##(六)Flex弹性盒模型

>\*\*1:基本概念\*\*

-\*\*2009年W3C新方案—-Flex 布局(弹性盒子模型),可以简便、完整、响应式地实现各种页面布局\*\*

-\*\*Flex布局元素为Flex容器(flex container),子元素为容器成员(flex item)\*\*

-\*\*容器默认存在两根轴\*\*

-水平主轴(main axis)

- main start

- main end

-垂直交叉轴(cross axis)

- cross start

- cross end

![image](./flex.png)

-\*\*默认沿主轴排列\*\*

-\*\*单个容器成员占据的主轴空间叫做main size,占据的交叉轴空间叫做cross size\*\*

>\*\*2:发展历程\*\*

-\*\*最老版本\*\*

- box (弹性伸缩盒)

- inline-box (内联块级弹性伸缩盒)

-\*\*过渡版本\*\*

- flexbox (弹性伸缩盒)

- inline-flexbox (内联块级弹性伸缩盒)

-\*\*最新版本\*\*

- flex (弹性伸缩盒)

- inline-flex (内联块级弹性伸缩盒)

>\*\*3:容器(flex container)属性\*\*

```css

flex-direction

flex-wrap

flex-flow

justify-content

align-items

align-content

```

-\*\*flex-direction (决定主轴方向)\*\*

- row (横向从左到右排列-左对齐)

- row-reverse (反转横向排列-右对齐)

- column (纵向排列)

- column-reverse (反转纵向排列)

```css

.main {

display: flex;

flex-direction: row;

}

```

-\*\*flex-wrap (换行)\*\*

- nowrap (子元素溢出时不换行)

- wrap (子元素溢出时自动换行)

- wrap-reverse (反转 wrap 排列)

```css

.main {

display: flex;

flex-wrap: nowrap;

}

```

-\*\*flex-flow (flex-direction与flex-wrap简写)\*\*

```css

.main {

display: flex;

flex-flow: wrap row;

}

```

-\*\*justify-content (容器成员(flex item)主轴上的对齐方式)\*\*

- flex-start (主轴起始位置对齐-左对齐)

- flex-end (主轴结束位置对齐-右对齐)

- center (主轴中间位置对齐-居中)

-未溢出:第一个元素与主轴起始位置的边距===最后一个元素与主轴起始位置的边距  
-溢出:第一个元素溢出===最后一个元素溢出

- space-between (平均地分布在主轴上)

-未溢出:两两之间间隔相等

-溢出

-只有一个子元素(flex-start)

-最左边的剩余空间是负数(flex-start)

- space-around (平均地分布在主轴上,两端保留子元素与子元素之间间距大小的一半)

-未溢出

-溢出

-只有一个子元素(center)

-最左边的剩余空间是负数(center)

- space-evenly (平均地分布在主轴上,两端保留子元素与子元素之间间距相同间距)

-未溢出

-溢出

-只有一个子元素(center)

-最左边的剩余空间是负数(center)

```css

.main {

display: flex;

justify-content: center;

}

```

-\*\*align-items (容器成员(flex item)交叉轴上的对齐方式)\*\*

- stretch (拉伸容器成员(flex item)适应容器-默认属性)

- flex-start (交叉轴起始位置对齐-上对齐)

- flex-end (交叉轴结束位置对齐-下对齐)

- center (交叉轴中间位置对齐-居中)

-未溢出:容器成员上边距交叉轴起始位置距离===容器成员上边距交叉轴起始位置距离  
-溢出:两个方向溢出相同的长度

- baseline (容器成员(flex item)基线对齐)

```css

.main {

display: flex;

align-items: center;

}

```

-\*\*align-content (多根轴线的对齐方式,单轴线不起作用-每一根轴线理解为一个容器成员(flex item))\*\*

- flex-start

- flex-end

- center

- space-between

- space-around

- stretch

>\*\*3:容器成员(flex item)属性\*\*

```css

order

flex-grow

flex-shrink

flex-basis

flex

align-self

```

-\*\*order (容器成员排列顺序- integer)\*\*

-0(默认-数值越小,排列越靠前,可以为负值)

```css

.item {

order:1;

}

```

-\*\*flex-grow (容器成员放大比例- number)\*\*

-0(默认-数值越大,占据的剩余空间越大,不允许负值)

```css

.item {

flex-grow:1;

}

```

-\*\*flex-shrink (容器成员缩小比例- number)\*\*

-1(默认-数值越大,压缩比例越大,不允许负值)

-最小宽度制约

```css

.item {

flex-shrink:0;

}

```

-\*\*flex-basis (分配多余空间之前,容器成员占据的主轴空间)\*\*

- auto (容器成员原本大小)

- px

-%

```css

.item {

flex-basis:50px;

}

```

-\*\*flex (flex-grow, flex-shrink 和 flex-basis的简写)\*\*

-顺序

- auto -(11 auto)

- none -(00 auto)

-1-(11 auto)

```css

.item {

flex: auto;

}

```

-\*\*align-self (允许单个容器成员有与其他容器成员不同的对齐方式)\*\*

- auto (默认)

- flex-start

- flex-end

- center

- baseline

- stretch

```css

.item {

align-self: auto;

}

```

>\*\*4:课后练习\*\*

- Flex container存在下列哪几个描述,分别代表什么

```css

A : Main axis

B : Cross axis

C : Row axis

D :Column axis

```

- Flex item存在下列哪几个描述,分别代表什么

```css

A : Main size

B : Cross size

C : Row size

D :Column size

```

- item元素的实际宽度是多少

```css

.main {

display: flex;

}

.main .item {

flex-basis:200px;

width:100px;

}

```

- Flex container具有哪几个属性,分别代表什么

- Flex item具有哪几个属性,分别代表什么

- class为two的容器宽度是多少?并说明原因(实操)

```css

.main {

display: flex;

background:#8A469B;

}

.main span{

width:800px;

height:50px;

border:1px solid #FFF;

box-sizing: border-box;

background:#EA7F26;

}

.zero {

flex-shrink:0;

}

.one {

flex-shrink:1;

}

.two {

flex-shrink:9;

}

<div class="main">

<span class="zero">1</span>

<span class="one">2</span>

<span class="two">3</span>

</div>

```

-计算出各个 li 元素实际宽度(计算)

```css

.main {

display: flex;

width:400px;

margin:0;

padding:0;

list-style: none;

}

.main li {

width:200px;

}

.main li:nth-child(1){

background:#888;

}

.main li:nth-child(2){

background:#ccc;

}

.main li:nth-child(3){

flex-shrink:3;

background:#aaa;

}

```

>\*\*`5:总结`\*\*

```

本节课讲解了Flex弹性盒模型的基本概念,简要介绍了Flex弹性盒模型的发展历程,最后重点介绍了Flex container六个属性以及Flex item的六个属性  
```

##(七)弹性盒模型(Flexbox)常见例子

>\*\*1:属性回顾\*\*

-\*\*flex container\*\*

```css

flex-direction

flex-wrap

flex-flow

justify-content

align-items

align-content

```

-\*\*flex item\*\*

```css

order

flex-grow

flex-shrink

flex-basis

flex

align-self

```

>\*\*2:垂直居中对齐\*\*

-\*\*不使用Flexbox\*\*

```css

/\* transform \*/

position: absolute;

left:50%;

top:50%;

transform: translate(-50%,-50%);

/\* margin \*/

margin: auto;

position: absolute;

top:0;

left:0;

bottom:0;

right:0;

/\* table \*/

/\* IE8推出的标准比较早期的布局技术\*/

display: table-cell;

vertical-align: middle;

text-align: center;

/\*-margin \*/

width:100px;

height:100px;

position: absolute;

left:50%;

top:50%;

margin:-50px 00-50px;

```

-\*\*使用Flexbox\*\*

```css

display: flex;

justify-content: center;

align-items: center;

```

>\*\*2:自适应导航\*\*

-\*\*不使用Flexbox\*\*

```css

.main {

text-align: right;

}

.main li {

display: inline-block;

}

/\*小于800px 大于600px \*/

@media screen and (max-width:800px){

.main {

text-align: justify;

text-align-last: justify;

}

}

/\*小于600px \*/

@media screen and (max-width:600px){

.main li {

display: block;

}

.main a {

display: block;

text-align: center;

text-align-last: center;

}

}

```

-\*\*使用Flexbox\*\*

```css

.main {

display: flex;

flex-flow: row wrap;

justify-content: flex-end;

}

/\*小于800px 大于600px \*/

@media screen and (max-width:800px){

.main {

justify-content: space-between;

}

}

/\*小于600px \*/

@media screen and (max-width:600px){

.main {

flex-flow: column nowrap;

}

}

```

>\*\*3:圣杯布局\*\*

-\*\*不使用Flexbox\*\*

```css

.left,

.middle,

.right {

position: relative;

float: left;

}

.container {

padding:0200px 0200px;

}

.left {

margin-left:-100%;

left:-200px;

width:200px;

}

.right {

margin-left:-200px;

right:-200px;

width:200px;

}

.middle {

width:100%;

}

```

-\*\*使用Flexbox\*\*

```css

.flex {

display: flex;

flex-wrap: nowrap;

}

.leftBar {

width:200px;

flex-shrink:0;

}

.container {

width:100%;

}

.rightBar {

width:200px;

flex-shrink:0;

}

```

>\*\*4:课后练习\*\*

-使用Flexbox实现下图样式

![image](./dice.png)

```css

/\*骰子外阴影\*/

box-shadow:

inset 05px white,

inset 0-5px #bbb,

inset 5px 0#d7d7d7,

inset -5px 0#d7d7d7;

/\*骰子点数阴影\*/

box-shadow: inset 03px #111, inset 0-3px #555;

```

-总结这两节课使用Flexbox感受

>\*\*5:总结\*\*

```

本节课从三个例子入手(垂直居中、响应式、圣杯)介绍了Flexbox的主要应用场景,并与传统方式对比,感受Flexbox布局带来的便捷开发体验  
```

##(八)CSS3渐变

>\*\*`1:基本概念`\*\*

-\*\*在两个或多个指定的颜色之间显示平稳的过渡\*\*

![image](./css.png)

-\*\*渐变效果由浏览器生成\*\*

-\*\*渐变类型\*\*

-线性渐变(Linear Gradients)-向下/向上/向左/向右/对角方向

-径向渐变(Radial Gradients)-由它们的中心定义

-\*\*作为元素的 background-image( background )使用\*\*

>\*\*`2:浏览器支持`\*\*

-\*\*完全支持该属性的第一个浏览器版本\*\*

![image](./borwer.png)

-\*\*浏览器前缀\*\*

```css

.header {

/\* Safari \*/

background:-webkit-linear-gradient(red, blue);

/\* Opera \*/

background:-o-linear-gradient(red, blue);

/\* Firefox \*/

background:-moz-linear-gradient(red, blue);

/\*标准的语法,放在最后\*/

background: linear-gradient(red, blue);

}

```

>\*\*`3:线性渐变(Linear Gradients)`\*\*

-\*\*基本语法\*\*

```css

background-image: linear-gradient([[<angle>| to <side-or-corner>],]?<color-stop>[,<color-stop>]+);

```

-\*\*从上到下-默认方向\*\*

```css

background-image: linear-gradient(#147B96,#E6D205);

```

-\*\*从左到右\*\*

```css

to top/bottom/left/right

background: linear-gradient(to right,#147B96,#E6D205);

```

-\*\*对角\*\*

```css

to left top/left bottom/right bottom/right top

background: linear-gradient(to bottom right,#147B96,#E6D205);

```

-\*\*使用角度\*\*

![image](./angle.png)

```css

background: linear-gradient(45deg,#147B96, transparent);

```

-\*\*多种颜色\*\*

```css

background-image: linear-gradient(to right,#147B96,#E6D20525%,#147B9650%,#E6D20575%,#147B96);

background-image: linear-gradient(to right,#147B96,#E6D20550px,#147B9650px,#E6D20550px,#147B96);

```

-\*\*重复渐变\*\*

```css

background-image: repeating-linear-gradient(to right,#147B96,#E6D2055%,#147B9610%,#E6D20520%,#147B96);

```

-\*\*声明多个-逗号分隔-最先声明优先级最高\*\*

```css

background: linear-gradient(green 10px, transparent 10px),

linear-gradient(90deg,green 10px, transparent 10px) repeat left top /40px,

linear-gradient(transparent 40px, green 4px);

```

>\*\*`4:线性渐变实例`\*\*

-\*\*炫彩文字\*\*

```css

.linear {

color: green;

background-image: linear-gradient(

to left,

rgb(16,121,148),

rgb(230,211,13)25%,

rgb(18,136,165)50%,

rgb(231,213,10)75%,

rgb(16,121,148)

);

background-size:200%100%;

-webkit-text-fill-color: transparent;

-webkit-background-clip: text;

animation: mask 4s infinite linear;

}

@keyframes mask {

0%{ background-position:00;}

100%{ background-position:-100%0;

}

```

-\*\*炫彩文字\*\*

```css

background: linear-gradient(-60deg, blue 50%, transparent 0) no-repeat top left /50%50%,

linear-gradient(60deg, blue 50%, transparent 0) no-repeat top right /50%50%,

linear-gradient(-120deg, blue 50%, transparent 0) no-repeat bottom left /50%50%,

linear-gradient(120deg, blue 50%, transparent 0) no-repeat bottom right /50%50%;

```

>\*\*`5:径向渐变(Radial Gradients)`\*\*

-\*\*基本语法\*\*

```css

background-image: radial-gradient([[<shape>||<size>][at <position>]?,| at <position>,]?<color-stop>[,<color-stop>]+);

```

-\*\*渐变中心-中心点\*\*

-30px (20%)20px (30%)-左侧30px (30%)距离上侧20px (20%)

```css

ackground-image: radial-gradient(0%0%, red, green, blue);

```

-\*\*渐变形状- ellipse (椭圆形)\*\*

- circle

- ellipse

```css

background-image: radial-gradient(circle at farthest-corner, red, green, blue);

```

-\*\*渐变大小- farthest-corner (最远)\*\*

- closest-side -渐变的半径长度为从圆心到离圆心最近的边

- farthest-side -渐变的半径长度为从圆心到离圆心最近的角

- closest-corner -渐变的半径长度为从圆心到离圆心最远的边

- farthest-corner -渐变的半径长度为从圆心到离圆心最远的角  
-...

```css

background-image: radial-gradient(circle at farthest-corner, red, green, blue);

```

-\*\*多种颜色\*\*

```css

...

```

-\*\*重复渐变\*\*

```css

...

```

-\*\*声明多个-逗号分隔-最先声明优先级最高\*\*

```css

...

```

>\*\*`6:课后练习`\*\*

-为什么针对Gradients属性特别推荐autoprefixer

-利用Linear Gradients实现下列样式(10px \*10px )(10\*5)

![image](./white.png)

-利用Radial Gradients实现下列样式(100px \*100px \*4)

![image](./picture.png)

>\*\*`7:总结`\*\*

```

本节课讲解了Linear Gradients与Radial Gradients的基本概念,介绍了两种渐变的常用属性,最后结合了Linear Gradients实例介绍了CSS3渐变在实际中的部分应用场景  
```

##(九)Grid网格布局(上)

>\*\*`1:基本概念(Grid Layout)`\*\*

-\*\*二维的基于网格的布局系统(同时处理列和行)\*\*

-\*\*第一个专门为解决布局问题而生的CSS模块\*\*

-\*\*`Grid Container`- Grid容器(设置了 display: gird 的元素)\*\*

```css

.main {

display: grid;

}

```

![image](./grid.png)

-\*\*`Grid Item`- Grid容器成员(直接子元素)\*\*

```css

<div class="main">

<div class="item"></div>

<div class="item"></div>

<div class="item"></div>

</div>

```

-\*\*`Grid Line`- Grid (行、列)网格线\*\*

-\*\*`垂直网格线(column grid lines)`\*\*

-\*\*`水平网格线(row grid lines)`\*\*

![image](./gridLine.png)

-\*\*`Grid Track`-两个相邻网格线之间的空间\*\*

![image](./gridTrack.png)

-\*\*`Grid Cell`-两个相邻的行和两个相邻的列网格线之间的空间,基础单元\*\*

![image](./gridCell.png)

-\*\*`Grid Area`-四个网格线包围的总空间,可以由任意数量的Grid Cell组成\*\*

![image](./gridArea.png)

>\*\*`2:浏览器兼容`\*\*

![image](./browser.png)

>\*\*`3:Grid容器(Grid Container)属性`\*\*

```css

display

grid-template-columns

grid-template-rows

grid-template-areas

grid-template

grid-column-gap

grid-row-gap

grid-gap

justify-items

align-items

justify-content

align-content

grid-auto-columns

grid-auto-rows

grid-auto-flow

grid

```

-\*\*`display`\*\*

-`grid (块级网格)`

-`inline-grid (行级网格)`

```css

.main {

display: grid;

}

.main {

display: inline-grid;

}

```

-\*\*`grid-template-columns/rows (定义网格的列/行)`\*\*

```css

.main {

grid-template-columns:[<Grid Line Name>]<Grid Track Size>...;

grid-template-rows:[<Grid Line Name>]<Grid Track Size>...;

}

```

-`基本用法`

```css

.main {

display: grid;

grid-template-columns:[columns-1]100px [columns-2]200px auto;

grid-template-rows:[rows-1]100px [rows-2]200px;

}

```

-`重复`

```css

.main {

display: grid;

grid-template-columns: repeat(4,100px [columns]) auto;

grid-template-rows: repeat(4,100px [rows]);

}

```

-`自由空间`

```css

.main {

display: grid;

grid-template-columns:100px 1fr 3fr;

grid-template-rows:100px 1fr 3fr;

}

```

-\*\*`grid-template-areas (定义网格模板)`\*\*

```css

.header {

grid-area: header;

background:#8A469B;

}

.left {

grid-area: left;

background:#EA7F26;

}

.right {

grid-area: right;

background:#EA7F26;

}

.footer {

grid-area: footer;

background:#8A469B;

}

.main {

height:500px;

display: grid;

grid-template-columns:100px 100px auto 100px 100px;

grid-template-rows:100px auto 100px;;

grid-template-areas:

"header header header header header"

"left left .. right"

"footer footer footer footer footer";

}

```

-\*\*`grid-template-areas (rows、columns、areas简写)`\*\*

```css

.main {

height:500px;

display: grid;

grid-template:

[title-left]"title title title"80px [title-right]

[content-left]"left content content"1fr [content-right]

[footer-left]"left footer footer"80px [footer-right]

/120px 1fr 120px;

}

```

-\*\*`grid-column/rows-gap (网格线宽度/高度)`\*\*

```css

.main {

display: grid;

grid-template-columns:100px 200px auto;

grid-template-rows:100px 200px;

grid-column-gap:20px;

grid-row-gap:20px;

}

```

-\*\*`grid-gap (rows、column缩写)`\*\*

```css

.main {

display: grid;

grid-template-columns:100px 200px auto;

grid-template-rows:100px 200px;

grid-gap:20px 30px;

}

```

-\*\*`justify-items (元素在Grid Cell横轴上的对齐方式)`\*\*

-`start`

-`end`

-`center`

-`stretch -默认值`

```css

.main {

display: grid;

justify-items: start;

}

```

-\*\*`align-items (元素在Grid Cell纵轴上的对齐方式)`\*\*

-`start`

-`end`

-`center`

-`stretch -默认值`

```css

.main {

display: grid;

align-items: start;

}

```

-\*\*`justify-content (Grid Cell在横轴上的对齐方式)`\*\*

-`start`

-`end`

-`center`

-`stretch`

-`space-around`

-`space-between`

-`space-evenly`

```css

.main {

display: grid;

justify-content: start;

}

```

-\*\*`align-content (Grid Cell在纵轴上的对齐方式)`\*\*

-`start`

-`end`

-`center`

-`stretch`

-`space-around`

-`space-between`

-`space-evenly`

```css

.main {

display: grid;

align-content: start;

}

```

-\*\*`grid-auto-columns/rows (自动生成网格(隐式)-网格溢出时使用)`\*\*

-`基本语法`

```css

.main {

display: grid;

grid-auto-columns:<Grid Track Size>...;

grid-auto-rows:<Grid Track Size>...;

}

```

-`不指定 grid-template-columns`

-`指定 grid-template-columns`

-\*\*`grid-auto-flow (自动放置Grid容器成员)`\*\*

-`row`-按行填充

-`column`-按列填充

-`dense`-按最小剩余空间填充(打乱布局,使用较少)

```css

.main {

display: grid;

grid-auto-flow: row;

}

