今天看到这篇文章。写的非常有意思。发现自己才处于”入室”阶段。骚年，路还很长，继续前行。

**【背景】**

如果你是刚进入web前端研发领域，想试试这潭水有多深，看这篇文章吧；  
如果你是做了两三年web产品前端研发，迷茫找不着提高之路，看这篇文章吧；  
如果你是四五年的前端开发高手，没有难题能难得住你的寂寞高手，来看这篇文章吧；

web前端研发工程师，在国内是一个朝阳职业，自07-08年正式有这个职业以来，也不过三四年的时间。这个领域没有学校的正规教育，没有行内成体系的理论指引，几乎所有从事这个职业的人都是靠自己自学成才。自学成才，一条艰辛的坎坷路，我也是这样一路走来。从2002年开始接触web前端研发至今已然有了9个年头，如今再回首，期间的走了很多弯路。推已及人，如果能让那些后来者少走些弯路，辛甚辛甚！

**【前言】**

所谓的天才，只不过是比平常人更快的掌握技能、完成工作罢了；只要你找到了正确的方向，并辅以足够的时间，你一样能够踏上成功彼岸。

本文将web前端研发编程能力划分了八个等级，每个等级都列举出了对应的特征及破级提升之方法，希望每位在看本文的同学先准确定位自己的等级（不要以你目前能力的最高点，而是以你当前能力的中档与之等级作对比，以免多走弯路），参考突破之法破之。

所谓的级别，只是你面对需求时的一种态度：能够完成、能够完美地完成、能够超出预期地完成。以追求完美的态度加以扎实的编程功力，那就是你的编程水平。  
切记心浮气燥，级别够了，那级别里的东西自然就懂了。悟了就是悟了，没悟也没关系，静下心来，投入时间而已。

**一.【入门】**

能够解决一些问题的水平。有一定的基础（比如最常见的html标签及其属性、事件、方法；最常见的css属性；基础的javascript编程能力），能够完成一些简单的web前端研发需求。  
举个例子：删除一字符串中指定的字符。

|  |  |
| --- | --- |
| 1  2  3  4  5 | var str="www.baidu.com/?page";  str=str.replace('?page',"");  alert(str);  str=str.substring(0,str.indexof("/"));  alert(str); |

首先不要苛责代码的对错严谨，毕竟每个程序员都有这样的一个过程；其次，这两段代码在这个实例里没有什么大过错，可能会有瑕疵，但能够解决问题（删除指定的字符），这就是这个级别的特征。  
再举个例子：

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | // 计算系统当前是星期几  var str = "";  var week = new date().getday();  if (week == 0) {  str = "今天是星期日";  } else if (week == 1) {  str = "今天是星期一";  } else if (week == 2) {  str = "今天是星期二";  } else if (week == 3) {  str = "今天是星期三";  } else if (week == 4) {  str = "今天是星期四";  } else if (week == 5) {  str = "今天是星期五";  } else if (week == 6) {  str = "今天是星期六";  }  // 或者更好一些  var str1 = "今天是星期";  var week = new date().getday();  switch (week) {  case 0 :  str1 += "日";  break;  case 1 :  str1 += "一";  break;  case 2 :  str1 += "二";  break;  case 3 :  str1 += "三";  break;  case 4 :  str1 += "四";  break;  case 5 :  str1 += "五";  break;  case 6 :  str1 += "六";  break;  }  alert(str);  alert(str1); |

入门”阶段是每个程序员的必经之路，只要”入门”，你就上路了。所谓”师傅领进门，修行靠个人”，有了这个”入门”的基础，自己就可以摸索着前进了。

**进阶之路**

将javascript、html、css之类的编码帮助手册里的每个方法/属性都通读几遍！只有将基础打好，以后的路才能走的顺畅。参考这些帮助文档，力争写出无瑕疵的代码。  
这些编码文档建议不仅是在入门提高期看，在你以后每个阶段破阶的时候都应该看看，最基础的东西往往也是最给力的东西，有时能够给你带来意想不到的收获。

