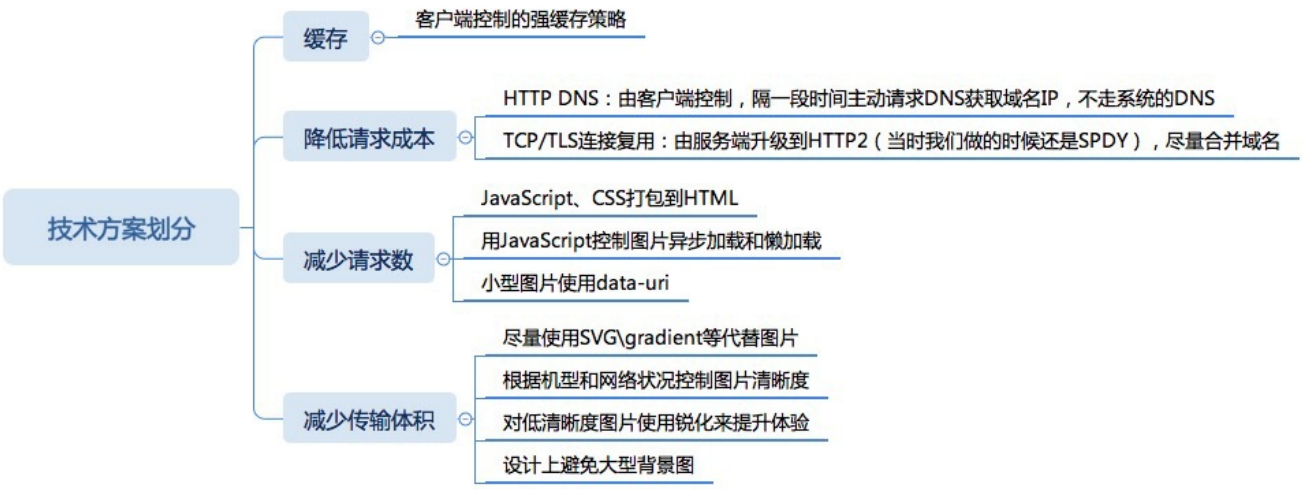
我们这里还是以加载过程为例，来讲解一下。

首先我们要简单分析一下，从输入URL后按下回车，到底发生了什么。

我们在浏览器的原理课程中，已经讲解了浏览器大致的工作过程，但是，我们必须理解几件事：

- 从域名到IP地址，需要用DNS协议查询；
- HTTP协议是用TCP传输的，所以会有TCP建立连接过程；
- 如果使用HTTPS，还有有HTTPS交换证书；
- 每个网页还有图片等请求。

从这个分析和实际试验的结果看，网页的加载时间，不但跟体积有关系，还跟请求数有很大关系，因此，我们最终设计的技术方案大约可以这样划分：



这里仅仅列出了性能优化的一部分技术方案，是我认为比较重要的部分，可以看到，这里涉及的并不仅仅是前端技术，有服务端、客户端、设计师团队，所以要想做好性能优化，绝对不能把自己限制在局部的视角，必须是整个业务一起考虑，才能有良好的收效。

执行

技术方案设计好了，它是不会自己变成线上页面的，所以，有了技术方案，我们只完成了一半的工作，接下来我们还需要一个执行过程。

执行也不简单，如果说方案主要靠技术，那么执行就是靠工程实施了。

根据公司的实际情况，工程实施可能有不同的程度，我把工程水平从低到高分成三个阶段：

- 纯管理；

- 制度化；
- 自动化。

纯行政管理，是由经理用纯粹的管理手段来执行方案，比如说，作为前端团队的Leader，我可以组织会议，要求整个团队使用我们前面谈的技术方案。

但是纯行政管理有一些问题，一方面，需要的行政资源不一定有，比如我没法强制让后端团队配合我，另一方面，纯粹的管理方式，团队本身的体验并不好，也不利于团队成长，最重要的是，纯粹管理方式容易造成执行不到位。这样的执行方式多数出现在非技术岗位。

制度化执行方式是用规则代替人的命令，指定责任人，通过培训、checklist、定期review等具体措施来保证实施。制度化执行可以极大地减轻管理工作量，一般现代互联网公司都会采用类似的方式。但是制度化执行方式还有很大成分是依靠人的主动性的，对程序员来说，还有更好的方式：自动化。

自动化的方式是在一些重要的操作路径上设置规则，针对我们的性能优化，有两个点适合做这件事：一个是把开发好的页面发布上线，另一个是开发好的页面URL投放到首页等处的链接。