```

-\*\*`grid (template-rows、template-columns、template-areas、auto-rows、auto-columns、auto-flow简写)`\*\*

-`template-areas template-rows auto-flow auto-rows / grid-template-columns auto-flow auto-columns`

```css

.main {

grid:[row1-start]"header header header"1fr [row1-end]

[row2-start]"footer footer footer"25px [row2-end]

/ auto 50px auto;

}

```

##(十)Grid网格布局(下)

>\*\*`4:Grid容器成员(Grid Item)属性`\*\*

```css

grid-column-start

grid-column-end

grid-row-start

grid-row-end

grid-column

grid-row

grid-area

justify-self

align-self

```

-\*\*`grid-column-start/end (根据网格线确定Grid Item位置)`\*\*

-`number`

-`name`

-`span number`

-`span name`

```css

.main {

display: grid;

grid-template-columns:100px 100px 100px;

grid-template-rows:100px 100px 100px;

}

.item {

grid-column-start:2;

grid-column-end: span 4;

}

```

-\*\*`grid-row-start/end (根据网格线确定Grid Item位置)`\*\*

-`number`

-`name`

-`span number`

-`span name`

```css

.main {

display: grid;

grid-template-columns:100px 100px 100px;

grid-template-rows:100px 100px 100px;

}

.item {

grid-row-start: span 2;

grid-row-end:4;

}

```

-\*\*`grid-column/row (grid-column/row-start/end简写)`\*\*

```css

.main {

display: grid;

grid-template-columns:100px 100px 100px;

grid-template-rows:100px 100px 100px;

}

.item {

grid-column: span 2/4;

grid-row:1/4;

}

```

-\*\*`grid-area (创建区域)`\*\*

-`命名引用`

```css

grid-area: name;

```

-`直接定义`

```css

grid-area: name | row-start | column-start | row-end | column-end   
```

```css

.main {

display: grid;

grid-template-columns:100px 100px 100px;

grid-template-rows:100px 100px 100px;

}

.item {

grid-area: item;

grid-area:3/1/4/6;

}

```

-\*\*`justify-self (元素在Grid Cell横轴上的对齐方式-只对单个元素生效)`\*\*

-`start`

-`end`

-`center`

-`stretch`

```css

.main {

display: grid;

grid-template-columns:100px 100px 100px;

grid-template-rows:100px 100px 100px;

}

.item {

justify-self: start;

}

```

-\*\*`align-self (元素在Grid Cell纵轴上的对齐方式-只对单个元素生效)`\*\*

-`start`

-`end`

-`center`

-`stretch`

```css

.main {

display: grid;

grid-template-columns:100px 100px 100px;

grid-template-rows:100px 100px 100px;

}

.item {

align-self: start;

}

```

>\*\*`5:Grid网格布局注意事项`\*\*

-`Grid网格布局适用于页面整体布局`

-`Grid容器成员float、table-cell、inline-block属性无效`

-`不推荐中文命名,中文命名会导致name无法匹配`

>\*\*`6:课后练习`\*\*

- Grid Cell与Grid Area的关系是

```css

A:并集关系

B:子集关系

C:交集关系

```

-与 grid-template-columns:1fr 1fr 1fr 等效的是  
```css

A: grid-template-columns: auto auto auto

B: grid-template: auto auto auto /100px 100px 100px

C: grid-template:100px 100px 100px / auto auto auto

D: grid: auto auto auto /100px 100px 100px   
```

-与图片中数字6对应的区域是

![image](./nine.png)

```css

A: grid-area:1/2/2/3

B: grid-area:2/3/3/4

C: grid-area:2/3/4/3

D: grid-area:1/2/3/4

```

-谈谈对CSS一维布局和二维布局的理解

-使用Grid网格布局方式实现双飞燕布局,要求使用尽量少的代码(评选出最佳布局)

```css

table

div

flex

grid

```

![image](./sanGreal.png)

```css

header

height:80px;

background-color:#cdcdcd;

left

width:200px;

background-color:#ffff99;

center

width: auto;

background-color:#ee8888;

right

width:200px;

background-color:#ffff99;

footer

height:40px;

background-color:#cdcdcd;

```

-使用Grid网格布局方式实现下图布局,要求文字垂直居中对齐,使用尽量少的代码实现  
![image](./book.png)

```css

蓝色:#02BCD4

绿色:#019688

紫色:#673AB7

最小方格尺寸:60px \*60px

字体大小:8px

```

>\*\*`7:总结`\*\*

```

本节课讲解了Grid网格布局的基本概念,重点介绍了Grid Container十四个常用属性以及Grid Item的九个常用属性  
```

##(十一)Transform变换

>\*\*`1:基本概念(Grid Layout)`\*\*

-\*\*`应用于元素的2D或3D转换`\*\*

-\*\*`允许将元素旋转,缩放,移动,倾斜`\*\*

-\*\*`透视投影`\*\*

![image](./camera.png)

-\*\*`正交投影`\*\*

![image](./squre.png)

-\*\*`坐标系`-左手坐标系\*\*

![image](./left.png)

-\*\*`Transform-origin`-基准点\*\*

-\*\*`Transform-style`-元素呈现方式\*\*

-\*\*`Rotate`-旋转\*\*

-\*\*`Scale`-缩放\*\*

-\*\*`Skew`-倾斜\*\*

-\*\*`Translate`-移动\*\*

-\*\*`Perspective`-透视\*\*

>\*\*`2:浏览器兼容`\*\*

-\*\*完全支持Transform第一个浏览器版本\*\*

![image](./bowser1.png)

-\*\*完全支持Transform-origin第一个浏览器版本\*\*

![image](./bowser2.png)

-\*\*完全支持Transform-style第一个浏览器版本\*\*

![image](./bowser3.png)

>\*\*`3:Rotate`\*\*

-`rotate(angle)`-2D 旋转

```css

transform: rotate(45deg);

```

-`rotateX(angle)`-沿 X 轴3D旋转

```css

transform: rotateX(45deg);

```

-`rotateY(angle)`-沿 Y 轴3D旋转

```css

transform: rotateY(45deg);

```

-`rotateZ(angle)`-沿 Z 轴3D旋转

```css

transform: rotateZ(45deg);

```

-`rotate3d(x, y, z, angle)`-3D旋转

- x y z -介于0-1之间,确定三维空间唯一坐标点

- angle -围绕原点与x y z坐标点连线的旋转角度

-欧拉角与四元数(了解)

```css

transform: rotate3d(1,1,1,45deg);

```

>\*\*`4:Scale`\*\*

-`scale(x[, y])`-2D 缩放

```css

transform: scale(2);

```

-`scaleX(x)`-沿 X 轴缩放

```css

transform: scaleX(2);

```

-`scaleY(y)`-沿 Y 轴缩放

```css

transform: scaleY(2);

```

-`scaleZ(z)`-沿 Z 轴缩放-单独使用时没有任何效果  
```css

transform: scaleZ(2);

```

-`scale3d(x, y, z)`-定义每个方向上的缩放- Z方向单独使用时没有任何效果  
```css

transform: rotate(2,2,2);

```

>\*\*`5:Skew`\*\*

-`skewX(angle)`-沿 X 轴的2D 倾斜

```css

transform: skewX(45deg);

```

-`skewY(angle)`-沿 Y 轴的2D 倾斜

```css

transform: skewY(45deg);

```

-`skew(angle-x, angle-y)`-沿 X Y 轴的2D 倾斜  
```css

transform: skewY(45deg,45deg);

```

>\*\*`6:Translate`\*\*

-`translateX(x)`-沿 X 轴位移

```css

transform: translateX(20px);

```

-`translateY(y)`-沿 Y 轴位移

```css

transform: translateY(20px);

```

-`translateZ(z)`-沿 Z 轴位移

```css

transform: translateZ(20px);

```

-`translate(x, y)`-沿 X Y 轴位移

```css

transform: translate(20px,20px);

```

-`translate3d(x, y, z)`-沿 X Y Z 轴位移  
```css

transform: translate(20px,20px,20px);

```

>\*\*`7:Perspective`-视觉和元素3D空间Z平面之间的距离\*\*

```css

transform: perspective(200px);

```

>\*\*`8:Transform-origin`-基准点\*\*

-`基本语法`

```css

transform-origin: x-axis y-axis z-axis;

```

-`默认值`-`50%50%0`

-`x-axis`

-`left`

-`center`

-`right`

-`length`

-`%`

-`y-axis`

-`top`

-`center`

-`bottom`

-`length`

-`%`

-`z-axis`

-`length`

>\*\*`9:Transform-style`-三维空间呈现形式\*\*

-`flat`-`2D形式`

-`preserve-3d`-`3D形式`

>\*\*`10:课后练习`\*\*

-\*\*`CSS中Transform采用哪种投影,有什么特点`\*\*

```css

A:正交投影

B:透视投影

```

-\*\*`尝试使用Transform实现下图样式`\*\*

![image](./photo.png)

```css

image宽度300px

image高度220px

image padding 10px

倾斜角度不限制,协调即可

```

-\*\*`使用Flex布局结合Transform实现下图样式`\*\*

![image](./touzi.png)

```css

边长400px \*400px

紫色背景#8A469B

棱(边)#CDCDCD

点数#000

面#FFF

```

-\*\*`尝试使用Transform实现下图样式(选做)`\*\*

![image](./three.png)

>\*\*`11:总结`\*\*

```

本节课讲解了Transform的基本概念,简要介绍了CSS3坐标系统与透视原理,重点介绍了Transform常用的Rotate(旋转)、Scale(缩放)、Skew(倾斜)、Translate(移动)、Perspective(透视)属性  
```

##(十二)Transition过渡(上)

>\*\*`1:基本概念`\*\*

-\*\*`在一定时间区间内平滑地过渡(圆滑地以动画效果改变CSS的属性值)`\*\*

-\*\*`鼠标单击、获得焦点、被点击或对元素改变中触发`\*\*

-\*\*`Transition-duration`-过渡时间\*\*

-\*\*`Transition-property`-过渡 CSS 属性名称\*\*

-\*\*`Transition-delay`-过渡开始时间\*\*

-\*\*`贝塞尔曲线`\*\*

-\*\*`Transition-timing-function`-过渡效果时间曲线\*\*

-\*\*`Transition`-简写(设置四个属性)\*\*

>\*\*`2:浏览器兼容`\*\*

-\*\*完全支持Transform第一个浏览器版本\*\*

![image](./browser.png)

>\*\*`3:Transition-duration`\*\*

```css

transition-duration: s | ms;

```

-`必须搭配transition-property同时使用`

-`默认值`-0

>\*\*`5:Transition-duration`\*\*

```css

transition-duration: s | ms;

```

-`必须搭配transition-property同时使用`

-`默认值`-0

>\*\*`4:Transition-property`\*\*

```css

transition-property: none | all | property;

```

-`默认值`- all

-`必须搭配transition-duration同时使用`

-`none`-没有过渡效果(突变)

```css

transition-property: none;

```

-`all`-所有属性获取过渡效果

```css

transition-property: all;

```

-`property`-特定属性获得过渡效果(多个属性用逗号隔开)

```css

transition-property: width, height;

```

-`所有支持属性`

```css

background-color

background-position

border-color

border-width

border-spacing

bottom

clip

color

crop

font-size

font-weight

height

left

letter-spacing

line-height

margin

max-height

max-width

min-height

min-width

opacity

outline-color

outline-offset

outline-width

padding

right

text-indent

text-shadow

top

vertical-align

visibility

width

word-spacing

z-index

```

>\*\*`5:Transition-duration`\*\*

```css

transition-duration: s | ms;

```

-`必须搭配transition-property同时使用`

-`默认值`-0

>\*\*`5:Transition-delay`\*\*

```css

transition-delay: s | ms;

```

-`默认值`-0

>\*\*`6:贝塞尔曲线`\*\*

-`应用于二维图形应用程序的数学曲线`

-`绘制贝塞尔曲线`

![image](./bezier1.png)

![image](./bezier2.png)

![image](./bezier3.png)

![image](./bezier4.png)

![image](./bezier5.png)

![image](./bezier6.png)

-`函数形式的贝塞尔曲线`

-一阶贝塞尔曲线

![image](./func1.gif)

![image](./move1.gif)

-二阶贝塞尔曲线

![image](./func2.gif)

![image](./move2.gif)

-三阶贝塞尔曲线

![image](./func3.gif)

![image](./move3.gif)

-`cubic-bezier(n1, n2, n3, n4)`

- P0(0,0)

- P1(1,1)

- P3(n1, n2)

- P4(n3, n4)

-`曲线上点的切线对应运动速度`

-`http://cubic-bezier.com/`-可视化cubic-bezier编辑

>\*\*`7:Transition-timing-function`\*\*

```css

transition-timing-function: linear | ease | ease-in | ease-out | ease-in-out | cubic-bezier(n1, n2, n3, n4);

```

-`默认值`- ease

-`n1、n2、n3、n4取值`-官方推荐(0-1)-实际浏览器已经完全支持

-`linear`-以相同速度过渡- cubic-bezier(0,0,1,1)/ cubic-bezier(1,1,0,0)

![image](./linear.png)

-`ease`-慢速开始->变快->慢速结束- cubic-bezier(0.25,0.1,0.25,1)

![image](./ease.png)

-`ease-in`-慢速开始- cubic-bezier(0.42,0,1,1)

![image](./ease-in.png)

-`ease-out`-慢速结束- cubic-bezier(0,0,0.58,1)

![image](./ease-out.png)

-`ease-in-out`-慢速开始、慢速结束、幅度比ease大- cubic-bezier(0.42,0,0.58,1)

![image](./ease-in-out.png)

-`cubic-bezier(n, n, n, n)`-自定义速度函数(n介于0和1)

##(十三)Transition过渡(下)

>\*\*`8:悬浮菜单栏特效`\*\*

-`拉伸展开`

![image](./1.gif)

-`初始状态`

```css

.origin1{

opacity:0;

transform: rotateY(90deg);

transform-origin: center;

transition-property: all;

}

```

-`结束状态`

```css

.main li:hover .menu1:nth-child(1){

opacity:1;

transition-duration:.5s;

transition-delay:.3s;

transform: rotateY(0);

}

.main li:hover .menu1:nth-child(2){

opacity:1;

transition-duration:.5s;

transition-delay:.6s;

transform: rotateY(0);

}

.main li:hover .menu1:nth-child(3){

opacity:1;

transition-duration:.5s;

transition-delay:.9s;

transform: rotateY(0);

}

.main li:hover .menu1:nth-child(4){

opacity:1;

transition-duration:.5s;

transition-delay:1.2s;

transform: rotateY(0);

}

```

-`旋转移入`

![image](./2.gif)

-`初始状态`

```css

.origin2{

opacity:0;

transform: translateX(50px) rotate(-90deg);

transform-origin: top center;

transition-property: all;

}

```

-`结束状态`

```css

.main li:hover .menu2:nth-child(1){

opacity:1;

transform: translateX(0) rotate(0);

transition-duration:.5s;

transition-delay:.3s;

}

.main li:hover .menu2:nth-child(2){

opacity:1;

transform: translateX(0) rotate(0);

transition-duration:.5s;

transition-delay:.6s;

}

.main li:hover .menu2:nth-child(3){

opacity:1;

transform: translateX(0) rotate(0);

transition-duration:.5s;

transition-delay:.9s;

}

.main li:hover .menu2:nth-child(4){

opacity:1;

transform: translateX(0) rotate(0);

transition-duration:.5s;

transition-delay:1.2s;

}

```

>\*\*`9:悬浮卡片特效`\*\*

![image](./6.gif)

-`下拉幕布`

```css

.avatar:before {

position: absolute;

bottom:100%;

content:"";

width:100%;

height:0;

border-radius:50%;

background:#8A469B;

opacity:0.2;

transform: scale(3);

transition: all 0.3s linear 0s;

}

.avatar:hover:before {

height:100%;

}

```

-`紫色外边框`

```css

.avatar:after {

position: absolute;

top:0;

left:0;

content:"";

width:100%;

height:100%;

border-radius:50%;

background:#8A469B;

z-index:-1;

}

```

-`图片外环`

```css

.avatar img {

width:100%;

height: auto;

border-radius:50%;

transform: scale(1);

transition: all 0.9s ease 0s;

}

.avatar:hover img {

box-shadow:00014px #FFFFFF;

transform: scale(0.7);

}

```

-`图片替换`

```css

.teacher3{

position: absolute;

left:0;

top:0;

opacity:0;

}

.team:hover .teacher3{

opacity:1;

transform: scale(0.7);

}

```

>\*\*`10:3D效果`\*\*

>\*\*`11:Transition特点`\*\*

-`简单易用`

-`需要事件触发,无法自动发生`

-`一次性的,不能重复`

-`只能定义开始状态和结束状态,不能定义中间状态`

>\*\*`12:课后练习`\*\*

-\*\*`使用Transform和Transition实现下图效果`\*\*

![image](./3.gif)

![image](./4.gif)

-\*\*`使用Transform和Transition实现下图效果`\*\*

![image](./8.gif)

>\*\*`13:总结`\*\*

```css

本节课介绍了Transition的基本概念, Transition相关的数学基础,在此基础上介绍了duration、property、delay、timing-function常用属性,最后结合常见实例分析Transition的优缺点  
```

##(十四)媒体查询及响应式

>\*\*`1:基本概念`\*\*

-`根据一个或多个基于设备类型、具体特点和环境的媒体查询来应用样式`

-`针对不同的媒体类型定义不同的样式`

-`可以针对不同的屏幕尺寸设置不同的样式`

-`实现响应式页面`

>\*\*`2:浏览器兼容`\*\*

-\*\*完全支持Transition第一个浏览器版本\*\*

![image](./brower.png)

>\*\*`3:语法`\*\*

-`优先级-遵循CSS层叠样式规则`

```css

@media mediatype and|not|only (media feature){

CSS-Code;

}

```

```css

<link rel="stylesheet" media="mediatype and|not|only (media feature)" href="mystylesheet.css">

```

>\*\*`4:媒体类型`\*\*

-`all -所有设备`

```css

@media all and|not|only (media feature){

CSS-Code;

}

```

-`screen -电脑屏幕,平板电脑,智能手机`

```css

@media screen and|not|only (media feature){

CSS-Code;

}

```

-`print -打印机和打印预览`

```css

@media print and|not|only (media feature){

CSS-Code;

}

```

-`speech -屏幕阅读器等发声设备`

```css

@media speech and|not|only (media feature){

CSS-Code;

}

```

>\*\*`5:运算符`\*\*

-`and`

```css

@media screen and (max-width:600px){

CSS-Code;

}

```

-`not`

```css

@media not screen {

CSS-Code;

}

```

-`only`

```css

@media only screen {

CSS-Code;

}

```

-`,-或`

```css

@media print ,(max-width :600px){

CSS-Code;

}

```

>\*\*`6:媒体功能`\*\*

-`width / height`

```css

@media screen and (width :600px),(width :800px){

CSS-Code;

}

```

-`min-width / min-height`

```css

@media screen and (min-width :600px){

CSS-Code;

}

```

-`max-width / max-height`

```css

@media screen and (max-width :600px){

CSS-Code;

}

```

-`device-height -屏幕可见高度`

-`min-device-height`

-`max-device-height`

-`device-width -屏幕可见宽度`

-`max-device-width`

-`min-device-width`

-`(color:8)-每一组彩色原件的个数`

-`max-color`

-`min-color`

-`(color-index:1500)-设备的彩色查询表条目数`

-`max-color-index`

-`min-color-index`

-`(orientation: landscape)-横纵`

-`(monochrome:0)-一个单色框架缓冲区中每像素包含的单色原件个数`

-`min-monochrome`

-`max-monochrome`

-`(grid:1)-是否使用栅格或点阵`

-`(scan: interlace)-扫描工序`

-`(device-aspect-ratio:8/5)-屏幕可见区域宽高比`

-`min-device-aspect-ratio`

-`max-device-aspect-ratio`

-`(aspect-ratio:8/5)-页面可见区域宽高比`

-`min-aspect-ratio`

-`max-aspect-ratio`

-`(resolution:996dpi)-分辨率`

-`min-resolution`

-`max-resolution`

>\*\*`7:媒体查询与响应式布局`\*\*

![image](./example1.png)

![image](./example2.png)