**二.【登堂】**

能够正确地解决问题。不管你是通过搜索网络，或者通过改造某些成品代码（jquery/dojo/ext/yui）案例，只要能够无错地完成需求。  
同样以上面的那段”字符串剪裁”代码为例：

|  |  |
| --- | --- |
| 1  2  3 | var str="www.baidu.com/?page";  str=str.replace(/?page/,"");  alert(str); |

仅仅解决问题对于”登堂”阶段来说已经不是问题，这个级别所给出方案不能是漏洞百出。以上面这段代码为例：replace方法的第一个参数虽然可以支持字符串，但最佳的类型是正则表达式；

|  |  |
| --- | --- |
| 1  2  3  4 | var a = new array("日", "一", "二", "三", "四", "五", "六");  var week = new date().getday();  var str = "今天是星期"+ a[week];  alert(str); |

对比”入门级”的代码，不管是从代码量、代码效率、代码优美性、代码思路来说，”登堂”级的这个日期处理代码都要优秀很多。

**进阶之路**

这个阶段虽然能够给出正确的解题方案，但是不一定是最优秀的方案。如何才能得到最优秀的方案呢？首先就是积累各种能够解决需求的方案，然后再验证每个方案，在这些方案中选择最好的一种。因此该阶段的进阶之路就是”行万里路，看万卷书”，积累各个需求的各个解决方案。

你可以扎身在专业论坛（蓝色理想、无忧、csdn）里，通读所有的faq及帖子；你可以打开搜索引擎，穷举所有的搜索结果。自己建立测试环境一一验证这些代码：去揣摩每段代码的意图，去比较每段代码之间的差异。这两条路可以让你快速完成原始积累，当你再面对大多数需求时能够说这些问题我以前做过，那你就水到渠成地晋阶了。

**三.【入室】**

最强代码，知道所有能够解决需求的各种方案，能够选择使用最优秀的方案满足需求。这个级别基本上产品开发编程中的代码主力。给出的一招一式，招招都是绝招。  
还以上面的那个例子为例，你能说出1、2、3之间的差别，以及适用于那种环境吗？

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var str="www.baidu.com/?page";  // 1、字符串剪裁  str.substring(0, str.indexof("?page"));  // 2、正则表达式  str.replace(/?page/, "");  // 3、字符串分拆、合并  str.split("?page").join(""); |

能够解决问题的方法会有很多，但是对于程序员来说应该选择最优秀的。上面这段代码从代码量来说”正则表达式”最优秀；从代码执行效率来说： “字符串剪裁”法最高（chrome中”正则表达式”效率最高），split法最次；从可扩展性上来说，”正则表达式”法最优。具体使用那种方案视具体的需求环境而定。

“入室”阶段，程序员应该能够肯定的回答：对于这个需求而言，我的代码就是最优秀的代码。

再以”今天是星期几”为例，”登堂”级的代码你敢说是最优秀的代码了吗？

|  |  |
| --- | --- |
| 1  2 | // 计算系统当前是星期几  var str = "今天是星期" + "日一二三四五六".charat(new date().getday()); |

对比”登堂”级的示例代码，上面这段代码给你什么感受？程序员追求的就是完美。”入室”级别追求的就是每一招每一式的完美无缺。

从web前端编程来说，通过2年左右的努力，很多人能够达到这个水平，但是，很大一部分人的编程能力也就止步于此。或限于产品的需求单一性，或限于需求开发的时间紧迫性，或限于人的惰性，能够完美地解决当前的需求就够了。

由于长期处于技术平台期，技术上得不到提高，通常这个级别的工程师会比较燥。技术上小有所成；或追求个人的突破；或追求产品差异性带来的新鲜感；或者只是想换个心情；因此很多此级别的工程师会经常换公司。

