差不多这就是最后的一个版本了，请你对下面的属性和方法进行注释，并解读下面代码实现了那些动画效果，谈一谈这些方法是如何实现动画的效果的。<template>

```
  <div class=”carousel-item”>
    <!-- 占位图 -->
    <img
      :src=”imgInfo.placeholder”
      alt=””
      class=”placeholderImg”
      v-if=”showPlaceholderImg”
    />
    <!-- 原图 -->
    <img
      :src=”imgInfo.src”
      alt=””
      class=”originalImg”
      @load=”handleMask”
      :style=”originalImgStyle”
    />
    <!-- 图片文本信息 -->
    <div class=”text-info”>
      <p class=”desc-title”>
        {{ displayedTitle }}<span v-show=”showTitleCursor”>|</span>
      </p>
      <p class=”desc-content”>
        {{ displayedContent }}<span v-show=”showContentCursor”>|</span>
      </p>
    </div>
  </div>
</template>

<script>
export default {
  props: {
    imgInfo: { type: Object, required: true },
    isActive: { type: Boolean, required: true },
    isHover: { type: Boolean, required: true },
  },
  data() {
    return {
```

```
    // 蒙版相关
    showOriginalImg: !this.isActive,
    showPlaceholderImg: this.isActive,
    duration: 2000,

    // 打字相关
    displayedTitle: ””,
    displayedContent: ””,
    showTitleCursor: false,
    showContentCursor: false,

    // 计时器相关
    typingTimer: null,
    currentPhase: ”idle”, // idle | typingTitle | typingContent | deleting
    currentIndex: 0,
    isPaused: false,
  };
},
computed: {
  originalImgStyle() {
    return {
      opacity: this.showOriginalImg ? 1 : 0,
      transition: `opacity ${this.duration}ms ease-in-out`,
    };
  },
},
watch: {
  isActive(newVal) {
    this.handleMask();
  },
  isHover(newVal) {
    if (newVal) {
      this.pauseTyping();
    } else {
      this.resumeTyping();
    }
  },
},
methods: {
  handleMask() {
    if (this.isActive) {
      this.showOriginalImg = true;
      setTimeout(() => {
        this.showPlaceholderImg = false;
        this.startTyping();
    }, this.duration);
  } else {
    this.showOriginalImg = false;
    this.showPlaceholderImg = true;
    this.stopTyping();
  }
},

// 开始打字动画
startTyping(options = {}) {
  this.stopTyping();

  const config = {
    typingSpeed: 150,
    deleteSpeed: 75,
    restartDelay: 1200,
    loop: true,
    ...options,
  };

  this.currentPhase = ”typingTitle”;
  this.currentIndex = 0;
  this.config = config;

  this.typeNextCharacter();
},

// 处理下一个字符
typeNextCharacter() {
  if (this.isPaused) return;

  const speed = this.currentPhase.includes(”typing”)
    ? this.config.typingSpeed
    : this.config.deleteSpeed;

  switch (this.currentPhase) {
    case ”typingTitle”:
      if (this.currentIndex < this.imgInfo.desc.title.length) {
        this.displayedTitle = this.imgInfo.desc.title.substring(
          0,
          this.currentIndex + 1
        );
        this.showTitleCursor = true;
        this.currentIndex++;
        this.typingTimer = setTimeout(
      () => this.typeNextCharacter(),
      speed
    );
  } else {
    this.showTitleCursor = false;
    this.currentPhase = ”typingContent”;
    this.currentIndex = 0;
    this.typingTimer = setTimeout(
      () => this.typeNextCharacter(),
      speed