-`常见的自适应布局方式`

>\*\*`8:课后练习`\*\*

-使用CSS媒体查询结合Transition与Transform实现响应式菜单栏  
![image](./9.gif)

```css

背景颜色 rgb(73,74,95)

菜单栏背景颜色#8A469B

左侧划出菜单栏背景颜色#222

鼠标悬浮左侧菜单划出

变换三角采用三条高度5px的div元素,悬浮旋转45deg

800px时菜单栏转换

```

>\*\*`9:总结`\*\*

```css

本节课介绍了媒体查询的基本概念,讲解了媒体类型、运算符和媒体功能的常用属性,最后介绍了媒体查询与响应式布局的相关内容  
```

##(十五)CSS动画

>\*\*`1:基本概念`\*\*

-\*\*`过渡动画- Transition`\*\*

-`初始状态`过渡到`结束状态`所产生的动画

-`只能定义初始和结束两个状态`

-`最简单的CSS动画`

-`被动触发`

-`不能重复发生`

-\*\*`关键帧动画- Animation -@keyframes `\*\*

```css

@keyframes 动画名称{

时间点{

元素状态

}

时间点{

元素状态

}

时间点{

元素状态

}

时间点{

元素状态

}

}

```

-`可以定义多个状态`

-`更复杂的动画效果`

-`可以主动触发`

-`可以重复发生`

>\*\*`2:浏览器支持`\*\*

-\*\*完全支持该属性的第一个浏览器版本\*\*

![image](./brower.png)

>\*\*`3:基本语法`\*\*

```css

.main {

animation: animationname duration timing-function delay iteration-count direction fill-mode play-state;

}

@keyframes animationname {

from { css-code }

to { css-code }

0%-100%{ css-code }

}

```

>\*\*`4:animationname -关键帧名称`\*\*

```css

.main {

animation-name: around;

animation-duration:5s;

animation-iteration-count: infinite;

}

@keyframes around {

from { css-code }

to { css-code }

0%-100%{ css-code }

}

```

>\*\*`5:duration -动画持续时间`\*\*

-`单位 s / ms`

```css

.main {

animation-name: around;

animation-duration:500ms;

animation-iteration-count: infinite;

}

@keyframes around {

from { css-code }

to { css-code }

0%-100%{ css-code }

}

```

>\*\*`6:timing-function -时间函数曲线`\*\*

```css

.main {

animation-name: around;

animation-duration:5s;

animation-iteration-count: infinite;

animation-timing-function: cubic-bezier(0.075,0.82,0.165,1);

}

```

-`时间区间两套CSS样式之间`

-`linear`

-`ease`

-`ease-in`

-`ease-out`

-`ease-in-out`

-`cubic-bezier(n, n, n, n)`

>\*\*`7:delay -动画开始延迟时间`\*\*

-`单位 s / ms`

-`仅定义第一次开始延迟时间`

```css

.main {

animation-name: around;

animation-duration:5s;

animation-iteration-count: infinite;

animation-timing-function: cubic-bezier(0.075,0.82,0.165,1);

animation-delay:3s;

}

```

>\*\*`8:iteration-count -动画播放次数`\*\*

```css

.main {

animation-name: around;

animation-duration:5s;

animation-iteration-count: infinite;

animation-timing-function: cubic-bezier(0.075,0.82,0.165,1);

animation-delay:3s;

}

```

-`n -具体的播放次数`

-`infinite -无限循环`

>\*\*`9:direction -动画播放方向`\*\*

```css

.main {

animation-name: around;

animation-duration:5s;

animation-iteration-count: infinite;

animation-timing-function: cubic-bezier(0.075,0.82,0.165,1);

animation-delay:3s;

animation-direction: alternate;

}

```

-`normal -正常播放`

-`reverse -反向播放`

-`alternate -奇数次正向,偶数次反向`

-`alternate-reverse -偶数次正向,奇数次反向`

>\*\*`10:fill-mode -动画不播放时应用到元素的样式`\*\*

```css

.main {

animation-name: around;

animation-duration:5s;

animation-iteration-count: infinite;

animation-timing-function: cubic-bezier(0.075,0.82,0.165,1);

animation-delay:3s;

animation-direction: alternate;

animation-fill-mode: forwards;

}

```

-`none -动画结束后恢复原有样式`

-`forwards -动画结束后动画将应用该属性值`

-`backwards -动画开始前应用第一帧的样式`

-`both`

>\*\*`11:play-state -控制动画运行或暂停`\*\*

-`paused`

-`running`

>\*\*`10:课后练习`\*\*

-关键帧动画动画的显著特点是

```css

A :连续播放

B :可以定义动画的中间状态

C :只能被动触发

D :实现复杂动画

```

-实现一个鼠标悬浮触发@keyframes的例子

```css

.main {

width:100px;

height:100px

}

@keyframes change {

from {

transform: rotateZ(0deg)

}

to {

transform: rotateZ(90deg)

}

}

```

-使用Animation实现其中一种轮播图

![image](./1.gif)

![image](./2.gif)

-使用CSS动画实现下图效果

![image](./3.gif)

-`要求`

```css

圆环外圈宽度高度:100px

圆环径向举例:20px

背景颜色:#8A469B

圆环颜色:#EA7F26

摇柄上下运动贝塞尔曲线:cubic-bezier(.5,0,.5,1)

```

-`提示`

```css

水平方向和垂直方向速度叠加

先完成垂直运动动画

```

![image](./run.png)

>\*\*`12:总结`\*\*

```

本节课介绍了CSS两种典型动画各自的特点,重点介绍了CSS帧动画的六个常用属性,最后结合练习题介绍了帧动画的常用场景  
```

##(十六)CSS常用动画库简介与JS动画

>\*\*`1:Animate.css`\*\*

-`cool, fun, and cross-browser animations`

-`抖动(shake)、闪烁(flash)、弹跳(bounce)、翻转(flip)、旋转(rotateIn/rotateOut)、淡入淡出(fadeIn/fadeOut)等多达60多种动画效果`

-`演示地址-`[https://daneden.github.io/animate.css/](https://daneden.github.io/animate.css/)

-`Github地址-`[https://github.com/daneden/animate.css](https://github.com/daneden/animate.css)

-`很值得学习内部代码的CSS动画库`

>\*\*`2:Animate.css使用`\*\*

-`引入Animate.css`

-`本地引入`

-`CDN引入-内容分发网络,快速找到最近的节点提供资源下载`

-`<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/3.7.0/animate.min.css">`

-`使用`

```css

<div class="animated bounce infinite"></div>

```

-`动态添加样式`

```css

document.getElementById("main").classList.add('bounce')

```

-`动态移除样式`

```css

document.getElementById("main").classList.remove('bounce')

```

-`重写样式`

```css

#main {

animate-duration:2s;

animate-delay:1s;

animate-iteration-count:2;

}

```

-`npm按需引入`

```css

npm install animate.css

```

```css

animate-config.json 文件

{

"scripts":{

"gulp":"./node\_modules/gulp/bin/gulp.js",

},

}

```

>\*\*`3:JS动画`\*\*

-`JS动画-使用Javascript控制元素的样式`

-`链式动画`

![image](./9.gif)

-`复杂图像+色彩变幻`

![image](./8.gif)

>\*\*`4:控制元素样式`\*\*

```css

function mouseOver(){

document.getElementById('image').src ="./1.jpg"

}

function mouseOut(){

document.getElementById('image').src ="./2.jpg"

}

```

>\*\*`5:返回顶部`\*\*

![image](./7.gif)

```css

var interval;

var distance = document.body.scrollHeight /1000;

var end = document.body.scrollHeight;

var current = document.body.scrollHeight - distance;

interval = setInterval(function(){

window.scrollTo(0, current);

current = current - distance;

if(window.scrollY <=0){

clearInterval(interval);

}

},10)

```

>\*\*`6: SetInterval 与 RequestAnimationFrame`\*\*

-`SetInterval (执行完一定记得clearInterval释放)`

-`定时器-制定时间间隔执行函数`

-`运行在JS主线程- setInterval与setTimeout时间间隔不准确-卡顿丢帧`

-`不释放始终运行-性能问题(浏览器窗口激活)`

-`RequestAnimationFrame`

-`高性能帧动画API`

-`把每一帧中的所有DOM操作集中起来,在一次重绘或回流中就完成`

-`重绘或回流的时间间隔紧紧跟随浏览器的刷新频率`

-`隐藏或不可见的元素不会进行重绘或回流(浏览器窗口激活)`

```css

var distance = document.body.scrollHeight /100;

var end = document.body.scrollHeight;

var current =0;

function bottom(){

window.scrollTo(0, current);

current = current + distance;

if(current <= end){

requestAnimationFrame(bottom);

} else {

cancelAnimationFrame(bottom);

}

}

requestAnimationFrame(bottom);

```

>\*\*`7: CSS动画与JS动画`\*\*

-`CSS动画`

-`浏览器对动画进行优化`

-`类requestAnimationFrame机制`

-`硬件加速`

-`代码相对简单`

-`帧速表现不好的低版本浏览器,可以自然降级`

-`运行过程控制较弱`

-`无法得知进度`

-`无法绑定函数回调`

-`代码冗长`

-`JS动画`

-`控制能力强`

-`动画效果丰富`

-`兼容性问题小`

-`主线程中运行-阻塞、丢帧`

-`代码复杂度高`

-`根据实际需求决定采用哪种方案`

>\*\*`8:课后练习`\*\*

-`使用Animate.css动画库实现下图效果`

![image](./1.gif)

-`实现下图小球的自由落体效果`

![image](./10.gif)

-`方式`

-`纯JS动画结合requestAnimationFrame`

-`JS动画结合贝塞尔曲线`

-`要求`

-`仅使用JS动画与CSS动画`

-`不得使用第三方库`

-`弹起时速度为反向速度的80%`

-`下落加速度大小参照重力加速度`

-`弹跳十次结束运动`

>\*\*`9:总结`\*\*

```

本节课介绍了CSS动画其中一个比较常用的动画库,简要介绍了Animate.css的使用,之后介绍了JS动画的使用方法,对比了CSS动画与JS各自的特点  
```

##(十七)REM布局原理

>\*\*`1:基本概念`\*\*

-`em -当前对象内文本的字体大小相对单位`

-`rem -字体大小相对单位`

-`物理像素`

-`逻辑像素`

-`设备像素比- dpr`

-`<meta>标签`

```css

<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0">

```

-`layout viewport -浏览器默认视口-可以显示PC界面`

-`clientWidth`

-`移动设备默认的viewport`

-`visual viewport -浏览器可视区域大小`

-`innerWidth`

-`ideal viewport -理想viewport`

-`iphone的ideal viewport宽度320px`

-`initial-scale - layout viewport相对于ideal viewport缩放比例`

-`取widthh和initial-scale中较大值`

-`initial-scale -0.5(左)-1(右)`

<img src="./scale1.jpg" width="45%"/>

&nbsp;&nbsp;&nbsp;&nbsp;

<img src="./scale2.jpg" width="45%"/>

>\*\*`2:适配设备屏幕(移动端)`\*\*

<img src="./taobao.jpg" width="89%" style="margin-left:40px"/>

-`最终目标-设计稿等比缩放在设备上`

-`动态处理Mate标签scale`

-`像素比-1- scale=1`

-`像素比-2- scale=2`

-`像素比-3- scale=3`

-`获取设备的像素比`

```css

var devicePixelRatio = window.devicePixelRatio;

var scale =1/ devicePixelRatio;

var metaElement = document.createElement('meta');

metaElement.setAttribute('name','viewport'); metaElement.setAttribute(

'content',

'initial-scale='+ scale +',

maximum-scale='+ scale +',

minimum-scale='+ scale +',

user-scalable=no'

);

document.documentElement.firstElementChild.appendChild(metaElement);

```

-`动态处理根节点font-size`

```css

var rem = document.documentElement.clientWidth /10

document.documentElement.style.fontSize = rem +'px'

```

>\*\*`3:还原设计图(移动端)`\*\*

-`缩放`

-`设计图尺寸640px \*750px`

-`兼容大多数设备的折中方案`

-`避免高清屏图片模糊`

-`1px边框问题-根据像素比缩放`

-`高清屏0.5缩放`

-`普通屏不缩放`

-`font-size`

-`将设计稿分成100份-预留 vw/vh 适配`

-`1rem =750px /10=75px`

-`等比例缩放`

>\*\*`4:还原设计图(PC端)`\*\*

-`缩放=1`

-`font-size`

-`设计图尺寸1920px \*1080px`

>\*\*`5:动态更改font-size`\*\*

```css

window.addEventListener('resize', resize)

```

>\*\*`6:vw / vh 与rem`\*\*

-`rem`

-`动态设置跟font-size`

-`用户无法设置更大字体`

![image](./rem.png)

-`vw / vh`

-`自动缩放`

-`不影响字体`

-`兼容性差`

![image](./vw.png)

>\*\*`7:课后练习`\*\*

>\*\*`8:总结`\*\*

```

本节课从移动端viewport切入,介绍了rem的基本概念,之后从viewport与font-size两个方面介绍了移动端自适应方案并扩展到PC端,最后介绍了css新特征vw与vh方案  
```

##(十八)淘宝Rem布局方案

>\*\*`1:lib-flexible`\*\*

```css

(function flexible (window, document){

var docEl = document.documentElement

var dpr = window.devicePixelRatio ||1  
function setBodyFontSize (){

if (document.body){

document.body.style.fontSize =(12\* dpr)+'px'

}

else {

document.addEventListener('DOMContentLoaded', setBodyFontSize)

}

}

setBodyFontSize();

function setRemUnit (){

var rem = docEl.clientWidth /10

docEl.style.fontSize = rem +'px'

}

setRemUnit()

window.addEventListener('resize', setRemUnit)

window.addEventListener('pageshow', function (e){

if (e.persisted){

setRemUnit()

}

})

if (dpr >=2){

var fakeBody = document.createElement('body')

var testElement = document.createElement('div')

testElement.style.border ='.5px solid transparent'

fakeBody.appendChild(testElement)

docEl.appendChild(fakeBody)

if (testElement.offsetHeight ===1){

docEl.classList.add('hairlines')

}

docEl.removeChild(fakeBody)

}

}(window, document))

```

-`手淘多年总结出的移动端适配方案`

-`动态改写<meta>标签`

-`并且动态改写font-size`

>\*\*`2:使用lib-flexible`\*\*

-`普通项目使用`

```css

<script src="http://g.tbcdn.cn/mtb/lib-flexible/0.3.4/??flexible\_css.js,flexible.js"></script>

```

-`工程化项目使用`

```css

npm install lib-flexible -save

入口文件引入

import 'lib-flexible/flexible.js'

```

>\*\*`3:手动px -> rem转换`\*\*

![image](./1.png)

-`设置基准-设置-扩展- px to rem`

-`Alt + z自动转换`

>\*\*`4:自动px -> rem转换`\*\*

- postcss-px2rem

>\*\*`5:字体不做rem处理`\*\*

-`避免出现奇数或小数px,导致文字模糊`

```css

.main {

font-size:12px;

}

[data-dpr="2"].main {

font-size:24px;

}

[data-dpr="3"].main {

font-size:36px;

}

```

>\*\*`6:课后练习`\*\*

-`使用淘宝Rem布局方案实现如下设计图`

![image](./2.png)

-`说明`

```css

请按照设计图高比例还原

```

-`使用淘宝vw/vh布局方案实现如下设计图`

![image](./2.png)

>\*\*`7:总结`\*\*

```

本节课介绍了淘宝移动端适配方案的基本原理和使用方法,介绍了实际开发中px到rem的转换方法,最后介绍了字体在Rem布局中的特殊处理  
```

##(十九)CSS Module

>\*\*`1:基本概念`\*\*

-`模块是可组合、分解和更换的单元`

```css

index.css

├─ header.css

│└─ reset.css

├─ content.css

│├─ left.css

││└─ nav.css

│└─ right.css

├─ fotter.css

└─...

```

-`所有的类名都只有局部作用域的CSS文件`

>\*\*`2:CSS Module解决了哪些问题`\*\*

-`全局污染`

-`!important`

-` inline`

-`复杂的选择器`

-`命名混乱`

-`避免样式冲突->选择器越来越复杂->命名越来越长`

-`层级结构不清晰`

-`代码难以复用`

-`很难从成千上万行代码中寻找`

-`代码压缩不彻底`

-`长的 class 名无能为力`

>\*\*`3:CSS Module原理`\*\*

-`AST语法树`

```css

{

nodes:[{

raws:{

before:'',

between:'',

semicolon: true,

after:'\r\n'

},

type:'rule',

source:{

start:{

line:1,

column:1

},

input:{

css:'.header {\r\n display: flex;\r\n lost-column:1/1;\r\n}\r\n',

hasBOM: false,

file:'../../../xiaokedao'

},

end:{

line:4,

column:1

}

},

selector:'.header',

lastEach:14,

indexes:{},

nodes:[]

}]

}

```

-`Vue scoped`

```css

<div data-v-2311c06a class="button-warp">

</div>

.button-warp[data-v-2311c06a]{

display: inline-block;

}

```

-`React Css module`

```css

<div class="wrapper\_\_\_2IPqp"></div>

```

>\*\*`4:局部作用域`\*\*

```css

module:{

rules:[

{

test:/\.css$/,

use: ExtractTextPlugin.extract({

fallback:"style-loader",

use:[

{

loader:'css-loader',

options:{

modules: true,

}

}

],

})

}

]

}

```

>\*\*`5:全局作用域`\*\*

```css

:global(.main){

display: flex;

}

```

>\*\*`6:哈希规则`\*\*

-`默认-[hash:base64]`

-`[path][name][local][hash:base64:n]`

-`[path][hash:base64:n]`

-`[name][hash:base64:n]`

-`[local][hash:base64:n]`

>\*\*`7:类组合`\*\*

```css

.header {

background:#8A469B;

}

.footer {

composes: header;

}

```

>\*\*`8:变量-搭配PostCSS`\*\*

```css

npm install postcss-modules-values

{

loader:"postcss-loader",

"options":{

plugins:[

require('postcss-modules-values'),

]

}

}

```

```css

@value color:#8A469B;

.header {

background: color;

}

.footer {

composes: header;

color:#FFF;

}

```

>\*\*`9:总结`\*\*

```

本节课介绍了淘宝移动端适配方案的基本原理和使用方法,介绍了实际开发中px到rem的转换方法,最后介绍了字体在Rem布局中的特殊处理  
```

##(二十)BEM命名规范

>\*\*`1:基本概念`\*\*

-`BEM 命名规范`

-`Block -更高级别的抽象或组件`

-`Element - Block 的后代`

-`Modifier -不同状态`

-`命名方法`

```css

.block\_\_element-modifier {

display: flex;

}

.block-modifier {

display: flex;

}

.block\_\_element {

display: flex;

}

<p class="header">

<p class="header\_\_body">

<button class="header\_\_button-primary"></button>

<button class="header\_\_button-default"></button>

</p>

</p>

```

-`清晰的描述`

-`清晰的结构`

>\*\*`2:Block`\*\*

-`负责描述功能-不包含状态`

```css

/\* correct \*/

.header {

}

/\* wrong \*/

.header-select {

}

```

-`不影响自身布局-不包含具体样式`

```css

/\* correct \*/

.header {

}

/\* wrong \*/

.header {

margin-top:50px;

}

```

-`不使用元素选择器和ID选择器`

```css

/\* correct \*/

.header {

}

/\* wrong \*/

.header a {

margin-top:50px;

}

```

>\*\*`3:Element`\*\*

-`双下划线分开`

```css

/\* correct \*/

.header\_\_body {

margin-top:50px;

}

/\* wrong \*/

.header .body {

margin-top:50px;

}

```

-`表示其目的-而不是其状态`

```css

.header\_\_body {

margin-top:50px;

}

.header\_\_logo {

margin-top:50px;

}

.header\_\_title {

margin-top:50px;

}

```

-`不能脱离Block独立使用`

```css

/\* correct \*/

<p class="header">

<p class="header\_\_body">

<button class="header\_\_button-primary"></button>

<button class="header\_\_button-default"></button>

</p>

</p>

/\* wrong \*/

<p>

<p class="header\_\_body">

<button class="header\_\_button-primary"></button>

<button class="header\_\_button-default"></button>

</p>

</p>

```

>\*\*`4:Modifier`\*\*

-`双横杠分开`

```css

.header\_\_button-default {

background: none;

}

```

-`Boolean`

```css

.header\_\_button-select {

background: blue;

}

```

-`枚举`

```css

.header\_\_button-primary {

background:#329FD9;

}

```

-`不能单独使用`

```css

/\* correct \*/

<p class="header">

<p class="header\_\_body">

<button class="header\_\_button-primary"></button>

<button class="header\_\_button-default"></button>

</p>

</p>

/\* wrong \*/

<p>

<p>

<button class="header\_\_button-primary"></button>

<button class="header\_\_button-default"></button>

</p>

</p>