在我之前的工作中，我们跟测试团队配合，开发了一套页面性能打分系统，它会自动扫面页面上的可优化点，并且跟发布平台和投放平台合作，把它加入日常机制中。现在多数公司都会采用制度化和自动化结合的执行方案。

结果评估和监控

执行完了之后，**就要向老板汇报争取升职加薪了**，还要有一定的结果总结，才是一个完整的工程实施，而且，凡是工程实施，肯定要有一定长效机制，不能优化完了退化，这些都要求有线上监控机制。

要想做线上监控，分两个部分：

- 数据采集；
- 数据展现。

数据采集部分，同样需要发布平台或者开发工具来配合，对性能数据来说，Performance API非常好用，它是浏览器记录的性能数据，一般来说，我们用统一的代码把它上传到服务器端就够用了。

数据的展现部分就比较自由了，可以用不同的数据可视化方案来展现性能数据，没有一定之规。一般的数据监控平台，会提供报警机制，对性能来说，报警需求不是特别强烈，但是也可以设置一些条件，针对秒开率特别低的网页报警。

有了监控，再配合一定制度，就可以保障整个团队产出的性能了，要注意，性能不是一个静态的事情，指标需要不断优化，技术方案还需要不断随着技术发展迭代，制度、自动化工具也需要不断改进，最终的监控平台产品也不能不做新需求，所以性能应该成为一个团队的日常工作的一部分，持续进行。

总结

今天我们学习了前端团队工程实施中的性能体系，首先我们介绍了总体思想：性能应该是基于业务和实际用户体验需求的一种工程实施，不是纯粹的技术游戏。

接下来我们分成四个步骤介绍了性能工程体系，首先介绍了现状评估和建立指标，建立指标应当从业务的角度考虑，接下来讲了技术方案设计，技术方案应当从整体角度，基于Profiling的结果分析来设计。

之后我们讲了实施，我们讲了工程实施的三个层次：纯管理、制度化、工程化，最后，我们讲了结果评估和线上监控，线上监控需要从数据采集和数据展现两个部分分别实现。

最后，留一个小问题，请你为自己的团队和业务设计一下性能的整体方案，欢迎来留言分享。

你好，我是winter。

从今天开始，我们就从前端知识学习的部分，过渡到了实践部分。这节课我来谈谈性能。

性能是个特别有意思的话题，在我之前的工作中，从入门的初级工程师到高级别的技术专家，大家都很喜欢谈性能，我以前参与晋升评审，每年总能听到很多关于性能的晋升述职。

那么，今天我就来谈谈我眼中的性能。

性能总论

> while循环快还是for循环快？
>
> |0 是不是比 Math.floor 性能好？

网上随处可以见到一类对性能的讨论。一些新人也非常热衷此类讨论。但是实际上，它们除了让你写代码的时候纠结之外，毫无意义。

为什么这样讲呢？我想讲一个小故事。

从前有个工程师，特别注重代码细节，有一天他发现系统中的一段代码写的性能很差，因此，他用汇编重写了整段代码，执行效率足足提升了三倍。但是最后，大家发现，用户反馈性能丝毫没有提高，因为他优化的那个进程名字叫“System Idle”。

所以你看，性能优化不能只着眼于局部的代码。这里，我要提出一个我的观点：**一切没有profiling的性能都是耍流氓**。凡是真正有价值的性能优化，必定是从端到端的业务场景建立体系来考虑的。

在我的认识中，性能体系的建立可以分成以下几部分：

- 现状评估和建立指标；

- 技术方案；
- 执行；
- 结果评估和监控。

下面，我就来为你一一讲解。

现状评估和建立指标

要想做好性能优化，正确地评估现状和建立指标是最关键的一步，它又往往是会被轻视的一步。

作为一个工程师，指标又要考虑两个因素。一方面，对用户来说，什么样的性能指标能更好地评估它的体验？另一方面，对公司来说，什么样的指标会影响业务价值呢？

在我公布答案之前，我希望你能思考一下，你所负责的业务，是否有前端性能指标？它是否能够满足我上面提到的两个要求？

在我之前的工作中，整个用了长达一年的时间来探索，才找到了合适的指标，并且回答好了两个问题。

性能问题可以分成很多方面，最重要的几个点是：

- 页面加载性能；
- 动画与操作性能；
- 内存、电量消耗。

注意，这里我们仅仅是对“性能”两个字的分析和解读，在对大量的用户数据分析后，我们发现，其实这三部分中，“页面加载性能”跟用户的流失率有非常强的关联性，而用户流失率，正是公司业务非常看重的指标。