    );
  }
  break;

case ”typingContent”:
  if (this.currentIndex < this.imgInfo.desc.content.length) {
    this.displayedContent = this.imgInfo.desc.content.substring(
      0,
      this.currentIndex + 1
    );
    this.showContentCursor = true;
    this.currentIndex++;
    this.typingTimer = setTimeout(
      () => this.typeNextCharacter(),
      speed
    );
  } else {
    if (this.config.loop) {
      this.currentPhase = ”deleting”;
      this.typingTimer = setTimeout(
        () => this.typeNextCharacter(),
        this.config.restartDelay
      );
    }
  }
  break;

case ”deleting”:
  if (this.displayedContent.length > 0) {
    this.displayedContent = this.displayedContent.substring(
      0,
      this.displayedContent.length - 1
    );
    this.typingTimer = setTimeout(
      () => this.typeNextCharacter(),
          speed
        );
      } else if (this.displayedTitle.length > 0) {
        this.displayedTitle = this.displayedTitle.substring(
          0,
          this.displayedTitle.length - 1
        );
        this.typingTimer = setTimeout(
          () => this.typeNextCharacter(),
          speed
        );
      } else {
        // 循环
        this.currentPhase = ”typingTitle”;
        this.currentIndex = 0;
        this.typingTimer = setTimeout(
          () => this.typeNextCharacter(),
          this.config.restartDelay
        );
      }
      break;
  }
},

// 暂停打字
pauseTyping() {
  if (this.currentPhase === ”idle” || this.isPaused) return;

  this.isPaused = true;
  clearTimeout(this.typingTimer);
  this.showTitleCursor = false;
  this.showContentCursor = false;
},

// 恢复打字
resumeTyping() {
  if (!this.isPaused) return;

  this.isPaused = false;
  this.typeNextCharacter();
},

// 停止打字
stopTyping() {
  clearTimeout(this.typingTimer);
```

```
      this.currentPhase = ”idle”;
      this.displayedTitle = ””;
      this.displayedContent = ””;
      this.showTitleCursor = false;
      this.showContentCursor = false;
      this.isPaused = false;
    },
  },
  beforeDestroy() {
    this.stopTyping();
  },
};
</script>

<style lang=”less” scoped>
/* 样式保持不变 */
.carousel-item {
  width: 100%;
  height: 100%;
  position: relative;
  overflow: hidden;

  .originalImg,
  .placeholderImg {
    width: 110%;
    height: 110%;
    position: absolute;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%);
    object-fit: cover;
  }

  .placeholderImg {
    filter: blur(2vw);
  }

  .text-info {
    position: absolute;
    left: 30px;
    top: 30%;
    letter-spacing: 3px;
    color: #fff;
    text-shadow: 1px 0 0 rgba(0, 0, 0, 0.5), -1px 0 0 rgba(0, 0, 0, 0.5),
      0 1px 0 rgba(0, 0, 0, 0.5), 0 -1px 0 rgba(0, 0, 0, 0.5);
  }

  .desc-title {
    font-size: 24px;
    margin-bottom: 15px;
    min-height: 28px;
  }

  .desc-content {
    font-size: 16px;
    line-height: 1.5;
    min-height: 20px;
  }

  .desc-content span,
  .desc-title span {
    animation: blink 0.7s infinite;
  }

  @keyframes blink {
    0%,
    100% {
      opacity: 1;
    }
    50% {
      opacity: 0;
    }
  }
}
</style>
```