```

>\*\*`5:Less中使用`\*\*

```css

@classname: header;

.@{classname}{

.@{classname}\_\_body {

padding:20px;

}

.@{classname}\_\_button {

.@{classname}\_\_button-primary {

background:#329FD9;

}

.@{classname}\_\_button-default {

background: none;

}

}

}

```

>\*\*`6:Sass中使用`\*\*

```css

.header {

&\_\_body {

padding:20px;

}

&\_\_button {

&-primary {

background:#329FD9;

}

&-default {

background: none;

}

}

}

```

>\*\*`7:课后练习`\*\*

-`谈一谈你对BEM命名规范的理解,命名是否过于繁琐?应如何解决`

-`使用BEM命名规范重新定义例子中的样式名`

![image](./6.gif)

-`要求-严格遵循BEM规范`

>\*\*`8:总结`\*\*

```

本节课从BEM命名的基本概念入手,介绍了Block、Element与Modifier常用的命名规范,最后介绍了BEM命名规范在预处理器语言中的使用  
```

##(二十一)Web存储技术(上)

>\*\*`1:背景介绍`\*\*

-`Cookie -网站身份证`

-`网站为了辨别用户身份、进行 session 跟踪而储存在用户本地终端上的数据`

-`存储电脑硬盘上的一个txt类型的小文件`

-`安全性问题-双刃剑`

-`...`

-`Cookie局限`

-`主流浏览器最大支持4096字节`

-`主流浏览器限制每个网站的Cookie个数-20`

-`Cookie默认会跟随所有Http请求发送-浪费网络资源`

-`部分浏览器限制Cookie总数-300`

-`...`

-`Web Storage应运而出`

-`存储大量数据`

-`支持复杂的本地数据库`

-`SessionStorage`

-`LocalStorage`

-`WebSQL`

>\*\*`2:Cookie`\*\*

![image](./cookie.jpg)

-`HTML4-一般不需要考虑兼容`

-`身份证-服务器可以针对不同用户,做出不同的响应`

-`浏览器储存在用户的机器上的纯文本,没有可执行代码`

-`浏览器默认携带当前访问网站的Cookie`

-`Cookie类型`

-`会话Cookie -退出浏览器即删除`

-`持久Cookie`

-`特定的过期时间(Expires)`

-`有效期(Max-Age)`

-`Cookie域- Domain`

-`控制Cookie使用权限`

-`Cookie路径- Path`

```css

www.baidu.com id="123456" domain="www.baidu.com"

www.baidu.com/user id="123456" user="eric" domain="www.baidu.com" path="/user/"

www.baidu.com/search id="123456";

www.baidu.com/user/search id="123456" user="eric";

```

-`Cookie - secure`

-`只有https情况下使用`

-`Cookie - HttpOnly`

-`不允许Javascript操作Cookie`

-`安全性`

-`服务器无法分辨用户和攻击者(攻击者窃取网站Cookie)`

-`跨站点脚本XSS`

-`跨站请求伪造CSRF`

>\*\*`3:SessionStorage`\*\*

![image](./sessionStorageValue.jpg)

-`基本概念`

-`Key - Value键值对`

-`HTML5新增的会话存储对象`

![image](./sessionStorage.jpg)

-`临时保存同一窗口(标签页)的数据`

-`关闭窗口(标签页)删除数据`

-`特点`

-`同源策略-同一协议、同一主机名和同一端口统一Tab`

-`单标签页限制`

-`只在本地存储-不会跟随HTTP请求发送到服务器`

-`存储方式采用key、value键值对-字符串类型`

-`存储上限限制-5MB`

-`超出覆盖`

-`属性`

-`sessionStorage.length -键值对数量`

-`sessionStorage.key(int index)-> null`

-`sessionStorage.getItem(string key)-> null`

-`sessionStorage[string key]`

-`sessionStorage.setItem(string key, string value)`

-`sessionStorage.removeItem(string key)`

-`sessionStorage.clear()`

-`Json对象`

-`JSON.stringify()`

-`JSON.parse()`

>\*\*`4:LocalStorage`\*\*

![image](./localStorage.png)

-`Key - Value键值对`

-`HTML5新增的存储对象`

![image](./localstoragesupport.jpg)

-`特点`

-`同源策略- localStorage`

-`无标签页限制`

-`只在本地存储-不会跟随HTTP请求发送到服务器`

-`存储方式采用key、value键值对-字符串类型`

-`存储上限限制-5MB`

-`无痕模式下不可读取`

-`永久性存储`

-`超出报错`

-`属性`

-`localStorage.length -键值对数量`

-`localStorage.key(int index)-> null`

-`localStorage.getItem(string key)-> null`

-`localStorage[string key]`

-`localStorage.setItem(string key, string value)`

-`localStorage.removeItem(string key)`

-`localStorage.clear()`

-`注意事项`

-`Web中不可靠`

-`IOS浏览器中不可重复setItem,先removeItem`

>\*\*`5:监听storage变化`\*\*

-`同源`

![image](./origin.png)

-`禁止不同源执行任何脚本`

-`监听同源网页-同一网页无效`

```css

window.addEventListener("storage", function (event){

console.log(event.key);

console.log(event.oldValue);

console.log(event.newValue);

console.log(event.url);

console.log(event.storageArea);

});

```

>\*\*`6:课后练习`\*\*

-`简要概括浏览器存储技术的发展历程`

-`使用LocalStorage实现下列数据的增删改查`

```css

key: Date.parse(new Date)

value:{

finished: false,

content:'完成侠客岛No.22小节学习',

}

```

-`要求`

-`包含一条数据`

-`包含增删改查四个Button`

-`新增时key为时间戳`

-`删除时根据key删除`

-`更新时根据key更新`

-`查询时根据key查询`

-`使用SessionStorage实现下列数据的增删改查`

```css

key: Date.parse(new Date)

value:{

finished: false,

content:'完成侠客岛No.22小节学习',

}

```

-`要求`

-`包含一条数据`

-`包含增删改查四个Button`

-`新增时key为时间戳`

-`删除时根据key删除`

-`更新时根据key更新`

-`查询时根据key查询`

##(二十二)Web存储技术(中)

>\*\*`1:IndexedDB背景`\*\*

-`Storage不适合储存大量数据`

-`Storage不提供搜索功能`

-`Storage不能建立索引`

-`IndexedDB -250MB 以上`

>\*\*`2:IndexedDB基本概念`\*\*

-`非关系型数据库- NoSQL`

-`特点`

-`键值对储存-允许所有类型-主键重复报错`

-`异步操作-不阻塞浏览器线程`

-`支持事务-一系列操作有一步失败,数据库回滚到事务发生之前的状态`

-`同源限制`

-`支持二进制储存`

-`IDBDatabase -数据库`

-`IDBObjectStore -对象仓库`

-`IDBIndex -索引`

-`IDBTransaction -事务`

-`IDBRequest -操作请求`

-`IDBCursor -指针`

-`IDBKeyRange -主键集合`

>\*\*`3:IndexedDB浏览器兼容`\*\*

![image](./indexedDB.jpg)

>\*\*`4:IDBDatabase`\*\*

-`数据的容器`

-`每个不同源可以新建多个数据库`

-`版本`

-`同一个时刻只能有一个版本存在`

-`修改数据库结构只能通过升级数据库版本`

-`打开数据库`

```css

/\* databaseName不存在则创建\*/

/\* version为整数,新建时为1\*/

let database;

let userStore;

const request = window.indexedDB.open(databaseName, version);

/\*成功打开数据库\*/

request.onsuccess = event =>{

database = request.result;

}

/\*打开数据库失败\*/

request.onerror = error =>{

console.log(error);

}

/\*版本号大于当前数据库版本\*/

request.onupgradeneeded = event =>{

database = event.target.result;

}

```

>\*\*`5:IDBObjectStore`\*\*

-`数据库表`

-`创建表-最好在upgradeneeded中执行`

-`指定主键`

-`autoIncrement: true`

-`keyPath:'id'`

```css

const createStore =()=>{

if(!db.objectStoreNames.contains('user')){

userStore = database.createObjectStore('user',{ keyPath:'id'});

}

}

```

-`指定索引`

```css

const createStore =()=>{

if(!database.objectStoreNames.contains('user')){

userStore = database.createObjectStore('user',{ keyPath:'id'});

userStore.createIndex('name','name',{ unique: true });

}

}

```

>\*\*`6:IDBTransaction`\*\*

-`数据增啥改查`

-`complete`

-`error`

-`abort`

-`属性`

-`IDBTransaction.db`

-`IDBTransaction.mode`

-`readonly`

-`readwrite`

-`IDBTransaction.objectStoreNames`

-`IDBTransaction.error`

-`新增数据- add`

-`已创建主键-新增必须包含主键`

-`已创建主键-主键不可重复`

```css

const add =()=>{

/\*创建事务\*/

/\*使用某个数据库\*/

/\* add新增\*/

transactionRequest = database.transaction(['user'],'readwrite')

.objectStore('user')

.add({ id:100, name:'Eric', age:28, email:'Ericlee00@163.com'});

/\*成功\*/

transactionRequest.onsuccess = event =>{

console.log('数据写入成功', event);

};

/\*失败\*/

transactionRequest.onerror = error =>{

console.log('数据写入失败', error);

}

}

```

-`读取数据- get`

```css

const read =()=>{

/\*创建事务\*/

transaction = database.transaction(['user']);

/\*选择数据库表\*/

table = transaction.objectStore('user');

/\*读取数据\*/

transactionRequest = table.get(2);

/\*成功\*/

transactionRequest.onerror = event =>{

console.log('数据读取失败', event);

};

/\*失败\*/

transactionRequest.onsuccess = event =>{

if (transactionRequest.result){

console.log('数据读取成功', transactionRequest.result);

} else {

console.log('未读取到数据');

}

};

}

```

-`更新数据- put`

-`更新不存在的数据会新建`

```css

const update =()=>{

transactionRequest = database.transaction(['user'],'readwrite')

.objectStore('user')

.put({ id: count, name:'David', age:35, email:'David@xiakedao.com'});

transactionRequest.onsuccess = function (event){

console.log('更新数据成功', event);

};

transactionRequest.onerror = error =>{

console.log('更新数据失败', error);

}

}

```

-`删除数据- delete`

```css

const delete =()=>{

transactionRequest = database.transaction(['user'],'readwrite')

.objectStore('user')

.delete(2);

transactionRequest.onsuccess = function (event){

console.log('删除数据成功', event);

};

transactionRequest.onerror = error =>{

console.log('删除数据失败', error);

}

}

```

-`清空数据- clear()`

>\*\*`7:IDBCursor`\*\*

-`指针`

-`遍历数据`

```css

const readAll =()=>{

table = database.transaction('user').objectStore('user');

table.openCursor().onsuccess =()=>{

let cursor = event.target.result;

if (cursor){

console.log('数据遍历', cursor);

cursor.continue();

} else {

console.log('数据遍历完成');

}

};

}

```

##(二十三)Web存储技术(下)

>\*\*`8:关闭IndexedDB数据库连接`\*\*

```css

const closeDataBase =()=>{

database.close();

}

```

>\*\*`9:删除IndexedDB数据库`\*\*

-`先关闭数据库连接`

```css

const deleteDataBase =()=>{

indexedDB.deleteDatabase('first\_database');

}

```

>\*\*`10:WebSQL基本概念`\*\*

-`非 HTML5规范-独立的规范`

-`SQL 语句操作客户端数据库`

-`openDatabase -使用现有数据库-新建数据库`

-`transaction -事务`

-`executeSql -执行SQL语句`

>\*\*`11:openDatabase`\*\*

-`数据库名称`

-`版本号`

-`描述文本`

-`数据库大小`

-`创建回调-只在第一次创建调用`

```css

const database = openDatabase('my\_database','1.0','first',2\*1024\*1024, function(){

});

```

>\*\*`12:transaction`\*\*

-`创建表`

```javascript

const createTable =()=>{

database.transaction(function (content){

content.executeSql('CREATE TABLE IF NOT EXISTS USER (id unique, name)');

});

}

```

-`添加数据`

```javascript

const addData =()=>{

database.transaction(function (content){

content.executeSql('INSERT INTO USER (id, name) VALUES (1,"Eric")');

});

}

```

-`查询数据`

```javascript

const searchData =()=>{

database.transaction(function (content){

content.executeSql('SELECT \* FROM USER');

});

}

```

-`更新数据`

```javascript

const updateData =()=>{

database.transaction(function (content){

content.executeSql('UPDATE USER SET name=\'David\' WHERE id=1');

});

}

```

-`删除数据`

```javascript

const deleteData =()=>{

database.transaction(function (content){

content.executeSql('DELETE FROM USER WHERE id =1');

});

}

```

-`删除数据库表`

```javascript

const deleteDataBase =()=>{

database.transaction(function (content){

content.executeSql('DROP TABLE USER');

});

}

```

>\*\*`13:课后练习`\*\*

-`使用IndexedDB创建下列数据库表并添加增删改查功能`

-`要求`

-`创建数据库表- mission\_finished`

-`创建数据库表- mission\_unfinished`

-`包含增删改查`

-`表结构`

```css

id: Date.parse(new Date())

content:'完成侠客岛No.23小节学习',

```

>\*\*`14:总结`\*\*

```css

本节课从浏览器存储技术的历史讲起,介绍了cookie、sessionStorage、localStorage、IndexedDb、WebSQL的技术背景和技术特征,最后结合实例了四种Web存储技术的使用方法  
```

##(二十四)初识WebWorker

>\*\*`1:基本概念`\*\*

-`JS单线程-同一时间只能做一件事`

-`JS宿主(浏览器 Node)支持多线程`

-`浏览器常驻线程`

-`UI线程-回流和重绘-与Javascript线程互斥`

-`JavaScript线程-单线程运行JavaScript`

-`GUI线程-交互线程`

-`...`

-`Network线程`

-`File线程`

-`定时器线程`

-`基于事件驱动`

-`Event Loop`

![image](./eventloop.png)

-`异步-将耗时长的任务放置到Event Queue尾部`

-`WebWorker`

-`JavaScript多线程模型-基于宿主`

-`子线程`

-`完全受主线程控制`

-`不能操作DOM`

>\*\*`2:浏览器兼容`\*\*

![image](./browser.png)

>\*\*`3:应用场景`\*\*

-`处理影响浏览器UI线程的JavaScript运算`

>\*\*`4:WebWorker特点`\*\*

-`一旦新建会始终运行,不会被主线程打断`

-`同源限制`

-`不能操作和访问DOM(window、document)`

-`不能使用包含交互的全局方法(alert、confirm)-可以使用XMLHttpRequest、setTimeout、setInterval`

-`不能读取本地文件`

-`dedicated web worker(专用线程)`

-`shared web worker(共享线程)`

>\*\*`5:基本用法`\*\*

-`创建WebWorker -资源只能来自网络`

```css

const webWorker = new Worker('main.js');

```

```css

let result =0;

const fibonacci =(n)=>{

if (n <=1) return 1;

return fibonacci(n -1)+ fibonacci(n -2);

}

result = fibonacci(10);

console.log('result', result);

```

-`向WebWorker发送消息(数据)`

```css

webWorker.postMessage({ number:10});

```

-`WebWorker接收消息`

```css

this.addEventListener('message', event =>{

console.log('received main worker data', event);

}, false);

```

-`WebWorker发送消息(数据)`

```css

this.postMessage(returnValue);

```

-`接收WebWorker消息`

```css

webWorker.addEventListener('message', event =>{

console.log('received webworker data', event.data);

}, false);

```

-`关闭WebWorker`

```css

webWorker.terminate();

关闭 WebWorker

```

```css

this.close();

不再接收新的 Macrotask

```

>\*\*`6:WebWorker调用脚本`\*\*

```css

importScripts('./one.js','./two.js');

```

>`7:WebWOrker错误监听`

```css

webWorker.addEventListener('error', error =>{

console.error(error.filename, error.lineno, error.message);

});

```

>\*\*`8:课后练习`\*\*

-`浏览器UI线程和JavaScript为什么是互斥的`

-`使用WebWorker实现下图效果`

![image](./4.gif)

-`要求`

```css

运算在WebWorker中执行

数字递增即可

输入框中可以正常输入

```

-`不使用WebWorker实现下图效果,比较输入框输入是界面效果`

![image](./4.gif)

>\*\*`9:总结`\*\*

```css

本节课从JavaScript单线程模型入手,介绍了WebWorker的产生背景,介绍了WebWorker从创建到销毁的整个生命周期  
```

##(二十五)WebWorker进阶

>\*\*`1:WebWorker与主线程通信`\*\*

-`通信内容`

-`字符串`

-`对象`

-`二进制`

-`...`

-`任何Javascript可以表示存储的内容`

-`通信是拷贝关系`

-`二进制数据转移-转移之后当前线程无法操作`

```css

webWorker.postMessage(arrayBuffer,[arrayBuffer]);

```

>\*\*`2:执行同页面的WebWorker`\*\*

-`未知的Script类型-代码不会执行`

```css

<script id="samePageWorker" type="unknow">

addEventListener('message',()=>{

postMessage('send message!');

}, false);

</script>

```

-`二进制读取并开启新的WebWorker`

```css

const blob = new Blob([document.getElementById('samePageWorker').text]);

const url = URL.createObjectURL(blob);

/\*获取保存在内存中的文件地址\*/

const webWorker = new Worker(url);

webWorker.postMessage(123);

```

>\*\*`3:WebWorker嵌套`\*\*

```css

const webWorker = new Worker('./main.js');

window.webWorker = webWorker;

webWorker.postMessage('Hello');

```

```css

this.addEventListener('message', event =>{

console.log('received index.html data', event.data);

}, false);

const webWorker = new Worker('./index.js');

webWorker.postMessage('Word');

```

```css

this.addEventListener('message', event =>{

console.log('received main data', event.data);

}, false);

```

>\*\*`4:Shared WebWorker -跨页面通信`\*\*

-`不能使用console`

```css

const shareWorker = new SharedWorker('./main.js');

window.shareWorker = shareWorker;

shareWorker.port.start();

shareWorker.port.postMessage(1);

shareWorker.port.onmessage = event =>{

console.log(event.data);

}

```

```css

const shareWorker = new SharedWorker('./main.js');

shareWorker.port.postMessage(1);

shareWorker.port.onmessage = event =>{

console.log(event.data);

};

shareWorker.port.start();

```

```css

let temp =1;

this.onconnect = event =>{

const port = event.ports[0];

port.start();

port.onmessage = e =>{

temp = temp + e.data;

port.postMessage(temp);

}

}

```

>\*\*`5:课后练习`\*\*

-使用WebWorker实现单词输入匹配功能(包含10000个英语单词)

![image](./word.png)

>\*\*`6:总结`\*\*

```css

本节课介绍WebWorker特殊应用场景,扩展了WebWorker的能力,为跨页面通信提供了另外一种思路  
```

##(二十六)SSE与WebSocket

>\*\*`1:基本概念`\*\*

-`广义短连接`

```css

建立连接——数据传输——关闭连接

建立连接——数据传输——关闭连接

```

-`广义长连接`

```css

建立连接

数据传输

...

...

关闭连接

```

-`单个 TCP 连接上全双工通讯协议-允许服务端主动向客户端推送数据`

-`SSE(Server-Sent Events)- WebSocket 轻量代替方案`

>\*\*`2:浏览器兼容`\*\*

![image](./browser.png)

>\*\*`3:SSE与WebSocket`\*\*

-`WebSocke`

-`全双工通道,可以双向通信`

-`全新协议,需要服务器端支持`

-`不会主动断开`

-`SSE`

-`单向通道,服务端向客户端推送`

-`部署在HTTP协议之上,都支持`

-`支持自定义数据类型`

-`轻量级协议`

-`静置15秒后自动断开(草案标准)`

>\*\*`4:基本用法`\*\*

-`建立连接`

```css

const SSEClient = new EventSource('http://localhost:3000/sse');

SSEClient.addEventListener('open',()=>{

console.log('连接成功!');

}, false);

```

>\*\*`5:数据格式`\*\*

-`纯文本的简单协议`

-`包含回车符和换行符的空行来分隔`

```css

data: Hello

data: Word

id:1

event: eventName

data: Hello

data: Word

id:2

