آموزش CSS

جلسه پنجم

آموزش کار با :

- position
- ارث بری
- اولویت بندی
- خاصیت آبشاری

کاربرد Position در Css

- ویژگی **position** به ما اجازه می دهد تا تعیین کنیم که آیا می خواهیم موقعیت یک عنصر در صفحه مشخص شود یا نه. اگر می خواهیم آن عنصر موقعیت مشخصی داشته باشد موقعیت عنصر را در صفحه تعیین می کند. چندین مقدار را می توان برای این ویژگی قرار داد.
  - **relative**

در این حالت ، چنانچه برای عنصر توسط یکی از خواص left , right , top و bottom مقداری تعیین شود ، آن مقدار به موقعیت عادی آن عنصر اضافه شده و نمایش داده می شود . برای مثال چنانچه مقدار خاصیت left = 20px در نظر گرفته شود ، 20px به موقعیت قرار گیری عنصر در سمت چپ اضافه می شود .

  - **Absolute**

این مقدار به ما اجازه می دهد تا یک عنصر از صفحه را در هر موقعیتی نسبت به بالا، راست، پایین یا چپ صفحه که بخواهیم نمایش دهیم.

  - **Fixed**

این گزینه به ما اجازه می دهد تا موقعیت یک عنصر صفحه را در پنجره مرورگر مشخص کنیم. در صورتی که برای یک عنصر از این ویژگی استفاده کنیم، موقعیت آن عنصر حتی در صورتی که به بالا و پایین صفحه برویم یا اندازه پنجره مرورگر را تغییر دهیم به همان صورت قبلی باقی می ماند. البته اینترنت اکسپلورر در ویندوز از این ویژگی پشتیبانی نمی کند ولی اینترنت اکسپلورر در مکینتاش همچنین مرورگرهای نت اسکیپ از این ویژگی پشتیبانی می کنند.

ارث بری در Css

ارث بری (وراثت) مفهومی است که در بحث شی گرایی در زبان های برنامه نویسی سطح بالا همچون زبان برنامه نویسی جاوا یا سی شارپ و ... کاربرد دارد اما در زبان CSS مفهموم ارث بری بسیار ساده تر می باشد و اگر بخواهیم آن را به زبان ساده بیان کنیم می توان آن را " ارث بری فرزند از برخی ویژگی های عنصر پدر " تعریف کرد.

همانطور که می دانید عناصر صفحه داخل تگ body قرار می گیرند،بنابراین هر عنصری که داخل صفحه قرار می دهیم فرزند تگ body می باشد،حال اگر در CSS استایل خاصی را برای تگ bodyدر نظر بگیریم،عناصر دیگر نیز برخی از این استایل ها را به ارث خواهند برد.

- **قابل ارث بری عناصر در CSS**

قابل ارث بری عناصر در CSS یعنی اینکه یک عنصر می تواند مقدار ویژگی هایش را از پدرش به ارث ببرد، و در کل تمامی عناصر در یک صفحه وب هم می توانند پدر باشند و هم می توانند فرزند باشند.

- **خب چجوری بفهمیم یک عنصر پدر است یا فرزند؟!**

فرض می گیریم یک عنصر داریم به اسم A و یه عنصر دیگر به اسم B، حال عنصر B را داخل عنصرA می گذاریم ، در نتیجه عنصر A پدر و B فرزند است و یا برعکس هیچ فرقی ندارد.

ارث بری در Css

برای درک بهتر موضوع به مثال زیر توجه کنید:

```
<html>
    <head>
        <title>کلاس طراحی وب</title>
        <style>
     body {
            direction: rtl;
            font-weight: bold;
            color: red;
       }
    </style>
    </head>
    <body>
        <p>آشنایی با وراثت در سی اس اس</p>
    </body>
</html>
```

ارث بری در Css

همانطور که در کد فوق مشاهده می کنید یک تگ p که حاوی عبارت " آشنایی با وراثت در سی اس اس " می باشد را داخل تگ body قرار داده ایم،اما به تگ p هیچ استایل خاصی را اختصاص نداده ایم. خروجی قطعه کد بالا در مرورگر به صورت زیر خواهد بود:

از آنجا که تگ p فرزند تگ body محسوب می شود، از اینرو استایلی که برای تگ bodyدر نظر گرفته ایم برای تگ p هم اعمال می شود. حال اگر برای تگ p نیز یک رنگ متفاوت در نظر بگیریم چه اتفاقی خواهد افتاد؟ کدهای CSSخود را تغییر می دهیم و نتیجه را با هم بررسی خواهیم کرد.