戒骄戒躁：

切勿以为自己能写一手漂亮的代码而自满；  
切莫以为别人”尊称”你一声”大侠”你就以 “大侠”自居；  
切莫以为自己积累了一些得意的代码就成了框架式开发。

细节决定成败，优秀的方案并不能保证最终的成功。还以”删除指定字符串”为例，原始字符串从格式上来看应该是了个url链接，在去除”pn=0″之后，最末尾处留了一个尾巴”?”；如果原始字符串是”[http://www.xxx.com/?pn=0&a=1](http://www.w3cfuns.com/)“，去除”pn=0″之后 ? 和 & 两个符号紧贴一起，这更是明显的bug。

**进阶之路**

此阶段进阶之路就是：切勿心浮气躁；你不再被需求牵着走，而是你牵着需求走。注重细节，注意那些当前需求里没有明文给出的细节：代码性能的差异、运行平台（浏览器）的差异、需求的隐性扩展、代码的向后兼容等等。

再通读几遍html/css/javascript帮助文档。

我建议这个级别的工程师做一做webtreeview控件，要求总节点量一万左右操作流畅，你的晋升之路就在这个控件的编码过程中。

**四.【入微】**

最强解决方案。你能够走在需求的前面，将当前需求里有的、没有直接提出来的、现在暂时没有但将来可能有的等等，及前端编程潜规则等各个方方面面都综合考虑，给出最优方案。以一招胜万招。

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21 | var str = "<http://www.xxx.com/?pn=0>"; // 删除指定字符 pn=0  // 我将这个字符串里所可能想到的各种情况都列举出来  var a = [  "<http://www.xxx.com/vmpn=?pn=0>"// pn= 可能出现在 ? 前  , "<http://www.xxx.com/vmpn=?pn=>"// url里允许pn 值为空  , "<http://www.xxx.com/vmpn=?pn=0&a=1>"// url 里可有多个字段  , "<http://www.xxx.com/vmpn=?a=1&pn=0>"// 可能排在最后  , "<http://www.xxx.com/vmpn=?a=1&pn=0&pn=1>"// 可能有多个 pn 字段  , "<http://www.xxx.com/vmpn=?a=1&pn=0&b=2>"// 可能在中间  , "<http://www.xxx.com/vmpn=?a=1&pn=0&pn=1&b=1>" // 可能在中间成组  , "<http://www.xxx.com/vmpn=?a=1&pn=0&b=1&pn=1>" // 可能零星分布  ];  /\* 需求的不言之秘：  ? 若出现在字符串最尾则要去之  ? & 两个符号不可重叠  \*/  var reg = /((\?)(pn=[^&]\*&)+(?!pn=))|(((\?|&)pn=[^&]\*)+$)|(&pn=[^&]\*)/g;  for (var i = 0; i < a.length; i++) {  alert(a + "\n" + a.replace(reg, "$2"));  } |

这个阶段已经不再追求一招一式，对你来说不是使用什么创新绝招解决需求，而是给出成熟稳重的方案，从根上解决问题。针对某个当前需求你的代码可能不是最优，但是针对此类的需求你的代码却是最优秀的代码。

**进阶之路**

很多web前端研发工程师在做了3-4年之后就会进入一个瓶颈期：产品开发需求是小菜一碟，没有新鲜的可以挑战的东西；代码开发中的稀奇的解题方法都已经尝试过。没有了可挑战的难题，失去了探索的激情，也就没有了再上升的动力，好不容易走过”入室”级别的人又会有八九成止步于此。或转做技术领导人，或转到其它的领域，或换公司。

这些人的上升之路在哪里呢？

这个阶段单单依靠技巧和数量的累积已经没有什么效果了，突破之路在第5层《化蝶》里会详细说明，我建议你在这个阶段末尾着重关注编程理论：面向对象/过程、代码组织形式、编译、代码规范、其它的框架设计等等。