```

-`数据类型`

-`空白-:开头-作为注释忽略-维持服务器和客户端的长连接`

-`data - data开头`

-`id -事件的标识符-重连标识(Last-Event-ID)`

-`event -消息事件类型-没有event字段默认触发messages事件-自定义事件不会触发message`

-`retry -重连等待时间(ms)-只接受整数`

```css

SSEClient.addEventListener('connecttime', event =>{

console.log('connecttime data', event.data);

}, false);

SSEClient.addEventListener('message', event =>{

console.log('message data', event.data);

}, false);

```

>\*\*`6:错误处理`\*\*

```css

SSEClient.onerror = error => console.error('连接出错!', error);

```

>\*\*`7:关闭连接`\*\*

```css

SSEClient.close();

```

>\*\*`8:只读属性`\*\*

-`readyState`

-`0 CONNECTING `

-`1 OPEN`

-`2 CLOSED`

```css

SSEClient.readyState

```

-`url`

```css

SSEClient.url

```

>\*\*`9:注意事项`\*\*

-`数据完整性- Last-Event-ID`

-`减少网络开销-必要时断开连接`

>\*\*`10:课后练习`\*\*

-`实现一个实时聊天系统`

![image](./9.gif)

-`要求`

```css

样式参考上图但不做限制,鼓励添加动画实现更好的效果

cmd进入server运行node bin/www

SSE调用地址- http://localhost:3000/sse

```

>\*\*`11:总结`\*\*

```css

本节课从广义上的网页长连接入手,介绍了WebSocket与SSE的主要特点,重点介绍了SSE的使用方法,最后结合课后练习展示了SSE与WebSocket最典型的应用场景  
```

##(二十七)Canvas简介与基础用法

>\*\*`1:基本概念`\*\*

-`<canvas>- HTML5新特征`

-`HTML代码配合高度和宽度属性而定义出的可绘制区域`

-`使用脚本(JavaScript)绘制图像的HTML元素`

-`制作照片-制作动画-实时视频处理`

-`HTML不存在一套二维绘图API -苹果公司提出Canvas前身`

>\*\*`2:浏览器兼容`\*\*

![image](./browser.png)

>\*\*`3:<canvas>元素`\*\*

-`width -宽度-默认300px`

-`height -高度-默认150px`

```css

<canvas width="300px" height="150px" id="canvas">

您的浏览器不支持Canvas

<img src="./backup.jpg" alt="">

</canvas>

```

>\*\*`4:坐标系统`\*\*

![image](./axios.png)

>\*\*`5:Rending Context(渲染上下文-画笔)`\*\*

```css

const canvas = document.getElementById('canvas');

/\*获得2d 上下文对象\*/

const ctx = canvas.getContext('2d');

```

>\*\*`6:原生绘制API`\*\*

-`fillRect(x, y, width, height)-填充矩形`

-`strokeRect(x, y, width, height)-矩形边框`

-`clearRect(x, y, widh, height)-清空`

```css

ctx.fillRect(0,0,50,50);

ctx.strokeRect(50,50,100,100);

ctx.clearRect(10,10,30,30);

```

>\*\*`7:绘制线段- Path`\*\*

```css

ctx.beginPath();

ctx.moveTo(0,0);

ctx.lineTo(300,150);

ctx.closePath();

ctx.stroke();

```

>\*\*`8:绘制三角形- Path`\*\*

```css

ctx.beginPath();

ctx.moveTo(0,0);

ctx.lineTo(300,0);

ctx.lineTo(150,150);

ctx.closePath();

ctx.fill();

```

>\*\*`9:绘制圆- Path`\*\*

-`arc(x, y, radius, startAngle, endAngle, anticlockwise)`

```css

ctx.beginPath();

ctx.moveTo(0,0);

ctx.arc(150,75,50,0, Math.PI /2, false);

ctx.stroke();

```

-`arcTo(x1, y1, x2, y2, radius)`

```css

ctx.beginPath();

ctx.moveTo(0,0);

ctx.arcTo(100,100,400,100,400);

ctx.stroke();

```

>\*\*`10:贝塞尔曲线- Path`\*\*

-`cubic-bezier(n1, n2, n3, n4)`

-`P0(0,0)`

-`P1(1,1)`

-`P2(n1, n2)`

-`P3(n3, n4)`

```css

ctx.beginPath();

ctx.bezierCurveTo(0,123,100,-9,100,100);

ctx.stroke();

```

>\*\*`11:课后练习`\*\*

-`使用Canvas绘制下图笑脸`

![image](./smile.png)

-`使用Canvas随机生成气泡`

![image](./buble.png)

```css

大小随机

位置随机

```

>\*\*`12:总结`\*\*

```css

本节课介绍了Canvas元素基本概念和坐标系统,从线段、三角形、圆和贝塞尔曲线四个示例介绍了Canvas元素的基本绘制方法  
```

##(二十八)丰富Canvas应用

>\*\*`1:添加样式`\*\*

-`fillStyle -填充`

-`颜色- CSS 颜色值`

```css

ctx.fillStyle ='#329FD9';

ctx.fillRect(0,0,400,100);

```

-`渐变对象- Gradient`

-`createLinearGradient`

```css

const linearGradient = ctx.createLinearGradient(0,300,400,0);

linearGradient.addColorStop(0,"#8A469B");

linearGradient.addColorStop(0.5,"#FFFFFF");

linearGradient.addColorStop(1,"#EA7F26");

ctx.fillStyle = linearGradient;

ctx.fillRect(0,100,400,100);

```

-`createRadialGradient`

![image](./radius.png)

```css

/\*径向渐变-同心圆\*/

const radialGradient1= ctx.createRadialGradient(200,250,10,200,250,60);

radialGradient1.addColorStop(0,"#8A469B");

radialGradient1.addColorStop(1,"#EA7F26");

ctx.fillStyle = radialGradient1;

ctx.fillRect(0,200,400,100);

/\*径向渐变-非同心圆\*/

const radialGradient2= ctx.createRadialGradient(60,350,10,350,350,60);

radialGradient2.addColorStop(0,"#8A469B");

radialGradient2.addColorStop(1,"#EA7F26");

ctx.fillStyle = radialGradient2;

ctx.fillRect(0,300,400,100);

```

-`图案(视频)对象- Pattern`

-`ctx.createPattern(image,"repeat | repeat-x | repeat-y | no-repeat");`

```css

const img = new Image();

img.src ="./backup.png";

img.onload =()=>{

const pattern = ctx.createPattern(img,'repeat');

ctx.fillStyle = pattern;

ctx.fillRect(0,400,400,100);

}

```

-`设置之后成为默认属性`

-`strokeStyle -轮廓`

-`颜色`

-`渐变对象`

-`图案(视频)对象`

-`设置之后成为默认属性`

-`transparency -透明度`

```css

ctx.globalAlpha =0.1;

```

-`line style -线条样式`

-`lineWidth -线条宽度`

```css

ctx.beginPath();

ctx.lineWidth =10;

ctx.moveTo(10,10);

ctx.lineTo(50,50);

ctx.closePath();

ctx.stroke();

```

-`lineCap -线条端样式`

```css

ctx.beginPath();

ctx.moveTo(20,150);

ctx.lineTo(20,250);

ctx.lineWidth =20;

ctx.lineCap ='round';

ctx.stroke();

```

-`butt -平直的边缘-默认`

-`round -圆形线帽-宽度一半`

-`square -正方形线帽-宽度一半`

-`lineJoin -线条连接处端样式-同一path`

-`miter -尖角-默认`

-`bevel -斜角`

-`round -圆角`

-`miterLimit -斜接长度(不做介绍)`

-`setLineDash -[实线长度,间隙长度]-虚线样式`

-`lineDashOffset -起始偏移量`

```css

ctx.lineWidth =1;

ctx.setLineDash([45,5]);

ctx.lineDashOffset =-5;

ctx.strokeRect(10,450,500,100);

```

>\*\*`2:绘制文本`\*\*

-`font`

-`font-style`

-`font-variant`

-`font-weight`

-`font-size`

-`font-family`

-`...`

-`textAlign`

-`start -默认`

-`end`

-`left`

-`right`

-`center`

-`textBaseline`

-`top`

-`hanging`

-`middle`

-`alphabetic -默认`

-`ideographic`

-`bottom`

-`direction`

-`ltr`

-`rtl`

-`inherit -默认`

-`fillText`

-`fillText(text, x, y [, maxWidth])`

-`strokeText`

-`strokeText(text, x, y [, maxWidth])`

>\*\*`3:绘制图片`\*\*

-`基本用法`

```css

const img = new Image();

img.src ="./backup.png";

img.onload =()=>{

const pattern = ctx.createPattern(img,'repeat');

ctx.fillStyle = pattern;

ctx.fillRect(0,400,400,100);

}

```

-`绘制img元素图片`

```css

const img = document.getElementById('image');

```

-`图片缩放`

```css

const img2= new Image();

img2.src ="./backup.png";

img2.onload =()=>{

const pattern = ctx.createPattern(img2,'repeat');

ctx.fillStyle = pattern;

ctx.drawImage(img2,0,200,100,100)

}

```

-`图片切片`

```css

const img3= new Image();

img3.src ="./backup.jpg";

img3.onload =()=>{

const pattern = ctx.createPattern(img3,'repeat');

ctx.fillStyle = pattern;

ctx.drawImage(img3,0,0,640,480);

ctx.drawImage(img3,320,240,640,480,0,0,640,480);

}

```

>\*\*`4:状态保存`\*\*

-`save -状态存储在栈中`

-`restore -栈中弹出恢复状态`

-`toDataURL(type, encoderOptions)-保存为图片`

-`type -默认image/png`

-`encoderOptions图片质量- image/jpeg/webp -0-1`

>\*\*`5:课后练习`\*\*

-`了解RGB、RGBA和Opacity的关系与区别`

-`使用Canvas绘制下图形状`

![image](./line.png)

-`宽度300px`

-`高度150px`

-`40% black -40% green -20% yellow`

-`使用Canvas绘制下图形状`

![image](./split.png)

-`宽度600px`

-`高度600px`

-`Math.floor(255-25.5\* i)`

-`将第27节气泡添加径向渐变(颜色不做限制)`

![image](./bubble.png)

>\*\*`6:总结`\*\*

```css

本节课介绍了Canvas填充、文本以及图片等多媒体样式应用,结合状态保存与恢复介绍了Canvas如何绘制多彩的内容  
```

##(二十九)Canvas变换

>\*\*`1:Translate -平移`\*\*

-`translate(x, y)`

-`x -左右偏移量`

-`y -上下偏移量`

>\*\*`2:Rotate -旋转`\*\*

-`rotate(angle)-顺时针`

>\*\*`3:Scale -缩放`\*\*

-`scale(x, y)-(x,y >0)`

-`本质-像素比`

>\*\*`4:Transform -矩阵变换`\*\*

-`transform(a, b, c, d, e, f)`

-`水平缩放`

-`水平倾斜`

-`垂直倾斜`

-`垂直缩放`

-`水平移动`

-`垂直移动`

-`对应矩阵`

```

a c e

b d f

001

```

-`x'= a\*x + c\*y +1\*e`

-`y'= b\*x + d\*y +1\*f`

-`实现Translate`

-`transform(1,0,0,1,100,100)`

-`translate(100,100);`

-`实现Scale`

-`transform(0.5,0,0,0.5,0,0)`

-`scale(0.5,0.5)`

-`实现Rotate`

-`transform(cosθ, sinθ,-sinθ, cosθ,0,0)`

-`rotate(θ)`

>\*\*`5:绘制阴影`\*\*

-`shadowOffsetX -阴影横向位移量`

-`shadowOffsetY -阴影纵向位移量`

-`shadowColor -阴影颜色`

-`shadowBlur -阴影模糊范围`

```css

/\* Shadow \*/

ctx.shadowOffsetX =5;

ctx.shadowOffsetY =5;

ctx.shadowColor ='#595959';

ctx.shadowBlur =2;

/\* Text \*/

ctx.font ="100px sans-serif";

ctx.fillText("侠课岛",10,400);

```

>\*\*`6:画布合成`\*\*

-`globalCompositeOperation -混合属性`

-`source-over -目标图像上显示源图像-默认属性`

-`source-atop -源图像位于目标图像之外部分不可见`

-`source-in -显示目标图像之内的源图像部分,目标图像透明`

-`source-out -显示目标图像之外的源图像部分,目标图像透明`

-`destination-over -源图像上显示目标图像`

-`destination-atop -源图像顶部显示目标图像。目标图像位于源图像之外的部分不可见`

-`destination-in -源图像中显示目标图像。只显示源图像之内的目标图像部分,源图像透明`

-`destination-out -源图像之外显示目标图像。只显示源图像之外的目标图像部分,源图像是透明的`

-`darken -保证重叠部分最暗(16进制数值最大)的像素`

-`lighter -保证重叠部分最亮(16进制数值最小)的像素`

-`copy -只保留目标图像`

-`xor -源图像与目标图像重叠部分透明`

>\*\*`7:路径裁剪`\*\*

-`clip -只显示裁剪路径内的区域`

```css

const polygon =(poly, context)=>{

context.beginPath();

context.moveTo(poly[0], poly[1]);

for (var i =2; i <= poly.length -2; i +=2){

context.lineTo(poly[i], poly[i +1]);

}

context.closePath();

context.stroke();

}

const canvas = document.getElementById('canvas');

/\*获得2d 上下文对象\*/

const ctx = canvas.getContext('2d');

const pointList =[300,0,366,210,588,210,408,342,474,546,300,420,126,546,192,342,12,210,234,210];

polygon(pointList, ctx);

ctx.clip();

const img = new Image();

img.src ="./logo.png";

img.onload =()=>{

const pattern = ctx.createPattern(img,'repeat');

ctx.fillStyle = pattern;

ctx.drawImage(img,0,0,610,610);

}

```

>\*\*`8:课后练习`\*\*

-`使用Canvas Cpip绘制下图八边形`

![image](./eight.png)

-`为气泡添加阴影效果并改变径向渐变中心`

![image](./shade.png)

>\*\*`9:总结`\*\*

```css

本节课介绍了Canvas变换相关内容,从Tramsform、Shadow和Clip三个方面拓展了Canvas的绘图能力  
```

##(三十)Canvas动画

>\*\*`1:Canvas动画原理`\*\*

-`快速切换的静态画面`

-`Frames Per Second`

-`人眼-24FPS`

-`游戏(屏幕)-60FPS`

>\*\*`2:基本步骤`\*\*

-`绘制-清空-绘制-清空-绘制...`

>\*\*`3:控制函数`\*\*

-`setInterval`

-`setTimeout`

-`requestAnimationFrame`

>\*\*`4:线性运动`\*\*

```css

ctx.clearRect(0,0,600,600);

ctx.beginPath();

ctx.moveTo(distance,300);

ctx.arc(distance,300,50,0,2\* Math.PI, false);

ctx.fill();

distance = distance + speed;

```

>\*\*`5:变速运动`\*\*

```css

ctx.clearRect(0,0,600,600);

ctx.beginPath();

ctx.moveTo(distance,300);

ctx.arc(distance,300,50,0,2\* Math.PI, false);

ctx.fill();

distance =9.8\*(Math.pow(count,2))/200;

```

>\*\*`6:函数运动-正弦`\*\*

```css

axis =300+ Math.sin(angle)\* range;

```

>\*\*`7:环形运动`\*\*

```css

const draw =(axisX)=>{

/\*清空画布(或部分清空)\*/

ctx.clearRect(0,0,600,600);

radialGradient = ctx.createRadialGradient(x, y,10, x, y,50);

radialGradient.addColorStop(0,"#FFFFFF");

radialGradient.addColorStop(1,"#EA7F26");

ctx.fillStyle = radialGradient;

ctx.beginPath();

ctx.moveTo(x, y);

ctx.arc(x, y,50,0,2\* Math.PI, false);

ctx.fill();

x = x + Math.cos(angle)\* scope;

y = y + Math.sin(angle)\* scope;

angle = angle +.1;

requestAnimationFrame(draw);

ctx.closePath();

ctx.beginPath();

ctx.arc(300,300,200,0,2\* Math.PI, false);

ctx.stroke();

}

requestAnimationFrame(draw);

```

>\*\*`8:课后练习`\*\*

-`使用Canvas绘制下图环形进度条`

![image](./9.gif)

-`使用Canvas实现下图始终效果`

![image](./2.gif)

-`为前面的Canvas气泡添加运动效果`

![image](./1.gif)

-`气泡向上做正弦运动`

>\*\*`9:总结`\*\*

```css

本节课介绍了Canvas动画效果的基本原理,从线性、变速、函数及环形运动介绍了Canvas常用的运动函数  
```

##(三十一)HTML地理坐标

>\*\*`1:定位基准`\*\*

-`IP`

-`ISP机房位置(登记位置)-小区大楼级别-十几米`

-`GPS`

-`卫星定位-精确-需要硬件支持-军方1米`

-`WIFI`

-`Mac地址-比较精确-1米`

-`GSM或CDMA`

-`设备基站-精度高-手机及通信设备-10米`

-`用户指定`

>\*\*`2:获取流程`\*\*

-`打开Web应用`

-`向浏览器请求地理位置,弹出询问窗口`

-`浏览器从设备或受信任服务器获取位置信息并返回`

>\*\*`3:浏览器兼容`\*\*

-`使用前检测浏览器是否支持`

![image](./browser.png)

>\*\*`4:获取用户当前位置(科学上网)`\*\*

-`getCurrentPosition(onSuccess, onError, options)`

-`options`

-`enableHighAccuracy -高精度标识`

-`timeout -超时时间-默认0(无穷大)`

-`maximumAge -缓存时间`

```javascript

const getLocation =()=>{

const options ={

enableHighAccuracy: false,

maximumAge:1000

}

if(navigator.geolocation){

//浏览器支持geolocation

navigator.geolocation.getCurrentPosition(onSuccess,onError,options);

} else {

//浏览器不支持geolocation

alert('当前浏览器不支持getLocation');

}

}

//成功回调

function onSuccess(position){

const longitude = position.coords.longitude;

//纬度

const latitude = position.coords.latitude;

console.log('position',{ longitude, latitude });

}

//失败回调

function onError(error){

switch(error.code){

case 1:

alert("位置服务被拒绝");

break;

case 2:

alert("暂时获取不到位置信息");

break;

case 3:

alert("获取信息超时");

break;

case 4:

alert("未知错误");

break;

}

}

```

>\*\*`5:持续获取用户当前位置(科学上网)`\*\*

-`watchCurrentPosition(onSuccess, onError, options)`

```javascript

let watchId = undefined;

const getLocation =()=>{

const options ={

enableHighAccuracy: false,

maximumAge:1000

}

if(navigator.geolocation){

//浏览器支持geolocation

watchId = navigator.geolocation.watchPosition(showPosition);

} else {

//浏览器不支持geolocation

alert('当前浏览器不支持getLocation');

}

}

function showPosition(position){

const longitude = position.coords.longitude;

const latitude = position.coords.latitude;

console.log('position',{ longitude, latitude });

}

const cancel =()=>{

if(watchId) navigator.geolocation.clearWatch(watchId);

}

```

>\*\*`6:coords`\*\*

-`coords.latitude -十进制数的纬度`

-`coords.longitude -十进制数的经度`

-`coords.accuracy -位置精度`

-`coords.altitude -海拔,海平面以上以米计`

-`coords.altitudeAccuracy -位置的海拔精度`

-`coords.heading -方向,从正北开始以度计`

-`coords.speed -速度,以米/每秒计`

-`timestamp -响应的日期/时间`

>\*\*`7:调用百度地图显示实时位置`\*\*

```css

<script type="text/javascript" src="http://api.map.baidu.com/api?v=2.0&ak=百度地图key"></script>

/\*创建Map实例\*/

const map = new BMap.Map("map");

/\*初始化地图,设置中心点坐标和地图级别\*/

map.centerAndZoom(new BMap.Point(116.404,39.915),11);

/\*开启鼠标滚轮缩放\*/

map.enableScrollWheelZoom(true);

```

>\*\*`8:Google地图(国内无法使用)`\*\*

>\*\*`9:课后练习`\*\*

-`使用手机浏览器获取当前地理位置-使用express简易服务器`

>\*\*`10:总结`\*\*

```css

本节课介绍了浏览器地理坐标的获取基准,结合具体百度地图展示了地理坐标从获取到展示的过程  
```

##(三十二)离线地图

>\*\*`1:瓦片地图原理`\*\*

-`地图瓦片-256px \*256px`

![image](./all.png)

![image](./wall.png)