ارث بری در Css

```
<html>
    <head>
        <title>کلاس طراحی وب</title>
        <style>
     body {
          direction: rtl;
          font-weight: bold;
          color: red;
       }
     p {
          color: blue;
       }

    </style>
    </head>
    <body>
        <p>آشنایی با وراثت در سی اس اس</p>
    </body>
</html>
```

خروجی کدهای مقابل در مرورگر به صورت زیر خواهد بود :

نکته :

با توجه به این که تگ p فرزند تگ body است از اینرو یک تضادی در کد فوق پیش می آید. در چنین مواقعی مقادیری که برای تگ فرزند در نظر گرفته ایم اعمال خواهند شد.

مثال هایی از ارث بری در Css

- **مثال های مختلف از نشان دادن پدر و فرزندی عناصر در یک صفحه وب**

❖ **مثال شماره ۱ :** تگ DIV پدر و تگ P فرزند است

```
<div>
    <p>من فرزند تگ دایو هستم</p>
</div>
```

❖ **مثال شماره ۲ :** تگ P پدر و تگ DIV فرزند است

```
<p>
    <div>من فرزند تگ پی هستم</div>
</p>
```

❖ **مثال شماره ۳ :** تگ Span فرزند تگ P است

```
<p>
    <span>من فرزند تگ پی هستم</span>
</p>
```

مثال هایی از ارث بری در Css

- **مثال های مختلف از نشان دادن پدر و فرزندی عناصر در یک صفحه وب**

- برای نمونه اگر یک **color** و **font-family** را روی عنصری تنظیم کنید، هر عنصر درون آن نیز با آن رنگ و فونت استایل‌بندی می‌شود مگر این که مقادیر رنگ و یا فونت دیگری مستقیماً برای آن‌ها تعیین شده باشد.

```
body {
    color: blue;
}

span {
    color: black;
}
```

```
<p>Hello. How are you? </p>
<p>This is a <span>Test</span>.</p>
```

و در نتیجه خروجی نهایی :

Hello. How are you?

This is a Test

ارث بری در Css

- **خواصی که به ارث برده می شوند**

تنها چند خاصیت CSS وجود دارند که می‌توانند مقدار خود را از والد به ارث ببرند که معمولا خواص مربوط به متن هستند.

- **کنترل کردن وراثت**

CSS چهار مقدار مشخصه برای کنترل کردن وراثت ارائه کرده است. هر مشخصه CSS سه مقدار می‌پذیرد.

- **Inherit** - مقدار مشخصه اعمال شده روی یک عنصر منتخب را به همان مقدار عنصر والد تعیین می‌کند. این وضعیت در عمل موجب فعال شدن ارث‌بری می‌شود.
- **Initial** – مقدار مشخصه اعمال شده روی یک عنصر منتخب را به همان مقدار تعیین شده برای عنصر در استایل‌شیت پیش‌فرض مرورگر تنظیم می‌کند. اگر هیچ مقداری از سوی استایل‌شیت پیش‌فرض مرورگر تعیین شده نباشد و مشخصه به صورت طبیعی ارث‌بری کند، در این صورت مقدار مشخصه به جای آن به صورت inherit خواهد بود.
- **unset** - مشخصه را به مقدار طبیعی‌اش ریست می‌کند. معنی این حرف آن است که اگر مشخصه به صورت طبیعی ارث‌بری داشته باشد، مانند inherit عمل می‌کند و در غیر این صورت مانند initial عمل خواهد کرد.