因此，在开始阶段，我们决定把性能优化的重点放在页面加载性能上。

那么，用什么指标来衡量页面加载性能呢？最容易想到的方案是“用户平均加载时间”，事实上，我们在相当长的一段时间，也都是在使用用户平均加载时间作为性能指标。

但是，很快我们发现，这个指标有严重的问题：

- 当加载时间低于一定数字，用户体感差别不大了，我们经过一定的研究，认为这个数字大约是1秒；
- 少数超长时间加载的用户（如2G），会极大影响整个指标，即指标不能反映大多数用户的体验。

于是，基于以上分析，我们设计了一个新的指标——秒开率，即一秒之内打开的用户占用户总量的百分比。这个指标后来逐渐推广到整个公司，甚至影响到了一些业内的其它企业，现在，谈秒开率已经是个非常自然的事情了，但是当初的设计确实走了不少弯路。

技术方案

有了指标，我们就有了优化的目标，接下来，就到了技术出场的环节了。

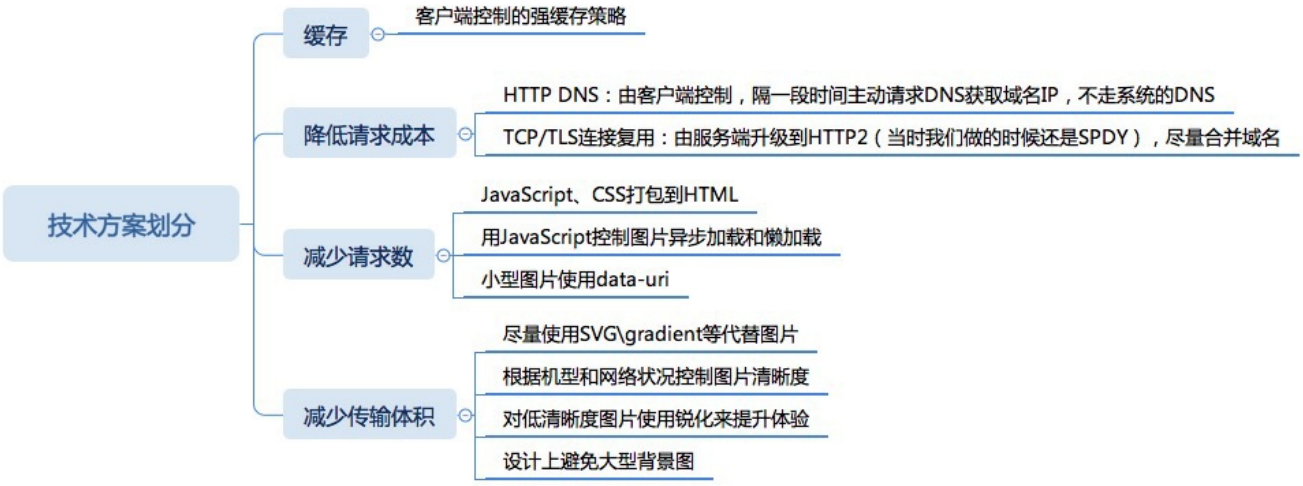
我们这里还是以加载过程为例，来讲解一下。

首先我们要简单分析一下，从输入URL后按下回车，到底发生了什么。

我们在浏览器的原理课程中，已经讲解了浏览器大致的工作过程，但是，我们必须理解几件事：

- 从域名到IP地址，需要用DNS协议查询；
- HTTP协议是用TCP传输的，所以会有TCP建立连接过程；
- 如果使用HTTPS，还有有HTTPS交换证书；
- 每个网页还有图片等请求。

从这个分析和实际试验的结果看，网页的加载时间，不但跟体积有关系，还跟请求数有很大关系，因此，我们最终设计的技术方案大约可以这样划分：



这里仅仅列出了性能优化的一部分技术方案，是我认为比较重要的部分，可以看到，这里涉及的并不仅仅是前端技术，有服务端、客户端、设计师团队，所以要想做好性能优化，绝对不能把自己限制在局部的视角，必须是整个业务一起考虑，才能有良好的收效。