-`瓦片坐标系统- x/y/z`

-`x,y -当前瓦片位置`

-`当前瓦片缩放级别`

-`第一级瓦片`

![image](./0-0-0.png)

-`第二级瓦片`

![image](./1-1-1.png)

-`瓦片地图优势`

-`高效缓存`

-`渐进加载`

-`简单易用`

>\*\*`2:Leaflet`\*\*

-`显示地图`

```css

var map = L.map('map').setView([51.505,-0.09],13);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png').addTo(map);

```

-`添加Marker、Popup`

```css

L.marker([51.5,-0.09]).addTo(map)

.bindPopup('FirstMarker')

.openPopup();

```

-`添加线条`

```css

var latlngs =[

[51.5,-0.09],

[55.5,-1.09],

[56.5,-2.09]

];

var polyline = L.polyline(latlngs,{

color:'red'

}).addTo(map);

```

-`添加多边形`

```css

var latlngs =[

[50,0],

[51,-1],

[52,-2],

[51,-3]

];

var polygon = L.polygon(latlngs,{

color:'red'

}).addTo(map);

```

>\*\*`3:离线地图`\*\*

![image](./offlineline.png)

>\*\*`4:其他应用`\*\*

![image](./others.png)

![image](./7.gif)

![image](./8.gif)

>\*\*`5:总结`\*\*

```css

本节课介绍了瓦片地图原理(标准),介绍了比较流行地图离线插件Leaflet的基本使用。

```

##(三十三)Audio

>\*\*`1:基本语法`\*\*

```css

<audio

src="xxxx.mp3"

controls="controls"

loop="loop"

autoplay="autoplay"

>

您的浏览器不支持html5的audio标签

</audio>

```

-`src -歌曲路径- mp3- ogg - wav`

-`controls -播放控制器`

-`loop -歌曲循环`

-`autoplay -自动播放-主流浏览器已关闭- JS触发`

-`preload -预加载- autoplay时忽略此属性`

>\*\*`2:Audio对象`\*\*

-`currentTime -获取当前播放时间`

-`duration -获取歌曲的总时间`

-`play -播放`

```css

audio.addEventListener("play",()=>{});

audio.onplay =()=>{}

```

-`pause -暂停`

-`loadstart -开始加载`

-`durationchange -时长数据变化`

-`loadedmetadata -元数据已加载`

-`loadeddata -当前帧的数据已加载,无法播放下一帧`

-`progress -加载中`

-`canplay -可以播放`

-`canplaythrough -边缓冲边播放`

-`play()-播放歌曲- Promise`

-`pause()-暂停歌曲`

-`load()-重新加载歌曲`

>\*\*`3:Audio.js`\*\*

-`http://kolber.github.io/audiojs/`

-`标准浏览器使用Audio标签`

-`旧浏览器使用Flash`

-`可以通过 CSS 定义外观`

```css

<script src="/audiojs/audio.min.js"></script>

<script>

audiojs.events.ready(function(){

const as = audiojs.createAll();

});

</script>

<audio style="display:none" src="./music.mp3" preload="auto"/>

```

>\*\*`4:总结`\*\*

```css

本节课介绍了Audio元素的基本使用方法,介绍了Audio对象的属性和方法  
```

##(三十四)Video

>\*\*`1:基本语法`\*\*

```css

<video

src="xxxx.mp4"

controls="controls"

loop="loop"

autoplay="autoplay"

>

您的浏览器不支持html5的video标签

</video>

```

-`src -源- mp4- webm - ogg`

-`autoplay -自动播放`

-`controls -控制器`

-`height`

-`width`

-`loop -循环播放`。

-`muted -静音`

-`poster -预览图`

-`preload -预加载- autoplay时失效`

>\*\*`2:Video对象`\*\*

-`currentTime -获取当前播放时间`

-`duration -获取视频的总时间`

-`play -播放`

```css

video.addEventListener("play",()=>{});

video.onplay =()=>{}

```

-`pause -暂停`

-`loadstart -开始加载`

-`durationchange -时长数据变化`

-`loadedmetadata -元数据已加载`

-`loadeddata -当前帧的数据已加载,无法播放下一帧`

-`progress -加载中`

-`canplay -可以播放`

-`canplaythrough -边缓冲边播放`

-`play()-播放视频- Promise`

-`pause()-暂停视频`

-`load()-重新加载视频`

>\*\*`3:视频格式与流媒体`\*\*

-`视频格式`

-`mp4`

-`ogg`

-`webm`

-`流媒体- Video标签不支持`

-`rtmp - Adobe - TCP协议-稳定性好`

-`rtsp - Netscape - DUP协议-实时性好`

-`flv -苹果公司- TCP协议-切成一段段m3u文件`

>\*\*`4:Video.js`\*\*

```css

<link href="https://cdnjs.cloudflare.com/ajax/libs/video.js/7.3.0/video-js.min.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/video.js/7.3.0/video.min.js"></script>

const player = videojs('video', options, function onPlayerReady(){

this.play();

this.on('ended', function(){

videojs.log('播放结束');

});

});

```

-`支持流媒体格式`

-`ftmp`

-`rtsp`

-`flv`

>\*\*`5:总结`\*\*

```css

本节课介绍了Video元素的基本使用方法,介绍了Video对象的属性和方法,最后结合流媒体格式介绍了Video.js播放器插件  
```

##(三十五)构造函数

>\*\*`1:基本概念`\*\*

-`创建对象`

-`对象字面量`

```css

const person ={

name:'Eric',

age:28,

school:'侠课岛',

getName: function(){

return this.name;

},

getAge: function(){

return this.age;

},

};

```

-`构造函数`

```css

function Person(name, age){

this.name = name ||'';

this.age = age ||0;

school:'侠课岛',

this.getName = function(){

return this.name;

}

this.getAge = function(){

return this.age

}

}

const Eric = new Person('Eric',26);

```

-`构造函数`

-`创建对象时初始化对象`

-`为对象成员变量赋初始值`

-`总与new运算符一起使用在创建对象语句中`

-`方便创建多个对象的实例`

-`实例化对象-创建对象的过程`

>\*\*`2:构造函数的特点`\*\*

-`首字母必须大写(区分于普通函数)`

-`this 指向生成的实例对象`

-`使用 new 关键字生成实例对象`

>\*\*`3:构造函数与普通函数的区别`\*\*

-`构造函数可以当做普通函数使用-尽量避免`

-`构造函数内部会创建实例-普通函数不会创建`

-`构造函数内部的this指向实例本身-普通函数指向调用者(默认window)`

```css

/\*构造函数\*/

function Person(){

console.log('this', this);

}

const A = new Person();

/\*普通函数\*/

function person(){

console.log('this', this);

}

const B = person();

```

>\*\*`4:构造函数返回值`\*\*

-`无return(void)-返回this`

```css

function Student(name){

this.name = name ||'';

}

const Tony = new Student('Tony');

console.log('Tony');

```

-`return 基本数据类型-返回this`

```css

function Teacher(name){

this.name = name ||'';

return 'teacher';

}

const Davy = new Teacher('Davy');

console.log(Davy);

```

-`return 对象-返回改对象`

```css

function Driver(name){

this.name = name;

return {

name:'xxxx',

};

}

const Micheal = new Driver('Micheal');

console.log(Micheal);

```

>\*\*`5:构造函数的创建过程`\*\*

-`创建一个空对象`

```css

var obj ={};

```

-`将空对象的\_\_proto\_\_指向构造函数对象的prototype`

```css

obj.\_\_proto\_\_= constructorFunction.prototype;

```

-`将构造函数的作用域赋给新对象`

```css

var result = constructorFunction.call(obj);

```

-`返回新对象`

```css

typeof result ==='object'? result : obj;

```

```css

var obj ={};

obj.\_\_proto\_\_= constructorFunction.prototype;

constructorFunction.call(obj);

return obj;

```

>\*\*`6:课后练习`\*\*

-`分析下列代码的执行结果(先分析再实际操作)`

```css

function Sum(a, b){

this.p = a + b;

this.getResult = function(){

return this.p;

}

return this.p;

}

const eight = new Sum(3,5);

console.log(eight.getResult());

const seven = Sum(3,5);

console.log(seven);

```

-`分析下列代码的执行结果(先分析再实际操作)`

```css

function Person(name, age){

this.name = name;

this.age = age;

this.eat = function(){

console.log(age +"岁的"+ name +"在吃饭。");

}

}

const p1= new Person("Eric",24);

const p2= new Person("Vico",24);

console.log(p1.eat === p2.eat);

```

-`分析下列代码,仅从代码重复改码量考虑构造函数(本身)的优点与缺点`

```css

/\*不使用构造函数\*/

const Eric ={

name:'Eric',

type:'student',

class:'侠课岛',

getName: function(){

return this.name;

}

}

const Davy ={

name:'Davy',

type:'student',

class:'侠课岛',

getName: function(){

return this.name;

}

}

const Micheal ={

name:'Micheal',

type:'teacher',

area:'长沙',

getName: function(){

return this.name;

}

}

const Vico ={

name:'Micheal',

type:'Vico',

area:'长沙',

getName: function(){

return this.name;

}

}

/\*使用构造函数\*/

function Student(name){

this.type ='student',

this.class ='侠课岛',

this.name = name,

getName: function(){

return this.name;

}

}

function Teacher(name){

this.type ='teacher',

this.name = name,

this.area ='长沙',

getName: function(){

return this.name;

}

}

const Eric = new Student('Eric');

const Davy = new Student('Davy');

const Micheal = new Teacher('Micheal');

const Vico = new Teacher('Vico');

```

>\*\*`7:总结`\*\*

```css

本节课从JavaScript创建对象入手,从基本用法、返回值及与普通函数的对比介绍了构造函数的基本用法,并抛出构造函数的执行过程  
```

##(三十六)JavaScript原型链

>\*\*`1:基本概念`\*\*

-`JavaScript 只有一种结构:对象`

-`JavaScript -基于原型的语言(prototype-based language)`

-`每个对象拥有一个原型对象,从中继承方法和属性`

-`原型对象也可能拥有原型,并从中继承方法和属性`

-`以此类推形成一个链路-原型链`

-`\_\_proto\_\_( Object.getPrototypeOf())-私有属性`

-`对象属性`

-`指向构造该对象的构造函数原型的指针(构造函数的prototype)`

```css

/\*定义数字类型\*/

const num =1234;

/\*包含toString()、toFixed()等方法\*/

num.toFixed(2);

/\*查看num的构造函数\*/

num.\_\_proto\_\_

/\*

Number {

toPrecision:ƒ,}

constructor:ƒ Number()

toExponential:ƒ toExponential()

toFixed:ƒ toFixed()

toLocaleString:ƒ toLocaleString()

toPrecision:ƒ toPrecision()

toString:ƒ toString()

valueOf:ƒ valueOf()

\_\_proto\_\_: Object

}

\*/

/\* Number的构造函数\*/

num.\_\_proto\_\_.\_\_proto\_\_

/\*

{

constructor:ƒ doSomething(),

\_\_proto\_\_:{

constructor:ƒ Object(),

hasOwnProperty:ƒ hasOwnProperty(),

isPrototypeOf:ƒ isPrototypeOf(),

propertyIsEnumerable:ƒ propertyIsEnumerable(),

toLocaleString:ƒ toLocaleString(),

toString:ƒ toString(),

valueOf:ƒ valueOf()

}

}

\*/

/\*继续查找\*/

num.\_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_

/\* null \*/

```

-`prototype -原型`

-`函数特有属性`

-`指向原型对象的指针`

-`包含所有实例共享的属性和方法`

```css

function fun(){}

fun.prototype

/\*

{

constructor:ƒ fun(),

\_\_proto\_\_:{

constructor:ƒ Object(),

hasOwnProperty:ƒ hasOwnProperty(),

isPrototypeOf:ƒ isPrototypeOf(),

propertyIsEnumerable:ƒ propertyIsEnumerable(),

toLocaleString:ƒ toLocaleString(),

toString:ƒ toString(),

valueOf:ƒ valueOf()

}

}

\*/

```

-`Object.getPrototypeOf(new Fun())=== Fun.prototype`

>\*\*`2:追溯原型链`\*\*

-`hasOwnProperty -判断一个对象是否具有某个属性或对象`

```css

function Person(name){

this.name = name,

this.getName = function(){

return this.name;

}

}

const Eric = new Person('Eric');

Eric.hasOwnProperty('getName');/\* true \*/

Eric.hasOwnProperty('name');/\* true \*/

Eric.hasOwnProperty('toString');/\* false \*/

```

-`isPrototypeOf -判断一个对象是否是另一个对象的原型`

-`对象直接量- Object.prototype`

-`构造函数-构造函数的prototype`

```css

function Person(name){

this.name = name,

this.getName = function(){

return this.name;

}

}

const Eric = new Person('Eric');

Person.prototype.isPrototypeOf(Eric);/\* true \*/

```

-`instanceof -判断一个对象是否是一个对象的实例`

```css

function Person(name){

this.name = name,

this.getName = function(){

return this.name;

}

}

const Eric = new Person('Eric');

Eric instanceof Person;/\* true \*/

```

>\*\*`3:修改原型`\*\*

```css

function Person(name){

this.name = name;

this.getName = function(){

return this.name;

}

}

const Eric = new Person('Eric');

Person.prototype.getFullName = function(){

return 'This is'+ this.name;

}

Eric.getFullName();/\* This is Eric \*/

Person.prototype;

/\*

{

constructor:ƒ fun(),

getFullName:ƒ fun(),

\_\_proto\_\_:{

constructor:ƒ Object(),

hasOwnProperty:ƒ hasOwnProperty(),

isPrototypeOf:ƒ isPrototypeOf(),

propertyIsEnumerable:ƒ propertyIsEnumerable(),

toLocaleString:ƒ toLocaleString(),

toString:ƒ toString(),

valueOf:ƒ valueOf()

}

}

\*/

```

>\*\*`6:课后练习`\*\*

-`将公用方法getName提取到prototype级`

```css

function Student(name){

this.type ='student',

this.class ='侠课岛',

this.name = name,

getName: function(){

return this.name;

}

}

function Teacher(name){

this.type ='teacher',

this.name = name,

this.area ='长沙',

getName: function(){

return this.name;

}

}

```

-`思考下列代码的执行结果`

```css

function Person(){

this.name ='Eric';

}

Person.prototype.getName = function(){

return this.name;

}

function Student(){}

Student.prototype = new Person();

Student.prototype ={

getAge: function(){

return 28;

}

}

const Eric = new Student();

Eric.name;

Eric.getName();

Eric.getAge();

```

-`思考下列代码的执行结果`

```css

function Person(){

this.name ='Eric';

}

Person.prototype.getName = function(){

return this.name;

}

function Student(){

this.getName = function(){

return 'Davy';

}

}

Student.prototype = new Person();

Student.prototype ={

getName: function(){

return 'Micheal';

}

};

const Eric = new Student();

Eric.name;

Eric.getName();

```

>\*\*`7:总结`\*\*

```css

本节课介绍了JavaScript原型链,从对象属性入手介绍了\_\_proto\_\_与prototype,讲解了原型链的追溯方式,最后结合练习题引入原型链继承概念  
```

##(三十七)JavaScript常用继承方式(上)

>\*\*`1:原型链继承`\*\*

-`重写原型对象,赋予一个新对象的实例`

```css

function Super(){

this.text ='Hello';

}

Super.prototype.getSuperText = function(){

return this.text;

}

function Sub(){

this.subText ='Word';

}

Sub.prototype = new Super();

const instance = new Sub();

console.log(instance);

```

-`优点-简单易操作`

-`缺点-对引用类型数据操作会互相影响`

```css

function Super(){

this.value =[1,2,3,4];

}

Super.prototype.getSuperValue = function(){

return this.value;

}

function Sub(){

this.subText ='Word';

}

Sub.prototype = new Super();

const instance1= new Sub();

const instance2= new Sub();

instance1.value.push(5);

console.log(instance2.value);

```

>\*\*`2:构造函数继承`\*\*

```css

function Super(){

this.value =[1,2,3,4];

}

Super.prototype.getSuperValue = function(){

return this.value;

}

function Sub(){

Super.call(this);

}

const instance1= new Sub();

instance1.value.push(5);

console.log(instance1.value);

const instance2= new Sub();

console.log(instance2.value);

```

-`构造函数执行时申请新的内存空间`

-`只能继承父类的实例属性和方法,不能继承原型属性和方法`

>\*\*`3:组合继承`\*\*

-`保留构造函数继承与原型链继承优点`

```css

function Person(name){

this.name = name;

this.value =["head","body","legs"];

}

Person.prototype.getName = function(){

return this.name;

};

/\*构造函数继承\*/

function Teacher(name, school){

Person.call(this, name);

this.school = school;

}

/\*原型链继承\*/

Teacher.prototype = new Person();

/\* prototype构造器指回自己\*/

Teacher.prototype.constructor = Teacher;

Teacher.prototype.getSchool = function(){

return this.school;

};

```

-`执行了两次Person -属性重复`

>\*\*`4:原型式继承`\*\*

-`利用一个空对象作为中介`

```css

const lakers ={

name:"lakers",

value:["Micheal","Wade","Kobe"]

};

const lakers1= Object.create(lakers);

const lakers2= Object.create(lakers);

lakers1.value.push('Fish');

console.log(lakers);

```

-`模拟Object.create()`

```css

Object.prototype.create = function(obj){

function Fun(){}

Fun.prototype = obj;

return new Fun();

}

```

-`无法传递参数`

-`引用类型存在污染`

##(三十八)JavaScript常用继承方式(下)

>\*\*`5:寄生式继承`\*\*

-`寄生新的方法属性`

-`Object.createNew()`

```css

Object.prototype.createNew = function(obj){

var newObj = Object.create(obj);

newObj.getLength = function(){...};

return newObj;

}

```

-`无法传递参数`

-`引用类型存在污染`

>\*\*`6:寄生组合继承`\*\*

```css

function Super(name){

this.name = name;

this.value =["Hello","Word"];

}

Super.prototype.getName = function(){

return this.name;

};

function Sub(name, age){

Super.call(this, name);

this.age = age;

}

let prototype = Object.create(Super.prototype);

prototype.constructor = Sub;

Sub.prototype = prototype;

Sub.prototype.getAge = function(){

return this.age;

}

const instance1= new Sub("Eric",23);

const instance2= new Sub("Vico",23);

instance1.value.push("!");

instance2.value.push("!!");

```

-`Super只执行一次`

>\*\*`7:继承多个对象`\*\*

```css

function

ClassOne.prototype = Object.create(SuperClass.prototype);

Object.assign(ClassOne.prototype, ClassTwo.prototype);

ClassOne.prototype.constructor = ClassOne;

```

>\*\*`8:课后练习`\*\*

-`下列代码的输出结果是`

```css

function A(){

this.name ='a';

this.color =['green','yellow'];

}

function B(){

}

B.prototype = new A();

var b1= new B();

var b2= new B();

b1.name ='change';

b1.color.push('black');

console.log(b1.name);

console.log(b2.name);

console.log(b1.color);

console.log(b2.color);

```

-`已知如下类Animal,要求设计一个Cat类继承自Animal,并实现如下功能(头条面试题)`

```css

/\* Animal \*/

function Animal(){

this.name ="Animal";

this.showName = function(){

console.log(this.name);

}

}

/\* Cat \*/

function Cat(){

this.name ="Cat";

this.showName1= function(){

console.log(this.name);

}

this.showName2= function(){

console.log(this.name);

}

this.showName3= function(){

console.log(this.\_\_super.name +"=>"+ this.name);

}

}

```

-`输出`

```css

var cat = new Cat();

console.log(cat instanceof Animal );/\* true \*/

cat.showName1();/\* Cat \*/

cat.showName2();/\* Animal \*/

cat.showName3();/\* Animal => Cat \*/

```

>\*\*`9:总结`\*\*

```css

本节课在JS原型链与JS构造函数基础上介绍了原型链继承、构造函数继承、组合继承、原型式继承、寄生式继承、寄生组合继承与继承多个父类  
```

##(三十九)JavaScript中的this

>\*\*`1:基本概念`\*\*

-`this -当前执行代码的环境对象-上下文`

-`严格模式-具有限制性JavaScript变体`

-`通过抛出错误来消除了一些原有静默错误`

-`修复了一些导致JavaScript引擎难以执行优化的缺陷`

-`禁用了在ECMAScript的未来版本中可能会定义的一些语法`

```css

/\*为脚本开启严格模式\*/

"use strict";

var v ="Hi! I'm a strict mode script!";

/\*为函数开启严格模式\*/

function strict(){

'use strict';

function nested(){

return "And so am I!";

}

return "Hi! I'm a strict mode function!"+ nested();

}