我建议这个级别的工程师做一做webeditor控件，不要求完整功能，但是该控件里的模块划分、代码组织、编程思想做到位，给出一个系统的解决方案。

**五.【化蝶】**

破茧重生，这个层次关注的是编程语言本身，而不再关心产品需求。什么是茧？产品需求就是茧。当你一招胜万招，打遍天下需求之时，你如果还拘泥于需求开发，那就是你限于茧中而不自知。要么就在这个茧里默默地老去，要么就破开茧获得新生。

还是以那个”字符串剪裁”的老例子：

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | /\*\*  \* 在拼接正则表达式字符串时，消除原字符串中特殊字符对正则表达式的干扰  \* @author:meizz  \* @version: 2010/12/16  \* @param {string} str 被正则表达式字符串保护编码的字符串  \* @return {string} 被保护处理过后的字符串  \*/  function escapereg(str) {  return str.replace(new regexp("([.\*+?^=!:\x24{}()|[\\]\/\\\\])", "g"), "\\\x241");  }  /\*\*  \* 删除url字符串中指定的 query  \* @author:meizz  \* @version:2010/12/16  \* @param {string} url url字符串  \* @param {string} key 被删除的query名  \* @return {string} 被删除指定 query 后的url字符串  \*/  function delurlquery(url, key) {  key = escapereg(key);  var reg = new regexp("((\\?)("+ key +"=[^&]\*&)+(?!"+ key +  "=))|(((\\?|&)"+ key +"=[^&]\*)+$)|(&"+ key +"=[^&]\*)", "g");  return url.replace(reg, "\x241")  }  // 应用实例  var str = "<http://www.xxx.com/?pn=0>"; // 删除指定字符 pn=0  delurlquery(str, "pn"); |

这段代码相对于层次4《入微》有什么区别吗？从代码实现上来说没有太大的区别，但是从思路上来说却有着本质的区别：1、不再是就事论事，头疼医头，而是把一类问题抽象理论化，一招破万招；2、有封装的概念，不再是每次从零开始，而是站在半山腰开始爬。

在web前端研发队伍里也有很大一部分人《入室》层次时就自我感觉良好，直接跨跃到《化蝶》，积累自己的代码库，抽象化问题。但没有基础，缺少强大的后劲，即使能够破茧也经受不了风吹雨打。一份不成熟的架构设计对团队开发带来的危害远大于它带来的好处，这种例子在业界屡见不鲜。不要拔苗助长，不要不会走就想着跑，夯实基础，水到渠成地成长，厚积薄发，强力地破茧而出。

**进阶之路**

你已经从原始积累，到厚积薄发，到破茧而出之后，你所关注的应该不再是一招一式、一个项目、一个模块，而应该是一种思路，一种理论。你可以做以下几个步骤以突破到更高层次：再仔细看几遍html/css/javascript接口帮助文档；选择一门强语言（c++/c#/java等）观察理解这些语言的组织结构，语言设计；看看原型链，链式语法编程，泛型，接口编程，dom遥控器等等；仔细阅读成熟的web前端开发框架的设计文档，看他们为什么要这样设计。

**六.【大侠】**

这里所说的大侠，不是大家互相吹捧的”大侠”，而是实至名归的高手。这个级别的人完全有能力写出不差于bindows/jquery/ext/yui/dojo的同等级别规模的前端开发框架。应用成熟的开发框架指导、解决问题。