执行

技术方案设计好了，它是不会自己变成线上页面的，所以，有了技术方案，我们只完成了一半的工作，接下来我们还需要一个执行过程。

执行也不简单，如果说方案主要靠技术，那么执行就是靠工程实施了。

根据公司的实际情况，工程实施可能有不同的程度，我把工程水平从低到高分成三个阶段：

- 纯管理；
- 制度化；
- 自动化。

纯行政管理，是由经理用纯粹的管理手段来执行方案，比如说，作为前端团队的Leader，我可以组织会议，要求整个团队使用我们前面谈的技术方案。

但是纯行政管理有一些问题，一方面，需要的行政资源不一定有，比如我没法强制让后端团队配合我，另一方面，纯粹的管理方式，团队本身的体验并不好，也不利于团队成长，最重要的是，纯粹管理方式容易造成执行不到位。这样的执行方式多数出现在非技术岗位。

制度化执行方式是用规则代替人的命令，指定责任人，通过培训、checklist、定期review等具体措施来保证实施。制度化执行可以极大地减轻管理工作量，一般现代互联网公司都会采用类似的方式。但是制度化执行方式还有很大成分是依靠人的主动性的，对程序员来说，还有更好的方式：自动化。

自动化的方式是在一些重要的操作路径上设置规则，针对我们的性能优化，有两个点适合做这件事：一个是把开发好的页面发布上线，另一个是开发好的页面URL投放到首页等处的链接。

在我之前的工作中，我们跟测试团队配合，开发了一套页面性能打分系统，它会自动扫面页面上的可优化点，并且跟发布平台和投放平台合作，把它加入日常机制中。现在多数公司都会采用制度化和自动化结合的执行方案。

结果评估和监控

执行完了之后，**就要向老板汇报争取升职加薪了**，还要有一定的结果总结，才是一个完整的工程实施，而且，凡是工程实施，肯定要有一定长效机制，不能优化完了退化，这些都要求有线上监控机制。

要想做线上监控，分两个部分：

- 数据采集；
- 数据展现。

数据采集部分，同样需要发布平台或者开发工具来配合，对性能数据来说，Performance API非常好用，它是浏览器记录的性能数据，一般来说，我们用统一的代码把它上传到服务器端就够用了。

数据的展现部分就比较自由了，可以用不同的数据可视化方案来展现性能数据，没有一定之规。一般的数据监控平台，会提供报警机制，对性能来说，报警需求不是特别强烈，但是也可以设置一些条件，针对秒开率特别低的网页报警。

有了监控，再配合一定制度，就可以保障整个团队产出的性能了，要注意，性能不是一个静态的事情，指标需要不断优化，技术方案还需要不断随着技术发展迭代，制度、自动化工具也需要不断改进，最终的监控平台产品也不能不做新需求，所以性能应该成为一个团队的日常工作的一部分，

持续进行。

总结

今天我们学习了前端团队工程实施中的性能体系，首先我们介绍了总体思想：性能应该是基于业务和实际用户体验需求的一种工程实施，不是纯粹的技术游戏。

接下来我们分成四个步骤介绍了性能工程体系，首先介绍了现状评估和建立指标，建立指标应当从业务的角度考虑，接下来讲了技术方案设计，技术方案应当从整体角度，基于Profiling的结果分析来设计。

之后我们讲了实施，我们讲了工程实施的三个层次：纯管理、制度化、工程化，最后，我们讲了结果评估和线上监控，线上监控需要从数据采集和数据展现两个部分分别实现。

最后，留一个小问题，请你为自己的团队和业务设计一下性能的整体方案，欢迎来留言分享。

你好，我是winter。

从今天开始，我们就从前端知识学习的部分，过渡到了实践部分。这节课我来谈谈性能。

性能是个特别有意思的话题，在我之前的工作中，从入门的初级工程师到高级别的技术专家，大家都很喜欢谈性能，我以前参与晋升评审，每年总能听到很多关于性能的晋升述职。

那么，今天我就来谈谈我眼中的性能。

性能总论

> while循环快还是for循环快？
>
> |0 是不是比 Math.floor 性能好？

网上随处可以见到一类对性能的讨论。一些新人也非常热衷此类讨论。但是实际上，它们除了让你写代码的时候纠结之外，毫无意义。

为什么这样讲呢？我想讲一个小故事。

从前有个工程师，特别注重代码细节，有一天他发现系统中的一段代码写的性能很差，因此，他用汇编重写了整段代码，执行效率足足提升了三倍。但是最后，大家发现，用户反馈性能丝毫没有提高，因为他优化的那个进程名字叫“System Idle”。

所以你看，性能优化不能只着眼于局部的代码。这里，我要提出一个我的观点：**一切没有profiling的性能都是耍流氓**。凡是真正有价值的性能优化，必定是从端到端的业务场景建立体系来考虑的。