```

>\*\*`2:全局环境`\*\*

-`无论是否在严格模式下,在全局执行环境中(任何函数体外部)this指向全局对象`

```css

/\*浏览器环境\*/

var name ='Eric';

console.log(window.name === this.name);/\* true \*/

console.log(window === this);/\* true \*/

```

>\*\*`3:函数体内部`\*\*

-`this的值取决于函数被调用的方式`

-`简单调用`

-`非严格模式- this默认指向全局对象`

```css

/\*浏览器环境\*/

function simple(){

return this;

}

console.log(simple()=== window);/\* true \*/

```

-`严格模式-保持进入执行环境时的值-没有指定时默认值为undefined`

```css

/\*浏览器环境\*/

function simple(){

"use strict";

return this;

}

simple()=== undefined;/\* true \*/

window.simple()=== window;/\* true \*/

```

-`this传递`

-`call / apply -绑定值是一个对象,存在ToObject过程`

```css

/\*浏览器环境\*/

var object ={

name:'Eric'

};

var name ='Iven';

function getName(arg){

return this.name;

}

getName();/\* Iven \*/

getName.call(object);/\* Eric \*/

getName.apply(object);/\* Eric \*/

```

-`bind -柯里化的call / apply -只能被绑定一次`

```css

name:'Davy';

function bindThis(){

return this.name;

}

var getName1= bindThis.bind({ name:"Eric"});

console.log(getName1());/\* Eric \*/

var getName2= getName1.bind({ name:"Iven"});

console.log(getName2());/\* Eric \*/

```

-`箭头函数- this与封闭词法环境(封闭作用域)的this保持一致- call / apply / bind将被忽略`

```css

/\*浏览器环境\*/

var globalThis = this;

var arrowsFunction =()=> this;

console.log(arrowsFunction()=== globalObject);/\* true \*/

```

-`作为对象方法被调用-靠近原则`

```css

/\*浏览器环境\*/

var object ={

name:'Eric',

getName: function(){

return this.name;

}

};

console.log(object.getName());/\* Eric \*/

function getName2(){

return this.name;

}

object.getName = getName2;

console.log(object.getName());/\* Eric \*/

object.object ={

getName: getName2,

name:'Iven'

};

console.log(object.object.getName());/\* Iven \*/

```

>\*\*`4:全局函数`\*\*

-`setTimeout`

-`setInterval`

-`alert`

>\*\*`5:课后练习`\*\*

-`下列代码的输出结果是`

```css

var name ="window";

function getName(){

var name ="inner";

console.log(this.name);

}

getName();

```

-`下列代码的输出结果是`

```css

var name ="window";

var person ={

name:"inner",

getName1: function (){

console.log(this.name);

},

getName2:()=>{

console.log(this.name);

}

}

person.getName1();

person.getName2();

```

-`下列代码的输出结果是`

```css

function foo(){

setTimeout(()=>{

console.log(this.id);

setTimeout(()=>{

console.log(this.id);

},100);

},100);

}

foo.call({id:123});

```

>\*\*`6:总结`\*\*

```css

本节课从严格模式与非严格模式两个角度出发,介绍了JavaScript中this在全局环境和函数体内部的指向  
```

##(四十)JavaScript调用栈

>\*\*`1:基本概念`\*\*

-`单线程模型`

-`Event Loop`

![image](./eventloop.png)

-`宏任务- Macro-task`

-`setTimeout`

-`setInterval`

-`I/O`

-`微任务- Micro-task`

-`promise`

>\*\*`2:Event Loop`\*\*

-`所有同步任务都在主线程上执行,形成一个执行栈`

-`主线程之外,存在一个任务队列(task queue),异步任务有了运行结果会在任务队列之中放置一个任务`

-`执行栈中的所有同步任务执行完毕后读取任务队列(先读取微任务、宏任务)`

-`不断重复上面的第三步`

```css

console.log('script start');

setTimeout(function(){

console.log('timeout1');

},10);

new Promise(resolve =>{

console.log('promise1');

resolve();

setTimeout(()=> console.log('timeout2'),10);

}).then(function(){

console.log('then1')

})

console.log('script end');

/\*

\* script start

\* promise1

\* script end

\* then1

\* timeout1

\* timeout2

\*/

```

>\*\*`3:宏任务`\*\*

-`由JavaScript线程外的宿主线程执行`

-`定时触发器线程`

-`异步HTTP请求线程`

-`JavaScript线程不空闲宏任务永远没有执行机会`

```css

for(let i=0; i<100000000; i++){}

setTimeout(function(){

console.log('setTimeout1');

},1000);

setTimeout(function(){

console.log('setTimeout2');

},2000);

```

>\*\*`4:微任务`\*\*

-`由JavaScript线程维护`

-`在主线程所有可执行代码执行完成后执行`

-`浏览器渲染Dom前会全部执行`

>\*\*`5:课后练习`\*\*

-`下列代码的输出结果是`

```css

for (var i =1;i <=5;i ++){

setTimeout(function timer(){

console.log(i)

}, i \*1000)

}

```

-`将上面代码稍作改动,使其输出变为1、2、3、4、5`

-`下列代码的输出结果是`

```css

console.log('global');

setTimeout(function (){

console.log('timeout1');

new Promise(function (resolve){

console.log('timeout1\_promise');

resolve();

}).then(function (){

console.log('timeout1\_then');

})

},2000)

for (var i=1; i<=5; i++){

setTimeout(function(){

console.log(i)

}, i \*1000);

console.log(i);

}

new Promise(function (resolve){

console.log('promise1');

resolve();

}).then(function (){

console.log('then1');

})

setTimeout(function (){

console.log('timeout2');

new Promise(function (resolve){

console.log('timeout2\_promise');

resolve();

}).then(function (){

console.log('timeout2\_then');

})

},1000)

new Promise(function (resolve){

console.log('promise2');

resolve();

}).then(function (){

console.log('then2');

})

```

>\*\*`6:总结`\*\*

```css

本节课从JavaScript单线程模型出发,介绍了浏览器环境下JavaScript线程的执行过程,结合具体例子介绍了宏任务与微任务结合的执行过程  
```

##(四十一)JavaScript函数式编程(上)

>\*\*`1:基本概念`\*\*

-`函数式编程`

-`一种编程思维方式`

-`对复用性高的功能代码进行函数封装`

-`实现代码的高复用性`

-`函数式编程特点`

-`函数是第一等公民`

-`只用表达式,不用语句`

-`没有副作用`

-`不修改状态`

-`引用透明`

>\*\*`2:函数是一等公民`\*\*

-`函数应用`

```css

/\*声明函数\*/

function getName(){

}

/\*表达式函数\*/

var getName = function(){

}

/\*匿名函数\*/

setTimeout(function(){

},1000);

/\*自执行函数(IIFE)\*/

(function(){

})();

```

-`函数声明优先级高于变量声明和函数表达式`

```css

console.log(getName);

getName();

var getName;

getName ='Eric';

function getName(){

console.log('function getName');

}

console.log(getName);

```

>\*\*`3:纯函数`\*\*

-`对于相同的输入,永远会得到相同的输出`

```css

function getNumber(num){

return num + Math.random();

}

```

-`不改变输入值`

```css

function getGirlGift(list){

/\*输入值改变\*/

list = list.map(girl =>{

girl.gift = girl.age >18?'big':'small';

});

return list;

}

```

-`不包含副作用(网络、I/O)`

```css

var array =[1,2,3,4,5];

array.slice(0,3);

array.slice(0,3);

/\*改变原数组\*/

array.splice(0,3);

array.splice(0,3);

/\*网络请求\*/

asiox.get('https://www.xxxx.com').then(res =>{

})

/\*时间\*/

function getDate(){

return new Date();

}

```

-`Array函数举例`

```css

/\*不纯\*/

array.push();/\*数组尾部插入\*/

array.pop();/\*删除并返回数组最后一个元素\*/

array.unshift();/\*数组头部插入\*/

array.shift();/\*删除并返回数组第一元素\*/

array.splice();/\*删除元素,并向数组添加元素\*/

array.reverse();/\*/颠倒数组元素的顺序\*/

array.sort();/\*排序数组元素\*/

/\*纯函数\*/

array.slice();/\*数组中返回选定的元素\*/

array.concat();/\*连接数组,并发挥新数组\*/

array.join();/\*按分隔符连接数组,返回字符串\*/

```

>\*\*`4:函数柯里化`\*\*

-`传递给函数一部分参数来调用它,让它返回一个函数去处理剩下的参数`

```css

/\*普通函数\*/

function add(x, y){

return x + y;

}

add(1,2);

/\*柯里化改编\*/

function addX(y){

return function (x){

return x + y;

};

}

addX(2)(1);

/\*函数不纯-硬编码-依赖min参数\*/

var min =90;

var isWell = score => score > min;

/\*柯里化改编\*/

var min =90;

var chekoLevel = baseLine =>(score => score > baseLine);

var isWell = chekoLevel(90);

```

-`一种对参数的缓存`

-`用在一些区分环境的函数预缓存`

-`已获取某些耗时操作结果缓存`

##(四十二)JavaScript函数式编程(下)

>\*\*`5:函数组合`\*\*

-`函数嵌套`

```css

function1(function2(function13(x)));

```

-`函数组合`

```css

var compose =(function1, function2)=>(x => function1(function2(x)));

var function1= param => param +1;

var function2= param => param +2;

var final = compose(function1, function2);

final(1);

/\*4\*/

```

>\*\*`6:Point Free`\*\*

-`不要命名转瞬即逝的中间变量`

```css

var getSplitWord = str => str.toUpperCase().split('');

```

-`柯里化封装`

```css

var toUpperCase = word => word.toUpperCase();

var split = x => str => str.split(x);

var getSplitWord = compose(split(''), toUpperCase);

```

>\*\*`7:声明式代码`\*\*

```javascript

var students =[{

name:'Eric',

score:99,

},

{

name:'Iven',

score:59,

}];

/\*命令式\*/

const getWell = students =>{

let result =[]

for (let i =0; i < students.length; i++){

if (students[i].score >=90){

result.push(students[i])

}

}

return result

};

/\*声明式\*/

const getWell = students => return students.filter(student => student.score >=90);

```

>\*\*`8:高阶函数`\*\*

-`把函数当参数,把传入的函数做一个封装`

```css

function add(x, y, f){

return f(x)+ f(y);

}

```

-`常见的高阶函数`

-`Array.map`

```css

var array =[1,2,3];

array.map(s => s +1);

```

-`Array.sort`

-`Array.filter`

-`react中应用比较广泛`

-`属性代理`

-`反向继承`

>\*\*`9:总结`\*\*

```css

本节课从函数式编程的基本概念出发,介绍了函数式编程的特点,最后结合实例介绍了函数式编程中核心概念  
```

##(四十三)JavaScript闭包

>\*\*`1:基本概念`\*\*

-`函数和声明该函数的词法环境的组合`

```javascript

function count(){

let count =0;

return function(){

count = count +1;

console.log(count);

}

}

var getCount = count();

getCount();

getCount();

```

-`将函数与其所操作的某些数据(环境)关联起来`

-`能够访问另一个函数作用域的变量的函数`

>\*\*`2:闭包的特点`\*\*

-`函数嵌套函数`

-`函数内部可以引用外部的参数和变量`

-`参数和变量不会被垃圾回收机制回收`

>\*\*`3:用途`\*\*

-`匿名自执行函数-只需要执行一次,其内部变量无需维护`

```javascript

(function(){

/\*...\*/

})();

```

-`封装回调保存作用域(缓存结果)`

```javascript

for(var i =1; i <5; i++){

setTimeout((function(i){

return function(){

console.log(i);

}

})(i), i \*1000)

}

```

-`模拟私有方法`

```javascript

var myNameSpace =(function (){

/\*私有计数器变量\*/

var myPrivateVar =0;

/\*记录所有参数的私有函数\*/

var myPrivateMethod = function (foo){

console.log(foo + myPrivateVar);

};

return {

/\*公有变量\*/

myPublicVar:'foo',

/\*调用私有变量和方法的公用函数\*/

myPublicFunction: function (bar){

/\*增加私有计数器值\*/

myPrivateVal ++;

myPrivateMethod(bar);

}

};

})();

```

>\*\*`4:常见误区`\*\*

-`在循环中创建闭包`

```javascript

for (var i =1; i <=5; i++){

setTimeout(function(){

console.log(i)

}, i \*1000)

}

```

-`添加监听事件`

```javascript

function func(){

var element = document.getElementById("app");

element.onclick = function(){

alert(element.id);

}

}

```

```javascript

function func(){

var element = document.getElementById("app");

var id = element.id;

element.onclick = function(){

alert(id);

}

element = null;

}

```

>\*\*`5:课后练习`\*\*

-`下列代码的输出结果是`

```javascript

function fn(){

var max =10;

function bar(x){

if (x > max){

console.log(x)

}

}

return bar;

}

var f1= fn();

var max =100;

f1(20);

```

-`下列代码的输出结果是`

```javascript

var object ={

name:'Eric',

getName: function(){

return function(){

console.log(this.name)

}

}

}

object.getName()();

```

-`下列代码的输出结果是`

```javascript

function circle(n, o){

console.log(o);

return {

circle: function(m){

return circle(m, n);

}

};

}

var a = circle(0);

a.circle(1);

a.circle(2);

a.circle(3);

var b = circle(0)

.circle(1)

.circle(2)

.circle(3);

var c = circle(0).circle(1);

c.circle(2);

c.circle(3);

```

>\*\*`6:总结`\*\*

```css

本节课从典型的闭包出发,介绍了闭包的基本概念、特点和应用场景,最后结合具体例子介绍了闭包的主要用途和常见的误区  
```

##(四十四)JavaScript正则表达式

>\*\*`1:基本概念`\*\*

-`正则表达式是用于匹配字符串中字符组合的模式`

-`正则表达式是对象`

-`普通字符`

-`a-z`

-`0-9`

-`A-Z`

-`...`

-`特殊字符`

-`^`

-`$`

-`.`

-`(x)`

-`...`

>\*\*`2:创建正则表达式`\*\*

-`对象字面量`

```javascript

var reg =/abc/;

reg.exec('abcd');

```

-`RegExp对象构造函数`

```javascript

var reg = new RegExp(/abc/);

reg.exec('abcd');

```

>\*\*`3:匹配符`\*\*

-`[a-z]-小写字母从a-z中的任一字符`

```javascript

var reg = new RegExp(/abc[a-z]/);

reg.exec('abc0');

```

-`[A-Z]-大写字母从A-Z中的任一字符`

-`[0-9]-数字从0-9任一字符,相当于\d`

-`[0-9a-z]-数字从0-9或小写字母从a-z任一字符`

-`[0-9a-zA-Z]-数字从0-9,小写字母从a-z或者是大写字母从A-Z的任一字符`

-`[abcd]-字符abcd中的任一字符`

-`[1234]-数字1234中的任一字符`

-`[^a-z]-除小写字母从a - z以外的任一字符`

-`[^0-9]-除数字从0-9以外的任一字符`

-`[^abcd]-除abcd以外的任一字符`

-`\d -一个数字字符,用字符簇表示[0-9]`

```javascript

var reg = new RegExp(/abc\d/);

reg.exec('abcd');

```

-`\D -一个非数字字符,也可以使用字符簇[^0-9]或[^\d]来表示`

-`\w -包括下划线的任何单词字符,用字符簇[0-9a-zA-Z\_]`

```javascript

var reg = new RegExp(/\w/);

reg.exec('abcd');

```

-`\W -任何非单词字符,[^0-9a-zA-Z\_]或[^\w]`

-`\s -任何空白字符,空格、制表符、换行符都可以通过\s匹配`

-`\S -任何非空白字符,[^\s]`

-`.-除"\n"之外的任何单个字符`

>\*\*`4:限定符`\*\*

-`\*-0到多,可以使用{0,}代替`

```javascript

var reg = new RegExp(/abc\*/);

reg.exec('abcccccccccc');

```

-`+-1到多,可以使用{1,}代替`

-`?-0或1,可以使用{0,1}代替`

-`{n}-匹配n次,如{18},连续匹配18次`

-`{n,}-至少匹配n次,如{1,},代表最少匹配1次`

-`{n,m}-最少匹配n次且最多匹配m次,如{1,7}代表最少匹配1最多匹配7次`

```javascript

var reg = new RegExp(/abc{4}/);

reg.exec('abccc');

```

>\*\*`5:定位符`\*\*

-`^匹配输入字符串的开始位置`

```javascript

var reg = new RegExp(/^abc{4}/);

reg.exec('dabccc');

```

-`$匹配输入字符串的结束位置`

-`\b 匹配一个单词边界`

-`\B 匹配非单词边界`

>\*\*`6:转义符`\*\*

-`\-遇到正则中特殊符号需要转义`

```javascript

var reg = new RegExp(/abc/g);

'abcabcabcabcabc'.match(reg)

```

>\*\*`7:选择匹配符`\*\*

-`|-或`

>\*\*`8:标志符`\*\*

-`g -全局搜索`

-`i -不区分大小写搜索`

>\*\*`9:正则表达式方法`\*\*

-`test -是否匹配的RegExp方法,返回true或false`

```javascript

var reg = new RegExp(/^abc{4}/);

reg.test('dabccc');

```

-`exec -检索字符串中的正则表达式的匹配,返回一个数组-可循环调用(/g)`

```javascript

var reg = new RegExp(/abc/g);

reg.exec('abcabcabc');

```

-`match - String对象方法,检索字符串中的正则表达式的匹配(/g时全部匹配)`

-`search - String对象方法`

-`replace - String对象方法`

-`split - String对象方法`

>\*\*`10:课后练习`\*\*

-`实现一个匹配11位电话号码的正则表达式`

-`以1为开头`

-`第二位可为3,4,5,7,8,中的任意一位`

-`最后以0-9的9个整数结尾`

-`实现一个匹配电子邮件的正则表达式`

-`@之前必须有内容且只能是字母(大小写)、数字、下划线(\_)、减号(-)、点(.)`

-`@和最后一个.之间必须有内容且只能是字母(大小写)、数字、点(.)、减号(-),且两个点不能挨着`

-`最后一个.之后必须有内容且内容只能是字母(大小写)、数字且长度为大于等于2个字节,小于等于6个字节`

>\*\*`11:总结`\*\*

```css

本节课介绍了Javascript中正则表达式的基本概念,介绍了常用的特殊符号,最后结合实例介绍了正则表达式的常用方法  
```

##(四十五)JavaScript面向对象编程(OOP)

>\*\*`1:基本概念`\*\*

-`用抽象方式创建基于现实世界模型的一种编程模式`

-`每个对象能够接收消息,处理数据和发送消息给其他对象`

-`每个对象都可以被看作是一个拥有清晰角色或责任的独立个体`

-`促进更好的灵活性和可维护性`

-`Namespace -命名空间`

-`Class -类-定义对象的特征(属性和方法)`

-`Object -对象-类的一个实例`

-`Property -属性-对象的特征`

-`Method -方法-对象的能力`

-`Constructor -构造函数-对象初始化的瞬间被调用的方法`

-`Inheritance -继承-继承另一个类的特征`

-`Encapsulation -封装-把数据和相关的方法绑定在一起使用`

-`Polymorphism -多态-不同类可以定义相同的方法或属性`

-`面向过程`

-`开门(冰箱)`

-`装进(冰箱,大象)`

-`关门(冰箱)`

-`面向对象`

-`冰箱.开门()`

-`冰箱.装进(大象)`

-`冰箱.关门()`

>\*\*`2:Namespace -命名空间`\*\*

-`命名空间只是另一个包含方法、属性和对象的对象`

-`创建全局变量`

-`所有的变量、方法和功能成为该对象的属性`

```css

var myNameScape = myNameScape ||{};

/\*子命名空间\*/

myNameScape.event ={};

/\*定义方法和属性\*/

myNameScape.commonMethods ={

name:'',

validateName: function(name){

/\* name格式验证\*/

},

};

/\*定义方法\*/

myNameScape.event ={

addListener: function(el, type, fn){

/\*...\*/

},

removeListener: function(el, type, fn){

/\*...\*/

},

}

```

-`内置对象`

-`Math`

-`Array`

-`Object`

-`...`

>\*\*`3:Class -类(构造函数)`\*\*

-`JavaScript是一种基于原型的语言,没有类声明语句`

-`用方法作类`

```css

function Person(){

}

var Person = function(){

}

```

>\*\*`4:Object -对象(类的实例)`\*\*

```css

function Person(){

}

var person1= new Person();

var person2= new Person();

```

>\*\*`5:构造器`\*\*

-`构造器是对象中的一个方法`

-`实例化时构造器被调用`

-`JavaScript中函数可以作为构造器使用`

-`每个声明的函数都可以在实例化后被调用执行`

```javascript

function Person(){

console.log('Person created!');

}

var person1= new Person();

var person2= new Person();

```

>\*\*`6:Property -属性`\*\*

-`类中包含的变量`

```javascript

function Person(firstName){

this.firstName = firstName;

console.log('Person created!');

}

var person1= new Person('Alice');

var person2= new Person('Bob');

```

>\*\*`7:Method -方法`\*\*

-`方法与属性相似`

-`方法是函数,属性可以被定义为函数`

```javascript

function Person(firstName){

this.firstName = firstName;

}

Person.prototype.sayHello = function(){

console.log("Hello, I'm "+ this.firstName);

};

var person1= new Person("Alice");

var person2= new Person("Bob");

```

>\*\*`8:Inheritance -继承`\*\*

```

...