代码注释与解读

1. 模板部分 (`<template>`)

HTML

```
<template>
 <div class=”carousel-item”>
  <!-- 占位图（模糊效果） -->
  <img :src=”imgInfo.placeholder” class=”placeholderImg” v-if=”showPlaceholderImg”/>

  <!-- 原图（带加载过渡效果） -->
  <img :src=”imgInfo.src” class=”originalImg” @load=”handleMask” :style=”originalImgStyle”/>

  <!-- 打字机效果文本区域 -->
  <div class=”text-info”>
   <p class=”desc-title”>
    {{ displayedTitle }}<span v-show=”showTitleCursor”>|</span> <!-- 标题光标 -->
   </p>
   <p class=”desc-content”>
    {{ displayedContent }}<span v-show=”showContentCursor”>|</span> <!-- 内容光标 -->
   </p>
  </div>
 </div>
</template>
```

2. 数据属性（`data()`）

JavaScript

```
data() {
 return {
  // 图片蒙版控制
  showOriginalImg: !this.isActive,    // 是否显示原图
  showPlaceholderImg: this.isActive,  // 是否显示占位图
  duration: 2000,                     // 图片切换动画时长(ms)

  // 打字效果状态
  displayedTitle: ””,         // 当前显示的标题文本
  displayedContent: ””,       // 当前显示的内容文本
  showTitleCursor: false,     // 标题光标可见性
  showContentCursor: false,   // 内容光标可见性

  // 计时器控制
  typingTimer: null,         // 打字效果计时器
  currentPhase: ”idle”,      // 当前阶段：idle/typingTitle/typingContent/deleting
  currentIndex: 0,           // 当前打字/删除的字符索引
  isPaused: false            // 是否暂停状态
   };
}
```

3. 计算属性（`computed`）

JavaScript

```
computed: {
 // 原图样式（透明度过渡效果）
 originalImgStyle() {
  return {
   opacity: this.showOriginalImg ? 1 : 0,
   transition: `opacity ${this.duration}ms ease-in-out`
     };
   }
}
```

4. 主要方法 (`methods`)

(1) `handleMask()` - 图片加载处理

JavaScript

```
// 处理图片蒙版切换逻辑
handleMask() {
 if (this.isActive) {
  // 激活状态：显示原图 → 隐藏占位图 → 开始打字
  this.showOriginalImg = true;
  setTimeout(() => {
   this.showPlaceholderImg = false;
   this.startTyping();
  }, this.duration);
 } else {
  // 非激活状态：隐藏原图 → 显示占位图 → 停止打字
  this.showOriginalImg = false;
  this.showPlaceholderImg = true;
  this.stopTyping();
   }
}
```

(2) `startTyping()` - 启动打字效果

JavaScript

```
// 初始化打字效果
startTyping(options = {}) {
 this.stopTyping(); // 先停止现有效果

 // 合并配置参数
 const config = {
  typingSpeed: 150,   // 打字速度(ms/字符)
  deleteSpeed: 75,    // 删除速度(ms/字符)
  restartDelay: 1200, // 循环间隔(ms)
  loop: true,         // 是否循环
     ...options
   };

 // 初始化状态
 this.currentPhase = ”typingTitle”; // 从标题开始
 this.currentIndex = 0;
 this.config = config;

 this.typeNextCharacter(); // 开始第一个字符
}
```

(3) `typeNextCharacter()` - 核心打字逻辑

JavaScript

[redacted]

```
// 处理每个字符的显示/删除
typeNextCharacter() {
  if (this.isPaused) return;

  // 根据当前阶段选择速度
  const speed = this.currentPhase.includes(”typing”)
    ? this.config.typingSpeed
    : this.config.deleteSpeed;

  switch (this.currentPhase) {
    // 标题打字阶段
    case ”typingTitle”:
      if (this.currentIndex < this.imgInfo.desc.title.length) {
        // 逐个显示标题字符
        this.displayedTitle = this.imgInfo.desc.title.substring(0, this.currentIndex + 1);
        this.showTitleCursor = true;
        this.currentIndex++;
        this.typingTimer = setTimeout(() => this.typeNextCharacter(), speed);
      } else {
        // 标题打完，切换到内容打字
        this.showTitleCursor = false;
        this.currentPhase = ”typingContent”;
        this.currentIndex = 0;
        this.typingTimer = setTimeout(() => this.typeNextCharacter(), speed);
              }
      break;

    // 内容打字阶段
    case ”typingContent”:
      if (this.currentIndex < this.imgInfo.desc.content.length) {
        // 逐个显示内容字符
        this.displayedContent = this.imgInfo.desc.content.substring(0, this.currentIndex + 1);
        this.showContentCursor = true;
        this.currentIndex++;
        this.typingTimer = setTimeout(() => this.typeNextCharacter(), speed);
      } else if (this.config.loop) {
        // 内容打完，切换到删除阶段
        this.currentPhase = ”deleting”;
        this.typingTimer = setTimeout(() => this.typeNextCharacter(), this.config.restartDelay);
              }
      break;

    // 删除阶段
    case ”deleting”:
      if (this.displayedContent.length > 0) {
        // 逐个删除内容字符
        this.displayedContent = this.displayedContent.substring(0, this.displayedContent.length - 1);
        this.typingTimer = setTimeout(() => this.typeNextCharacter(), speed);
      } else if (this.displayedTitle.length > 0) {
        // 内容删完，删除标题
        this.displayedTitle = this.displayedTitle.substring(0, this.displayedTitle.length - 1);
        this.typingTimer = setTimeout(() => this.typeNextCharacter(), speed);
      } else if (this.config.loop) {
        // 全部删完，重新开始循环
        this.currentPhase = ”typingTitle”;
        this.currentIndex = 0;
        this.typingTimer = setTimeout(() => this.typeNextCharacter(), this.config.restartDelay);
              }
      break;
    }
}
```