在我的认识中，性能体系的建立可以分成以下几部分：

- 现状评估和建立指标；
- 技术方案；
- 执行；
- 结果评估和监控。

下面，我就来为你一一讲解。

现状评估和建立指标

要想做好性能优化，正确地评估现状和建立指标是最关键的一步，它又往往是会被轻视的一步。

作为一个工程师，指标又要考虑两个因素。一方面，对用户来说，什么样的性能指标能更好地评估它的体验？另一方面，对公司来说，什么样的指标会影响业务价值呢？

在我公布答案之前，我希望你能思考一下，你所负责的业务，是否有前端性能指标？它是否能够满足我上面提到的两个要求？

在我之前的工作中，整个用了长达一年的时间来探索，才找到了合适的指标，并且回答好了两个问题。

性能问题可以分成很多方面，最重要的几个点是：

- 页面加载性能；
- 动画与操作性能；
- 内存、电量消耗。

注意，这里我们仅仅是对“性能”两个字的分析和解读，在对大量的用户数据分析后，我们发现，其实这三部分中，“页面加载性能”跟用户的流失率有非常强的关联性，而用户流失率，正是公司业务非常看重的指标。

因此，在开始阶段，我们决定把性能优化的重点放在页面加载性能上。

那么，用什么指标来衡量页面加载性能呢？最容易想到的方案是“用户平均加载时间”，事实上，我们在相当长的一段时间，也都是在使用用户平均加载时间作为性能指标。

但是，很快我们发现，这个指标有严重的问题：

- 当加载时间低于一定数字，用户体感差别不大了，我们经过一定的研究，认为这个数字大约是1秒；
- 少数超长时间加载的用户（如2G），会极大影响整个指标，即指标不能反映大多数用户的体验。

于是，基于以上分析，我们设计了一个新的指标——秒开率，即一秒之内打开的用户占用户总量的百分比。这个指标后来逐渐推广到整个公司，甚至影响到了一些业内的其它企业，现在，谈秒开率已经是个非常自然的事情了，但是当初的设计确实走了不少弯路。

技术方案

有了指标，我们就有了优化的目标，接下来，就到了技术出场的环节了。

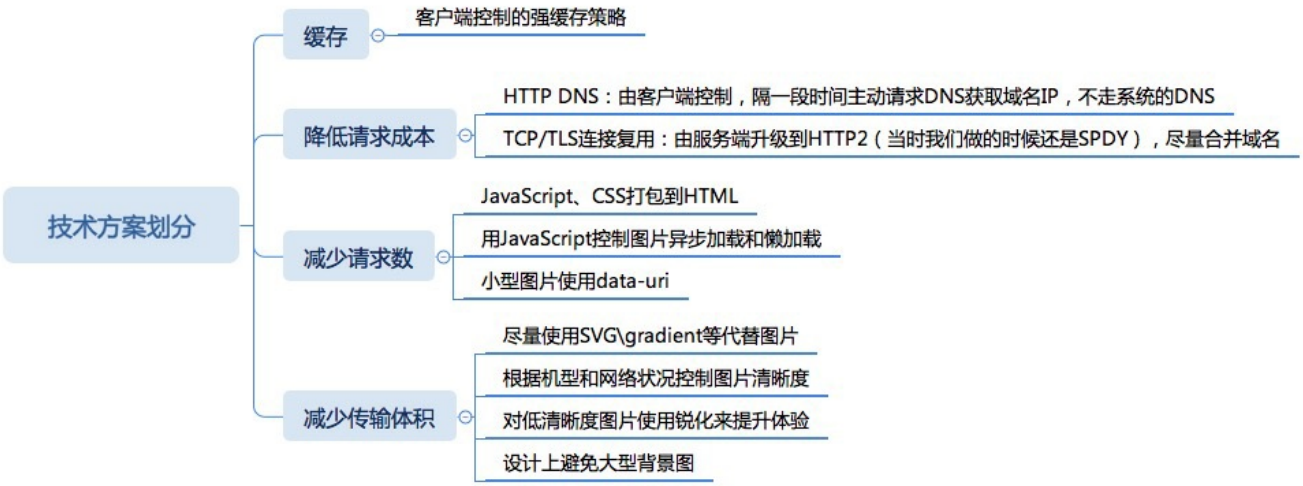
我们这里还是以加载过程为例，来讲解一下。

首先我们要简单分析一下，从输入URL后按下回车，到底发生了什么。

我们在浏览器的原理课程中，已经讲解了浏览器大致的工作过程，但是，我们必须理解几件事：

- 从域名到IP地址，需要用DNS协议查询；
- HTTP协议是用TCP传输的，所以会有TCP建立连接过程；
- 如果使用HTTPS，还有有HTTPS交换证书；
- 每个网页还有图片等请求。