نکته: یک مقدار جدیدتر به نام revert نیز وجود دارد که پشتیبانی مرورگر محدودی دارد.

اولویت بندی عناصر در Css

سطح خصوصیت یا Specificity به شیوه تصمیم‌گیری مرورگر در مورد اعمال قواعد چندگانه با سلکتورهای مختلف گفته می‌شود، اما می‌تواند روی یک عنصر واحد نیز اعمال شود. Specificity اساساً معیاری برای میزان خاص بودن انتخاب یک سلکتور به صورت زیر است:

- اگر سلکتور یک عنصر سطح خصوصیت پایین‌تری داشته باشد – همه عناصر از نوع آن که در یک صفحه ظاهر می‌شوند را انتخاب می‌کند و از این رو امتیاز پایین‌تری به دست می‌آورد.
- یک سلکتور کلاس که سطح خصوصیت بالاتری دارد – تنها عناصری را روی صفحه انتخاب می‌کند که یک مقدار خصوصیت class خاص داشته باشند و از این رو امتیاز بالاتر می‌گیرد.

اینک نوبت به بررسی این مفاهیم با مثال می‌رسد. در ادامه دو قاعده را می‌بینید که می‌توانند روی h1 اعمال شوند. h1 زیر در نهایت به رنگ قرمز در می‌آید. زیرا سلکتور کلاس قاعده خود را به سطح خصوصیت بالاتر می‌دهد و از این رو علی‌رغم این که قاعده با همان عنصر در ادامه کد مجدداً ظاهر می‌شود، اما سلکتور اول اعمال خواهد شد.

```
<h1 class="main"> This is my Heading </h1>
```

```
.main {
color: red;
}
h1 {
color: blue;
}
```

و در نتیجه خروجی نهایی :

This is my Heading.

اولویت بندی عناصر در Css

اولویت بندی به چه معنا است:

به طور مثال ما می آییم بصورت داخلی (استفاده از تگ style در قسمت head صفحه) برای عناصرمان در CSS استایل تعریف می کنیم ، مثلا می گوییم رنگ زمینه تگ p آبی باشد.

حال در همین زمان باز مثلا در فایل CSS خارجی برای همون تگ p تنظیم می کنیم که رنگ زمینه اش قرمز باشد؟! خب حالا شما بگویید کدام دستور باید اجرا شود؟؟

اینجاست که بحث اولویت بندی وسط می اید، یعنی ما باید بدانیم این اولویت بندی ها یا قوانین در CSS چه ها هستند و بتوانیم به بهترین حالت ممکن از آنها استفاده نماییم.

یک المنت HTML می تواند توسط چندین دستور CSS مورد هدف قرار گیرد. بگذارید از یک پاراگراف ساده بعنوان مثال استفاده کنیم:

```
<p class="message" id="introduction">
کلاس طراحی وب
</p>
```

اولویت بندی عناصر در Css

در این پاراگراف ما می توانیم فقط با استفاده از نام تگ آن، در رنگ آن تغییر ایجاد کنیم:

```
p { color: blue;}
```

یا می توانیم از نام کلاس آن استفاده کنیم:

```
.message { color: green;}
```

و یا از شناسه آن استفاده کنیم:

```
#introduction { color: red;}
```

از آنجا که مرورگر تنها می تواند یک رنگ را برای اعمال برروی این پاراگراف، انتخاب کند، پس باید تصمیم بگیرد که کدام دستور CSS اولویت بیشتری نسبت به مابقی دستورات دارد. این همان اولویت بندی در CSS است.

در مثال ما، پاراگراف **قرمز** خواهد بود بدین دلیل که انتخابگر شناسه # id خاص تر از مابقی موارد است، بنابراین مهم تر از انتخاب های دیگر است.