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | // 库文件 /mz/string/escapereg.js  /\*\*  \* 在拼接正则表达式字符串时，消除原字符串中特殊字符对正则表达式的干扰  \* @author:meizz  \* @version: 2010/12/16  \* @param {string} str 被正则表达式字符串保护编码的字符串  \* @return {string} 被保护处理过后的字符串  \*/  mz.string.escapereg = function (str) {  return str.replace(new regexp("([.\*+?^=!:\x24{}()|[\\]\/\\\\])", "g"), "\\\x241");  }  // 库文件 /mz/url/delquery.js  /// include mz.string.escapereg;  /\*\*  \* 删除url字符串中指定的 query  \* @author:meizz  \* @version:2010/12/16  \* @param {string} url url字符串  \* @param {string} key 被删除的query名  \* @return {string} 被删除指定 query 后的url字符串  \*/  mz.url.delquery = function (url, key) {  key = mz.string.escapereg(key);  var reg = new regexp("((\\?)("+ key +"=[^&]\*&)+(?!"+ key +  "=))|(((\\?|&)"+ key +"=[^&]\*)+$)|(&"+ key +"=[^&]\*)", "g");  return url.replace(reg, "\x241")  }  // 应用实例  /// include mz.url.delquery;  var str = "<http://www.xxx.com/?pn=0>"; // 删除指定字符 pn=0  mz.url.delquery(str, "pn"); |

自成体系，有基础，也有理论高度。知道为什么这样设计，也知道什么样的设计最好。比如这个例子可以有这样的封装：

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21 | // 库文件 /mz/url/delquery.js  /// include mz.string.escapereg;  /\*\*  \* 删除url字符串中指定的 query  \* @author:meizz  \* @version:2010/12/16  \* @param {string} url url字符串  \* @param {string} key 被删除的query名  \* @return {string} 被删除指定 query 后的url字符串  \*/  string.prototype.delquery = function ( key) {  key = mz.string.escapereg(key);  var reg = new regexp("((\\?)("+ key +"=[^&]\*&)+(?!"+ key +  "=))|(((\\?|&)"+ key +"=[^&]\*)+$)|(&"+ key +"=[^&]\*)", "g");  return this.replace(reg, "\x241")  }  // 应用实例  /// include mz.url.delquery;  var str = "<http://www.xxx.com/?pn=0>"; // 删除指定字符 pn=0  str.delquery("pn"); |

而为什么不采用下面的那种封装呢？经过了《知微》和《化蝶》你就懂了。

**进阶出路**

道法自然，从根上去寻找突破的契机。你可以研读html解析引擎设计与实现，js解析引擎设计与实现，其它语言的代码解析与编译实现等等。  
或者出些书。低级别的人写的书要么是一大抄，空无一物；要么是害人。

**七.【宗师】**

这个级别的人已然到了无招胜有招的境界。项目开发中的难题？没有难题！运行平台的差异？从根本上搞定！代码规范、开发模式，早已经被抛在身后。这个级别的人已经不再关注于某个前端开发框架，而是应对具体的环境给出最佳的理论指导。

这个级别的人所注意的应该是以最合理的系统架构引领着整个团队的进步，在什么样的场景下该用什么样的架构设计。3个、10个、50个、100个人的团队最应该用哪种模式？等你到了宗师级别，你再来回答吧。

**进阶出路**

每一个宗师就是一个高山，就是一个领域里的神，但是你仅满足于在一群比你弱的群体展现你的强大吗？如果还你是止步原地，那总会有人乘着飞机、宇宙飞船从你的头领掠过，高处不胜寒！

要突破这片领域，那就必须跳出这片领域。要想突破web前端研发的宗师级，那就跳出web前端吧，上面还有web开发。即使你是web前端的宗师，但没有快速的数据响应，没有高速的网络架构，没有优美的系统支持，你又能如何？所以突破之路就是把目光投到整条web开发的链条中去。

**八.【飞升】**

其实严格来说，飞升已经不是原领域的范围了。在web研发领域，对于这个层次的有一个很好听的称谓：架构师。当然那些”伪架构师”另当别论。

一法通，万法通。在其它的技术领域，也可以按照《入门》《登堂》《入室》《入微》《化蝶》《大侠》《宗师》来划分等级；一样也可以按照我这里所写的每个级别的【进阶之路】来快速提升。

祝贺你再获辉煌！