(4) 控制方法

JavaScript

```javascript
// 暂停打字效果
pauseTyping() {
  if (this.currentPhase === ”idle” || this.isPaused) return;

  this.isPaused = true;
  clearTimeout(this.typingTimer); // 清除计时器
  this.showTitleCursor = false;  // 隐藏光标
  this.showContentCursor = false;
}

// 恢复打字效果
resumeTyping() {
  if (!this.isPaused) return;

  this.isPaused = false;
  this.typeNextCharacter(); // 从暂停点继续
}

// 完全停止打字效果
stopTyping() {
  clearTimeout(this.typingTimer);
  this.currentPhase = ”idle”;
  this.displayedTitle = ””;
  this.displayedContent = ””;
  this.showTitleCursor = false;
  this.showContentCursor = false;
  this.isPaused = false;
}
```

5. 动画效果实现原理

(1) 图片过渡效果

- **实现方式**：通过 `originalImgStyle` 计算属性控制 `opacity` 的 CSS 过渡
- **触发条件**：图片加载完成时触发 `handleMask`
- **效果**：原图淡入，占位图淡出（通过 `v-if` 控制）

(2) 打字机效果

- **核心机制**：递归调用 `setTimeout` 的 `typeNextCharacter` 方法
- **三个阶段**：
  - **typingTitle**：逐个显示标题字符
  - **typingContent**：逐个显示内容字符
  - **deleting**：反向逐个删除字符
- **速度控制**：通过 `typingSpeed` 和 `deleteSpeed` 参数控制每个字符的显示/删除时间
- **光标效果**：通过 `v-show` 动态显示/隐藏光标，配合 CSS 动画实现闪烁

(3) 暂停/恢复功能

- **暂停**：清除计时器并记录状态
- **恢复**：从当前状态继续执行 `typeNextCharacter`
- **悬停控制**：通过 `watch` 监听 `isHover` 属性自动触发

(4) 循环播放

- **实现方式**：当完成删除阶段后，通过 `config.loop` 判断是否重新开始打字
- **间隔控制**：`restartDelay` 参数控制重新开始前的停顿时间

6. 样式亮点

- **光标闪烁**：通过 `@keyframes blink` 实现
- **文本阴影**：`text-shadow` 增强文字可读性
- **占位图模糊**：`filter: blur(2vw)` 创建毛玻璃效果
- **响应式布局**：使用 `vw` 单位和百分比尺寸

总结

这段代码实现了三种主要动画效果：

1. **图片过渡动画**：通过 CSS 透明度过渡实现平滑切换
2. **打字机效果**：通过 JavaScript 定时器模拟逐字打印和删除
3. **光标闪烁动画**：通过 CSS 关键帧动画实现

所有动画都可以通过配置参数精确控制速度，并且完美支持暂停/恢复功能。相比 `requestAnimationFrame` 版本，这个实现更简洁直观，适合大多数需要打字机效果的场景。