اولویت بندی عناصر در Css

- **نحوه اولویت بندی عناصر در CSS**

بطور کلی می توانیم اولویت بندی هایمان را به ۴ دسته تقسیم کنیم یا در واقع می توانیم بگوییم ۴ دسته اصلی انتخابگر داریم و به ترتیب از شماره ۱ تا ۴ اولویت بندی می شوند :

۱. استایل های درون خطی یا Inline که با استفاده از صفت Style ایجاد می شوند (اولویت اول)
۲. آی دی ها یا ID (اولویت دوم)
۳. (کلاس ها یا Class) و (صفت ها یا Attributes) و (شبه کلاس ها یا Pseudo Classes) (اولویت سوم)
۴. تگ ها یا عناصر (مثه تگ p,h1,divو..) و شبه عناصر یا Pseudo Elements (اولویت چهارم)

مقیاس گذاری در اولویت بندی عناصر در Css

یک روش سریع برای پی بردن به اینکه یک دستور CSS به چه میزان "مهم" است، اندازه گیری میزان خاص بودن انتخابگرها است:

ما برای صفت Style عدد ۱۰۰۰ رو در نظر می گیریم و عدد ۱۰۰ برای ID و عدد ۱۰ برای صفات، کلاس ها و شبه کلاس ها و عدد ۱ برای تگ ها و شبه عناصر.

عنصر اولویت دار	ارزش عددی
صفت Style	۱۰۰۰
صفت ID	۱۰۰
صفت Class	۱۰
انتخابگر صفت	۱۰
شبه کلاس ها	۱۰
تگ ها	۱
شبه عناصر	۱

مقیاس گذاری در اولویت بندی عناصر در Css

مثال:

ما یک سری دستور داریم که می خواهیم با استفاده از اعداد بالا مشخص کنیم که اولویت اول با کدام عنصر می باشد، دستورات ما عبارتند از :

A : h1

B : #test h1

C : <h1 style="color:blue">کلاس طراحی وب</h1>

در بخش A ما یک تگ داریم به اسم h1 ، همانطور که گفتیم ارزش تگ ها عدد ۱ می باشد، پس فعلا یک ۱ را کنار می گذاریم، بعد در بخش B ما یک ID داریم به اسم #test و یه تگ به اسم h1.

ارزش ID ۱۰۰ و ارزش هر تگ ۱ پس جمع بخش B می شود ۱۰۱ و حالا بخش C را بررسی می کنیم، در بخش C داخلی ترین دستور صفت Style هست که ارزشش ۱۰۰۰ هست.

پس به اینصورت داریم :

جمع کل ارزش بخش A می شود ۱

جمع کل ارزش بخش B می شود ۱۰۱

ارزش صفت Style موجود در بخش C هم می شود ۱۰۰۰ ،

پس **اولویت** اجرای دستورات اول با **بخش C** می باشد.

خاصیت آبشاری بودن در Css

استایل‌شیت‌های آبشاری در ساده‌ترین سطح به این معنی است که ترتیب قواعد CSS مهم هستند. زمانی که دو قاعده که Specificity یکسانی دارند روی یک عنصر اعمال می‌شوند، آن که در CSS بعدتر می‌آید، همان است که در عمل اعمال خواهد شد.

در مثال زیر دو قاعده داریم که روی h1 اعمال می‌شوند. h1 در نهایت به رنگ آبی در می‌آید. این قواعد یک سلکتور یکسان دارند و از این رو واجد specificity یکسانی هستند. بنابراین آن که در ترتیب بعدتر می‌آید برنده می‌شود.

```
<h1> This is my Heading </h1>
```

```
h1 {
color: red;
}
h1 {
color: blue;
}
```

و در نتیجه خروجی نهایی :

This is my Heading.

درک آبشاری بودن در Css

حالا نگاهی دقیق خواهیم داشت به طرز کار آبشاری بودن در مورد این که کدام قواعد CSS در زمان اعمال بیش از یک استایل‌بندی روی یک عنصر عمل می‌کنند.

در این زمینه باید سه عامل را در نظر بگیریم که در ادامه به ترتیب کاهش اهمیت رتبه‌بندی شده‌اند. در واقع موارد بالاتر بر موارد پایین‌تر تقدم دارند.