```

>\*\*`9:Encapsulation -封装`\*\*

-`把数据和相关的方法绑定在一起使用`

```javascript

function Person(name, age, sex){

this.name = name;

this.age = age;

this.sex = sex;

}

Person.prototype.show = function (){

console.log(this.name +''+ this.sex +''+ this.age);

}

var person = new Person('Eric',18,'男');

```

>\*\*`10:Polymorphism -多态`\*\*

-`同一操作作用于不同的对象,可以有不同的解释,产生不同的执行结果`

```javascript

function Person(name, age){

this.name = name;

this.age = age;

}

Person.prototype.toString = function(){

return "I am a Person, my name is "+ this.name;

}

function Man(name, age){

Person.apply(this, arguments);

}

Man.prototype = Object.create(Person.prototype);

Man.prototype.toString = function(){

return "I am a Man, my name is"+this.name;

}

var person = new Person("Neo",19);

var man1= new Man("Davin",18);

var man2= new Man("Jack",19);

```

>\*\*`11:课后练习`\*\*

-`下列函数的输出结果是`

```css

function Person(firstName){

this.firstName = firstName;

}

Person.prototype.sayHello = function(){

alert("Hello, I'm "+ this.firstName);

};

var person1= new Person("Alice");

var person2= new Person("Bob");

var helloFunction = person1.sayHello;

person1.sayHello();

person2.sayHello();

helloFunction();

```

-`下列代码的输出结果是`

```javascript

function Person(firstName){

this.firstName = firstName;

}

Person.prototype.walk = function(){

alert("I am walking!");

};

Person.prototype.sayHello = function(){

alert("Hello, I'm "+ this.firstName);

};

function Student(firstName, subject){

Person.call(this, firstName);

this.subject = subject;

};

Student.prototype = Object.create(Person.prototype);

Student.prototype.constructor = Student;

Student.prototype.sayHello = function(){

console.log("Hello, I'm "+ this.firstName +". I'm studying "+ this.subject +".");

};

Student.prototype.sayGoodBye = function(){

console.log("Goodbye!");

};

var student1= new Student("Janet","Applied Physics");

student1.sayHello();

student1.walk();

student1.sayGoodBye();

console.log(student1 instanceof Person);

console.log(student1 instanceof Student);

```

-`使用面向对象思想实现一个购物车模块`

-`item格式-{ item:'bread', price:'0.5'}`

-`addItem -添加到购物车`

-`getItemCount -获取购物车的item数目`

-`getTotal -获取总价格`

>\*\*`12:总结`\*\*

```css

本节课介绍了Javascript面向对象编程的基本概念,从Namespace、Class、Object、Property、Method、Constructor、Inheritance、Encapsulation和Polymorphism几个方面介绍了面向对象编程中的核心概念  
```

##(四十五六)JavaScript自定义事件

>\*\*`1:基本概念`\*\*

-`系统内发生的动作或者发生的事情`

-`系统会在事件出现时触发某种信号`

-`提供一个自动加载某种动作的机制`

```css

var myButton = document.getElementById('button');

myButton.onclick = function(){

alert('点击了button');

}

myButton = null;

```

>\*\*`2:事件三要素`\*\*

-`事件源-触发(被)事件的元素`

-`事件类型-事件的触发方式(例如鼠标点击或键盘点击)`

-`事件处理程序-事件触发后要执行的代码`

>\*\*`3:事件冒泡(DOM)`\*\*

![image](./bubble.png)

-`触发阶段`

-`捕获阶段`

-`冒泡阶段`

>\*\*`4:创建自定义事件`\*\*

```javascript

var event = new Event('myEvent');

/\*

var event = document.createEvent('Event');

event.initEvent('myEvent', true, true);

\*/

myButton.addEventListener('myEvent', function (e){

debugger;

}, false);

myButton.dispatchEvent(event);

```

>\*\*`5:添加自定义数据`\*\*

```javascript

var event = new CustomEvent('myEvent',{'detail':{

time: new Date().toLocaleDateString()

}});

myButton.addEventListener('myEvent', function (e){

debugger;

}, false);

myButton.dispatchEvent(event);

```

>\*\*`6:仿JQ封装自定义事件(标准浏览器)`\*\*

```javascript

var $= function(el){

return new \_$(el);

};

var \_$= function(el){

this.el =(el && el.nodeType ===1)? el: document;

};

\_$.prototype ={

constructor:\_$,

addEvent: function(type, fn, capture){

var el = this.el;

if (window.addEventListener){

el.addEventListener(type, fn, capture);

var ev = document.createEvent("HTMLEvents");

ev.initEvent(type, capture || false, false);

}

return this;

},

fireEvent: function(type){

var el = this.el;

if (typeof type ==="string"){

if (document.dispatchEvent){

if (el["ev"+ type]){

el.dispatchEvent(el["ev"+ type]);

}

}

}

return this;

},

removeEvent: function(type, fn, capture){

var el = this.el;

if (window.removeEventListener){

el.removeEventListener(type, fn, capture || false);

}

return this;

}

};

```

>\*\*`7:总结`\*\*

```css

本节课介绍了Javascript中事件的基本概念及特点,通过两个例子介绍了自定义事件的使用方法,最后结合仿照JQ实现一个事件绑定对象  
```

##(四十七)JavaScript实现Ajax

>\*\*`1:基本概念`\*\*

-`XMLHttpRequest -网络请求模块(浏览器网络线程)`

-`用于从URL获取数据`

-`页面的无刷新请求数据`

-`可以用于获取多种类型的数据(ftp、文件)`

>\*\*`2:实例化XMLHttpRequest`\*\*

```css

var xhr = new XMLHttpRequest();

```

>\*\*`3:初始化请求`\*\*

```css

/\* method -请求方式\*/

/\* url -请求地址\*/

/\* async -是否异步\*/

/\* user -用户名\*/

/\* password -密码\*/

xhr.open(method, url, async, user, password);

```

>\*\*`4:发送请求(encodeURIComponent)`\*\*

```css

xhr.send(data);

```

>\*\*`5:接收网络请求返回`\*\*

-`responseText -请求返回的数据内容`

-`responseXML -如果响应内容是text/xml、application/xml,这个属性将保存响应数据的XML、DOM文档`

-`status -响应的HTTP状态,如200304404等`

-`statusText - HTTP状态说明`

-`readyStatus -请求/响应过程的当前活动阶段`

-`0-未调用open`

-`1-已调用open但未发送`

-`2-已调用send`

-`3-已接收到请求返回的数据`

-`4-请求已完成`

-`timeout -设置请求超时时间`

```javascript

xhr.onreadystatechange =()=>{

if (xhr.readyStatus ===4){

/\* HTTP 状态在200-300之间表示请求成功\*/

/\* HTTP 状态为304表示请求内容未发生改变,可直接从缓存中读取\*/

if (xhr.status >=200&&

xhr.status <300||

xhr.status ==304){

console.log('请求成功', xhr.responseText)

}

}

}

```

>\*\*`6:封装Promise`\*\*

```javascript

function ajax (options){

/\*获取请求地址\*/

let url = options.url;

/\*获取请求方法\*/

const method = options.method.toLocaleLowerCase()||'get';

/\*默认异步\*/

const async = options.async != false;

/\*获取请求request数据\*/

const data = options.data;

/\*实例化XMLHttpRequest \*/

const xhr = new XMLHttpRequest();

/\*设置超时时间\*/

if (options.timeout && options.timeout >0){

xhr.timeout = options.timeout

}

return new Promise ((resolve, reject)=>{

/\*添加超时回调\*/

xhr.ontimeout =()=> reject && reject('请求超时');

/\*成功回调\*/

xhr.onreadystatechange =()=>{

if (xhr.readyState ==4){

if (xhr.status >=200&& xhr.status <300|| xhr.status ==304){

resolve && resolve(xhr.responseText);

} else {

reject && reject();

}

}

}

/\*失败回调\*/

xhr.onerror = err => reject && reject(err);

/\*拼接参数\*/

let paramArr =[];

let encodeData;

if (data instanceof Object){

for (let key in data){

paramArr.push( encodeURIComponent(key)+'='+ encodeURIComponent(data[key]));

}

encodeData = paramArr.join('&');

}

/\* get请求\*/

if (method ==='get'){

const index = url.indexOf('?')

if (index ===-1) url +='?'

else if (index !== url.length -1) url +='&'

url += encodeData

}

/\*初始化请求\*/

xhr.open(method, url, async);

/\* get直接发送拼接的URL \*/

if (method ==='get') xhr.send(null);

if (method ==='post'){

xhr.setRequestHeader('Content-Type','application/x-www-form-urlencoded;charset=UTF-8')

xhr.send(encodeData)

}

})

}

```

>\*\*`7:总结`\*\*

```css

本节课介绍了Javascript中XMLHttpRequest对象的基本属性和方法,从实例化、初始化、发送和接受四个阶段完成了Ajax网络请求核心内容封装  
```

##(四十八)排序算法

>\*\*`1:基本概念`\*\*

-`时间复杂度-算法执行所耗费的时间`

```css

/\* O(N1)\*/

for (var i =0; i < data.length; i++){

...

}

/\* O(N2)\*/

for (var i =0; i < data.length; i++){

for (var j =0; j < data.length; j++){

...

}

}

```

-`空间复杂度-运行一个程序所需内存的大小`

-`内排序-所有排序操作都在内存中完成`

>\*\*`2:冒泡排序(Bubble Sort)`\*\*

![iamge](./bubble.gif)

-`复杂度- O(N2)`

-`实现原理`

-`重复地走访过要排序的数列`

-`一次比较两个元素`

-`如果它们的顺序错误就把它们交换过来`

-`代码实现`

-`两层循环嵌套`

-`外循环遍历数组的每一项`

-`内循环用于两两元素比较`

-`每次内循环减1`

```javascript

function bubbleSort(arr){

var length = arr.length;

for (var i =0; i < length; i++){

for (var j =0; j < length -1- i; j++){

if (arr[j]> arr[j+1]){

[arr[j], arr[j+1]]=[arr[j+1], arr[j]];

}

}

}

return arr;

}

```

>\*\*`3:选择排序(Selection Sort)`\*\*

![iamge](./selection.gif)

-`复杂度- O(N2)`

-`实现原理`

-`从未排序序列中找到最小(大)存放到起始位置`

-`再从未排序序列中重复上一步`

-`代码实现`

-`两层循环嵌套`

-`外循环遍历数组的每一项`

-`内循环记录找到的最小或最大值`

-`每次内循环减1`

```javascript

function selectionSort(arr){

var length = arr.length;

var minIndex, temp;

for (var i =0; i < length -1; i++){

minIndex = i;

for (var j = i +1; j < length; j++){

if (arr[j]< arr[minIndex]){

minIndex = j;

}

}

[arr[minIndex], arr[i]]=[arr[i], arr[minIndex]];

}

return arr;

}

```

>\*\*`4:快速排序(Quick Sort)`\*\*

![iamge](./quick.gif)

-`复杂度- O(N \* log2N)`

-`实现原理`

-`通过递归的方式将数据依次分解为包含较小元素和较大元素的不同子序列`

-`代码实现`

-`找到一个数作为参考`

-`比这个数字大的放在数字左边,比它小的放在右边`

-`分别再对左边和右变的序列做相同的操作`

```javascript

function partition(arr, low, high){

let pivot = arr[low];

while (low < high){

while (low < high && arr[high]> pivot){

-high;

}

arr[low]= arr[high];

while (low < high && arr[low]<= pivot){

++low;

}

arr[high]= arr[low];

}

arr[low]= pivot;

return low;

}

function quickSort(arr, low, high){

if (low < high){

let pivot = partition(arr, low, high);

quickSort(arr, low, pivot -1);

quickSort(arr, pivot +1, high);

}

return arr;

}

```

>\*\*`5:插入排序(Insertion Sort)`\*\*

![iamge](./insertion.gif)

-`复杂度- O(N2)`

-`实现原理`

-`构建有序序列`

-`未排序数据在已排序序列中从后向前扫描`

-`找到正确位置插入`

-`代码实现`

-`两层循环嵌套`

-`创建已排序数组,起始包含数组的第一个元素`

-`比这个数字大的放在数字左边,比它小的放在右边`

```javascript

function insertSort(arr){

for(let i =1; i < arr.length; i++){

for(let j = i; j >0; j-){

if(arr[j]< arr[j-1]){

[arr[j], arr[j-1]]=[arr[j-1], arr[j]];

} else {

break;

}

}

}

return arr;

}

```

>\*\*`6:课后练习`\*\*

-`将下列数组使用不同的方法进行排序,查看所花费时间并思考排序结果`

<details>

<summary><font>详情</font></summary>

var originArray =[3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36,26,27,2,46,4,19,50,48,3,44,38,5,47,15,36];

</summary>

</details>

>\*\*`7:总结`\*\*

```css

本节课介绍了常用四种排序算法的基本原理和代码实现思路,并使用JavaScript代码实现了四种排序函数  
```

##(四十八)JavaScript模块化开发

>\*\*`1:发展历史`\*\*

-`早期的Javascript -<script>标签引入-没有模块化`

-`简单粗暴`

-`逻辑混乱`

-`页面复杂,可维护性差`

-`全局变量暴露`

-`文件引入顺序`

-`JavaScript引入模块化`

-`避开全局污染`

-`模块复用,提高开发效率与协作`

-`模块功能单一职能方便维护`

-`解决文件依赖顺序`

-`CommonJs`

-`AMD -异步模块`

-`CMD -通用模块`

>\*\*`2:CommonJs`\*\*

-`CommonJs社区贡献`

-`规范`

-`定义模块标识规范`

-`全局函数require -传入模块标识来引入其他模块(暴露API)`

-`模块嵌套-依次加载引用模块`

-`通过exports暴露API`

-`定义模块`

```css

var basic =1;

var increase = function(value){

return value + basic;

}

module.exports.increase = increase;

```

-`引入模块`

```css

var myModule = require('./myModule.js');

console.log(myModule.increase(1));

```

>\*\*`3:AMD(Asynchronous Module Definition)`\*\*

-`异步加载模块`

-`回调函数中执行逻辑`

-`使用define定义模块`

-`define(id?, dependencies?, factory)`

-`模块的标识(名字)`

-`模块依赖(数组)-['module1','module2']`

-`模块逻辑`

-`return`

-`exports`

-`module.exports`

-`定义模块`

```javascript

define(['increase','reduce'], function(increase, reduce){

return function(value){

return increase(value)+ reduce(value);

}

});

```

-`引入模块`

```javascript

require.config({

baseUrl:"./js",

paths:{

"myModule":"myModule.js"

}

});

require(['myModule'],(myModule)=> console.log(myModule(1)));

```

>\*\*`3:CMD(Common Module Definition)`\*\*

-`玉伯- sea.js`

-`AMD改进版本`

-`按需加载`

-`定义模块`

```javascript

define(function(require, exports, module){

var increase = require('increase');

var reduce = require('./reduce');

module.exports = function(value){

return increase(value)+ reduce(value);

}

});

```

-`引入模块`

```javascript

seajs.use('./myModule', function(myModule){

console.log(myModule(1));

});

```

>\*\*`4:课后练习`\*\*

-`分析AMD与CMD规范对于网页的响应速度,从首次加载与后续操作两个角度考虑`

-`尝试使用AMD规范引入Jquery,并重命名为myJQuery`

>\*\*`5:总结`\*\*

```css

本节课从JavaScript发展历史讲起,介绍了模块化开发的发展历程,并结合实例介绍了CommonJS、AMD、CMD三种规范的基本用法  
```

##(五十)ES6-变量扩展与解构

>\*\*`1:基本概念`\*\*

-`ES6-> ECMAScript6`

-`1996年网景(Netscape)提交给国际标准组织(ECMA)的新语言,希望成为国际标准。`

-`第二年发布了Javascript并称为ECMAScript`

-`商标问题,只有网景公司可以使用Javascript`

-`体现标准的制作者不是个人或者公司`

-`ES6在5.1版本之后更新迭代`

-`ES6第一个版本在2015年6月发布,后续每年六月份发布一次,ES2015、ES2016、ES2017、ES2018`

-`ES6泛指下一代Javascript语言`

>\*\*`2:为什么学习ES6`\*\*

-`大幅度提高开发效率`

-`流行的JavaScript开源框架都使用ES6编写`

-`ES6是下一代JavaScript标准`

>\*\*`3:变量的扩展`\*\*

-`变量的扩展`

-`let -声明变量-只在let命令所在的代码块内有效-块级作用域-不存在变量提升`

```javascript

{

let a =10;

var b =10;

}

a /\* ReferenceError: a is not defined.\*/

b /\*10\*/

for(var i =0; i <5; i++){

setTimeout(function(){

console.log(i);

}, i \*100);

}

```

-`const -只读常量-保证该变量指向得内存地址不变(复杂类型只读特征可能失效)`

```javascript

const a =10;

a =3;/\* TypeError: Assignment to constant variable.\*/

```

>\*\*`4:解构`\*\*

-`解构-按照一定模式从数组和对象中提取值对变量进行赋值-解构失败返回undefined`

-`数组的解构`

```javascript

let [a, b, c]=[1,2,3];

/\* a =1; b =2; c =3\*/

let [week,[[month], year]]=[1,[[2],3]];

/\* week =1; month =2; year =3\*/

let [,, third]=["week","month","year"];

/\* third ='year'\*/

let [x,, y]=[1,2,3];

/\* x =1; y =3\*/

let [head,...tail]=[1,2,3,4];

/\* head =1; tail =[2,3,4]\*/

let [a,[b], d]=[1,[2,3],4];

/\* a =1; b =2; d =4\*/

[a, b]=[b, a];

/\*位置互换\*/

/\*指定默认值\*/

let [x, y ='b']=['a'];

/\* x ='a'; y ='b'\*/

function f(){

console.log('参数错误!');

}

let [x = f()]=[];

```

-`对象的解构-没有顺序,匹配属性名`

```javascript

let { month, week }={ week:'aaa', month:'bbb'};

/\* week ='aaa'; month ='bbb'\*/

let { log, sin, cos }= Math;

/\* log = Math.log; sin = Math.sin; cos = Math.cos \*/

let obj ={

p:[

'Hello',

{ y:'World'}

]

};

let { p:[x,{ y }]}= obj;

/\* x ='hello'; y ='word';\*/

let { week:{ month }}={ year:'year'};

/\* TypeError: Cannot destructure property `month` of 'undefined' or 'null'.\*/

```

>\*\*`5:课后练习`\*\*

-`下列代码的执行结果是`

```javascript

let obj ={

a:1,

b:2,

c:{

d:{

e:5,

},

},

};

let { a, b, c:{ d }}= obj;

console.log(d);

```

-`下列代码的执行结果是`

```javascript

let [x =1, y = x]=[];

console.log(x);

console.log(y);

let [x =1, y = x]=[2];

console.log(x);

console.log(y);

let [x =1, y = x]=[1,2];

console.log(x);

console.log(y);

let [x = y, y =1]=[];

console.log(x);

console.log(y);

```

-`改写下列函数,使输出变为0-9`

```javascript

var funcs =[];

for (var i =0; i <10; i++){

funcs.push(function(){

console.log(i)l

})

};

funcs.forEach(function(func){

func();

});

```

>\*\*`6:总结`\*\*

```css

本节课从ES6的发展历史入手,简要介绍了ES6的主要优势,重点介绍了ES6中变量的扩展与解构两个常用概念  
```