2018年03月13日 | Unknown author

前端面试遇到的算法题

Q: 解析 URL Params 为对象

尽可能的全面正确的解析一个任意 url 的所有参数为 Object，注意边界条件的处理。

```
let url = 'http://www.domain.com/?user=anonymous&id=123&id
parseParam(url)
```

复制代码

答:

```
function parseParam(url) {
  const paramsStr = /.+\?(.+)$/.exec(url)[1];
  const paramsArr = paramsStr.split('&');
  let paramsObj = {};

  paramsArr.forEach(param => {
    if (/=/.test(param)) {
```

```
      let [key, val] = param.split('=');
      val = decodeURIComponent(val);
      val = /^\d+$/.test(val) ? parseFloat(val) : val;

      if (paramsObj.hasOwnProperty(key)) {
        paramsObj[key] = [].concat(paramsObj[key], val);
      } else {
        paramsObj[key] = val;
      }
    } else {
      paramsObj[param] = true;
    }
  })

  return paramsObj;
}
```

复制代码

模板渲染

Q: 实现一个简单的模板引擎:

```
let template = '我是{{name}}，年龄{{age}}，性别{{sex}}';
let data = {
  name: '姓名',
  age: 18
}
```

```
render(template, data);
```
复制代码

答案:

- 简单实现

```
function render(template, data) {
  const reg = /\{\{(\w+)\}\}/;
  if (reg.test(template)) {
    const name = reg.exec(template)[1];
    template = template.replace(reg, data[name]);
    return render(template, data);
  }
  return template;
}
```
复制代码

- 一行代码实现方式

```
function render(template, data) {
  return template.replace(new RegExp('{{(.*?)}}', 'g')
}
```
复制代码

参考

一行代码实现一个简单的模板字符串替换

Q: 实现一个简单的虚拟 DOM 渲染

```
let domNode = {
  tagName: 'ul',
  props: { class: 'list' },
  children: [{
    tagName: 'li',
    children: ['item1']
  }, {
    tagName: 'li',
    children: ['item1']
  }]
};

// 构建一个 render 函数，将 domNode 对象渲染为 以下 dom
<ul class="list">
    <li>item1</li>
    <li>item2</li>
</ul>
复制代码
```

答案:

```
function render(domNode) {
  if (!domNode) return document.createDocumentFragment();
  let $el
  if (typeof domNode === 'object') {
    $el = document.createElement(domNode.tagName);
```

```
    if (domNode.hasOwnProperty('props')) {
      for (let key in domNode.props) {
        $el.setAttribute(key, domNode.props[key]);
      }
    }

    if (domNode.hasOwnProperty('children')) {
      domNode.children.forEach(val => {
        const $childEl = render(val);
        $el.appendChild($childEl);
      })
    }
  } else {
    $el = document.createTextNode(domNode);
  }

  return $el;
}
```

复制代码

Q: 字符串查找

请使用最基本的遍历来实现判断字符串 a 是否被包含在字符串 b 中，并返回第一次出现的位置（找不到返回 -1）。

例子：

```
a='34';b='1234567';
a='35';b='1234567';
a='355';b='12354355';
isContain(a,b);
```

复制代码

答案:

```
function isContain(a, b) {
  for (let i in b) {
    if (a[0] === b[i]) {
      let tmp = true;
      for (let j in a) {
        if (a[j] !== b[~~i + ~~j]) {
          tmp = false;
        }
      }
      if (tmp) {
        return i;
      }
    }
  }
  return -1;
}
```

复制代码

Q: 将一个任意长的数字变成逗号分割的格式

例子:

```
parseToMoney(1234.56);
parseToMoney(123456789);
parseToMoney(1087654.321);
```
复制代码

答:

```
function parseToMoney(num) {
  num = parseFloat(num.toFixed(3));
  let [integer, decimal] = String.prototype.split.call(num
  integer = integer.replace(/\d(?=(\d{3})+$)/g, '$&,');
  return integer + '.' + (decimal ? decimal : '');
}
```
复制代码

Q: 数据绑定最基本的实现

```
let obj = {
  key_1: 1,
  key_2: 2
}
function func(key) {
```

```
  console.log(key + ' 的值发生改变: ' + this[key]);
}
bindData(obj, func);
obj.key_1 = 2;
obj.key_2 = 1;
```

复制代码

答:

```
function bindData(obj, fn) {
  for (let key in obj) {
    Object.defineProperty(obj, key, {
      set(newVal) {
        if (this.value !== newVal) {
          this.value = newVal;
          fn.call(obj, key);
        }
      },
      get() {
        return this.value;
      }
    })
  }
}
```

复制代码

Q: 数据结构处理

有一个祖先树状 json 对象，当一个人有一个儿子的时候，其 child 为其儿子对象，如果有多个儿子，child 为儿子对象的数组。

请实现一个函数，找出这个家族中所有有多个儿子的人的名字（name），输出一个数组。

```
let data = {
  name: 'jack',
  child: [
    { name: 'jack1' },
    {
      name: 'jack2',
      child: [{
        name: 'jack2-1',
        child: { name: 'jack2-1-1' }
      }, {
        name: 'jack2-2'
      }]
    },
    {
      name: 'jack3',
      child: { name: 'jack3-1' }
    }
  ]
}
```

复制代码

答案:

- 用递归

```
function findMultiChildPerson(data) {
  let nameList = [];

  function tmp(data) {
    if (data.hasOwnProperty('child')) {
      if (Array.isArray(data.child)) {
        nameList.push(data.name);
        data.child.forEach(child => tmp(child));
      } else {
        tmp(data.child);
      }
    }
  }
  tmp(data);
  return nameList;
}
```

复制代码

- 非递归

```
function findMultiChildPerson(data) {
  let list = [data];
  let nameList = [];

  while (list.length > 0) {
    const obj = list.shift();
```

```
    if (obj.hasOwnProperty('child')) {
      if (Array.isArray(obj.child)) {
        nameList.push(obj.name);
        list = list.concat(obj.child);
      } else {
        list.push(obj.child);
      }
    }
  }
  return nameList;
}
```

复制代码