- **اهمیت**
- **سطح خصوصیت**
- **ترتیب در سورس کد**

ما سه عامل فوق را به ترتیب معکوس بررسی می‌کنیم تا ببینیم مرورگرها در زمان اعمال CSS دقیقاً چگونه عمل می‌کنند.

ترتیب در سورس کد

پیش‌تر دیدیم که ترتیب در سورس چه اهمیتی در آبشاری بودن دارد. اگر بیش از یک قاعده داشته باشید که وزن دقیقاً یکسانی داشته باشند، آن قاعده‌ای برنده خواهد بود که در CSS متاخرتر است. این وضعیت را می‌توان این گونه تصور کرد که قواعد نزدیک‌تر به خود عنصر انواع قدیمی‌تر را لغو می‌کنند تا این که آخری در زمینه استایل‌بندی عنصر برنده می‌شود.

درک آبشاری بودن در Css

سطح خصوصیت

حال که این واقعیت را درک کردید که ترتیب در سورس کد اهمیت دارد، باید بدانید برخی موقعیت‌ها وجود دارند که قاعده‌ای در سورس کد در مراحل متأخرتر می‌آید، اما یک قاعده متقدم و معارض اعمال می‌شود. دلیل این وضع آن است که قاعده متقدم سطح خصوصیت (Specificity) بالاتری دارد، یعنی خاص‌تر است و از این رو از سوی مرورگر به عنوان قاعده‌ای که باید عنصر را استایل‌بندی کند انتخاب می‌شود.

چنان که دیدیم، سلکتور کلاس، وزن بیشتری از یک سلکتور عنصر دارد و از این رو مشخصه‌های تعریف شده روی کلاس، انواعی را که مستقیماً روی آن عنصر اعمال شده‌اند نقض می‌کنند.

نکته‌ای که باید در این جا توجه کرد این است که گرچه در این جا در مورد سلکتورها و قواعدی که روی عنصر منتخب اعمال می‌شوند صحبت می‌کنیم، اما این کل قاعده نیست که بازنویسی می‌شود، بلکه فقط مشخصه‌هایی که تطبیق می‌یابند بازنویسی خواهند شد.

این رفتار از بروز تکرار در CSS جلوگیری می‌کند. یک رویه معمول این است که استایل‌های عمومی برای عناصر ابتدایی تعریف می‌شوند و سپس کلاس‌هایی برای آن موارد که تفاوت دارند ایجاد می‌شود.

درک آبشاری بودن در Css

به مثال زیر توجه کنید.

```
<h2>Heading with no class</h2>
<h2 class="small">Heading with class of small</h2>
<h2 class="bright">Heading with class of bright</h2>
```

و در نتیجه خروجی نهایی :

```
h2 {
    font-size: 2em;
    color: #000;
    font-family: Georgia, 'Times New Roman', Times, serif;
}

.small {
    font-size: 1em;
}

.bright {
    color: rebeccapurple;
}
```

درک آبشاری بودن در Css

کلید واژه important

یک بخش خاصی در CSS وجود دارد که می‌تواند برای کنار زدن همه محاسبات فوق مورد استفاده قرار گیرد و آن !important است، با این حال باید در مورد استفاده از آن هوشیار باشیم. این کلیدواژه برای این طراحی شده است که یک مشخصه و مقدار آن بالاترین سطح خصوصیت را به دست آورند و از این رو قواعد عادی آبشار را بر هم می‌زند.

درک آبشاری بودن در Css

به مثال زیر توجه کنید. در این جا دو پاراگراف داریم که یکی از آن‌ها یک ID دارد.

```
<p class="better">This is a paragraph.</p>
<p class="better" id="winning">One selector to rule them all!</p>

#winning {
    background-color: red;
    border: 1px solid black;
}

.better {
    background-color: gray;
    border: none !important;
}

p {
    background-color: blue;
    color: white;
    padding: 5px;
}
```

و در نتیجه خروجی نهایی :