从这个分析和实际试验的结果看，网页的加载时间，不但跟体积有关系，还跟请求数有很大关系，因此，我们最终设计的技术方案大约可以这样划分：



这里仅仅列出了性能优化的一部分技术方案，是我认为比较重要的部分，可以看到，这里涉及的并不仅仅是前端技术，有服务端、客户端、设计师团队，所以要想做好性能优化，绝对不能把自己限制在局部的视角，必须是整个业务一起考虑，才能有良好的收效。

执行

技术方案设计好了，它是不会自己变成线上页面的，所以，有了技术方案，我们只完成了一半的工作，接下来我们还需要一个执行过程。

执行也不简单，如果说方案主要靠技术，那么执行就是靠工程实施了。

根据公司的实际情况，工程实施可能有不同的程度，我把工程水平从低到高分成三个阶段：

- 纯管理；
- 制度化；
- 自动化。

纯行政管理，是由经理用纯粹的管理手段来执行方案，比如说，作为前端团队的Leader，我可以组织会议，要求整个团队使用我们前面谈的技术方案。

但是纯行政管理有一些问题，一方面，需要的行政资源不一定有，比如我没法强制让后端团队配合我，另一方面，纯粹的管理方式，团队本身的体验并不好，也不利于团队成长，最重要的是，纯粹管理方式容易造成执行不到位。这样的执行方式多数出现在非技术岗位。

制度化执行方式是用规则代替人的命令，指定责任人，通过培训、checklist、定期review等具体措施来保证实施。制度化执行可以极大地减轻管理工作量，一般现代互联网公司都会采用类似的方式。但是制度化执行方式还有很大成分是依靠人的主动性的，对程序员来说，还有更好的方式：自动化。

自动化的方式是在一些重要的操作路径上设置规则，针对我们的性能优化，有两个点适合做这件事：一个是把开发好的页面发布上线，另一个是开发好的页面URL投放到首页等处的链接。

在我之前的工作中，我们跟测试团队配合，开发了一套页面性能打分系统，它会自动扫面页面上的可优化点，并且跟发布平台和投放平台合作，把它加入日常机制中。现在多数公司都会采用制度化和自动化结合的执行方案。

结果评估和监控

执行完了之后，**就要向老板汇报争取升职加薪了**，还要有一定的结果总结，才是一个完整的工程实施，而且，凡是工程实施，肯定要有一定长效机制，不能优化完了退化，这些都要求有线上监控机制。

要想做线上监控，分两个部分：

- 数据采集；
- 数据展现。

数据采集部分，同样需要发布平台或者开发工具来配合，对性能数据来说，Performance API非常好用，它是浏览器记录的性能数据，一般来说，我们用统一的代码把它上传到服务器端就够用了。

数据的展现部分就比较自由了，可以用不同的数据可视化方案来展现性能数据，没有一定之规。一般的数据监控平台，会提供报警机制，对性能来说，报警需求不是特别强烈，但是也可以设置一些条件，针对秒开率特别低的网页报警。

有了监控，再配合一定制度，就可以保障整个团队产出的性能了，要注意，性能不是一个静态的事情，指标需要不断优化，技术方案还需要不断随着技术发展迭代，制度、自动化工具也需要不断改进，最终的监控平台产品也不能不做新需求，所以性能应该成为一个团队的日常工作的一部分，持续进行。

总结

今天我们学习了前端团队工程实施中的性能体系，首先我们介绍了总体思想：性能应该是基于业务和实际用户体验需求的一种工程实施，不是纯粹的技术游戏。

接下来我们分成四个步骤介绍了性能工程体系，首先介绍了现状评估和建立指标，建立指标应当从业务的角度考虑，接下来讲了技术方案设计，技术方案应当从整体角度，基于Profiling的结果分析来设计。

之后我们讲了实施，我们讲了工程实施的三个层次：纯管理、制度化、工程化，最后，我们讲了结果评估和线上监控，线上监控需要从数据采集和数据展现两个部分分别实现。

最后，留一个小问题，请你为自己的团队和业务设计一下性能的整体方案，欢迎来留言分享。